Scaling up Multi-agent Reinforcement Learning with Mean Field Games and Vice-versa

Sarah Perrin

Université de Lille - Faculté des Sciences et Technologies
École Doctorale Mathematics and Digital Sciences

Thèse de Doctorat

Spécialité **Informatique**

présentée par
Sarah Perrin

Scaling up Multi-agent Reinforcement Learning with Mean Field Games and Vice-versa

---

Mise à l'échelle de l'apprentissage par renforcement multi-agent grâce aux jeux à champ moyen et vice-versa

---

sous la direction d'**Olivier Pietquin**
et de **Romuald Élie**

A été soutenue publiquement à **Villeneuve d'Ascq**, le **8 décembre 2022** devant le jury composé de

M. François **Charpillet**	Directeur de Recherche, Université de Lorraine, CNRS, Inria, LORIA	Président du Jury
M. François **Delarue**	Professeur, Université Côte d'Azur, CNRS	Rapporteur
M. Marcello **Restelli**	Associate Professor, Politecnico di Milano	Rapporteur
M^{me} Émilie **Kaufmann**	Chargée de Recherche, CNRS, Université de Lille	Examinatrice
M. Olivier **Pietquin**	Professeur, Google Research	Directeur de thèse
M. Romuald **Élie**	Professeur, DeepMind	Co-directeur de thèse
M. Gergely **Neu**	Research Assistant Professor, Universitat Pompeu Fabra	Invité
M. Matthieu **Geist**	Professeur, Google Research	Invité

Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStAL),
UMR 9189 Équipe Scool, 59650, Villeneuve d'Ascq, France

Université de Lille

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

Inria

À mes grands-parents, Charles-Joël et Pierrette.

Remerciements

Je tiens tout d'abord à remercier très sincèrement mes deux directeurs de thèse, Olivier et Romuald, qui m'ont toujours poussée à me dépasser tout en me soutenant avec bienveillance. Sachez que je vous en veux toujours de m'avoir forcée à rentrer en France quand nous étions à Berkeley et que la pandémie du covid a éclaté. Heureusement que j'ai pu aller à Montréal ! Olivier, tu as su m'aider à toujours garder les bonnes questions de recherche en tête. Merci pour avoir passé des nuits blanches à reformuler les abstracts et introductions de nos papiers juste avant de les soumettre, en améliorant grandement leur qualité au passage. Tu as toujours fait en sorte que je puisse grandir sur le plan professionnel en m'offrant des opportunités et surtout en m'apprenant à être greedy, un comble pour un chercheur en reinforcement learning ! Merci en particulier pour ce séjour à Montréal qui restera gravé dans ma mémoire. Sur une note moins formelle, merci de m'avoir faite découvrir les meilleurs restaurants et bars de Lille, et pour toutes ces soirées où tu nous as accueillis chez toi, qui se terminaient bien trop tard mais qui laissent des souvenirs ineffaçables (bien que flous). Heureusement que les parties de billard étaient accompagnées de musique de qualité, je devrais d'ailleurs remercier au passage Corinne Charby et Bibi Flash. Romuald, merci pour ton soutien et merci d'avoir toujours fait en sorte que je me sente bien tout au long de ma thèse. Tu as veillé à ce que je sois entourée par les bonnes personnes pour pouvoir grandir autant sur le plan académique que personnel. Tu as su me garder motivée et me faire voir le bon côté des choses même dans les moments difficiles.

Je tiens également à remercier profondément mes plus proches collaborateurs et mentors, Mathieu, Julien et Matthieu, sans qui ces trois années auraient été beaucoup moins fructueuses. Mathieu, tu as été un soutien quotidien pendant ces trois années. Merci pour ces heures de discussions autour des jeux à champ moyen, pour tout le temps passé, de jour comme de nuit, à faire avancer nos projets. Merci pour ta bienveillance, ta bonne humeur et ta gentillesse sans égales, même lorsque tu n'avais dormi que cinq heures cumulées sur les trois derniers jours. Je n'en serais clairement pas là aujourd'hui sans toi. Julien, au début de mon doctorat et malgré le covid, tu t'es impliqué quasi quotidiennement en passant parfois de longs moments à m'aider. Étant donné la qualité de mon code à l'époque, c'était sacrément gentil. Tu m'as guidée académiquement tout en m'offrant de précieux conseils, et m'a intégrée dans tes projets qui sont, par la suite, devenus plusieurs publications de ce manuscrit. Merci Matthieu pour ta disponibilité et merci de m'avoir appris ta rigueur scientifique. J'admire sincèrement ta capacité

d'abstraction et la manière dont tu crées des liens entre différents domaines de recherche. Enfin, merci à mes collaborateurs Paul, Sertan, Mark, Karl, Geogios et Théophile avec qui j'ai eu la chance de travailler.

J'en viens à mes très chers collègues, et maintenant amis, Mathieu, Dorian et Nathan. Mathieu, merci pour ces sessions d'escalades souvent suivies d'un bon dîner et pour nos discussions sans fin. Dorian et Nathan, merci d'avoir été mes compagnons du bureau A06. Le covid nous a empêchés de passer tout le temps que l'on aurait du ensemble mais cela ne nous a pas empêchés d'avoir le temps de faire une virée en Belgique sur un déjeuner pour se fournir les meilleurs bières du monde. Merci également à Marc, Antoine, Jules, Yannis, Reda, Xuedong, Edouard, Matheus, Patrick et Fabien d'avoir égayé mes déjeuners et d'avoir fait vivre nos interminables pauses café. Merci Odalric pour ta bienveillance et de m'avoir intégrée à l'équipe bien que nos sujets étaient différents. Merci Émilie pour ces goûters préparés par tes soins, et merci d'avoir accepté d'être examinatrice lors de ma soutenance. Je remercie Philippe de m'avoir accueillie au sein de Scool. Enfin, je remercie François Delarue, Marcello Restelli, François Charpillet et Gergely Neu d'avoir accepté de faire partie de mon jury.

Mais ma thèse aurait été bien différente sans mon expérience de quelques mois à Montréal, au Mila, l'institut québécois d'intelligence artificielle. Je remercie en particulier Marc G. Bellemare de m'avoir accueillie parmi ses étudiants et m'avoir fait découvrir une autre manière de faire de la recherche. Je tiens également à remercier tous mes collègues et amis rencontrés là-bas. Alexandre, merci pour tous ces déjeuners qui commençaient souvent un peu trop tôt à ton goût. Je n'oublie pas notre promesse de collaborer. Bao, merci de m'avoir fait tant rire et pour ta sensibilité que tu m'as laissée entrevoir. Je remercie également Max, Johan, Nathan, Pierluca, Harley, Jesse, Charline, Linda et Rishab pour nos discussions scientifiques ainsi que pour leur accueil chaleureux au Mila. Merci à Joey, Aarash, Julien et Alex pour cette semaine à Baltimore lorsque nous avons assisté à ICML. Enfin, merci à Antoine, Camille, Joséphine, Clara, Gabriel, François, Chloé, Alex, Vic, Tom, Grégoire et Sarah. Vous êtes désormais de vrais amis et vous m'avez fait voir la vie différemment.

Bien évidemment, tout cela n'aurait pas été possible sans ma famille, qui m'a toujours soutenue et sans qui je n'en serais pas là aujourd'hui. Maman, tu as toujours fait en sorte que je puisse travailler dans les meilleures conditions possibles, en allant jusqu'à sacrifier tes propres vacances pour me tenir compagnie pendant mes révisions de concours. Papa, merci d'avoir éveillé mon intérêt scientifique, en m'apprenant par exemple à résoudre des équations du premier degré quand j'avais dix ans. Je crois que mes amis de CM1 m'ont par la suite trouvée un peu bizarre, en me voyant manipuler des x sur un coin de table. Thierry, merci pour ces soirées passées à m'expliquer mes cours de maths, et de m'avoir convaincue de faire une thèse lorsque j'hésitais. Te voir travailler aussi dur est une grande source d'inspiration. Ensuite, je n'ai pas les mots pour remercier mes grands parents, papi et mamie, qui malheureusement ne sont plus présents. Vous rendre fiers a été ma plus grande motivation. Merci papi de m'avoir

convaincue qu'il fallait absolument aller à Polytechnique si je voulais être astronaute, ce qui ferait de moi une "ingénieure polytechnicienne astronaute cosmonaute". Mamie, je n'oublie pas la promesse que je t'ai faite juste avant que tu partes. Je remercie également ma soeur Lucie pour tous les rires pendant notre enfance, et plus récemment en soirée. Merci à Théo et Eva pour nos nombreuses batailles, dans la neige comme dans l'eau. Enfin, je remercie mes oncles et tantes Sophie, Ekke, Sabine et Alexancre, d'avoir participé à ce qu'on garde une famille aussi soudée, et mes cousins Laurène, Gabriel, Alexandre, Camille, Lucas, Noah et Salomé, pour nos vacances passées à La Féclaz.

J'en viens désormais à mon cercle d'amis. Merci à DH et Vic qui ont vu les débuts un peu difficiles de ma thèse et qui m'ont toujours soutenue. Je n'oublierai jamais nos moments de vie chez nous, passés à mixer, jouer à la switch ou simplement à discuter dans la cuisine. Diane, je tiens à te remercier sincèrement pour ton soutien et de m'avoir toujours poussée vers le haut. J'admire ton énergie et ta détermination. Je remercie également mes amis les plus proches, Antoine, Manon, Jenny, Lina, Stella, Elie, Gauthier, mes amies d'enfance Adèle, Marianne et Clara ainsi que mes amis de Phasm, Thomas, Richard, Barth, Vincent, Manon, Chloé et Clément. Les moments de vie en dehors de la thèse étaient une vraie bouffée d'air frais qui m'ont apporté un équilibre précieux.

Enfin, merci mon Emma.

Résumé

De la propagation d'une épidémie à l'optimisation du trafic routier, en passant par l'étude des environnements biologiques, les systèmes multi-agent sont omniprésents dans la nature et en ingénierie. Cependant, si les progrès en intelligence artificielle et en particulier en apprentissage par renforcement ont permis de résoudre des jeux complexes tels que le Go, Starcraft et le Poker, les méthodes récentes ont toujours du mal à s'attaquer à des applications de plus d'une douzaine de joueurs. Cette difficulté est connue sous le nom de la malédiction des nombreux agents : quand le nombre d'agents augmente, le jeu devient bien plus difficile à résoudre car le nombre d'interactions à étudier entre les joueurs devient intraitable.

Dans cette thèse, nous étudions comment l'apprentissage par renforcement et les jeux à champ moyen peuvent bénéficier mutuellement l'un de l'autre. D'un côté, les jeux à champ moyen peuvent permettre à l'apprentissage par renforcement multi-joueurs de passer à l'échelle en termes de nombre d'agents, étant donné qu'ils comportent par définition une infinité de joueurs. De l'autre côté, l'apprentissage par renforcement s'est avéré efficace pour résoudre des jeux stochastiques et complexes et pourraient ainsi permettre de trouver des équilibres de Nash dans des jeux à champ moyen compliqués, sans avoir à connaître le modèle ou à résoudre un système forward-backward d'équations stochastiques ou aux dérivées partielles.

Au cours de cette dissertation, nous définissons précisément les jeux à champ moyen, les processus décisionnels de Markov et l'apprentissage par renforcement, avant de détailler les différentes configurations que le lecteur peut rencontrer en cherchant à entremêler l'apprentissage par renforcement avec les jeux à champ moyen. Puis, nous présentons une approche unifiée des algorithmes, aussi appelés méthodes itératives, servant à résoudre des jeux à champ moyen, soit à l'aide de la programmation dynamique quand le modèle est connu, soit avec de l'apprentissage par renforcement lorsqu'il ne l'est pas.

Puis, nous zoomons sur deux méthodes itératives : Fictitious Play (FP) et Online Mirror Descent (OMD). Nous prouvons leur convergence vers l'équilibre de Nash sous la condition de monotonicité dans le cas exact, avec ou sans bruit commun, dans des jeux à champ moyen à une ou plusieurs populations. Nous démontrons numériquement leur convergence dans un large set d'exemples et soulignons qu'OMD converge plus rapidement.

Dans la dernière partie, nous proposons trois contributions démontrant que l'apprentissage par renforcement profond peut résoudre des jeux à champ moyen. La première présente comment des agents qui utilisent ce paradigme apprennent à se regrouper ensemble, dans un environnement continu et multi-dimensionnel. Puis, nous nous attaquons à la généralisation par rapport à la distribution initiale, et démontrons que l'apprentissage par renforcement profond permet le calcul de politiques population-dépendantes. Enfin, nous proposons deux algorithmes permettant à FP et OMD de passer à l'échelle, ne requérant pas de sommer ou moyenner des réseaux de neurones.

Abstract

From understanding the spreading of an epidemic to optimizing traffic flow or biological swarming, multi-agent systems are ubiquitous in nature and engineering. However, if recent progress in artificial intelligence and in particular reinforcement learning has allowed to solve complex games such as Go, Starcraft and Poker, recent methods still struggle to tackle applications with more than a dozen a players. This difficulty is known as the curse of many agents: when the number of agents increases, the game becomes computationally way harder to solve as interactions among players become intractable.

In this Ph.D. thesis, we study how reinforcement learning and mean field games can benefit mutually from each other. On one hand, Mean Field Games (MFGs) can help to scale up games and Multi-agent Reinforcement Learning (MARL) in terms of number of agents, as they naturally deal with an infinite number of players. On the other hand, Reinforcement Learning (RL) has proven very efficient to solve complex stochastic games and could thus be used to find Nash equilibria in complicated MFGs without having to know the model or to solve forward-backward system of partial or stochastic differential equations.

During the dissertation, we define precisely mean field games, Markov Decision Processes (MDPs) and reinforcement learning, before detailing the different settings the reader may encounter when intertwining MFGs with RL. Then, we present a unified approach of algorithms, or iterative methods, to solve MFGs, either with Dynamic Programming (DP) when the model is known or reinforcement learning when it is not.

Then, we zoom in two iterative methods: Fictitious Play (FP) and Online Mirror Descent (OMD). We prove their convergence to the Nash equilibrium under the monotonicity condition in the exact case, with or without common noise, in single and multi-populations mean field games. We demonstrate numerically their convergence in a various set of examples and highlight that OMD converges faster.

In the last part, we propose three contributions to demonstrate that Deep Reinforcement Learning (DRL) can solve mean field games. The first one presents how agents using deep reinforcement learning and fictitious play learn to flock in a complex multi-dimensional continuous environment. Then, we tackle the question of generalization in mean field games, and how DRL allows to compute population-dependant policies able to be near-optimal against many distributions. Finally, we propose new state-of-the-art algorithms which are a deep scalable version of both fictitious play and online mirror descent and do not require to average or sum neural networks compared to their previous counterparts.

Contents

List of Acronyms xv

List of Symbols xvii

1 Introduction 1

1.1 Context and Scope 2

1.2 Outline and Contributions 9

I Background and Settings 15

2 Background 17

2.1 From Markov Decision Process to Deep Reinforcement Learning 18

2.2 Mean Field Games: Definition and Settings 28

3 Iterative Methods, Reinforcement Learning for Mean Field Games and Metrics 45

3.1 Iterative methods 46

3.2 Reinforcement learning for Mean Field Games 56

3.3 Metrics and Numerical Experiments 63

II Deep Dive to Iterative Methods: Fictitious Play and Online Mirror Descent 71

4 Fictitious Play 73

4.1 Motivation 74

4.2 Continuous Time Fictitious Play in Mean Field Games 78

4.3 Experiments on Fictitious Play in the Finite Horizon Case . . . 79
4.4 Finite Horizon Mean Field Games with Common Noise . . . 82
4.5 Experiments with Common Noise . . . 83
4.6 Experiment at Scale . . . 85
4.7 Conclusion of the Chapter . . . 85

5 Online Mirror Descent **87**
5.1 Motivation . . . 88
5.2 Preliminaries on Multi-Population Mean Field Games . . . 89
5.3 Online Mirror Descent: Algorithm and Convergence . . . 93
5.4 Numerical Experiments . . . 97
5.5 Conclusion of the Chapter . . . 102

Conclusion of Part I and II **105**

III Deep Reinforcement Learning for Mean Field Games **107**

6 Flocking **109**
6.1 Motivation . . . 110
6.2 The Model of Flocking . . . 111
6.3 Our Approach . . . 112
6.4 Experiments . . . 116
6.5 Conclusion of the Chapter . . . 119

7 Generalization in Mean Field Games **121**
7.1 Motivation . . . 121
7.2 Background and Related Works . . . 122
7.3 Master Policies for MFGs . . . 126
7.4 Algorithm . . . 127
7.5 Numerical Experiments . . . 131
7.6 Conclusion of the Chapter . . . 134

8 Scalable Algorithms **137**

8.1 Motivation . . . 138

8.2 Background . . . 139

8.3 Deep Reinforcement Learning for MFGs . . . 142

8.4 Experiments . . . 146

8.5 Conclusion of the Chapter . . . 152

Conclusion of Part III **155**

9 General Conclusion and Perspectives **157**

9.1 Conclusion on our Contributions . . . 157

9.2 Future Work and Perspectives . . . 159

A Complements on Chapter 2 **163**

A.1 Some applications . . . 163

A.2 An introduction to MFGs in OpenSpiel . . . 164

B Complements on Chapter 4 **167**

B.1 Continuous Time Fictitious Play in Finite Horizon . . . 167

B.2 Continuous Time Fictitious Play in Finite Horizon with Common Noise . . . 170

B.3 Continuous Time Fictitious Play: the γ-discounted case . . . 171

B.4 Algorithms . . . 176

B.5 Linear Quadratic Model . . . 177

B.6 Common Success Metrics in Mean Field Games . . . 179

C Complements on Chapter 5 **181**

C.1 Separability and Monotonicity Imply Weak Monotonicity . . . 181

C.2 Multi-Population Reward . . . 182

C.3 Fictitious Play . . . 183

C.4 Online Mirror Descent Dynamics . . . 186

C.5 Weak monotonicity . . . 188

C.6 Strictly weak monotonicity implies uniqueness . . . 188

C.7 Online Mirror Descent Convergence . . . 189
C.8 Numerical Experiments . . . 190

D Complements on Chapter 6 **197**
D.1 More numerical tests . . . 197
D.2 Normalizing Flows . . . 199
D.3 Visual Rendering with Unity . . . 202

E Complements on Chapter 7 **203**
E.1 Notations . . . 203
E.2 Experimental Details . . . 203
E.3 Learning a Population-dependent Policy with Deep RL . . . 205
E.4 Proof of Theorem 7.1 . . . 205
E.5 On the Convergence of Master Fictitious Play . . . 207

F Complements on Chapter 8 **213**
F.1 Algorithms in the exact case . . . 213
F.2 Deep RL Algorithms . . . 214
F.3 Details on the link between MOMD and regularized MDPs . . . 217
F.4 Hyperparameters sweeps . . . 218

List of Figures **221**

List of Algorithms **226**

List of Tables **227**

List of References **229**

List of Acronyms

A

ADP	Approximate Dynamic Programming

C

CNN	Convolutional Neural Network
CTOMD	Continuous Time Online Mirror Descent , 94

D

DP	Dynamic Programming , 10, 45
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning , 10, 17, 109

F

FCN	Fully-Connected Network
FP	Fictitious Play , 73, 109

I

i.i.d.	independent and identically distributed

L

LQ	Linear Quadratic , 11

M

MARL Multi-agent Reinforcement Learning

MCTS Monte-Carlo Tree Search

MDP Markov Decision Process , 17

MFC Mean Field Control , 5, 17, 45

MFG Mean Field Games , 2, 17, 45

ML Machine Learning

MP-MFG Multi Population-Mean Field Games , 89

N

NE Nash Equilibrium , 11, 109

NF Normalizing Flow , 109

NN Neural Network

O

OMD Online Mirror Descent , 87

P

PI Policy Iteration

POMDP Partially Observable Markov Decision Process

R

RL Reinforcement Learning , 17, 45

S

SAC Soft Actor Critic , 116

V

VI Value Iteration

List of Symbols

Mathematical notations

$\mathbb{N}$	set of integers
$\mathbb{R}$	set of real numbers
$\lvert E\rvert$	Cardinal of a set E
$\lvert x\rvert$	Euclidean norm for a vector $x \in \mathbb{R}^n$
Δ_E	Set of probability distributions on a set E; when E is finite it is the corresponding simplex in the Euclidean space of dimension $\lvert E\rvert$, we view probability distributions as normalized vectors of $\mathbb{R}^{\lvert E\rvert}$
2^E	Set of subsets of a set E
argmax	Set of all maximizers
$\pi, \mu, \ldots$	Unless otherwise specified, bold symbols denote time-dependent objects, which can beviewed as functions of time or as sequences indexed by time steps
$M^\top$	transpose of a matrix M
$\mathbb{E}$	expectation under a probabilistic model. Given a probability distribution p on a set $\mathcal{X}$ and a function $\varphi : \mathcal{X} \to \mathbb{R}$, $\mathbb{E}_{x\sim p}[\varphi(x)] = \mathbb{E}[\varphi(x) \,\lvert\, x \sim p] = \sum_{x\in\mathcal{X}} p(x)\varphi(x)$.
$\mathbb{P}$	probability under a probabilistic model
$\mathcal{N}$	Normal distribution

Markov Decision Processes

$\mathcal{X}$	set of states $x \in \mathcal{X}$
$\mathcal{A}$	set of actions $a \in \mathcal{A}$

$r(x, a)$	reward function for single-agent RL $R : x, a \to r(x, a)$
$P(x' \mid x, a)$	transition distribution $x' \sim P(x' \mid x, a)$
γ	discount factor in $[0, 1]$
μ	population's distribution
m_0	initial population's distribution
$k \in 1, ..., K$	iteration index inside an algorithm, used as superscript
N	Number of agents
K	Number of iterations in iterative methods or algorithms
N_p	Number of populations
π	policy
N_T	Time horizon inside an episode
$n \in 1, ..., N_T$	time index inside an episode, used as subscript
$\pi^\star$	optimal policy
V	state value function (* for optimal value, $^\pi$ for policy value)
Q	state-action value function (* for optimal value, $^\pi$ for policy value)
B	Bellman operator (* for optimality, $^\pi$ for evaluation)

Chapter 1

Introduction

When I took the decision of starting a Ph.D., I knew I wanted to do research in Machine Learning, preferably in Reinforcement Learning. At the time, I was doing a Master of Science specialized in Artificial Intelligence at Polytechnique and Olivier Pietquin was teaching us a course on Deep Reinforcement Learning (RL). I was amazed by the fact that an artificial intelligence could play Atari games, often better than humans. How was this even possible? At this point, I decided to pursue my education with a Ph.D. in Reinforcement Learning. I contacted Olivier to know if he would be interested in working with me. He answered positively and proposed several possible subjects. Among them, he offered me to work on Mean Field Games, in collaboration with Romuald Élie, who became my co-advisor. At first, I had exactly the same reaction than you probably have at this point: what are Mean Field Games (MFGs)? I understood roughly that MFGs are games with an infinite number of identical players, and that they have numerous applications such as crowd motion, energy management, economics... Reinforcement learning is already quite a tricky subject by itself, Multi-agent Reinforcement Learning (MARL) is even more complicated. Now, I had the opportunity to do research on multi-Agent reinforcement learning with in infinite number of agents! This choice, that could have appeared frightening for some people, got me really curious and excited. Was it reckless to accept to work on mean field games and reinforcement learning? Without further delay, let me dig into the heart of the subject and explain in more details what are mean field games, what is reinforcement rearning, and why RL and MFGs can benefit from each other.

1.1 Context and Scope

1.1.1 The curse of many agents

From understanding biological swarming, ant colonies or forest ecosystems to optimizing traffic flow, market economies or spreading of epidemics, multi-agent systems are ubiquitous in nature and engineering. However, if recent progress in artificial intelligence and in particular deep reinforcement learning has allowed to solve complex games such as Go, Starcraft and Poker, recent methods still struggle to tackle applications with more than a dozen players. This difficulty is known as the curse of many agents: when the number of agents increases, the game becomes computationally way harder to solve as interactions among players become intractable (L. Wang, Z. Yang, and Z. Wang, 2020). However, real-world applications such as population dynamics, epidemics control or economics in general often involve hundreds if not thousands of players. Are we doomed to solve multi-agent systems with only a few players?

Introduced concurrently by M. Huang, R. P. Malhamé, and Caines (2006) and Lasry and Lions (2007), Mean Field Games could offer a way out, as they deal by definition with an infinite number of identical and anonymous players. MFGs have the particularity of replacing the interactions between all players by the interaction between a so-called *representative player* and the distribution of the continuum of players. As a result, MFGs do not suffer the curse of many agents anymore. Furthermore, they offer a new mathematical framework to study large-population systems and they have found a wide variety of applications in finance, engineering or crowd modeling[1]. Thus, it seems there is still hope of scaling multi-agent systems thanks to MFGs. This observation has been the first motivation guiding my choice of subject when I decided to start a Ph.D. more than three years ago. Besides scaling up games in terms of number of agents, what is the benefit of solving mean field games with reinforcement learning? I will try to provide answers to this vast (and for now ill-posed) question.

In the above, I took as granted that many applications of our everyday life can be expressed as *games*. This is the goal of *Game Theory*, which studies strategic interactions among rational agents. As soon as a problem can be expressed as a game, *i.e.* as soon as there is a notion of reward than can be maximized (or of a cost than can be minimized) and rational agents interacting, a problem can be turned into a game. However, what notions of convergence can we look at when many different players come into play? Indeed, players must anticipate what the others might do because other players' decisions will affect the final payoff and potentially the dynamics. In order to study such interactive systems, classical game theory is based on the rationality of players. This central hypothesis means that agents are able to act optimally with respect to a criterion, seeking to accumulate as much reward as possible along time. In such context, how can we study multi-agent systems? Nash (1950) introduced the concept of Nash

[1] We refer the reader to Section A.1 for an extensive list of examples

equilibrium in order to study the outcomes of games where the players only act in their own interest, best responding to their belief about the behavior of others. These games, also known as competitive games, are a useful framework of which Mean Field Games are a subclass. We will define more formally what a Nash equilibrium is in the context of Mean Field Games in the sequel. For now, let us just underline that Nash equilibrium, although being very useful for studying what kind of equilibrium may emerge in competitive multi-agent systems, are quite unsatisfactory as they can lead to undesirable situations. The example of the prisoner's dilemma is maybe the most famous instance of this kind of situation, in which individuals receive the greatest payoffs if they betray the group rather than cooperate. It is thus well-known that Nash equilibrium can result in bad outcomes in terms of what is socially optimal for the group. Nonetheless, Nash equilibria provide a useful framework for studying the behavior in large systems of rational and interactive agents and this is the main concept of solutions we look for in the thesis.

We can now phrase more formally the goal of the dissertation: *How can we use Reinforcement Learning to find Nash equilibria in Mean Field Games?*

In the rest of the introduction, I start by defining more formally what are mean field games, and what we mean by learning and (multi-agent) reinforcement learning. I will then provide the first insights of how reinforcement learning can scale up mean field games (and vice-versa), before presenting the outline and our contributions.

1.1.2 Quick introduction to mean field games

General intuition. At a high level, a mean field game is a game with an infinite number of identical and anonymous players. All players have a similar behavior, *i.e.* they are symmetric: we do not need to retain the identity of a player as part of its state. Furthermore, as we have an infinite number of players, we can replace the atomic players by their distribution over the state (and sometimes action) space. This idea is borrowed from statistical physics and we thus name this distribution the *mean field* distribution (Figure 1.1). All the individual interactions can then be replaced by the interaction between a representative player and the mean field distribution, which considerably simplifies the model and the analysis. This approximation relies on the assumption that the population is homogeneous and that the interactions are symmetric and anonymous in the sense that each player interacts only with the empirical distribution of the other players, see Figure 1.2 for a schematic view. As already mentioned, our ultimate goal is to compute a Nash equilibrium, which corresponds to the situation where no player has any interest in deviating from its current behavior, provided that the other players do not deviate either.

Figure 1.1 – The mean field approximation: atomic players are replaced by a distribution of players over the state space, here a two dimensional domain.

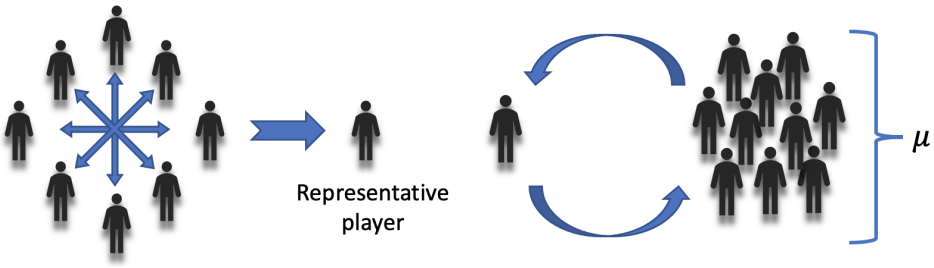


Figure 1.2 – Reading order: **Left** All identical and anonymous players allow to elect a representative player; **Right** The representative player adapts its policy to the distribution of players μ. In return, the distribution is influenced by all the players using the policy of the representative player.

Example. As a typical example, we can consider crowd motion. Each player is an agent represented by its position and is able to control its velocity so as to reach a target destination while minimizing the effort made to move. Typically, passing through a crowded region, *i.e.* a region with a high density of players, requires extra efforts or reduces the velocity, which creates some congestion. If we assume that the number of agents is extremely large and that these agents are homogeneous and have symmetric interactions, then we can approximate the empirical distribution of positions by the mean field distribution corresponding to the limiting regime with an infinite population. This allows to simplify tremendously the computation of a Nash equilibrium because we only need to compute the optimal policy of the representative player instead of having one different policy per player.

Mean Field Games vs. Mean Field Control. While computing a Nash equilibrium is the main focus of the thesis, another popular notion of equilibrium is the Pareto (or social) optimum. In this formulation, one usually supposes that a central entity can control all the agents, which has

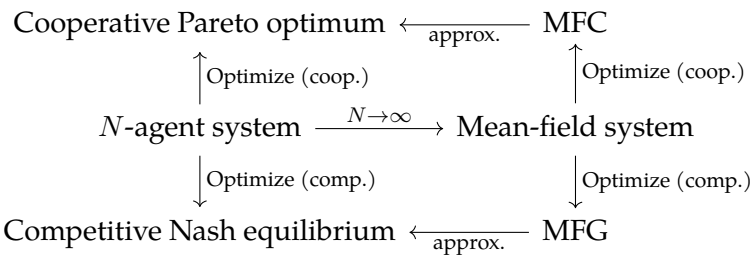


Figure 1.3 – Figure from Cui, Tahir, et al. (2022). Scheme of the links between multi-agent systems, mean field games, mean field control and there solutions.

the benefit of allowing to converge to a better equilibrium *in terms of social benefit*. We refer to this setting as Mean Field Control (MFC) (or control of McKean-Vlasov dynamics) (Bensoussan, Frehse, and P. Yam, 2013; Carmona and Delarue, 2018b). In both cases of MFG and MFC, the solutions are typically characterized through optimality conditions taking the form of a coupled system of forward-backward equations. The forward equation describes the evolution of the population distribution while the backward equation represents the evolution of the value function (i.e. the utility of its behaviour) for a representative player. In the continuous time and continuous space setting, the equations can be partial differential equations (PDEs) (Lasry and Lions, 2007) or stochastic differential equations (SDEs) of McKean-Vlasov type (Carmona and Delarue, 2013) depending on whether one relies on the analytical approach or the probabilistic approach. We refer to Bensoussan, Frehse, and P. Yam (2013), Carmona and Delarue (2018b), Carmona and Delarue (2018c), and Achdou, Cardaliaguet, et al. (2020) for more details. In this thesis, we will focus on the discrete time case, which is closer to the framework of Markov Decision Processes (MDP) (Bertsekas and Shreve, 1996; Puterman, 2014) and mainly on Nash equilibria and mean field games even if we will also discuss the mean field control setting in the first two chapters.

Another dichotomy between MFG and MFC is the type of games. MFGs often focus on competitive games as each player is trying to optimize its own reward. On the other hand, we can consider that MFC is focusing on collaborative games: we argue that it is equivalent to consider that a central controller is optimizing for the whole population than to consider players are willing to collaborate and optimizing a common reward (supposing perfect communication and information). Figure 1.3 summarizes the links between multi-agent systems, MFG and MFC.

1.1.3 Learning and Reinforcement Learning

Two notions of learning. The second question we need to answer before diving more into the thesis is what learning means in our context. There are mainly two interpretations of learning. The first one comes from game theory and economics. According to Fudenberg and Levine, 1998a, *"The theory of **learning in games** [...] examines how, which, and what kind of equilibrium might arise as a consequence of along-run non equilibrium process of learning, adaptation, and/or imitation."* From this point of view, the main focus is on how the players iteratively adjust their behavior until convergence to an equilibrium. The second interpretation of learning is mainly used in Machine Learning (ML) and in **reinforcement learning**. As Mitchell (1997) puts it, *"a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."* In this concept, the concept of learning is very related to the idea of improving one's performance by using data or samples. In this dissertation, we are interested in delineating these two notions of learning while explaining how they can be combined.

Learning in games. Solving decision making problems involving multiple agents has been the topic of intensive research in artificial intelligence for decades (Shoham, 1993; Wooldridge and N. R. Jennings, 1995). This research direction finds its roots in the literature on learning in games, which goes back to the works of Shannon (1959) and Samuel (1959). Despite the vast literature on game theory and numerous fundamental results, application to real-world problems remains a challenge. Recently, successes of combining game theory and machine learning (especially deep learning (Goodfellow, Bengio, and Courville, 2016) and reinforcement learning (Sutton and Barto, 2018) led to solutions for large scale games such as chess (Campbell, Hoane Jr, and Hsu, 2002), Checkers (Schaeffer et al., 2007; Samuel, 1959), Hex (Anthony, Tian, and Barber, 2017), Go (Silver, A. Huang, et al., 2016; Silver, Schrittwieser, et al., 2017; Silver, T. Hubert, et al., 2018), Poker (N. Brown and T. Sandholm, 2017; N. Brown and T. Sandholm, 2019; Moravcik et al., 2017; Bowling et al., 2015), complex video games like StarCraft II (Vinyals et al., 2019) and more recently complex imperfect information games such as Stratego (Perolat, Vylder, et al., 2022) and no-press Diplomacy (Bakhtin et al., 2022). Although this allowed for tackling problems involving large state spaces, the number of agents still remains limited and scaling up to large populations of players remains intractable without the mean-field approximation.

Reinforcement learning. The primary goal of reinforcement learning is to learn to act in a (complex) environment. It can be described as a general framework for learning-based sequential decision-making. More formally, at each time step n, the system is described by its state x_n in which the agent is. The agent takes an action a_n and receives a reward r_n which often depends on its state and action, before transitioning to a new state x_{n+1}, drawn

from a conditional distribution $p(x_{n+1}|x_n, a_n)$ that we call the transition kernel (or transition probabilities). The ultimate goal for the agent is to learn to act optimally in such an environment through sampled-based interactions, *i.e.* to learn a policy mapping its state to an action: $\pi(x_n|a_n) \in \Delta_{\mathcal{A}}$. It is generally formalized as an MDP, *i.e.* a tuple $(\mathcal{X}, \mathcal{A}, P, r, \gamma)$ where adequate behavior consists in optimizing the long term return of the agent, weighted with the discount factor γ. We describe precisely MDP and reinforcement learning in Chapter 2.

Multi-agent reinforcement learning. While it is fairly straightforward to define what should be learned in the single agent case thanks to the reward function, it is more tricky in the multi-agent case because the optimal behaviour of a player must take into account the other players. If one agent changes its strategy, other agents must adapt to this new strategic interaction, making the environment from the agent point of view non stationary with respect to the behavior of others. To circumvent this difficulty, multi-agent reinforcement learning often keeps track of all agents' states and use the framework of Markov games which generalizes MDPs to multi-agent systems. We denote the number of the player i, with $i \in [1, \dots, N]$ (not to be confused with the time step $n \in [1, \dots, N_T]$). In every state x of a state space $\mathcal{X}$, players simultaneously take an action a^i in a set $\mathcal{A}^i(x)$. As a result of this joint action $(a^1, \dots, a^N)$, each player receives a reward $r^i(x, a^1, \dots, a^N) = r^i(x, \mathbf{a})$ and the system moves to the following state x' with a probability defined by the transition kernel $p(x'|x, a^1, \dots, a^N) = p(x'|x, \mathbf{a})$. Usually in these settings, each player's goal is to find a strategy $\pi^i(\cdot|x)$ to maximize the expected sum of γ-discounted rewards. As explained earlier, this framework suffers from the curse of many agents: increasing the number of agents increases exponentially the action space which makes the model intractable when N becomes too large.

From multi-agent reinforcement learning to mean field games. While the general multi-agent learning case might seem out of reach, considering interactions within a very large population of players may lead to tractable models. We summarize below the main differences between single-agent, multi-agent learning and mean-field interactions.

Table 1.1 – Notations for various number of agents

	Single Agent	N-player Game	Mean Field Game
State space	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{X}$
Number of players	1	N	∞
Action Space	$\mathcal{A}$	$\mathcal{A}^i{}_{i\in\{1,\dots,N\}}$	$\mathcal{A}$
Reward	$r(x, a)$	$r^i(x, \mathbf{a})$	$r(x, a, \mu)$
Policy	$\pi(a\|x)$	$\pi^i(a\|x)_{i\in\{1,\dots,N\}}$	$\pi(a\|x)$
Transition kernel	$p(x'\|x, a)$	$p(x'\|x, \mathbf{a})$	$p(x'\|x, a, \mu)$

In particular, we do not need to keep track of all agents' states inside the policy π because having an infinite number of agents makes the evolution of the distribution deterministic thanks to the law of large numbers. This simplifies tremendously the computation of an equilibrium. Furthermore, most of the MFG literature assumes the representative player to be fully informed about the game dynamics and the associated reward mechanisms.

1.1.4 Reinforcement Learning for Mean Field Games and vice-versa

In the dissertation, we argue that mean field games and reinforcement learning are two paradigms that can be merged efficiently into what we call *Mean Field Reinforcement Learning* (Figure 1.4). On one hand, MFGs can scale multi-agent reinforcement learning in terms of number of agents. The first reason, that we described before, is because mean field games allow to focus on policies that depend only on the representative agent's state without keeping track of all agents. A key property is that the solution to the MFG provides an ε-Nash equilibrium for the corresponding N-player game, with ε going to 0 as N goes to infinity. Furthermore, under suitable assumptions, N-player Nash equilibria or social optima converge to the corresponding mean field equilibrium or social optimum. Such results build on the idea of propagation of chaos (Sznitman, 1991) but are more subtle since the players are not simple particles but rational agents making optimal decisions and reacting to other players' decisions (Lacker, 2017; Cecchin and Pelino, 2019; Lacker, 2020; Cardaliaguet, Delarue, et al., 2019).

We now argue that on the other hand, reinforcement learning can help to scale mean field games in terms of model complexity. Except Delarue and Vasileiadis (2021), which identifies common noise as an exploration noise, a key concept in reinforcement learning, few works have tried to use reinforcement learning concepts to solve MFGs. In fact, most existing numerical methods for MFGs and MFC problems are based on the aforementioned optimality conditions phrased in terms of PDEs or SDEs, *without* reinforcement learning. Such approaches typically rely on suitable discretizations, *e.g.*, by finite differences (Achdou and Capuzzo-Dolcetta, 2010; Achdou, Camilli, and Capuzzo-Dolcetta, 2012) primal-dual methods (Briceño-Arias, Kalise, and Silva, 2018; Briceño-Arias, Kalise, Kobeissi, et al., 2019), semi-Lagrangian schemes (Carlini and Silva, 2014; Carlini and Silva, 2015), or probabilistic approaches (Chassagneux, Crisan, and Delarue, 2019; Angiuli, Graves, et al., 2019). We refer to Achdou and Laurière (2020) and Laurière (2021) for recent surveys on these methods. Although these methods are very well understood and very successful in small dimension, they cannot tackle MFGs with high dimensional states or controls due to the curse of dimensionality (Bellman, 1957). To address this limitation, stochastic methods based on approximations by neural networks have recently been introduced by Carmona and Laurière (2021), Carmona and Laurière (2019), Fouque and Z. Zhang (2020), and Germain, Mikael, and Warin (2022) using optimality conditions for general mean field games, by Ruthotto et al. (2020) for MFGs which can be written as a control

problem, and by Cao, Guo, and Laurière (2020) and Lin et al. (2020) for variational MFGs in connection with generative adversarial networks (GANs) (Goodfellow, Pouget-Abadie, et al., 2014). We refer to R. Hu and Laurière (2022) for a recent survey on machine learning methods for control and games, with applications to MFGs and MFC problems. However, these methods still try to solve the problems in an exact way by relying on exact computations of gradients *i.e.* exploiting the full knowledge of the model. In this thesis, we focus rather on learning methods which aim at solving MFGs in a model-free fashion to foster the scalability of numerical methods for these problems. Furthermore, solving MFGs without explicitly solving a forward-backward system of continuous equations allows to deal with much more complex domains, with possible many boundary conditions as it often occurs in three-dimensional real-world applications, *e.g.*, when obstacles are part of the environment.

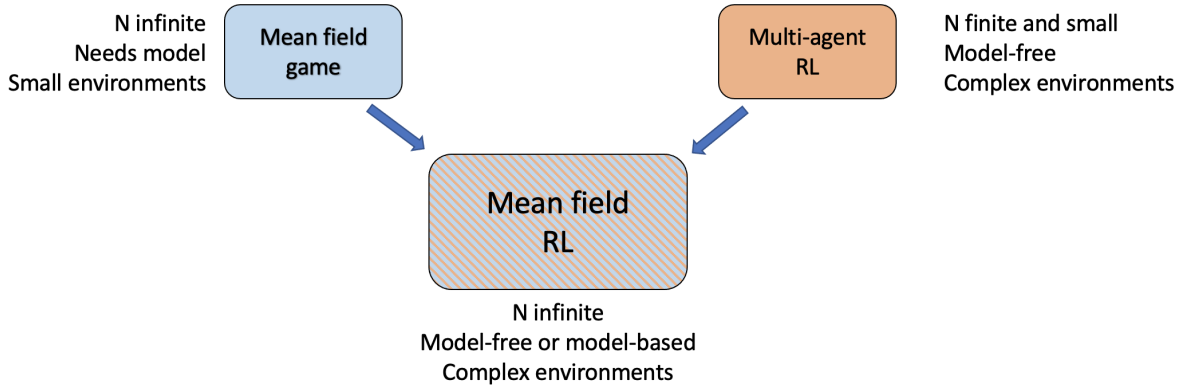


Figure 1.4 – Mean field Reinforcement Learning.

1.2 Outline and Contributions

The goal of this thesis is to propose new methods combining mean field approximations and reinforcement learning. We want to tackle large population games in highly complex environments, *i.e.* to solve multi-agent systems at a very large scale, both in terms of population size and model complexity. To do so, we can ask ourselves the following questions:

- *How to design algorithms to find Nash equilibria in mean field games?*
- *How to adapt these algorithms to a model-free setting, using deep reinforcement learning?*

With this clear objective in mind, we now present the outline of the thesis.

In Part I, we provide the necessary background on Reinforcement Learning and Mean Field Games. This part is mainly dedicated to answering the first question, to reviewing the

literature and to laying the foundation for the rest of the dissertation. It is largely based on M. Laurière, S. Perrin, M. Geist, and O. Pietquin (2022). Learning Mean Field Games: A Survey. *arXiv:2205.12944*, divided in two chapters.

Chapter 2 starts by giving a detailed background on Markov decision processes for a single agent learning in an environment, both in stationary and finite horizon settings. The stationary setting is the most encountered one in reinforcement learning, but we will stress out that the finite horizon setting encompasses more interesting interactions when learning in an MFG. We define the Bellman operator for the stationary setting and introduce the value and state-value functions. We provide the equations to both evaluate a policy or find an optimal one. We then describe the value iteration and policy iteration algorithms that both converge to an optimal policy in these two settings. The rest of the chapter is dedicated to the Approximate Dynamic Programming (ADP) framework that appears when the transition kernel and reward are not known. We will describe reinforcement learning methods that have been successfully developed in this model-free setting, with a precise characterization of what we consider as an environment. Finally, we review classical Deep Reinforcement Learning (DRL) methods that have proved successful to solve complex games. In the second part of the chapter, we define mean field games in the stationary and finite horizon settings and provide several extensions of these two frameworks. We introduce relevant notations that will be used throughout the rest of the dissertation. We adapt the single-agent MDP framework detailed before to mean field games and discuss the precise nature of the objects of interests, namely the policy π of the optimal player and the distribution μ of players. In particular, we provide Bellman operator and Fokker-Plank equation (adapted to discrete time).

Chapter 3 provides an overview of iterative methods, *i.e.* algorithms, using Dynamic Programming (DP) for mean field games. The first part of the chapter is dedicated to introducing a general framework to study iterative methods. We state that fixed point approaches often fail by lack of contraction of the operators, which motivates the use of various techniques of regularization to ensure convergence to a Nash equilibrium. We detail several algorithms and their connections to single-agent reinforcement learning. Building on these methods and the connection between MDPs and RL, we explain in Section 3.2 how RL and DRL methods can be adapted to solve MFGs and MFC problems, when the model is unknown. Section 3.3 discusses metrics, and in particular the *exploitability*, that can be used to assess the numerical convergence of algorithms. We finish the chapter with an illustration of the algorithms on a representative MFG example, and introduce the OpenSpiel library in which many games and algorithms for MFGs have been implemented.

Having introduced all the necessary ingredients for combining efficiently reinforcement learning with mean field games, Part II takes a deep dive into two iterative methods: Fictitious Play (FP) and Online Mirror Descent (OMD).

In Chapter 4, we study a continuous-time version of fictitious play, prove its convergence to a Nash equilibrium under the monotonicity condition and even provide a rate of convergence. This chapter is based on S. Perrin, J. Perolat, M. Laurière, M. Geist, R. Elie, and O. Pietquin (2020). Fictitious Play for Mean Field Games: Continuous Time Analysis and Applications. In *Advances in Neural Information Processing Systems* (*NeurIPS 2020*). Vol. 33. Curran Associates, Inc., pp. 13199–13213. We also study the convergence under a common noise, a source of randomness shared by all the agents. We introduce the discrete-time version of the algorithm, implement it and test it in a Linear Quadratic (LQ) game and in a game named the beach bar process. We test model-based algorithms, relying on dynamic programming, and model-free algorithms such as Q-learning, and highlight that it manages to converge in simple games. We finish our numerical study with a more involved example in two dimensions, adapting the beach bar problem to a maze where the agents must learn to go the center while avoiding interactions.

Chapter 5 studies Online Mirror Descent and proves the convergence of the algorithm to a Nash Equilibrium with continuous-time updates, also under the monotonicity condition. It uses the results of J. Perolat, S. Perrin, et al. (2021). Scaling up Mean Field Games with Online Mirror Descent. *Autonomous Agents and Multiagent Systems* (*AAMAS 2022*) (*arXiv preprint arXiv:2103.00623*). Rather than computing a full best response at each iteration, OMD only evaluates the current policy which is computationally cheaper. Here, we introduce the theoretical results under multi-population mean field games, a slightly different framework that allows to have several populations competing against each other, while extending the classical MFG setting. In particular, the results are still valid for a single-population MFG. We show numerically that OMD converges faster than FP in all the games we consider. In these two chapters, the theory considers that the model is known and that quantities can be computed up to an infinite precision (apart when using Q-learning in Chapter 4). This part is thus mainly dedicated to answering the second question, namely to design algorithms that converge to Nash equilibria in mean field games.

Finally, Part III is dedicated to three contributions involving model-free deep reinforcement learning methods, and clearly answers to the second question. Compared to Part II, we do not suppose anymore that the model is known.

This part begins with Chapter 6, which provides an illustration of how deep reinforcement learning can be used alongside fictitious play in a complex multi-dimensional continuous example. It is based on S. Perrin, M. Laurière, J. Pérolat, M. Geist, R. Élie, and O. Pietquin (2021). Mean Field Games Flock! The Reinforcement Learning Way. *International Joint Conference of Artificial Intelligence* (*IJCAI 2021*) (*arXiv preprint arXiv:2105.07933*). We study flocking, a collective behavior that emerges naturally in group of animals. We show that modelling this behavior as a MFG (Cucker and Mordecki, 2008) allows the agents to all align their velocities.

We propose several experiments in two and three dimensions and provide a rich visualization thanks to an integration in Unity, a rich game engine to create three dimensional games.

Then, Chapter 7 studies the question of generalization and summarizes the results of S. Perrin, M. Laurière, J. Pérolat, R. Élie, M. Geist, and O. Pietquin (2021). Generalization in Mean Field Games by Learning Master Policies. *Proceedings of the AAAI Conference on Artificial Intelligence* (*AAAI 2022*) (*arXiv preprint arXiv:2109.09717*). A natural way of computing Nash equilibria in MFGs is to suppose that the agents are distributed according to a fixed initial distribution. We argue that this hypothesis is not natural in machine learning as there is no reason it would happen in real-world applications which require policies able to be reactive to the current distribution of agents. Thus, we propose a way of surpassing this issue, that relies on three ingredients: adding the distribution of agents to the policy, using deep reinforcement learning and learning the policy from many initial distributions. We call the resulting object the *master policy* and show numerically that this approach is sound. However, it still uses the fictitious play algorithm in a non-scalable way.

This leads us to our last contribution. In Chapter 8, we propose two scalable DRL adaptations of fictitious play and online mirror descent. This chapter uses the results of M. Lauriere et al. (2022). Scalable Deep Reinforcement Learning Algorithms for Mean Field Games. In *Proceedings of the 39th International Conference on Machine Learning* (*ICML 2022*). Vol. 162, pp. 12078–12095. We recall that FP and OMD were not considered scalable, as they require to average or sum quantities. However, this simple operation is not trivial in non-linear functions such as neural networks, as simply averaging weights does not work. We thus propose Deep Averaged Fictitious Play (D-AFP) and Deep Munchausen Online Mirror Descent (D-MOMD) that surpass this difficulty. D-AFP keeps a buffer of all previous iterations and minimizes a categorical loss in order to learn directly an average policy without having to keep in memory all past best responses. D-MOMD uses a reparameterization based on Legendre-Fenschel transform, which regularizes directly the Q-function and does not require anymore to sum all previous Q-functions. To the best of our knowledge, D-MOMD is the state-of-the-art in many different games as it is the fastest algorithm to converge to a Nash equilibrium that uses deep reinforcement learning.

List of publications

Publications in international conferences with proceedings

- S. Perrin, J. Perolat, M. Laurière, M. Geist, R. Elie, and O. Pietquin (2020). Fictitious Play for Mean Field Games: Continuous Time Analysis and Applications. In *Advances in Neural Information Processing Systems* (*NeurIPS 2020*). Vol. 33. Curran Associates, Inc., pp. 13199–13213 (used in Chapter 4)

- J. Perolat, S. Perrin, et al. (2021). Scaling up Mean Field Games with Online Mirror Descent. *Autonomous Agents and Multiagent Systems* (*AAMAS 2022*) (*arXiv preprint arXiv:2103.00623*) (used in Chapter 5)
- S. Perrin, M. Laurière, J. Pérolat, M. Geist, R. Élie, and O. Pietquin (2021). Mean Field Games Flock! The Reinforcement Learning Way. *International Joint Conference of Artificial Intelligence* (*IJCAI 2021*) (*arXiv preprint arXiv:2105.07933*) (used in Chapter 6)
- S. Perrin, M. Laurière, J. Pérolat, R. Élie, M. Geist, and O. Pietquin (2021). Generalization in Mean Field Games by Learning Master Policies. *Proceedings of the AAAI Conference on Artificial Intelligence* (*AAAI 2022*) (*arXiv preprint arXiv:2109.09717*) (used in Chapter 7)
- M. Lauriere et al. (2022). Scalable Deep Reinforcement Learning Algorithms for Mean Field Games. In *Proceedings of the 39th International Conference on Machine Learning* (*ICML 2022*). Vol. 162, pp. 12078–12095 (used in Chapter 8)

Preprint

- M. Laurière, S. Perrin, M. Geist, and O. Pietquin (2022). Learning Mean Field Games: A Survey. *arXiv:2205.12944* (used in Chapter 2 and Chapter 3)

Collaborations not presented in this thesis

As these papers are not presented extensively in the thesis, we provide some background here.

- M. Geist, J. Pérolat, et al. (2021). Concave utility reinforcement learning: the mean-field game viewpoint. *Autonomous Agents and Multiagent Systems* (*AAMAS 2022*) (*arXiv preprint arXiv:2106.03787*). This paper received an honorable mention as a runner up for the AAMAS 2022 Best Paper Award. Concave Utility Reinforcement Learning (CURL) extends RL from linear to concave utilities in the occupancy measure induced by the agent's policy. This encompasses not only RL but also imitation learning and exploration, among others. Yet, this more general paradigm invalidates the classical Bellman equations, and calls for new algorithms. Our core contribution consists in showing that CURL is a subclass of MFGs. We think this important to bridge together both communities. It also allows to shed light on aspects of both fields: we show the equivalence between concavity in CURL and monotonicity in the associated MFG, between optimality conditions in CURL and Nash equilibrium in MFG, or that fictitious play for this class of MFGs is simply Frank-Wolfe, bringing the first convergence rate for discrete-time FP for MFGs. We also experimentally demonstrate that, using algorithms recently introduced for solving MFGs, we can address the CURL problem more efficiently.

Introduction

- T. Cabannes et al. (2021). Solving N-player dynamic routing games with congestion: a mean field approach. *Extended abstract at AAMAS 2022* (*long version: arXiv preprint arXiv:2110.11943*). This article introduces a new N-player dynamic routing game with explicit congestion dynamics, as well as the corresponding mean field game. Experiments reproduce several classical benchmark networks of the traffic community: the Pigou, Braess, and Sioux Falls networks with heterogeneous origin, destination and departure time tuples. The Pigou and Braess examples reveal that the mean field approximation is generally very accurate and computationally efficient as soon as the number of vehicles exceeds a few dozen. On the Sioux Falls network, this approach enables learning traffic dynamics with more than 14,000 vehicles.

- P. Muller, R. Elie, et al. (2022). Learning Correlated Equilibria in Mean-Field Games. *arXiv:2208.10138*. In this work, we provide an alternative route for studying mean field games, by developing the concepts of mean field correlated and coarse-correlated equilibria, another type of equilibria beyond Nash. We show that they can be efficiently learnt in *all games*, without requiring any additional assumption on the structure of the game, using three classical algorithms. Furthermore, we establish correspondences between our notions and those already present in the literature, derive optimality bounds for the mean field to N-player transition, and empirically demonstrate the convergence of these algorithms on simple games.

Others

- I co-organized the *Gamification and Multiagent solutions* workshop at ICLR 2022 with Andrea Tacchetti, Ian Gemp, Satpreet Singh, Arash Mehrjou, Noah Golowich and Nina Vesseron.

- I contributed to the OpenSpiel library (Lanctot, Lockhart, et al., 2019).

Part I

Background and Settings

Chapter 2

Background

This chapter provides the background for understanding the rest of the thesis. We start by giving an overview of Markov Decision Process, Reinforcement Learning and Deep Reinforcement Learning to the reader, before presenting several settings of Mean Field Games and Mean Field Control problems that have appeared in the literature (Section 2.2). We stress the similarities and the differences, in terms of definitions and in terms of solutions.

Contents

2.1 From Markov Decision Process to Deep Reinforcement Learning 18

2.2 Mean Field Games: Definition and Settings 28

2.1 From Markov Decision Process to Deep Reinforcement Learning

2.1.1 Markov Decision Processes

We recall a few important concepts pertaining to optimal control in discrete time for a single agent. We will only review the main ideas and we refer to *e.g.* Bertsekas and Shreve (1996) and Puterman (2014) for more details. The notion of Markov decision processes will play a key role in the description of dynamic MFGs.

Stationary MDP

A **stationary Markov Decision Process** (MDP) is a tuple $(\mathcal{X}, \mathcal{A}, p, r, \gamma)$ where $\mathcal{X}$ is a state space, $\mathcal{A}$ is an action space, $p : \mathcal{X} \times \mathcal{A} \to \Delta_{\mathcal{X}}$ is a transition kernel, $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is a reward function and $\gamma \in (0, 1)$ is a discount factor. Using action a when the current state is x leads to a new state distributed according to $p(\cdot|x, a) \in \Delta_{\mathcal{X}}$ and produces a reward $r(x, a)$. The reward could be stochastic but to simplify the presentation, we consider that r is a deterministic function of the state and the action. A **stationary policy** $\pi : \mathcal{X} \to \Delta_{\mathcal{A}}$, $x \mapsto \pi(\cdot|x)$ provides a distribution over actions for each state. The goal of the MDP is to find an optimal policy π^* which maximizes the total return defined as the expected (discounted) sum of future rewards:

$$J(\pi) = \mathbb{E}\Big[\sum_{n\geq 0} \gamma^n r(x_n, a_n)\Big],$$

subject to:

$$\begin{cases} x_0 \sim m_0, \\ a_n \sim \pi(\cdot|x_n), \quad x_{n+1} \sim p(\cdot|x_n, a_n), \quad n \geq 0, \end{cases}$$

where m_0 is an initial distribution whose choice does not influence the set of optimal policies.

Assuming the model is fully known to the agent, the problem can be solved using for instance dynamic programming. The **state-action value function**, or Q-function, associated to a stationary policy π is defined as:

$$Q^\pi(x, a) = \mathbb{E}\left[\sum_{n\geq 0} \gamma^n r(x_n, a_n) \Big| x_0 = x, a_0 = a, a_n \sim \pi(\cdot|x_n), x_{n+1} \sim p(\cdot|x_n, a_n)\right]. \tag{2.1}$$

By dynamic programming, it satisfies the following fixed point equation with unknown $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$:

$$Q = B^\pi Q,$$

where B^π denotes the **Bellman operator** associated to π:

$$(B^\pi Q)(x,a) = r(x,a) + \gamma \mathbb{E}_{x' \sim p(\cdot|x,a), a' \sim \pi(\cdot|x')}[Q(x',a')]. \tag{2.2}$$

We recall that the expectation is to be understood as:

$$\mathbb{E}_{x' \sim p(\cdot|x,a), a' \sim \pi(\cdot|x')}[Q(x',a')] = \sum_{x'} p(x'|x,a) \sum_{a'} \pi(a'|x') Q(x',a'). \tag{2.3}$$

The **optimal state-action value function** is defined as:

$$Q^*(x,a) = \sup_\pi Q^\pi(x,a). \tag{2.4}$$

It satisfies the fixed point equation:

$$Q = B^* Q, \tag{2.5}$$

where B^* denotes the **optimal Bellman operator**:

$$(B^* Q)(x,a) = r(x,a) + \gamma \mathbb{E}_{x' \sim p(\cdot|x,a)}[\max_{a'} Q(x',a')], \tag{2.6}$$

with

$$\mathbb{E}_{x' \sim p(\cdot|x,a)}[\max_{a'} Q(x',a')] = \sum_{x'} p(x'|x,a) \max_{a'} Q(x',a'). \tag{2.7}$$

It is also convenient to introduce the (state only) **value function** associated to a policy $V^\pi : x \mapsto \mathbb{E}_{a \sim \pi(\cdot|x)}[Q^\pi(x,a)]$ and the (state only) **optimal value function** $V^* : x \mapsto \mathbb{E}_{a \sim \pi^*(\cdot|x)}[Q^*(x,a)]$, where π^* is an optimal policy. These value functions can also be characterized as fixed points of two Bellman operators. Note that these objects are all independent of time, as we search for a stationary solution.

Finite Horizon MDP

One can also consider problems set with a finite time horizon. A **finite-horizon Markov decision process** (MDP) is a tuple $(\mathcal{X}, \mathcal{A}, p, r, N_T)$ where $\mathcal{X}$ is a state space, $\mathcal{A}$ is an action space, N_T is a time horizon, $p : \{0, \dots, N_T - 1\} \times \mathcal{X} \times \mathcal{A} \to \mathcal{P}(\mathcal{X})$ is a transition kernel, and $r : \{0, \dots, N_T\} \times \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is a reward function. At time n, using action a when the current state is x leads to a new state distributed according to $p_n(\cdot|x,a) \in \Delta_\mathcal{X}$ and produces a reward $r_n(x,a) \in \mathbb{R}$. A policy $\boldsymbol{\pi} : \{0, \dots, N_T - 1\} \times \mathcal{X} \to \mathcal{P}(\mathcal{A})$, $(n,x) \mapsto \boldsymbol{\pi}_n(\cdot|x)$ provides a distribution over actions for each state at time n. The goal of the MDP is to find a policy $\boldsymbol{\pi}^*$ which maximizes the total return defined as the expected (discounted) sum of future rewards:

$$J(\boldsymbol{\pi}) = \mathbb{E}\Big[\sum_{n=0}^{N_T} r_n(x_n, a_n)\Big],$$

subject to:

$$\begin{cases} x_0 \sim m_0, \\ a_n \sim \boldsymbol{\pi}_n(\cdot|x_n), \quad x_{n+1} \sim p_n(\cdot|x_n, a_n), \quad n = 0, \dots, N_T, \end{cases}$$

where m_0 is an initial distribution whose choice does not influence the set of optimal policies.

Here again, assuming the model is known to the agent, the problem can be solved using for instance dynamic programming. The **state-action value function** associated to a stationary policy $\boldsymbol{\pi}$ is defined as:

$$\begin{cases} \mathbf{Q}^{\boldsymbol{\pi}}_{N_T}(x, a) = r_{N_T}(x, a) \\ \mathbf{Q}^{\boldsymbol{\pi}}_{n}(x, a) = \mathbb{E}\left[\sum_{n' \geq n} r_{n'}(x_{n'}, a_{n'}) \middle| x_n = x, a_n = a, a_{n'} \sim \boldsymbol{\pi}_{n'}(\cdot|x_{n'}), x_{n'+1} \sim p_{n'}(\cdot|x_{n'}, a_{n'}) \right], \\ \qquad\qquad n = N_T - 1, \dots, 0. \end{cases}$$

The **optimal state-action value function** is defined as:

$$\mathbf{Q}^*(x, a) = \sup_{\boldsymbol{\pi}} \mathbf{Q}^{\boldsymbol{\pi}}(x, a). \tag{2.8}$$

Here again, we can introduce the (state only) **value function** associated to a policy: $\mathbf{V}^{\boldsymbol{\pi}}_n(x) = \mathbb{E}_{a \sim \boldsymbol{\pi}_n(\cdot|x)}[\mathbf{Q}^{\boldsymbol{\pi}}_n(x, a)]$, and the **optimal value function**: $\mathbf{V}^*_n(x) = \mathbb{E}_{a \sim \boldsymbol{\pi}^*_n(\cdot|x)}[\mathbf{Q}^*_n(x, a)]$, $\boldsymbol{\pi}^*$ is an optimal policy.

Formally, the finite-horizon MDP can be restated as a stationary MDP by incorporating the time n in the state. However, it can be simpler to directly tackle this MDP using techniques that are specific to the finite-horizon setting. In particular we stress that, in contrast with the stationary setting presented above, the value functions are here characterized by equations which are not fixed point equations but backward equations. They can be solved by backward induction, as we will discuss in the sequel (see Section 3.1). For more details on finite-horizon MDP we refer to *e.g.* (Puterman, 2014).

In the rest of this section, we first recall value iteration and policy iteration methods in standard MDPs. We then explain later in this chatper how these methods are adapted in the MFG setting. For the sake of simplicity, we focus on two settings: the stationary setting and the finite horizon evolutive setting. We stress the main similarities and differences between the methods to solve these types of MFGs. The methods in these two settings can be adapted to tackle the static setting and the γ-discounted setting, which are thus omitted for the sake of brevity.

2.1.2 Solving standard MDPs

We recall here two fundamental families of methods to compute optimal policies: value iteration and policy iteration. For more details on these methods, we refer to Sutton and Barto (2018), Bertsekas and Shreve (1996), Bertsekas (2012), Puterman (2014), and Meyn (2022).

Value iteration

One way to obtain an optimal policy is to first compute the optimal value function by using the optimal Bellman operator, and then consider a greedy policy with respect to this optimal value function. Since we are motivated by applications to RL algorithms, we focus on the state-action value function.

Stationary MDP. In a stationary MDP, the **value iteration** method can be expressed as follows: Q^0 is given, and for $k = 0, \dots, K-1$,

$$Q^{k+1} = B^* Q^k. \tag{2.9}$$

At the end we use the following policy as an approximation of the optimal policy:

$$\pi^K \in GQ^K,$$

where G denotes the greedy policy operator defined by:

$$GQ = \left\{ \pi \,:\, \forall x \in \mathcal{X},\ \sum_{a \in \mathcal{A}} \pi(a|x) Q(x,a) = \arg\max_a Q(x,a) \right\}. \tag{2.10}$$

Thanks to the fact that the Bellman operator is a γ-contraction, when $K \to +\infty$, $\pi^K \to \pi^*$ under suitable conditions on the MDP. Equivalently, the above iterations can also be written as follows, by introducing the greedy policy at every iteration: Q^0 is given, and for $k = 0, \dots, K-1$,

$$\begin{cases} \pi^k = GQ^k, \\ Q^{k+1} = B^{\pi^k} Q^k, \end{cases}$$

where the Bellman operator B^{π^k} associated to the current policy π^k is defined in (2.2).

Finite horizon MDP. In a finite horizon MDP, the optimal value function $\mathbf{Q}^*$ can be computed by dynamic programming since it satisfies the **optimal Bellman equation**:

$$\begin{cases} \mathbf{Q}^*_{N_T}(x,a) = r_{N_T}(x,a), & (x,a) \in \mathcal{X} \times \mathcal{A}, \\ \mathbf{Q}^*_n(x,a) = r_n(x,a) + \gamma \mathbb{E}\left[\max_{a \in \mathcal{A}} \mathbf{Q}^*_{n+1}(x_{n+1},a) \middle| x_{n+1} \sim p_n(\cdot|x,a) \right], \\ \qquad (x,a) \in \mathcal{X} \times \mathcal{A}, n = N_T - 1, \dots, 0. \end{cases} \tag{2.11}$$

Computing $\mathbf{Q}^*$ using the above equation is called **backward induction**. Once it has been computed, an optimal policy can be found by taking the greedy policy at each step, i.e.:

$$\boldsymbol{\pi}^* = \mathbf{G}\mathbf{Q}^*,$$

where $\mathbf{G}$ is the finite-horizon greedy policy operator defined as:

$$(\mathbf{G}\mathbf{Q})_n = G\mathbf{Q}_n, \qquad n = 0, \dots, N_T - 1. \tag{2.12}$$

Remark 1. *Notice that the Bellman equation* (2.11) *is a backward equation and not a fixed-point equation, contrary to Eq.* (2.5) *characterizing the optimal value function in the stationary case. Since the horizon is finite, the optimal value function is computed with a finite number of steps, which is an important difference with the stationary MDP setting.*

Policy iteration

The optimal policy can also be computed by successive improvements of a policy. Starting from an initial policy, at each iteration, we evaluate the performance of this policy by computing the associated value function, and then we take a greedy step to improve the policy. These two steps are called **policy evaluation** and **policy improvement**, and the overall algorithm is called **policy iteration**.

Stationary MDP. In a stationary MDP, the method consists in applying the Bellman operator B^{π^k} associated to the current policy π^k (see (2.2)) and then applying the greedy policy operator defined in (2.10). Thus, this method takes the following form: π^0 is given, and for $k = 0, \dots, K-1$:

$$\begin{cases} Q^{k+1} = Q^{\pi^k}, \\ \pi^{k+1} \in GQ^{k+1}. \end{cases}$$

At the end, we return π^K and use it as an approximation of π^*. As $K \to +\infty$, we have $\pi^K \to \pi^*$ under suitable assumptions on the MDP.

At iteration k, the value function Q^{π^k} can be computed by applying repeatedly the Bellman operator B^{π^k} until convergence to its fixed point, or until an approximation of Q^{π^k} is obtained with a finite number of iterations: with $Q^{k,0}$ given, we repeat for $m = 0, \dots, M-1$,

$$Q^{k,m+1} = B^{\pi^k} Q^{k,m}, \tag{2.13}$$

and we use $Q^{k,M}$ as an approximation of Q^{π^k}.

Finite horizon MDP. In a finite horizon, we can define the following method by analogy with the stationary case: $\boldsymbol{\pi}^0$ is given, and for $k = 0, \dots, K-1$:

$$\begin{cases} \mathbf{Q}^{k+1} = \mathbf{Q}^{\boldsymbol{\pi}^k}, \\ \boldsymbol{\pi}^{k+1} \in \mathbf{G}\mathbf{Q}^{k+1}. \end{cases}$$

where $\mathbf{G}$ is the finite-horizon greedy policy operator defined in (2.12). At the end, we return $\boldsymbol{\pi}^K$ and use it as an approximation of $\boldsymbol{\pi}^*$.

At each iteration, the state-action value function associated to the current policy can be computed by backward induction. Indeed, for a given policy $\boldsymbol{\pi}$, $\mathbf{Q}^{\boldsymbol{\pi}}$ satisfies the following Bellman equation, which holds by dynamic programming:

$$\begin{cases} \mathbf{Q}^{\boldsymbol{\pi}}_{N_T}(x,a) = 0, \qquad (x,a) \in \mathcal{X} \times \mathcal{A}, \\ \mathbf{Q}^{\boldsymbol{\pi}}_{n}(x,a) = r_n(x,a) + \gamma \mathbb{E}\left[\mathbf{Q}^{\boldsymbol{\pi}}_{n+1}(x_{n+1}, a_{n+1}) \middle| x_{n+1} \sim p_n(\cdot|x,a), a_{n+1} \sim \boldsymbol{\pi}_n(\cdot|x) \right], \\ \qquad\qquad (x,a) \in \mathcal{X} \times \mathcal{A}, n = N_T - 1, \dots, 0. \end{cases} \tag{2.14}$$

2.1.3 Reinforcement Learning

Until now we have assumed that the model is fully known and that there are no numerical approximations in the computation of the rewards or the transitions. In this context, the only approximations that we have to cope with are in situations where an infinite number of iterations would be needed but we can only afford a finite number of iterations (*e.g.*, to compute a stationary distribution or a stationary value function).

However, in many situations, these methods cannot be implemented as such. A typical scenario is when the model is not completely known from the agent that is trying to learn an optimal behavior. Another instance is when the model is known, but the state space or the action space are too big to compute the solution on the whole domain. In such cases, exact dynamic programming cannot be used. Instead (**model-free**) **reinforcement learning** (**RL**) methods have been developed. Here we will focus on methods relying on **approximate**

dynamic programming (ADP). The question of exploring efficiently the state-action domain plays a crucial role.

RL ideas have first been developed for finite and small state and action spaces, in which case the algorithms are called **tabular methods** since the value function can be described by a table (*i.e.*, a matrix). However, many of the recent breakthrough applications of RL have been obtained thanks to a combination of RL methods with neural network approximations and deep learning techniques, which leads to **deep reinforcement learning (DRL) methods**. The flexibility and the generalization capabilities of deep neural networks allow us to efficiently learn solution to highly complex problems. In the context of games, some striking examples that were successfully tackled are ALE (Atari Learning Environment) (Mnih et al., 2013; Bellemare et al., 2013), Go (Silver, A. Huang, et al., 2016), poker (N. Brown and T. Sandholm, 2017; Moravcik et al., 2017) or Starcraft (Vinyals et al., 2019).

In the context of MFGs, we will build on the iterative methods presented in Section 3.1. These methods boil down to alternating mean-field updates and policy updates, and the policy updates stem from standard MDP techniques. As a consequence, standard RL techniques can readily be injected at this level to learn policies or value functions.

In this section, starting from exact dynamic programming, we discuss some key ideas underlying ADP and RL methods. We then move on to neural network approximations and DRL. We explain later how these ideas can be adapted to the MFG setting.

Environment. Traditional RL aims at solving a stationary MDP, see Section 2.1.1. In the typical setting, the agent who is trying to find an optimal policy for the MDP interacts with an environment through experiments that can be summarized as follows:

1. The agent observes the current state x of the MDP (which is referred to as the state of the environment but could be for instance its own state or the state of the world).
2. The agent takes an action a, which is going to influence the state of the MDP through the transition kernel p and produces a reward through the function r.
3. The agent observes the new state $x' \sim p(\cdot|x, a)$ as well as the reward $r(x, a)$ resulting from its action.

The agents can repeat such experiments. We provide in Fig. 2.1 a schematic representation of this setting. It is often assumed that the agent can reboot the environment from time to time. To avoid remaining stuck in local maxima, it is common to assume that the new state is picked randomly, which is referred to an exploring start.

We stress that the agent does not observe directly the functions p and r that are used to compute the new state and the reward. The agent only observes the outputs of these functions. In some cases, recovering the functions p and r from such observations is feasible, leading to

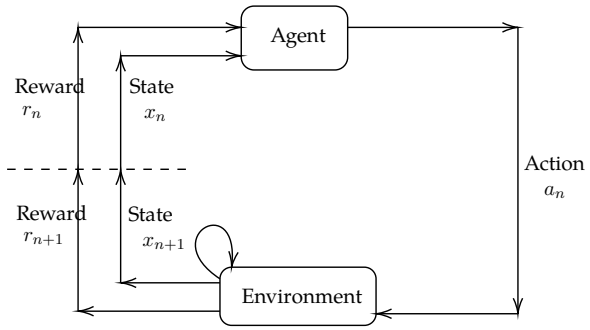


Figure 2.1 – Reinforcement learning environment: classical single-agent setup. Here, at iteration n, the current state of the MDP is x_n, the action taken by the agent is a_n, the new state is $x_{n+1} \sim p(\cdot|x_n, a_n)$ and the reward is $r_n = r(x_n, a_n)$. The new state x_{n+1} is observed by the agent and is also used for the next step of the environment's evolution.

the concept of **model-based RL**. However, for complex environments (*i.e.*, complex p and r), recovering the functions would require such a large number of observations that we generally prefer to directly aim for an optimal policy, which leads to the concept of **model-free RL**. The agent needs to interact multiple times to figure out the most suitable actions for a given state of the world. For more details, we refer the interested reader to Sutton and Barto (2018), Bertsekas (2012), Szepesvári (2010), and Meyn (2022).

Approximate dynamic programming. Some of the most popular RL methods are based on approximations of the exact dynamic programming equations satisfied by the value functions. Focusing on a stationary MDP, let us recall that an optimal policy can be computed for instance by value iteration or policy iteration (see Section 2.1.2), which require computing the state-action value functions Q^* or Q^π respectively. These two functions satisfy fixed-point equations whose solutions can be approximated by repeatedly applying the corresponding Bellman operators B^* and B^π, see (2.9) and (2.13). This amounts to repeating:

$$Q^\pi(x,a) \leftarrow r(x,a) + \gamma \mathbb{E}_{x' \sim p(\cdot|x,a), a' \sim \pi(\cdot|x')}[Q(x',a')], \qquad \forall (x,a) \in \mathcal{X} \times \mathcal{A}$$
$$Q^*(x,a) \leftarrow r(x,a) + \gamma \mathbb{E}_{x' \sim p(\cdot|x,a)}[\max_{a'} Q^*(x',a')], \qquad \forall (x,a) \in \mathcal{X} \times \mathcal{A}.$$

The arrow is used to denote that the value of $Q^\pi(x,a)$ is replaced by the value in the right hand side.

In the context of RL, we assume that the agent does not know r or p, so it cannot perform the above updates. However, these updates can be performed approximately provided we assume that the agent can query the environment and ask, for any pair (x,a), the value of $r(x,a)$ and a sample $x' \sim p(\cdot|x,a)$ (picked independently at each realization). Then to update $Q^\pi(x,a)$ and $Q^*(x,a)$, the agent can query multiple times the pair (x,a) and replace the expectations by

empirical averages:

$$\tilde{Q}^{\pi}(x,a) \leftarrow r(x,a) + \gamma \frac{1}{I} \sum_{i=1}^{I} \tilde{Q}(x^i, a^i), \qquad x^i \sim p(\cdot|x,a), a^i \sim \pi(\cdot|x^i), \quad \forall (x,a) \in \mathcal{X} \times \mathcal{A}$$

$$\tilde{Q}^{*}(x,a) \leftarrow r(x,a) + \gamma \frac{1}{I} \sum_{i=1}^{I} \max_{a'} \tilde{Q}^{*}(x^i, a'), \qquad x^i \sim p(\cdot|x,a), \quad \forall (x,a) \in \mathcal{X} \times \mathcal{A},$$

where the Monte Carlo samples x^i and a^i are independent. However, it is generally too computationally expensive to update every pair (x,a) using a batch of I samples. Furthermore, in many scenarios the agent does not have the freedom to query any state x. Instead, it is usually bound to observe the state of the environment, which is updated iteration after iteration in a sequential manner by following the dynamics of the state. The agent can influence the evolution of the state, but it cannot pick any new state that it wants at every iteration. In such scenarios, the agent can only perform updates by using the available information at each iteration.

To be specific, let us assume that the agent has a policy $\tilde{\pi}$ that it uses to generate a trajectory by interacting with the environment:

$$\begin{cases} x_0 \sim m_0, \\ a_n \sim \tilde{\pi}(\cdot|x_n), \quad x_{n+1} \sim p(\cdot|x_n, a_n), \quad n \geq 0. \end{cases}$$

The fixed-point equation satisfied by Q^* says that $\tilde{Q}^*(x,a)$ is well estimated if:

$$\tilde{Q}^{*}(x,a) = r(x,a) + \gamma \mathbb{E}_{x' \sim p(\cdot|x,a), a' \sim \pi(\cdot|x')} \left[\max_{a'} \tilde{Q}^{*}(x', a') \right]. \tag{2.15}$$

So it is natural to use the discrepancy between the right hand side and the left hand side to improve the estimate $\tilde{Q}^*$ of Q^*. Since we are bound to follow the trajectory, we cannot get the expectation over x' and, instead, we perform sampled-based updates using one sample at each step and a learning $\alpha > 0$:

$$\tilde{Q}^{*}(x_n, a_n) \leftarrow \tilde{Q}^{*}(x_n, a_n) + \alpha \left[r(x_n, a_n) + \gamma \max_{a'} \tilde{Q}^{*}(x_{n+1}, a') - \tilde{Q}^{*}(x_n, a_n) \right].$$

This leads to the celebrated **Q-learning** algorithm introduced by C. Watkins (1989) and whose convergence under suitable conditions has been proved by C. J. Watkins and Dayan (1992). Estimating correctly the whole function Q^* can be ensured if every pair (x,a) is visited infinitely often, which can be guaranteed under some assumptions on the dynamics of the state and by taking $\tilde{\pi}$ for instance as an ε-greedy policy (according to which in every state, every action has some probability to be selected). To estimate $\tilde{Q}^{\pi}$, a similar strategy can be used.

Deep reinforcement learning. When state and action spaces are finite, a state-action value function is simply a matrix, which can be stored in memory and processed easily when the spaces are small enough. Thanks to this, we can update the value function point by point (one point being a state-action pair in the case of Q-functions). However, when the spaces are very large or even continuous, it becomes impossible to evaluate precisely every pair (x, a). Furthermore, it is also impossible to visit all pairs during training, impliying that the question of **generalization** (*i.e.*, performance on unvisited pairs) cannot be avoided. Motivated by both efficiency and generalization capabilities, we can approximate the state-action value functions by parameterized non-linear functions such as neural networks. For example, let us approximate Q^* by a neural network Q_θ with a given architecture and parameters θ. Going back to (2.15), we note that now, not only we do not know the expected value, but also we cannot update the function only at a specific pair without changing its value at other pairs. Instead, we use the discrepancy between the left hand side and an estimation of the right hand side to define a loss function that can be used to train the neural network Q_θ. Since this neural network appears in both sides of the equation, to make the learning process more stable, we introduce an auxiliary neural network $Q_{\theta_{\text{target}}}$ called **target network** and we use it to replace Q_θ in the right hand side. The parameters θ_{target} are fixed when we update θ, and are updated from time to time but less frequently than θ. To be specific, we define the loss:

$$L(\theta; \theta_{\text{target}}) = \mathbb{E}_{x,a}\left[\left|Q_\theta(x,a) - r(x,a) - \mathbb{E}_{x'\sim p(\cdot|x,a)}\left[\max_{a'} Q_{\theta_{\text{target}}}(x',a')\right]\right|^2\right], \tag{2.16}$$

where the expectation is over state-action pairs. We do not specify here the distribution of this pair, but in practice it typically comes from played trajectories stored in a replay buffer. In practice, this loss is minimized using stochastic gradient descent on mini-batches sampled from this replay buffer. This leads to the **DQN algorithm** introduced by Mnih et al. (2013).

The above approach uses that we can easily compute the maximal value of $Q_{\theta_{\text{target}}}(x', \cdot)$ over the action space. This is typically possible only if the action space is finite and not too large. Otherwise, we can use another neural network to encode a policy and train this neural network so that it approximates an optimal policy, using a so called policy loss. Then, in loss (2.16), the term $\max_{a'} Q_{\theta_{\text{target}}}(x', a')$ is replaced by the expectation of the target Q-value according to the learnt policy. The resulting agent is called an actor-critic (the actor is the policy, the critic is the state-action value function). Since our goal is not to present an exhaustive list of methods, we simply mention below three popular approaches, to give an idea of the variety of algorithms:

- If we consider only deterministic policies, we can replace the policy by a parameterized function $\varphi_\omega : \mathcal{X} \to \mathcal{A}$ with parameters ω. In this case, the corresponding policy loss optimises for
$$\max_\omega \mathbb{E}_{x'}[Q_{\theta_{\text{target}}}(x', \varphi_\omega(x'))].$$

Its gradient can be obtained using the chain rule. This leads to an algorithm that is reminiscent of the **Deep Deterministic Policy Gradient (DDPG)** of Lillicrap et al. (2016).

- Another approach is to look for a general stochastic policy π, in which case we have the interpretation :

$$\mathbb{E}_{a'\sim\pi(\cdot|x')}[Q_{\theta_{\text{target}}}(x',a')] = \int_{\mathcal{A}} Q_{\theta_{\text{target}}}(x',a')\pi(a'|x')da'.$$

Taking the gradient and using the log trick yields the state-wise gradient:

$$\mathbb{E}_{a'\sim\pi(\cdot|x')}\left[Q_{\theta_{\text{target}}}(x',a')\nabla\log\pi(a'|x')\right].$$

The resulting algorithm is reminiscent of **Reinforce** (Williams, 1992). The related empirical gradient requires sampling from the learnt policy and is usually of high variance. An alternative approach consists in using the reparameterization trick. For example, we can restrict our attention to Gaussian policies $\pi_\omega(\cdot|x) = \Phi_{\mathcal{N}(m_\omega(x),\sigma_\omega(x)I)}(\cdot)$, where $\Phi_{\mathcal{N}(m_\omega(x),\sigma_\omega(x)I)}$ denotes the density function of the normal distribution $\mathcal{N}(m_\omega(x),\sigma_\omega(x)I)$, with m_ω and σ_ω being two parameterized functions with parameters ω. Here I denotes the identity matrix on $\mathcal{A}$. This leads to the following approximation:

$$\begin{aligned}&\mathbb{E}_{a'\sim\pi(\cdot|x')}[Q_{\theta_{\text{target}}}(x',a')] \approx \mathbb{E}_{a'\sim\pi_\omega(\cdot|x')}[Q_{\theta_{\text{target}}}(x',a')]\\ &= \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,I)}[Q_{\theta_{\text{target}}}(x',m_\omega(x)+\sigma_\omega(x)\varepsilon)].\end{aligned}$$

Here again, we can replace the expectation by an average over a finite number of realizations of ε. In this way, we obtain an algorithm which is similar to **TD3** (Fujimoto, Hoof, and Meger, 2018).

Since the goal of this section is simply to describe some of the key ideas behind DRL, we do not go further into a detailed presentation of the variety of existing methods. We refer the interested reader to e.g. Arulkumaran et al. (2017) and François-Lavet et al. (2018).

2.2 Mean Field Games: Definition and Settings

In this section, we present several settings of MFGs which depend on how time is involved (or not) in the problem. These settings correspond to different applications and different notions of Nash equilibrium. Here, we focus on four settings, that can be summarized as follows. We start with games in which the agents take a single decision. There is no notion of time intrinsic to the game so we call them **static MFGs**. We then turn to games in which there is a dynamical aspect. In such games, each agent has a state that evolves along time, and it can act on this evolution. At the level of the population, in some situations, we can expect the

distribution of states to converge to a stationary regime, in which the population is stable at a macroscopic level, even though each agent's state is possibly changing. We refer to this setting as a **stationary MFG**. In other cases, one wants to understand not only the stationary regime, but how the population evolves, starting from an initial configuration. This is relevant for applications in which the agents' behaviors change along time, for instance because there is a finite horizon at which the game stops. We call such games **evolutive MFGs**. This setting comes at the expense of having policies and mean-field terms that depend on time and are thus harder to compute. To mitigate this complexity while not falling completely into the stationary regime, an intermediate model has been introduced. The idea is to try to keep the best of the stationary and evolutive settings by considering a proxy for the whole evolution of the distribution. We call this setting **discounted stationary MFGs**. In the rest of this section, we define each setting as well as the corresponding notion of Nash equilibrium, along with relevant concepts.

Remark 2. ***Mean field games vs. non-atomic anonymous games.*** *Games modeling infinite populations of agents have also been studied in the framework of non-atomic anonymous games, which have founds applications particularly in economics, see e.g. Aumann (1964), Schmeidler (1973), and Aumann and Shapley (2015). In such games, there is typically a continuum of players, indexed by, say, real numbers in $I = [0, 1]$ and the population is represented by a non-atomic measure on I. Each player perceives the other players through some aggregate quantity. Although this is very similar to the MFG framework, the key difference is that the MFG approach completely avoids representing the continuum of players. The main idea is to exploit the homogeneity of the population and the symmetry of interactions to simplify the characterization of an equilibrium: it is sufficient to understand the behavior of a single representative player facing a distribution representing the aggregate information available to this player. The analysis is greatly simplified, particularly when it comes to stochastic games. Defining rigorously a continuum of random variables with nice measurability properties is not trivial, as noticed for instance by Duffie and Y. Sun (2012) and Y. Sun (2006) who used the concept of rich Fubini extension to develop an exact law of large numbers. The MFG framework allows to carry out the mathematical analysis of Nash equilibria without requiring such sophistication.*

2.2.1 Static MFG

Let $\mathcal{A}$ be a finite action space. The behavior of one player, called a **strategy**, is an element of $\Delta_{\mathcal{A}}$, that is a distribution over the action set. In this setting, the behavior of the population is also an element of $\Delta_{\mathcal{A}}$. We denote a generic element of $\mathcal{A}$ by a, and we denote a generic individual behavior and a generic population behavior by π and ξ respectively.

Besides the action space $\mathcal{A}$, the model is completely defined by a reward function $r : \mathcal{A} \times \Delta_{\mathcal{A}} \to \mathbb{R}$. Given a population behavior $\xi \in \Delta_{\mathcal{A}}$, the reward of a player using $\pi \in \Delta_{\mathcal{A}}$ is

defined as the expected reward when sampling an action according to π:

$$J_{\text{static}}(\pi; \xi) = \mathbb{E}_{a \sim \pi} \left[r(a, \xi) \right].$$

The reward function $r : \mathcal{A} \times \Delta_{\mathcal{A}} \to \mathbb{R}$ can typically encode crowd aversion or attraction towards a population action of interest.

Example 1. *One of the first examples in the MFG literature is the problem of choosing a starting time for a meeting, introduced and solved explicitly by Guéant, Lasry, and Lions (2011). In this problem, the players choose at what time they want to arrive to the meeting room so that they are neither too late nor too early. The global outcome is the time at which the meeting actually starts, which is not known in advance and depends on the everyone's arrival time. Despite its name, there is no dynamic aspect in the original formulation of the example. Another popular example is the problem in which each agent chooses a location on a beach, see e.g. Perrin, Perolat, et al. (2020). They want to put their towel close to a bar (or an ice cream stall) but not in a very crowded area. The global outcome is the distribution of towels on the beach. To be specific, a simple model can be as follows: $\mathcal{A} = [0, 1]$, which represents possible positions on the beach, $a_{\text{bar}} \in \mathcal{A}$ is the position of the bar, and the reward is $r(a, \xi) = -|a - a_{\text{bar}}| - \ln(\xi(a))$, where the first term penalizes the distance to the bar and the second term penalizes choosing a location a at which the density $\xi(a)$ of people is high.*

A central concept is the notion of best response. Let us define the (set-valued) **best response map** by:

$$\texttt{BR}_{\text{static}} : \Delta_{\mathcal{A}} \to 2^{\Delta_{\mathcal{A}}}, \xi \mapsto \texttt{BR}_{\text{static}}(\xi) := \arg\max_{\pi \in \Delta_{\mathcal{A}}} J_{\text{static}}(\pi; \xi).$$

Definition 1 (Static MF Nash Equilibrium)**.** *$\hat{\pi} \in \Delta_{\mathcal{A}}$ is a **static mean field Nash equilibrium** (static MFNE) if it satisfies the following condition: $\hat{\pi} \in \texttt{BR}_{\text{static}}(\hat{\pi})$.*

The above definition has the advantage to clearly show that the equilibrium is a fixed point of $\texttt{BR}_{\text{static}}$.

Another point of view, which will be useful in the dynamic settings presented in the sequel, consists in saying that the equilibrium is given by a pair of a representative agent's behavior and the population's behavior. Here, it means that the equilibrium is a pair $(\hat{\pi}, \hat{\xi}) \in \Delta_{\mathcal{A}} \times \Delta_{\mathcal{A}}$ such that:

1. $\hat{\pi}$ is optimal for the representative agent facing $\hat{\xi}$, i.e., $\hat{\pi} \in \texttt{BR}_{\text{static}}(\hat{\xi})$,
2. $\hat{\xi}$ corresponds to the population behavior when all every agent uses $\hat{\pi}$, i.e., $\hat{\xi} = \hat{\pi}$.

The second point represents the fact that all the agents are "rational in the same way" and hence, at equilibrium, adopt the same behavior. This viewpoint is unnecessarily complicated in this setting as $\hat{\pi}$ alone is enough to define the MFNE, but will be useful in dynamic settings.

Remark 3. *Consistently with the literature on normal-form games (Fudenberg and Tirole, 1991), each player chooses a distribution over actions* without *seeing the strategy chosen by other players and the resulting distribution at the population level. Each agent thus tries to anticipate, in a rational way, the distribution generated by other players' actions.*

A Nash equilibrium corresponds to a situation in which no selfish player has any incentive to deviate unilaterally. However, it is not necessarily a situation that is optimal from the point of view of the whole population. The notion of social optimum is discussed below in Section 2.2.6.

Remark 4. *Although we provided an intuitive explanation for ξ in terms of a continuum of players, we want to stress that in the definition of an MFG equilibrium or social optimum, we actually never need to consider a continuum of players. As already pointed out in Remark 2, this shortcut is one of the main advantages of the MFG paradigm compared with non-atomic anonymous games.*

2.2.2 Notations for the dynamic setting

In contrast with the static case, in the dynamic setting, each agent has a state which evolves in time. The agent can influence the evolution of their own state using actions. The population's state is the distribution of the agents' states, the joint distribution of their states and actions. This is what constitutes the mean field, with which every agent interacts through its dynamics and its rewards.

As far as the population distribution is concerned, we will consider mainly two types of settings: one in which the population distribution is fixed, and one in which it can also evolve. Typically, the former is conceptually simpler and easier to compute but the latter is more realistic since many real world applications involve a population evolving in time. In each cases, several variants can be considered. For the sake of brevity, we shall focus only on the main ideas.

We will consider discrete time models, with time typically denoted by $n \in \mathbb{N}$. If a **time horizon** is imposed, we will typically use the notation N_T. Let $\mathcal{X}$ be a finite **state space**. A **stationary policy** is an element of $\Pi := (\Delta_{\mathcal{A}})^{\mathcal{X}}$. In this setting, we assume that the interactions occur through an aggregate variable which represents the behavior of the population. A **mean field state** is an element of $\Delta_{\mathcal{X}}$, which is the set of probability distributions on the state space. It represents the state of the population at a given time. We denote generic elements of $\mathcal{X}$, $\mathcal{A}$, Π, and $\Delta_{\mathcal{X}}$ respectively by x, a, π, and μ.

Depending on the setting, we might consider policies that depend on time or on an initial distribution. More details will be provided below, as we introduce several setups.

Remark 5. *To alleviate the presentation, we choose to focus on finite state and action spaces and discrete time. In some cases, continuous space or continuous time models might be more relevant. They are typically analyzed using partial differential equations or stochastic differential equations. Suitable*

discretizations can lead to (possibly non-trivial) approximations of these models with discrete ones, as presented in this survey, see e.g. Hadikhanloo and Silva (2019) for more details on the convergence of a finite MFG to a continuous one. We do not discuss in detail the continuous settings here and we refer the interested reader to the literature, e.g., M. Huang, R. P. Malhamé, and Caines (2006), Lasry and Lions (2007), Bensoussan, Frehse, and P. Yam (2013), Carmona and Delarue (2018b), and Carmona and Delarue (2018c).

2.2.3 Stationary setting

Here we consider an infinite horizon model, meaning that there is no terminal time. We assume that the players interact through a *stationary* distribution, which represents a steady state of the population. The model is defined by a tuple $(\mathcal{X}, \mathcal{A}, p, r, \gamma)$ consisting of:

- a state space $\mathcal{X}$ and an action space $\mathcal{A}$,
- a one-step transition probability kernel $p : \mathcal{X} \times \mathcal{A} \times \Delta_{\mathcal{X}} \to \Delta_{\mathcal{X}}$,
- a one-step reward function $r : \mathcal{X} \times \mathcal{A} \times \Delta_{\mathcal{X}} \to \mathbb{R}$,
- and a discount factor $\gamma \in [0, 1]$.

The main difference with standard MDPs as recalled in Section 2.1.1 is the presence of a third input for p and r, which is an element of the mean field state space $\Delta_{\mathcal{X}}$. It plays the role of the population's state, which influences the dynamics and the rewards.

Assume the state of the population is given by $\mu \in \Delta_{\mathcal{X}}$ and consider a representative agent using policy $\pi \in \Pi$. The total, discounted reward of this player is given by:

$$J_{\text{statio}}(\pi; \mu) = \mathbb{E}\left[\sum_{n=0}^{\infty} \gamma^n r(x_n, a_n, \mu)\right], \tag{2.17}$$

where the state of the agent evolves according to:

$$\begin{cases} x_0 \sim \mu, \\ x_{n+1} \sim p(\cdot|x_n, a_n, \mu), \quad a_n \sim \pi(\cdot|x_n), \quad n \geq 0. \end{cases} \tag{2.18}$$

Remark 6 (State-action distribution)**.** *An extension of the above model is to consider that the agents interact through the state-action distribution. In this case, assume the state of the population is given by $\xi \in \Delta_{\mathcal{X}\times\mathcal{A}}$ and consider a representative agent using policy $\pi \in \Pi$. The total, discounted reward of a representative player is given by:*

$$J_{\text{statio}}(\pi; \xi) = \mathbb{E}\left[\sum_{n=0}^{\infty} \gamma^n r(x_n, a_n, \xi)\right],$$

where the state of the agent evolves according to:

$$\begin{cases} x_0 \sim \mu = \xi_1, \\ x_{n+1} \sim p(\cdot|x_n, a_n, \xi), \quad a_n \sim \pi(\cdot|x_n), \quad n \geq 0. \end{cases}$$

*with $\mu = \xi_1 \in \Delta_{\mathcal{X}}$ denoting the first marginal of ξ. This setting is considered for instance by Guo, A. Hu, et al. (2019) and Guo, A. Hu, et al. (2020). MFG with interactions through state-action distributions have first been studied by D. A. Gomes, Patrizi, and V. Voskanyan (2014) and D. A. Gomes and V. K. Voskanyan (2016) and are sometimes referred to as **extended MFGs** or **MFG of controls**, see Cardaliaguet and Lehalle (2018) and Kobeissi (2022). Let us stress that a state-action distribution is not always a product distribution, meaning that for some $\xi \in \Delta_{\mathcal{X}\times\mathcal{A}}$ there is no $\mu \in \Delta_{\mathcal{X}}$ and $\nu \in \Delta_{\mathcal{A}}$ such that $\xi = \mu \otimes \nu$. In fact, in general the actions of a player are given by a function of the player's states, and hence the joint distribution cannot be written as a product. To simplify the presentation, we restrict our attention to interactions through state-only distributions but most of the ideas can be extended to state-action distributions. The interested reader is referred to Carmona and Delarue (2018b, Section 4.6) and the references therein.*

This stationary MFG setting has been studied for instance by J. Subramanian and Mahajan (2019) with applications to malware spread and investments in product quality, by Guo, A. Hu, et al. (2019) and Guo, A. Hu, et al. (2020) with applications to auctions and by Angiuli, Fouque, and Laurière (2022) in the context of linear-quadratic MFGs.

Example 2 (Repeated auction game)**.** *As an example, Guo, A. Hu, et al. (2019) consider a repeated game in which the players bid in an auction game. At a given time, a player's state and action are respectively its budget and its bid for the next auction.*

The evolution of the population is given by a transition matrix defined by: for all $\tilde{\mu} \in \Delta_{\mathcal{X}}$, $\pi \in \Pi$, $\mu \in \Delta_{\mathcal{X}}$ and $x \in \mathcal{X}$,

$$(P^{\top}_{\tilde{\mu},\pi}\mu)(y) = \sum_x \mu(x) \sum_a \pi(a|x)p(y|x,a,\tilde{\mu}). \tag{2.19}$$

In words, $P^{\top}_{\tilde{\mu},\pi}\mu$ is the next state distribution for a representative agent starting with state distribution μ and using policy π while the population has state distribution $\tilde{\mu}$.

Given a population state, the goal for a representative agent, is to find the best reaction, i.e., a policy that maximizes their total reward. We define the (set-valued) **best response map** by:

$$\mathtt{BR}_{\mathrm{statio},\gamma} : \Delta_{\mathcal{X}} \to 2^{\Pi}, \quad \mu \mapsto \mathtt{BR}_{\mathrm{statio},\gamma}(\mu) := \arg\max_{\pi\in\Pi} J_{\mathrm{statio}}(\pi;\mu) \subseteq \Pi,$$

and the (set-valued) **population behavior map** by:

$$\mathbf{M}_{\text{statio}} : \Pi \to 2^{\Delta_{\mathcal{X}}}, \quad \pi \mapsto \mathbf{M}_{\text{statio}}(\pi) := \{\mu \in \Delta_{\mathcal{X}} \,|\, \mu = P_{\mu,\pi}^{\top}\mu\}, \tag{2.20}$$

which is the **stationary distribution** obtained when using π (that we assume to be unique).

Remark 7. *Note that solving the equation is not trivial since μ is involved in the transition matrix $P_{\mu,\pi}$. We come back to this point in Section 3.1.2.*

Definition 2 (Stationary MF Nash Equilibrium)**.** *A pair $(\hat{\pi}, \hat{\mu}) \in \Pi \times \Delta_{\mathcal{X}}$ is a **stationary mean field Nash equilibrium** (stationary MFNE) if it satisfies the following two conditions:*

- $\hat{\pi} \in \texttt{BR}_{\text{statio},\gamma}(\hat{\mu})$;
- $\hat{\mu} \in \mathbf{M}_{\text{statio}}(\hat{\pi})$.

*Alternatively, an equilibrium can be defined as a fixed point: $\hat{\pi}$ is a **stationary MFNE policy** if it is a fixed point of $\texttt{BR}_{\text{statio},\gamma} \circ \mathbf{M}_{\text{statio}}$, and $\hat{\mu}$ is a **stationary MFNE distribution** if it is the stationary distribution of a stationary MFNE policy.*

In this setting, the **state-action value function** associated to a stationary policy π for a given distribution μ is defined as:

$$Q^{\pi,\mu}(x,a) = \mathbb{E}\left[\sum_{n\geq 0} \gamma^n r(x_n, a_n, \mu) \Big| x_0 = x, a_0 = a, x_{n+1} \sim p(\cdot|x_n, a_n, \mu), a_n \sim \pi(\cdot|x_n)\right].$$

The problem then reduces to a standard stationary MDP, parameterized by μ. By dynamic programming, $Q^{\pi,\mu}$ satisfies the fixed point equation:

$$Q = B^{\pi,\mu}Q, \tag{2.21}$$

where $B^{\pi,\mu}$ denotes the **Bellman operator** associated to π and μ:

$$(B^{\pi,\mu}Q)(x,a) = r(x,a,\mu) + \gamma \mathbb{E}_{x'\sim p(\cdot|x,a,\mu), a'\sim\pi(\cdot|x')}[Q(x',a')], \tag{2.22}$$

where

$$\mathbb{E}_{x'\sim p(\cdot|x,a,\mu), a'\sim\pi(\cdot|x')}[Q(x',a')] = \sum_{x'} p(x'|x,a,\mu) \sum_{a'} \pi(a'|x')Q(x',a'). \tag{2.23}$$

The **optimal state-action value function** is defined as:

$$Q^{*,\mu}(x,a) = \sup_{\pi} Q^{\pi,\mu}(x,a).$$

It satisfies the fixed point equation:

$$Q = B^{*,\mu}Q, \tag{2.24}$$

where $B^{*,\mu}$ denotes the **optimal Bellman operator** associated to μ:

$$(B^{*,\mu}Q)(x,a) = r(x,a,\mu) + \gamma \mathbb{E}_{x' \sim p(\cdot|x,a,\mu)}[\max_{a'} Q(x',a')], \tag{2.25}$$

with

$$\mathbb{E}_{x' \sim p(\cdot|x,a,\mu)}[\max_{a'} Q(x',a')] = \sum_{x'} p(x'|x,a,\mu) \max_{a'} Q(x',a'). \tag{2.26}$$

Note that the functions $Q^{\pi,\mu}$ and $Q^{*,\mu}$, and the operators $B^{\pi,\mu}$ and $B^{*,\mu}$ are all independent of time, as we search for stationary equilibria.

2.2.4 Evolutive setting

We next turn our attention to a model in which not only the agents' state can evolve, but the population's distribution too. In this case, the mean field is not stationary. At each time step, the transition and the reward of every agent depends on the *current* distribution instead of the stationary one. The model is defined by a tuple $(\mathcal{X}, \mathcal{A}, m_0, N_T, p, r)$ consisting of:

- a state space $\mathcal{X}$ and an action space $\mathcal{A}$,
- an initial distribution $m_0 \in \Delta_{\mathcal{X}}$,
- a time horizon $N_T \in \mathbb{N} \cup \{+\infty\}$,
- a sequence of one-step transition probability kernels $p_n : \mathcal{X} \times \mathcal{A} \times \Delta_{\mathcal{X}} \to \Delta_{\mathcal{X}}$, $n \geq 0$,
- a sequence of one-step reward functions $r_n : \mathcal{X} \times \mathcal{A} \times \Delta_{\mathcal{X}} \to \mathbb{R}$, $n \geq 0$.

In this context, a population behavior is a **mean field flow**, generally denoted by $\boldsymbol{\mu}$, which is an element of $\Delta_{\mathcal{X}}^{N_T}$. A **policy** is an element of Π^{N_T}, generally denoted by $\boldsymbol{\pi}$. We use bold letter to stress that these are sequences, which can also be viewed as functions of time.

The total reward is:

$$J_{\text{evol}}(\boldsymbol{\pi}; \boldsymbol{\mu}) = \mathbb{E}\left[\sum_{n=0}^{N_T} r_n(x_n, a_n, \boldsymbol{\mu}_n)\right],$$

subject to the following evolution of the agent's state:

$$\begin{cases} x_0 \sim m_0, \\ x_{n+1} \sim p_n(\cdot|x_n, a_n, \boldsymbol{\mu}_n), \quad a_n \sim \boldsymbol{\pi}_n(\cdot|x_n), \quad n \geq 0. \end{cases} \tag{2.27}$$

This evolution is analogous to (2.18) except that the stationary mean-field state is replaced by the current mean-field state at time n.

We assume that the transition p and the reward r are such that the total reward is well defined.

Remark 8 (Finite and infinite horizon discounted settings)**.** *Our notation covers two very common settings:*

- *Finite horizon:* $N_T < +\infty$.
- *Infinite horizon:* $N_T = +\infty$. *In this case, it is common to assume that* p *is independent of time, and that* r *is of the form* $r_n(x, a, \mu) = \gamma^n \tilde{r}(x, a, \mu)$ *where* $\tilde{r}$ *is independent of time and bounded.*

Note that even in the infinite horizon setting and even if p *and* r *are stationary (constant with respect to the time parameter* n*), in general the optimal policy still depends on time. This is in contrast with the stationary setting (section 2.2.3) and is due to the fact that the population distribution starts from* m_0 *and evolves. The player needs to take that into account in its decisions. To be specific, even if* $r_n(x, a, \mu) = \tilde{r}(x, a, \mu)$ *is independent of time, the reward associated to a fixed state-action pair* (x, a) *is* $\tilde{r}(x, a, \boldsymbol{\mu}_n)$ *at time* n *and* $\tilde{r}(x, a, \boldsymbol{\mu}_{n'})$ *at time* n'*. Unless the mean-field state is stationary, in general the two reward values will be different.*

Example 3 (Crowd motion)**.** *This setting is probably the most commonly studied one in the MFG literature. As a typical example, we can think of a model for crowd motion in the spirit of e.g. Achdou and Lasry (2019): the agents start from an initial position and want to reach a point of interest while avoiding crowded areas. Because the population distribution changes as the agents move, looking for a stationary solution is not satisfactory if we want to compute the evolution of the whole population. This is because a stationary solution would only give the optimal policy (from the Nash equilibrium perspective) against the stationary distribution, and would not be able to recover the full evolution of the agents. In contrast, a time-dependent policy in the evolutive setup is able to adjust the agents' behavior step by step.*

Remark 9. *Discrete time finite state mean field games have been introduced by D. A. Gomes, Mohr, and Souza (2010). In the model analyzed therein, the players can directly control their transition probabilities. Note that the model we consider here is a bit more general since the transition probabilities are functions of the actions, but they are not necessarily chosen freely by the players.*

We define the (set-valued) **best response map** by:

$$\mathtt{BR}_{\mathrm{evol},m_0,N_T}(\boldsymbol{\mu}) := \operatorname*{arg\,max}_{\boldsymbol{\pi}\in\Pi^{N_T}} J_{\mathrm{evol}}(\boldsymbol{\pi};\boldsymbol{\mu}) \subseteq \Pi^{N_T}.$$

Let us define the **population behavior map** by:

$$\mathbf{M}_{\mathrm{evol},m_0,N_T} : \Pi^{N_T} \to \Delta_{\mathcal{X}}^{N_T}, \quad \mathbf{M}_{\mathrm{evol},m_0,N_T}(\boldsymbol{\pi}) := \text{ mean field flow when using } \boldsymbol{\pi} \text{ and starting from } m_0, \tag{2.28}$$

where this flow is defined by:

$$\begin{cases} \boldsymbol{\mu}_0 = m_0, \\ \boldsymbol{\mu}_{n+1} = P^{\top}_{n,\boldsymbol{\mu}_n,\boldsymbol{\pi}_n} \boldsymbol{\mu}_n, \quad n \geq 0. \end{cases} \tag{2.29}$$

When the context is clear, we will simply write $\boldsymbol{\mu}^{m_0,\boldsymbol{\pi}}$.

Definition 3 (Evolutive MFG Nash Equilibrium)**.** *A pair $(\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\mu}}) \in \Pi^{N_T} \times \Delta_{\mathcal{X}}^{N_T}$ is an **evolutive mean field Nash equilibrium** (evolutive MFNE) if it satisfies the following two conditions:*

- *Best response: $\hat{\boldsymbol{\pi}} \in \mathtt{BR}_{\mathrm{evol},m_0,N_T}(\hat{\boldsymbol{\mu}})$;*
- *Mean field flow: $\hat{\boldsymbol{\mu}} = \mathbf{M}_{\mathrm{evol},m_0,N_T}(\hat{\boldsymbol{\pi}})$.*

*Alternatively, an evolutive MFNE can be defined as a fixed point: $\hat{\boldsymbol{\pi}}$ is an **evolutive MFNE policy** if it is a fixed point of $\mathtt{BR}_{\mathrm{evol},m_0,N_T} \circ \mathbf{M}_{\mathrm{evol},m_0,N_T}$, and $\hat{\boldsymbol{\mu}}$ is an **evolutive MFNE flow** if it is the mean field flow generated by an evolutive MFNE policy.*

The state-action value function associated to a policy $\boldsymbol{\pi}$ and the optimal state-action value function are defined analogously to standard MDP but parameterized by $\boldsymbol{\mu}$. We denote them respectively by $\mathbf{Q}^{\boldsymbol{\pi},\boldsymbol{\mu}}$ and $\mathbf{Q}^{*,\boldsymbol{\mu}}$.

For the sake of completeness, let us provide more details in the finite horizon setting. Assume $N_T < +\infty$. The **state-action value function** associated to a policy $\boldsymbol{\pi}$ for a given distribution $\boldsymbol{\mu}$ is defined as:

$$\begin{cases} \mathbf{Q}^{\boldsymbol{\pi},\boldsymbol{\mu}}_{N_T}(x,a) = r_{N_T}(x,a,\boldsymbol{\mu}_{N_T}) \\ \mathbf{Q}^{\boldsymbol{\pi},\boldsymbol{\mu}}_{n}(x,a) = \mathbb{E}\left[\displaystyle\sum_{n'=n}^{N_T} r_{n'}(x_{n'},a_{n'},\boldsymbol{\mu}_{n'}) \Big| x_n = x, a_n = a, x_{n'+1} \sim p_{n'}(\cdot|x_{n'},a_{n'},\boldsymbol{\mu}_{n'}), a_{n'} \sim \boldsymbol{\pi}_{n'}(\cdot|x_{n'}) \right], \\ \qquad\qquad n = N_T - 1, \dots, 0. \end{cases}$$

The **optimal state-action value function** is defined as:

$$\mathbf{Q}^{*,\boldsymbol{\mu}}(x,a) = \sup_{\boldsymbol{\pi}} \mathbf{Q}^{\boldsymbol{\pi},\boldsymbol{\mu}}(x,a).$$

2.2.5 Discounted stationary setting

We now discuss a setting that is somehow between the stationary and the evolutive ones. Note that in the stationary setting, we focus on the stationary distribution of the population while in the evolutive setting, we care about the entire sequence starting from m_0. In the first case, we can restrict our attention to stationary policies, whereas this is not possible in the second case, as highlighted in Remark 8. An intermediate approach consists in replacing the distribution

$\boldsymbol{\mu}_n^{m_0,\boldsymbol{\pi}}$ at time n by an aggregate which keeps some memory of m_0, instead of the stationary distribution.

The model is defined by a tuple $(\mathcal{X}, \mathcal{A}, m_0, p, r, \gamma)$ consisting of:

- a state space $\mathcal{X}$ and an action space $\mathcal{A}$,
- an initial distribution $m_0 \in \Delta_{\mathcal{X}}$,
- a one-step transition probability kernel $p : \mathcal{X} \times \mathcal{A} \times \Delta_{\mathcal{X}} \to \Delta_{\mathcal{X}}$,
- a one-step reward function $r : \mathcal{X} \times \mathcal{A} \times \Delta_{\mathcal{X}} \to \mathbb{R}$,,
- a discount factor $\gamma \in [0, 1)$.

We define the **discounted distribution** as:

$$\mathbf{M}_{\text{statio},\gamma,m_0}(\pi) := \mu_\gamma^{m_0,\pi} := (1-\gamma) \sum_{n \geq 0} \gamma^n \boldsymbol{\mu}_n^{m_0,\pi} \in \Delta_{\mathcal{X}},$$

where $\boldsymbol{\mu}^{m_0,\pi}$ follows the dynamics (2.29) but with $\boldsymbol{\pi}_n = \pi$ for all $n \geq 0$ after starting from m_0 at time 0, and with the mean-field term replaced by $\mu_\gamma^{m_0,\pi}$, i.e.,

$$\begin{cases} \boldsymbol{\mu}_0^{m_0,\pi} = m_0, \\ \boldsymbol{\mu}_{n+1} = P^\top_{\mu_\gamma^{m_0,\pi},\pi} \boldsymbol{\mu}_n^{m_0,\pi}, \quad n \geq 0. \end{cases}$$

This formulation allows us to work with a single distribution, which plays the role of a summary of what happens along the mean field flow. In contrast with the stationary MFG setting, here the initial distribution m_0 still influences the mean field term, namely, $\mu_\gamma^{m_0,\pi}$. However, we can restrict our attention to stationary policies just as in the stationary MFG setting.

Example 4 (Exploration)**.** *In Perrin, Perolat, et al. (2020) and Geist, Pérolat, et al. (2021), this setting has been used for an MFG in which the agents explore the spatial domain. From the point of view of the population, it amounts to maximizing the entropy of the distribution. The discounted stationary distribution can be used as a proxy to evaluate with a single distribution how well the population explore the state space.*

Remark 10. *The discounted distribution can be interpreted as the stationary distribution of an agent who starts with distribution m_0, uses policy π but has a probability to stop at any time step. To be specific, let τ be a random variable with geometric distribution on $\{0, 1, 2 \ldots, \}$ with parameter $(1-\gamma)$.*

Let us denote by $\boldsymbol{\mu}_n^{\gamma,m_0,\pi}$ is the distribution of x_n, where:

$$\begin{cases} x_0 \sim m_0, \\ x_{n+1} \sim p(\cdot|x_n, a_n, \mu_\gamma^{m_0,\pi}), \quad a_n \sim \pi(\cdot|x_n), \quad 0 \leq n \leq \tau \\ x_{n+1} = x_n, \quad \tau \leq n \end{cases}$$

Then we have: for every $x \in \mathcal{X}$,

$$\begin{aligned} \mathbb{P}(x_n = x) &= \sum_{k \leq n} \mathbb{P}(\tau = k)\mathbb{P}(x_k = x|\tau = k) + \mathbb{P}(\tau > n)\mathbb{P}(x_n = x|\tau > n) \\ &= (1-\gamma)\sum_{k \leq n} \gamma^k \boldsymbol{\mu}_k^{\gamma,m_0,\pi}(x) + \mathbb{P}(\tau > n)\mathbb{P}(x_n = x|\tau > n). \end{aligned}$$

When $n \to +\infty$, $\mathbb{P}(\tau > n) \to 0$, so we obtain that:

$$\mu_\gamma^{m_0,\pi}(x) = \lim_{n \to +\infty} \mathbb{P}(x_n = x) = (1-\gamma)\sum_k \gamma^k \boldsymbol{\mu}_k^{\gamma,m_0,\pi}(x).$$

Definition 4 (Discounted stationary MFG Nash Equilibrium)**.** *A pair $(\hat{\pi}, \hat{\mu}) \in \Pi \times \Delta_{\mathcal{X}}$ is a **discounted stationary mean field Nash equilibrium** (discounted stationary MFNE) if it satisfies the following two conditions:*

- $\hat{\pi} \in \texttt{BR}_{\text{statio},\gamma}(\hat{\mu})$;
- $\hat{\mu} = \mathbf{M}_{\text{statio},\gamma,m_0}(\hat{\pi})$.

*Alternatively, $\hat{\mu}$ is a **discounted stationary mean field Nash equilibrium distribution** if it is a fixed point of: $\mathbf{M}_{\text{statio},\gamma,m_0} \circ \texttt{BR}_{\text{statio},\gamma}$.*

2.2.6 Social optimum and Mean Field Control

The notions of MFNE discussed above correspond to non-cooperative games, in which each player maximizes their own reward while trying to anticipate the behavior of other selfish agents. A different question consists in considering that the agents are cooperative and try to maximizer a social welfare criterion by choosing together a suitable policy. This situation can also be interpreted as an optimization problem from the point of view of a social planner, who tries to figure out which behavior is optimal from the society standpoint.

Static setting. The social welfare function is defined as the reward obtained on average by the agents:

$$\pi \mapsto J_{\text{static}}^{\text{social}}(\pi) := J_{\text{static}}(\pi; \pi).$$

A strategy π^* is a **static mean field social optimum** (static MFSO) if it maximizes the social welfare function $J^{\text{social}}_{\text{static}}$.

Stationary case. The total, discounted social welfare associated to a policy π is:

$$J^{\text{social}}_{\text{statio}}(\pi) = J_{\text{statio}}(\pi; \mu^\pi) = \mathbb{E}\left[\sum_{n=0}^{\infty} \gamma^n r(x_n, a_n, \mu^\pi)\right]$$

subject to:

$$\begin{cases} x_0 \sim \mu^\pi, \\ x_{n+1} \sim p(\cdot|x_n, a_n, \mu^\pi), \quad a_n \sim \pi(\cdot|x_n), \quad n \geq 0, \end{cases}$$

where μ^π is the stationary distribution induced by π. Here we see that perturbing π changes μ^π, which is reflected in the third argument of the transition function and the reward function. An **optimal stationary policy** is a $\pi \in \Pi$ maximizing $J^{\text{social}}_{\text{statio}}$. This setting has been considered by J. Subramanian and Mahajan (2019) or by Angiuli, Fouque, and Laurière (2022).

Evolutive case. The total social welfare is:

$$J^{\text{social}}_{\text{evol}}(\boldsymbol{\pi}) = J_{\text{evol}}(\boldsymbol{\pi}; \boldsymbol{\mu}^{m_0,\boldsymbol{\pi}}) = \mathbb{E}\left[\sum_{n=0}^{N_T-1} r_n(x_n, a_n, \boldsymbol{\mu}_n^{m_0,\boldsymbol{\pi}})\right],$$

subject to the following evolution of the agent's state:

$$\begin{cases} x_0 \sim m_0, \\ x_{n+1} \sim p_n(\cdot|x_n, a_n, \boldsymbol{\mu}_n^{m_0,\boldsymbol{\pi}}), \quad a_n \sim \boldsymbol{\pi}_n(\cdot|x_n), \quad n \geq 0. \end{cases}$$

An **optimal policy** is a $\boldsymbol{\pi}$ maximizing $J^{\text{social}}_{\text{evol}}$.

Discounted stationary case. The total social welfare is:

$$J^{\text{social}}_{\text{d-statio}}(\boldsymbol{\pi}) = J_{\text{evol}}(\boldsymbol{\pi}; \boldsymbol{\mu}_\gamma^{m_0,\boldsymbol{\pi}}) = \mathbb{E}\left[\sum_{n=0}^{N_T-1} r_n(x_n, a_n, \boldsymbol{\mu}_\gamma^{m_0,\boldsymbol{\pi}})\right],$$

subject to:

$$\begin{cases} x_0 \sim m_0, \\ x_{n+1} \sim p_n(\cdot|x_n, a_n, \boldsymbol{\mu}_\gamma^{m_0,\boldsymbol{\pi}}), \quad a_n \sim \boldsymbol{\pi}_n(\cdot|x_n), \quad n \geq 0, \end{cases}$$

where $\boldsymbol{\mu}_\gamma^{m_0,\boldsymbol{\pi}} = (1-\gamma)\sum_{n\geq 0}\gamma^n \boldsymbol{\mu}_n^{m_0,\pi} \in \Delta_{\mathcal{X}}$ is the discounted distribution introduced in Section 2.2.5.

Price of anarchy. The average reward obtained by a representative player can only be higher in an MFSO than in an MFNE, by the very definition of a social optimum. The discrepancy between the two situations is quantified by the following notion the **price of anarchy** (PoA). In the static setting, it is defined as:

$$\texttt{PoA}_{\text{static}} = \frac{\sup_\pi J^{\text{social}}_{\text{static}}(\pi)}{\inf_{\hat{\pi}\in\mathcal{NE}_{\text{static}}} J^{\text{social}}_{\text{static}}(\hat{\pi})}.$$

In the denominator, $\mathcal{NE}_{\text{static}}$ denotes the set of static MFNE. The PoA can be defined analogously in the other settings.

The term "Price of Anarchy" has been coined by Koutsoupias and Papadimitriou (1999). Since then, this notion has been widely studied in game theory and can be viewed as a way to measure the inefficiency of Nash equilibria (Roughgarden and Tardos, 2007). In the context of MFGs, it has been studied e.g. by Lacker and Ramanan (2019) in a static setting, and by Carmona, Graves, and Tan (2019) in a dynamic setting.

2.2.7 Bridging Mean-Field Control and Games: Introducing Mean-Field (Coarse) Correlated Equilibria

We have discussed so far either fully decentralized decision makings, via Nash equilibria, or fully centralized control, via Mean-Field Control. *Equilibria* denote situations of play where agents do not have any incentive to change their behavior while *Centralized* implies a central coordination of agents. Uniqueness of Mean Field Nash equilibria does not hold in general and, in such situation, the proper coordination of agents becomes crucial for selecting a relevant equilibrium. For this purpose, we now turn towards the in-between notion of centralized equilibrium.

(Coarse) Correlated Equilibria are widely studied in the game theory literature (Blum and Mansour, 2005; Morrill, D'Orazio, Sarfati, et al., 2020; Morrill, D'Orazio, Lanctot, et al., 2021; Aumann, 1987; Neumann, 1928; Neumann and Morgenstern, 1944) and have recently been introduced by Muller, Elie, et al. (2022), Muller, Rowland, et al. (2021), Degl'Innocenti (2018), and Campi and Fischer (2020) in mean field games settings. They exactly fit the intuition above: a centralized instance provides a *population recommendation*, *i.e.* a distribution over policies according to which the population will play. For each player, a policy is sampled from the *population recommendation* and sent to said player. Players only observe their policies, but they know the probabilities that the the central instance assigns to each *population recommendation*. From there, there are two deviation types that players may consider :

- Deviate given a policy - answering the question "Given that I have been tasked to play policy π, should I play something else ?". Equilibria for which there is no incentive to deviate this way are called *correlated equilibria*.
- Deviate in general - answering the question "Given what typically happens, should I play π' all the time instead of listening to recommendations ?". Equilibria for which there is no incentive to deviate this way are called *coarse-correlated equilibria*.

More formally, the sets of relevant policy deviations are $\mathcal{U}_{CE} = \{u \mid u : \Pi \to \Pi\}$ and $\mathcal{U}_{CCE} = \{u \mid u : \Pi \to \Pi, u \text{ constant}\}$. We also write, given a *population recommendation* $\nu \in \Delta(\Pi)$, $\mu(\nu)$ the mean-field state distribution when the population's policies are distributed according to ν. A correlated equilibrium is a distribution over state distributions, therefore it belongs to $\mathcal{P}(\Delta(\Pi))$.

Definition 5 (MF (Coarse) Correlated Equilibrium)**.** *A **MF correlated equilibrium** defines as a distribution $\rho \in \mathcal{P}(\Delta(\Pi))$ such that*

$$\mathbb{E}_{\nu\sim\rho,\pi\sim\nu}\left[J(u(\pi),\mu(\nu)) - J(\pi,\mu(\nu))\right] \leq 0 \qquad \forall u \in \mathcal{U}_{CE} \ .$$

*Similarly, $\rho \in \mathcal{P}(\Delta(\Pi))$ is a **MF coarse-correlated equilibrium** if*

$$\mathbb{E}_{\nu\sim\rho,\pi\sim\nu}\left[J(u(\pi),\mu(\nu)) - J(\pi,\mu(\nu))\right] \leq 0 \qquad \forall u \in \mathcal{U}_{CCE} \ .$$

As detailed in Muller, Elie, et al. (2022), similarly to Nash equilibria, plugging using a mean-field (coarse) correlated equilibrium in an N-player game yields a $\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$-approximate (coarse) correlated equilibrium. We also demonstrate that learning algorithms can converge to coarse correlated equilibrium, whenever uniqueness of Nash equilibrium is not in force in the mean field game.

Given that the set of correlated equilibria strictly includes the set of Nash equilibria, its price of anarchy will be at least as high. However, its **Price of Stability** (PoS) will, thanks to the possibility of coordinating strategies, typically be lower than Nash. The PoS is the ratio between the best objective function value of one of its equilibria and that of an optimal outcome (the PoA being the ratio between the worst equilibrium and the optimal outcome).

2.2.8 Conclusion and Extensions

In this chapter, we have given background on MDPs, RL and DRL and reveiwed the different settings that one can encounter when studying MFGs with discrete-time and discrete state and actions spaces. We conclude this chapter by mentioning a few extensions. For the sake of

readability, we use the basic settings described above in the sequel. However, several variants have been considered in the literature.

Multiple populations. Mean field theory allows us to approximate a homogeneous population of individuals by the limiting probability distribution. In multi-population MFGs, there is a finite number of sub-populations, each of them representing a homogeneous group of agents. The transition function and the reward function are the same for all the agents of one sub-population, but may be different from one group to the other. In this way, the MFG paradigm can still be used. We refer for instance to M. Huang, R. P. Malhamé, and Caines (2006), Feleqi (2013), Cirant (2015), and Bensoussan, T. Huang, and Laurière (2018) for an analytical approach and to Carmona and Delarue (2018b, Section 7.1.1) for a probabilistic formulation. In the context of reinforcement learning, multi-population MFGs have been studied e.g. by S. G. Subramanian et al. (2020) and in our work (Perolat, Perrin, et al., 2021).

Population-dependent policies. In all the previous settings, the policies are *independent* of the population distribution. This aspect is classical in the MFG and MFC literature because, if a player anticipates correctly the policy used by the rest of the population, they can anticipate the whole population behavior without uncertainty. As a consequence, the distribution needs not be an input to the agent's policy. However, this aspect might be counter-intuitive from a learning perspective, because it means that the agents react optimally only to the equilibrium population behavior but they cannot adjust their behavior if the distribution deviates from this equilibrium.

Population-dependent policies are tightly connected with population-dependent value functions, and the so-called **Master equation** in MFGs. This equation has been introduced by P.-L. Lions in the continuous setting (continuous time, state and action) (Lions, 2012). There, it is a partial differential equation (PDE) which corresponds to the limit of systems of Hamilton-Jacobi-Bellman PDEs characterizing Nash equilibria in symmetric N-player games. For more details in the continuous setting, we refer the interested reader to Bensoussan, Frehse, and S. C. P. Yam (2015) and Cardaliaguet, Delarue, et al. (2019). In the discrete time and space setting, population-dependent value functions and policies have been studied by Mishra, Vasal, and Vishwanath (2020) and in our work (Perrin, Laurière, Pérolat, Élie, et al., 2021), where a deep RL method to learn such policies is developed (Chapter 7).

Common noise. Besides idiosyncratic noise affecting the evolution of each agent independently, it is possible to consider macroscopic shocks affecting the whole population. This is referred to as **common noise** in the MFG literature. Because the whole population's evolution is stochastic, using policies functions of the player's state only is in general suboptimal. This is because even if the player knows the policy used by all the other players, it cannot predict with

certainty the evolution of the distribution. In this case, it is more efficient to use population-dependent policies. We refer to Carmona, Delarue, and Lacker (2016a) and to Cardaliaguet, Delarue, et al. (2019) for respectively a probabilistic and an analytical treatment of MFGs with common noise and we will discuss it in Chapter 4.

Chapter 3

Iterative Methods, Reinforcement Learning for Mean Field Games and Metrics

Having defined properly the different settings in Mean Field Games and Reinforcement Learning, we now summarize the main directions that researchers have taken to tackle this problem and provide an overview of computing equilibria in mean field games. In Section 3.1, we start by introducing a general framework that unify the different algorithms to solve Mean Field Games (or Mean Field Control) with Dynamic Programming. We describe mainly two classes of algorithms to find Nash equilibria in MFGs. These algorithms are based on iteratively updating the mean field and the policy, so we refer to them as iterative methods. Building on these methods and the connection between MDPs and RL, we explain in Section 3.2 how RL and DRL methods can be adapted to solve MFGs and MFC problems. Section 3.3 discusses metrics, namely the exploitability and Wasserstein distance, that can be used to assess the numerical convergence of algorithms and illustrate some of the methods on a representative MFG example.

Contents

3.1 Iterative methods 46
3.2 Reinforcement learning for Mean Field Games 56
3.3 Metrics and Numerical Experiments 63

3.1 Iterative methods

We turn our attention to the question of computing mean field Nash equilibria in the settings presented in Chapter 2. The goal is to compute a pair consisting of a policy and a mean field which form a fixed point. A simple strategy is, starting with some initial pair, to update alternatively the policy and the mean field until convergence to an equilibrium. In this section, we assume that the model is completely known. We call the algorithms presented here **iterative methods** for the sake of convenience and to distinguish them from the RL algorithms discussed later on. As we will discuss in the sequel, these methods rely on fixed point-type iterations. In contrast with the MDP setting, the underlying operator for these iterations is not always contractive, which triggers the introduction of variants to help ensuring convergence.

3.1.1 Overview of the methods

As explained above, the main idea is to alternate an update of the population distribution and an update of the representative agent's policy, which can be represented as:

$$\dots \to \mu^{\ell} \xrightarrow{\text{policy update}} \pi^{\ell+1} \xrightarrow{\text{mean field update}} \mu^{\ell+1} \to \dots \tag{3.1}$$

At a high level, we expect the scheme described in (3.1) to converge towards a fixed point $(\hat{\mu}, \hat{\pi})$ which is a Nash equilibrium.

The **mean field update** is done using the population distribution or the sequence of distributions induced by the current policy. Notice that, since the dynamics is known, it is straightforward to compute the mean field induced by a given policy. The converse is more challenging: except in some special cases, given a mean field, it is hard to find which policy generated it as many policies can generate the same mean field. Thus, computing not only the mean field but also an equilibrium policy is a crucial point.

The **policy update** can typically be done in two different ways. In the first family of methods, the policy is updated by computing a best response against the mean field. In the second family, the policy is updated based on the evaluation of the previous policy. We call these two families **best-response based** and **policy-evaluation based** respectively. In fact, this distinction stems from an analogous distinction between two families of methods to solve standard MDPs, respectively value iteration and policy iteration.

3.1.2 Solving MFGs

As explained at the beginning of this section (see Equation (3.1)), the main idea underlying the methods we present below is to alternate an update of the population distribution and an update the representative agent's policy.

Inspired by the above methods for standard MDPs, we can distinguish two families of methods for MFGs, depending on whether the policy update consists in computing an optimal policy against μ^ℓ or simply improving the current policy. We call these two family of methods **best-response based** and **policy-evaluation based**.

Best response-based methods

Since an MFG equilibrium can be defined as the fixed point of a mapping, a basic strategy consists in repeatedly applying this mapping. Under suitable conditions, this method converges and the limit is automatically a fixed point.

Stationary MFG. In the stationary MFG setting (see Section 2.2.3), we recall that a Nash equilibrium consists of a stationary distribution $\hat{\mu} \in \Delta_{\mathcal{X}}$ and a stationary policy $\hat{\pi} \in \Pi$. The policy $\hat{\pi}$ is characterized as an optimal policy for a representative player facing the population distribution $\hat{\mu}$. This problem can be phrased in the framework of MDPs.

If the stationary mean field is μ, then the MDP that a representative player needs to solve is:

$$(\mathcal{X}, \mathcal{A}, p(\cdot, \cdot, \mu), r(\cdot, \cdot, \mu), \gamma), \tag{3.2}$$

where the transition and the reward functions are given by:

$$p(\cdot, \cdot, \mu) : \mathcal{X} \times \mathcal{A} \to \mathcal{P}(\mathcal{X}), \qquad r(\cdot, \cdot, \mu) : \mathcal{X} \times \mathcal{A} \to \mathbb{R}.$$

The optimal policy for this MDP is the best response against μ, which is denoted by $\mathtt{BR}_{\mathrm{statio},\gamma}(\mu)$. It can be obtained for example by applying the policy iteration or the value iteration algorithms as recalled in Section 2.1.2. Conversely, given a policy π, the induced mean field is the stationary distribution (assuming it is unique for simplicity) induced by π and denoted by $\mathbf{M}_{\mathrm{statio}}(\pi)$, see Equation (2.20).

This is summarized as follows: μ^0 is given, and for $\ell = 0, \dots, L-1$,

$$\begin{cases} \pi^{\ell+1} = \mathtt{BR}_{\mathrm{statio},\gamma}(\mu^\ell) \\ \mu^{\ell+1} = \mathbf{M}_{\mathrm{statio}}(\pi^{\ell+1}). \end{cases} \tag{3.3}$$

At the end, we use (π^L, μ^L) as a proxy for the MFG equilibrium. Under suitable conditions, it is close to $(\hat{\pi}, \hat{\mu})$ when L is large enough. We come back to the question of convergence in Section 3.1.3 below.

In the above iterative method, we update the mean field term by using the operator $\mathbf{M}_{\text{statio}}$, which can be approximated by applying a large number of times the transition matrix defined in (2.29). In other words, in practice, $\mu^{\ell+1}$ is often defined by first computing:

$$\boldsymbol{\mu}_{n+1} = P^{\top}_{n,\boldsymbol{\mu}_n,\pi^{\ell+1}} \boldsymbol{\mu}_n, \quad n = 0, \dots, M-1,$$

with $\boldsymbol{\mu}_0$ a given initial distribution. For instance we can take $\boldsymbol{\mu}_0 = \mu^{\ell}$ from the previous iteration. As $M \to +\infty$, we expect $\boldsymbol{\mu}_M \to \mathbf{M}_{\text{statio}}(\pi^{\ell+1})$, so we use $\boldsymbol{\mu}_M$ as an approximation of $\mu^{\ell+1}$.

In fact, taking M relatively small can have some advantages. In some sense, it amounts to slowing down the updates of the mean-field term. This can bring more stability to the iterative method, particularly when the policy $\pi^{\ell+1}$ is computed approximately (*e.g.*, in a reinforcement learning setup). We will come back to this idea of damping the update of the mean field in Section 3.1.3 below, but let us immediately emphasize that a variant of the above iterative method consists in doing only one application of the transition matrix at each iteration ℓ. This can be summarized as: μ^0 is given, and for $\ell = 0, \dots, L-1$,

$$\begin{cases} \pi^{\ell+1} = \texttt{BR}_{\text{statio},\gamma}(\mu^{\ell}) \\ \mu^{\ell+1} = P^{\top}_{n,\mu^{\ell},\pi^{\ell+1}} \mu^{\ell}. \end{cases}$$

This method has been used for instance by Guo, A. Hu, et al. (2019) and Anahtarcı, Karıksız, and Saldi (2020a). It is also in line with the idea of using a two-timescale approach for mean field Nash equilibria (J. Subramanian and Mahajan, 2019; Mguni, J. Jennings, and Munoz de Cote, 2018; Angiuli, Fouque, and Laurière, 2022; Xie et al., 2021). A similar method has been analyzed in (Anahtarcı, Karıksız, and Saldi, 2019b; Anahtarcı, Karıksız, and Saldi, 2020b) for average cost MFGs (in the latter work, it is referred to as **value iteration** algorithm for MFGs).

Finite-horizon MFG. In the evolutive MFG setting with a finite horizon N_T (see Section 2.2.4), an equilibrium is a sequence of distributions $\hat{\boldsymbol{\mu}} = (\hat{\boldsymbol{\mu}}_n)_{n=0,\dots,N_T}$ and a sequence of policies $\hat{\boldsymbol{\pi}} = (\hat{\boldsymbol{\pi}}_n)_{n=0,\dots,N_T}$, indexed by the time steps in the game. Given a sequence of distributions $\hat{\boldsymbol{\mu}} = (\hat{\boldsymbol{\mu}}_n)_{n=0,\dots,N_T}$, a representative player needs to solve the following finite-horizon MDP (see Section 2.1.1):

$$(\mathcal{X}, \mathcal{A}, p_{\boldsymbol{\mu}}, r_{\boldsymbol{\mu}}, N_T),$$

where:

$$p_{\boldsymbol{\mu}} : \{0, \dots, N_T - 1\} \times \mathcal{X} \times \mathcal{A} \to \mathcal{P}(\mathcal{X}), \qquad p_{\boldsymbol{\mu}} : (n, x, a) \mapsto p_n(\cdot | x, a, \boldsymbol{\mu}_n)$$

and

$$r_{\boldsymbol{\mu}} : \{0, \dots, N_T\} \times \mathcal{X} \times \mathcal{A} \to \mathbb{R}, \qquad r_{\boldsymbol{\mu}} : (n, x, a) \mapsto r_n(x, a, \boldsymbol{\mu}_n).$$

The optimal policy for this MDP is the best response against $\boldsymbol{\mu}$, which is denoted by $\texttt{BR}_{\text{evol},m_0,N_T}(\boldsymbol{\mu})$.

It can be obtained as a greedy policy for the optimal value function $\mathbf{Q}^{*,\boldsymbol{\mu}}$, which can be computed by backward induction as described in Section 2.1.2. Alternatively, the optimal policy can be computed by policy iteration as described in Section 2.1.2. Conversely, given a policy $\boldsymbol{\pi} = (\boldsymbol{\pi}_n)_{n=0,\dots,N_T}$, the induced mean-field is the sequence of distributions generated by starting from m_0 (remember that m_0 is fixed in the definition of the MFG, see Section 2.2.4) and using $\boldsymbol{\pi}_n$ at time step n, $n = 0, \dots, N_T - 1$. The resulting mean-field sequence is denoted by $\mathbf{M}_{\text{evol},m_0,N_T}(\boldsymbol{\pi})$, see Equation (2.28).

This is summarized below, using the notation introduced in Section 2.2.4: $\boldsymbol{\mu}^0$ is given, and for $\ell = 0, \dots, L-1$,

$$\begin{cases} \boldsymbol{\pi}^{\ell+1} = \texttt{BR}_{\text{evol},m_0,N_T}(\boldsymbol{\mu}^{\ell}) \\ \boldsymbol{\mu}^{\ell+1} = \mathbf{M}_{\text{evol},m_0,N_T}(\boldsymbol{\pi}^{\ell+1}). \end{cases} \tag{3.4}$$

At the end, we use $(\boldsymbol{\pi}^L, \boldsymbol{\mu}^L)$ as a proxy for the MFG equilibrium. Under suitable conditions on the MFG, this pair is close to $(\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\mu}})$ when L is large enough.

For the sake of completeness and future reference, we provide here the Bellman equations satisfied by $\mathbf{Q}^{*,\boldsymbol{\mu}}$ and $\mathbf{Q}^{\boldsymbol{\pi},\boldsymbol{\mu}}$, which can be derived by dynamic programming:

$$\begin{cases} \mathbf{Q}^{*,\boldsymbol{\mu}}_{N_T}(x,a) = r_{N_T}(x, a, \boldsymbol{\mu}_{N_T}) \\ \mathbf{Q}^{*,\boldsymbol{\mu}}_{n}(x,a) = r_n(x, a, \boldsymbol{\mu}_n) + \mathbb{E}\left[\max_{a \in \mathcal{A}} \mathbf{Q}^{*,\boldsymbol{\mu}}_{n+1}(x_{n+1}, a) \middle| x_{n+1} \sim p_n(\cdot | x_n, a_n, \boldsymbol{\mu}_n) \right], \\ \qquad\qquad n = N_T - 1, \dots, 0, \end{cases} \tag{3.5}$$

and

$$\begin{cases} \mathbf{Q}^{\boldsymbol{\pi},\boldsymbol{\mu}}_{N_T}(x,a) = r_{N_T}(x, a, \boldsymbol{\mu}_{N_T}) \\ \mathbf{Q}^{\boldsymbol{\pi},\boldsymbol{\mu}}_{n}(x,a) = r_n(x, a, \boldsymbol{\mu}_n) + \mathbb{E}\left[\mathbf{Q}^{\boldsymbol{\pi},\boldsymbol{\mu}}_{n+1}(x_{n+1}, a_{n+1}) \middle| x_{n+1} \sim p_n(\cdot | x_n, a_n, \boldsymbol{\mu}_n), a_{n+1} \sim \boldsymbol{\pi}_{n+1}(\cdot | x_{n+1}) \right], \\ \qquad\qquad n = N_T - 1, \dots, 0. \end{cases} \tag{3.6}$$

In Perrin, Perolat, et al. (2020) and Perrin, Laurière, Pérolat, Élie, et al. (2021) (Chapter 4 and Chapter 7), we use backward induction to compute the optimal value function for finite-horizon MFG (embedded in fictitious play iterations, see Section 3.1.3), which served as a baseline to assess the performance of RL-based methods (see next section). Cui and Koeppl (2021) solved finite-horizon MFG using fixed point iterations combined with RL methods and entropy regularization (we come back to this point in Section 3.1.3 below). Mishra, Vasal, and

Vishwanath, 2020 also solved MFGs based on a best-response computation, but by computing a best response backward in time in the spirit of dynamic programming, which requires solving for all possible distributions since the equilibrium mean field sequence is not known a priori. The aforementioned two-timescale approach originally studied in the stationary setting has been extended by Angiuli, Fouque, and Laurière (2021) to solve finite-horizon MFGs.

Remark 11. *In the stationary regime, we can view iterations as time steps. Taking a large number of iterations amounts to looking at the long time behavior. However, in the finite-horizon MFG setting, the index of iterations does not coincide with the index of time in the game. At each iteration ℓ, the policy and the distributions are updated for all time steps, $n = 0, \dots, N_T$.*

Policy evaluation-based methods

Instead of computing a full-fledged best response for the policy update at each iteration of (3.1), we can simply do one step of policy improvement. Intuitively, evaluating the current policy should be computationally faster than computing an optimal policy (except when the state space is small or when we have an explicit formula for the optimizer of the value function). To improve the policy, we can first evaluate the current policy given the latest mean field, and then take a greedy policy.

Stationary MFG. In a stationary MFG, we can proceed as follows: we first compute the state-action value function associated to the current policy against the current population distribution (policy evaluation step). We then define the new policy as a greedy policy for the newly computed value function (policy improvement step). Last, we deduce the stationary population distribution induced by this policy (mean field update step). Concretely, the method is: μ^0 and π^0 are given, and for $\ell = 0, \dots, L-1$,

$$\begin{cases} Q^{\ell+1} = Q^{\pi^\ell, \mu^\ell} \\ \pi^{\ell+1} \in GQ^{\ell+1} \\ \mu^{\ell+1} = \mathbf{M}_{\text{statio}}(\pi^{\ell+1}). \end{cases} \tag{3.7}$$

This method is referred to as the **Policy Iteration** (PI) algorithm for MFGs and was introduced by Cacace, Simone, Camilli, Fabio, and Goffi, Alessandro, 2021 for continuous time, continuous space MFGs. It is not to be confused with the method that consists in using standard policy iteration to compute a best response against a given distribution (*i.e.*, replacing $\texttt{BR}_{\text{statio},\gamma}(\mu^\ell)$ in (3.3) by the result of a policy iteration method).

In practice, the evaluation step can be done by applying a finite number of times the Bellman operator B^{π^ℓ, μ^ℓ} as defined in Eq. (2.22). Thanks to the contraction property of this operator,

we obtain an approximation of Q^{π^ℓ,μ^ℓ}. Furthermore, as discussed above, $\mathbf{M}_{\text{statio}}(\pi^{\ell+1})$ can be approximated by applying a large but finite number of times the transition matrix.

Finite-horizon MFG. In a finite-horizon MFG, the same strategy can be applied, except that we need to take into account the evolutive aspect of the game. Each of the step is done for all the time steps. The method can be summarized as follows: μ^0 and π^0 are given, and for $\ell = 0, \dots, L-1$,

$$\begin{cases} \mathbf{Q}^{\ell+1} = \mathbf{Q}^{\boldsymbol{\pi}^\ell,\boldsymbol{\mu}^\ell} \\ \boldsymbol{\pi}^{\ell+1} \in \mathbf{G}\mathbf{Q}^{\ell+1} \\ \boldsymbol{\mu}^{\ell+1} = \mathbf{M}_{\text{evol},m_0,N_T}(\boldsymbol{\pi}^{\ell+1}). \end{cases} \tag{3.8}$$

In this setting, $\mathbf{Q}^{\boldsymbol{\pi}^\ell,\boldsymbol{\mu}^\ell}$ can be computed by backward induction, thanks to the dynamic programming equation (3.6). Similarly, $\mathbf{M}_{\text{evol},m_0,N_T}(\boldsymbol{\pi}^{\ell+1})$ can be computed by following N_T transitions, see (2.29).

Cacace, Simone, Camilli, Fabio, and Goffi, Alessandro (2021), mentioned above, also studied policy iteration in the finite-horizon setting and proved convergence under suitable conditions. Still in the finite-horizon setting, the convergence results were extended to other settings by Camilli and Tang (2022) and Laurière, Song, and Tang (2021). Using a purely greedy policy often leads to instabilities, particularly in the finite state case; see *e.g.*, (Cui and Koeppl, 2021) and the next section for more details. For this reason, variants with regularized policies have been introduced, such as the online mirror descent, as we explain below.

3.1.3 Convergence and variants

Convergence of fixed point iterations. Intuitively, the scheme described in (3.1) indeed converges towards a fixed point if the mapping $(\mu^\ell, \pi^\ell) \mapsto (\mu^{\ell+1}, \pi^{\ell+1})$ is a strict contraction on a suitably defined space. In a stationary setting, this property can be ensured by assuming that the reward function and the transition function are smooth enough. Typically, this amounts to assuming that they are Lipschitz continuous with small enough Lipschitz constants. In a finite horizon setting, this condition can sometimes be replaced by an assumption on the smallness of the time horizon. One advantage of having a contraction is that it provides a constructive way to get the equilibrium through Banach-Picard iterations. This technique is commonly used in the literature on MFGs, both to show existence and to derive algorithms. See *e.g.*, M. Huang, R. P. Malhamé, and Caines (2006) in the context of existence and uniqueness of the equilibrium or Carlini and Silva (2014) in the context of numerical methods. It is in general difficult to formulate sufficient conditions on the model (*i.e.*, the reward and the transition) to ensure the strict contraction property because the mapping involves the policy update step, for which there is in general no explicit formula. In the linear-quadratic case, several sufficient conditions

are formulated by R. Hu (2021, Proposition 3.1). Furthermore, regularizing the policy can help to alleviate some of the conditions ensuring the contraction property, see *e.g.*, Guo, A. Hu, et al. (2019) and Anahtarcı, Karıksız, and Saldi (2020a). Using regularization of the policy, Guo, A. Hu, et al. (2020) have proved convergence and analyzed the complexity of value-based and policy-based algorithms.

However, conditions guaranteeing the strict contraction property are generally very restrictive and fails to hold for many games. For example, Cui and Koeppl (2021, Theorem 2) show that non-contractivity is the rule rather than the exception. Without contractivity, Banach-Picard iterations typically lead to oscillations, see *e.g.*, Chassagneux, Crisan, and Delarue (2019, Figure 3) in the context of a method based on the probabilistic interpretation of MFGs, or Laurière (2021, Figure 4) in the context of linear-quadratic MFGs.

To address this issue, several variants of the pure Banach-Picard fixed point iterations have been proposed in the literature, relying on a few key principles.

Before describing these principles, let us mention that besides the aforementioned class of assumptions to ensure contractivity which are somehow *quantitative assumptions* since they boil down to smallness of some coefficients, an alternative class of hypotheses are in some sense *qualitative assumptions* which pertain to the structure of the game. For example, potential structure and MFG satisfying Lasry-Lions monotonicity (Lasry and Lions, 2007) can be used to prove convergence of best-response based and policy evaluation based algorithms, see respectively (Cardaliaguet and Hadikhanloo, 2017; Perrin, Perolat, et al., 2020; Geist, Pérolat, et al., 2021) and (Hadikhanloo, 2017; Perolat, Perrin, et al., 2021). In particular, the Lasry-Lions monotonicity condition, which basically refers to the fact that players tend to avoid crowded regions, has been interpreted in terms of exploitability (see Section 3.3.1). These convergence results do not rely on smallness conditions on the coefficients. However, even for MFGs with such nice structure, pure fixed point iterations rarely converge and smoothing the iterations if typically required to ensure convergence.

Remark 12. *Regularization has also been the subject of extensive studies in reinforcement learning, most of the time for exploration or robustness purposes. See Geist, Scherrer, and Pietquin (2019) or Neu, Jonsson, and Gómez (2017) for a general framework of entropy-regularized MDPs in the average reward setting.*

Smoothing the mean field updates. First, a simple modification consists in using damping to slow down the updates of the mean field term. Even if the mapping $\mu^\ell \to \pi^\ell \to \mu^{\ell+1}$ is not contractive, we can hope that the following mapping is contractive, at least for small enough values of $\alpha \in (0,1)$:

$$\bar{\mu}^\ell \to \pi^\ell \to \bar{\mu}^{\ell+1} := (1-\alpha)\bar{\mu}^\ell + \alpha\mu^{\ell+1}. \tag{3.9}$$

Here $\mu^{\ell+1}$ is the mean field associated to policy π^ℓ while $\bar{\mu}^\ell$ is an average over past mean field terms. See Laurière (2021, Section 2) for an example in which damping with a constant coefficient helps ensuring numerical convergence. We also refer to Tembine, Tempone, and Vilanova (2012) for more algorithms developed along these lines and presented in the context of static games.

We can also let α change with the iteration index, *i.e.* take a different α^ℓ for $\ell = 1, 2, \dots$. One of the most popular versions consist in taking $\alpha^\ell = 1/(\ell+1)$ and is called **Fictitious Play**. It was first introduced in two-player games by G. W. Brown (1951) and Robinson (1951) and extended to MFG by Cardaliaguet and Hadikhanloo (2017), Hadikhanloo (2018), and Hadikhanloo and Silva (2019). In the context of stationary MFGs for example, (3.3) is replaced by: μ^0 is given, and for $\ell = 0, \dots, L-1$,

$$\begin{cases} \pi^{\ell+1} = \mathtt{BR}_{\mathrm{statio},\gamma}(\bar{\mu}^\ell) \\ \mu^{\ell+1} = \mathbf{M}_{\mathrm{statio}}(\pi^{\ell+1}) \\ \bar{\mu}^{\ell+1} = \dfrac{\ell}{\ell+1}\bar{\mu}^\ell + \dfrac{1}{\ell+1}\mu^{\ell+1}. \end{cases} \tag{3.10}$$

Under suitable assumptions, $\bar{\mu}^\ell$ converges to a stationary MFG equilibrium distribution. It is important to note that in general the last iterate π^ℓ of the policy does *not* generate $\bar{\mu}^\ell$ and hence does not converge towards an equilibrium policy. If one cares about the equilibrium policy, it is thus required to learn a policy generating $\bar{\mu}^\ell$. In some cases, convergence of the last iterate towards an equilibrium holds, see *e.g.*, Cardaliaguet and Hadikhanloo (2017).

For finite-state MFGs, we have obtained a rate of convergence (Perrin, Perolat, et al., 2020) for continuous-time FP (see Chapter 4) under monotonicity condition and in Geist, Pérolat, et al. (2021) and Bonnans, Lavigne, and Pfeiffer (2021) respectively for discrete-time FP in some potential MFGs. In linear-quadratic MFGs, a rate of convergence has been obtained by Delarue and Vasileiadis (2021), who also studied the impact of common noise.

Slowing down the updates of the mean field term is also in line with the idea of using a two-timescale approach for mean field Nash equilibria (J. Subramanian and Mahajan, 2019; Mguni, J. Jennings, and Munoz de Cote, 2018; Angiuli, Fouque, and Laurière, 2022; Xie et al., 2021). Here, the distribution and the policy (or the value function) are both updated at every iteration but the distribution is updated at a slower rate than the policy. Intuitively, this implies that the representative agent has enough time to compute an approximate best response before the distribution changes too much.

Whenever Nash equilibria are not unique, one can use a slight alteration of the Fictitious Play algorithm, **Joint Fictitious Play**, that we have defined in Muller, Elie, et al. (2022) as a continuous time algorithm following

$$\pi_\tau^{BR} = \arg\max_{\pi' \in \Pi} \int_{s=0}^{\tau} \langle \mu^{\pi'}, r^{\pi'}(\cdot, \mu^{\pi_s}) \rangle ds \quad \text{and} \quad \mu^{\pi_\tau}(x) = \frac{1}{\tau} \int_0^{\tau} \mu^{\pi_s^{BR}}(x) ds,$$

which is proven to converge to a Mean-Field Coarse Correlated Equilibrium. More precisely, the distribution which uniformly samples π_t^{BR} with $t \in [0, T]$, and recommends this policy to the whole population, converges towards a Mean-Field Coarse Correlated Equilibrium at a rate of $\mathcal{O}\left(\frac{1}{T}\right)$.

Smoothing the policy updates. Another way to bring more stability to the iterative method is to regularize the policy update. For instance, the greedy policy operator defined in (2.10) is very sensitive to perturbations of the state-action value function. Small changes in this value function might lead to significant changes in the induced greedy policy. To mitigate this problem, it is common to replace the $\arg\max$ by a $\mathrm{softmax}$, meaning that we can define:

$$\pi^{(k+1)}(\cdot|x) = \mathrm{softmax}_\tau Q(x, \cdot), \tag{3.11}$$

where $\tau > 0$ is an inverse temperature parameter and $\mathrm{softmax} : \mathbb{R}^{|\mathcal{A}|} \to \Delta_{\mathcal{A}}$ is defined by: for $q = (q_1, \ldots, q_{|\mathcal{A}|})$,

$$\mathrm{softmax}_\tau(q) = \left(\frac{e^{\tau q_i}}{\sum_{j=1}^{|\mathcal{A}|} e^{\tau q_j}} \right)_{i=1,\ldots,|\mathcal{A}|}.$$

It transforms a vector of Q-values into a discrete probability distribution on the action space in which the actions with larger value have a higher probability. Using a softmax instead of the argmax generally yields smoother and more stable learning curves, see *e.g.*, Guo, A. Hu, et al. (2019) and Anahtarcı, Karıksız, and Saldi (2020a).

In fact, finite-state finite-action MFGs typically admit only randomized policy equilibria and no pure equilibria. This is also the reason why we generally allow for randomized policies in finite-player games (Nash, 1950; Nash, 1951). Hence, iterative methods with pure greedy policies cannot be expected to converge to Nash equilibria in general, and using mixed policies is unavoidable.

Regularized policies can be obtained *e.g.*, by directly changing the way the policy is obtained from the value function (Guo, A. Hu, et al., 2019; Perolat, Perrin, et al., 2021) or by adding a penalty in the reward function, which changes the value function and hence the policy, see Anahtarcı, Karıksız, and Saldi (2019a), Guo, Xu, and Zariphopoulou (2020), Cui and Koeppl (2021), Firoozi and Jaimungal (2022), and Lauriere et al. (2022). However, it should be noted that regularizing the policies also has drawbacks: if π^ℓ is forced to be smooth, this constraint might prevent the iterative method from converging towards the Nash equilibrium since π^ℓ can only be smooth version of the equilibrium policy.

One way to circumvent this limitation and to allow the regularized policy to concentrate on optimal actions is to let the underlying Q-function take larger and larger values. This can be achieved by considering a cumulative Q-function, which leads to the **Online Mirror Descent** (**OMD**) algorithm (Hadikhanloo, 2017; Perolat, Perrin, et al., 2021):

$$\begin{cases} Q^{\ell+1} = Q^{\pi^\ell, \mu^\ell} \\ \tilde{Q}^{\ell+1} = \tilde{Q}^\ell + \alpha Q^{\ell+1} \\ \pi^{\ell+1} = \mathrm{softmax}_\tau \tilde{Q}^{\ell+1} \\ \mu^{\ell+1} = \mathbf{M}_{\mathrm{statio}}(\pi^{\ell+1}). \end{cases} \tag{3.12}$$

where $\alpha > 0$ is a parameter which determines the cumulative factor. This algorithm can be viewed as a modification of the policy evaluation method described in (3.7) with a cumulative Q-function and a regularized greedy policy. Instead of the softmax, we can more generally take the gradient of the convex conjugate of a strongly convex regularizer, see Perolat, Perrin, et al. (2021) (Chapter 5) for more details.

In situations where the MFG does not admit a unique Nash equilibrium, Muller, Elie, et al. (2022) verifies that recommending an OMD policy at uniformly-sampled times to the whole population yields a Mean-Field Coarse-Correlated Equilibrium as presented in Definition 5.

3.1.4 Iterative methods for Mean Field Control

We recall that the MFC problem introduced in Section 2.2.6 corresponds to the maximization of a social reward. In the evolutive case, it can be reformulated as an MDP by considering the population distribution as the state. Indeed, we can rewrite:

$$J_{\mathrm{evol}}^{\mathrm{social}}(\boldsymbol{\pi}) = \sum_{n=0}^{N_T-1} \underbrace{\sum_{x\in\mathcal{X}} \sum_{a\in\mathcal{A}} r_n(x, a, \boldsymbol{\mu}_n^{m_0,\boldsymbol{\pi}}) \boldsymbol{\mu}_n^{m_0,\boldsymbol{\pi}}(x) \boldsymbol{\pi}_n(a|x)}_{=:\bar{r}_n(\boldsymbol{\mu}_n^{m_0,\boldsymbol{\pi}}, \boldsymbol{\pi}_n)},$$

subject to the following evolution of the mean field state:

$$\begin{cases} \boldsymbol{\mu}_0^{m_0,\boldsymbol{\pi}} = m_0, \\ \boldsymbol{\mu}_{n+1}^{m_0,\boldsymbol{\pi}} = P^\top_{n,\boldsymbol{\mu}_n^{m_0,\boldsymbol{\pi}},\boldsymbol{\pi}_n} \boldsymbol{\mu}_n^{m_0,\boldsymbol{\pi}}, \quad n \geq 0. \end{cases}$$

This is an MDP with:

- state space $\Delta_{\mathcal{X}}$,
- action space Π,
- probability transition function: $\bar{p}_n(\cdot|\mu, \pi) = (P_n^{\mu,\pi})^\top \mu$,

- reward function: $\bar{r}_n(\mu, \pi) = \sum_{x \in \mathcal{X}} \sum_{a \in \mathcal{A}} r_n(x, a, \mu) \mu(x) \pi(a|x)$.

We will refer to this MDP as the **mean field MDP** (MFMDP). An action, taken by the central planner or collectively by the population, is an element of $\overline{\mathcal{A}} = (\Delta_{\mathcal{A}})^{\mathcal{X}}$. A one-step policy at the level of the population is a function from $\overline{\mathcal{X}}$ to $\Delta_{\overline{\mathcal{A}}}$. Note that, even if $\mathcal{X}$ and $\mathcal{A}$ are finite, the state space $\overline{\mathcal{X}}$ and the action space $\overline{\mathcal{A}}$ of the MFMDP are continuous and hence rigorously defining and analyzing this MDP requires a careful formulation. We refer to the work of Gast, Gaujal, and Le Boudec (2012), Gu et al. (2019), Gu et al. (2021a), Motte and Pham (2019), Carmona, Laurière, and Tan (2019b), and Bäuerle (2021) for more details on MFMDP.

Let us stress that this MFMDP is not to be confused with the MDP arising in MFGs, which is the MDP for a single representative player when the mean field term is given. In the latter case, the state is simply the agent's state and not the population state.

With this reformulation, the evolutive MFC problem can be analyzed and solved using methods developed from MDP. However, notice that the policies are, in general, functions of both the representative agent's state and the mean field state. The main challenges thus pertain to the numerical implementation of these methods, since we need to represent efficiently the distribution and the policy. We will come back to this question in Section 3.2.1.

Remark 13. *Note that, in the present model, the evolution of $\boldsymbol{\mu}^{m_0, \boldsymbol{\pi}}$ is in fact completely deterministic once m_0 and $\boldsymbol{\pi}$ are given. Noise affecting the distribution and making its evolution stochastic is referred to as common noise. We refer to Motte and Pham (2019) and Carmona, Laurière, and Tan (2019b) for more details. Furthermore, since an action is an element of Π, a policy is a function $\bar{\pi} : \Delta_{\mathcal{X}} \ni \mu \mapsto \bar{\pi}(\mu) \in \Delta_{\Pi}$. Sampling from $\bar{\pi}(\mu)$ amounts to sample an element π to be used by the whole population. Carmona, Laurière, and Tan (2019b) referred to this as **common randomness**.*

3.2 Reinforcement learning for Mean Field Games

As for the single-agent case, the iterative methods developed for MFGs in the previous section are described with exact updates, meaning that we assume that the model is fully known and that there are no numerical approximations in the computation of the rewards or the transitions. In this context, the only approximations that we have to cope with are in situations where an infinite number of iterations would be needed but we can only afford a finite number of iterations (*e.g.*, to compute a stationary distribution or a stationary value function).

In the context of MFGs, in this section we will build on the iterative methods presented in Section 3.1. These methods boil down to alternating mean-field updates and policy updates, and the policy updates stem from standard MDP techniques. As a consequence, standard RL techniques can readily be injected at this level to learn policies or value functions, in a model-free fashion.

Environments with mean field interactions

To study RL methods for MFGs, the first question is the definition of the environment. In MFGs, the transitions and the rewards depend on the population distribution, which should thus be part of the environment. Since we are going to focus on how a representative agent learns an equilibrium policy, we also include the state of this representative agent in the state of the environment.

The next question is: What is the information available to the agent who is learning? In other words, we should decide what the output of one query to the environment is. Remember that for a given population distribution (or sequence of distributions in the evolutive setting), the agent tries to solve an MDP parameterized by this distribution but the policy is not a function of the population distribution (see *e.g.*, Section 2.1.3 in the stationary MFG case). From this point of view, to learn an optimal policy, the agent does not need to observe the rest of the population: it is sufficient to observe the result of the reward function and some samples of transitions.

Remark 14. *The fact that the representative agent does not need to observe the rest of the population in order to learn an optimal policy is specific to the mean-field setting with an infinite number of players. In a finite-player game, the equilibrium policy of each player generally depends on the configuration of the rest of the population, even when the interactions are symmetric or when a mean-field approximation is used, see e.g., Y. Yang, Luo, et al. (2018), Y. Yang and J. Wang (2020), and K. Zhang, Z. Yang, and Başar (2021) in the context of MARL. This is because in a mean-field setting, the law of large numbers allows to get rid of the randomness of the evolution of the crowd, provided that the crowd is always starting from the same initial distribution and all the players are anonymous, identical and use the same policy (which is exactly the mean-field setting). With a finite number of players, even if the players are always starting from the same set of states, having a stochastic environment (e.g., a stochastic transition kernel) compels the agents to keep track of the current states of other agents. Thus, focusing on population-independent policies (which are sometimes referred to as decentralized policies) is one of the main advantage of the MFG approach compared with a finite-player game framework. We stress that this is possible because the agents always start from the same initial distribution. Relaxing this constraint requires to consider population-dependant policies (Perrin, Laurière, Pérolat, Élie, et al., 2021) see Chapter 7.*

We summarize the environment in Figure 3.1, which is very similar to the classical RL setup described in Figure 2.1 except that the distribution is involved in the environment.

Remark 15 (On the implementation of the environment)**.** *In some cases, the environment is truly based on the mean field state corresponding to the regime with an infinite number of agents. This can be the case for example when the state space and the action space are finite and small, and the evolution of the distribution or the stationary distribution can be computed exactly using the transition matrix. However,*

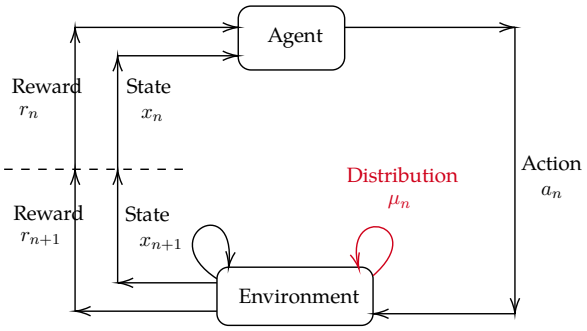


Figure 3.1 – Environment for MFGs: Here, the current state of the MDP is the representative agent's state x_n and the population distribution μ_n, the action taken by the agent is a_n, the new state is $x_{n+1} \sim p(\cdot|x_n, a_n, \mu_n)$ and the reward is $r_n = r(x_n, a_n, \mu_n)$. The new state x_{n+1} is observed by the agent and is also used for the next step of the environment's evolution along with μ_n.

in general, the environment relies on some approximate version of the population distribution (e.g., using an empirical distribution with a finite number of agents, or using some function approximations). Compared with the ideal environment with the true mean-field distribution, this adds an extra layer of approximation which can be neglected if one is purely interested in the performance of RL algorithms. We come back to this point in Section 3.2.1 below, in the context of MFC.

Reinforcement learning for MFGs

We focus on two settings: stationary and finite horizon. The ideas developed in these cases can be adapted to tackle static and infinite horizon MFGs.

Stationary MFG setting. In a nutshell, in the stationary MFG setting, when using one of the iterative methods presented in Section 3.1, at each iteration the representative agent faces a stationary MDP parameterized with a fixed distribution μ. We can thus use off-the-shelf RL methods.

To be more specific, we assume that a representative agent is encoded by a stationary policy $\pi \in \Pi$, either explicitly or implicitly (through a Q-function) and can interact with the environment in the following way: at each step, the agent observes its current state x, chooses action $a \sim \pi(\cdot|x)$, and the environment returns a realization of $x' \sim p(\cdot|x, a, \mu)$ and $r(x, a, \mu)$. Note that the agent does not need to observe directly the mean field flow μ, which is stored in the environment and simply enters as a parameter of the transition and reward functions p and r. In this stationary setting, in Figure 3.1, μ_n is constant equal to μ for all n. Based on such samples, the representative agent can implement any of the RL methods (*e.g.*, the ones discussed in Section 2.1.3) for standard MDPs.

Notice that, at each new step of the iterative method described in (3.1), after the mean field update the environment needs to be updated with the new population distribution. That is to say, when the mean field state is $\mu^{(\ell)}$, the agent uses the environment of Figure 3.1 with $\mu_n = \mu^{(\ell)}$ for every n to learn a best-response or evaluate a policy. Here n is the index of the RL method iteration. Then, the new mean field $\mu^{(\ell+1)}$ is computed. For the next iteration, the MDP is updated so the agent interacts with the environment of Figure 3.1 but now with $\mu_n = \mu^{(\ell)}$ for every n.

For stationary MFG equilibria, Guo, A. Hu, et al. (2019) introduced a best-response based iterative method with tabular Q-learning to compute the best response at each iteration. Moreover, they proved convergence using bounds on classical Q-learning, combined with a strict contraction argument. Guo, A. Hu, et al. (2020) generalized this idea notably using a policy gradient approach in lieu of Q-learning. A similar algorithm but combined with fitted Q-learning instead of tabular Q-learning was analyzed and proved to converge by Anahtarcı, Karıksız, and Saldi (2019a). Furthermore, Anahtarcı, Karıksız, and Saldi (2020a) and Anahtarci, Kariksiz, and Saldi (2021) showed that some of the conditions to obtain convergence in the tabular Q-learning case can be relaxed if the MDP is regularized.

J. Subramanian and Mahajan (2019) and Angiuli, Fouque, and Laurière (2022) used a two-timescale approach combined with model-free RL to compute stationary equilibria. The convergence has been proved under suitable conditions on the underlying ODEs by using stochastic approximation techniques (Borkar, 2009).

In the γ-discounted setting, Elie, Perolat, et al. (2020) analysed the propagation of error in fictitious play (*i.e.*, how errors made in the computation of the best response propagate through the algorithm) and implemented this scheme with an actor-critic DRL method (namely, DDPG (Lillicrap et al., 2016)) to compute the best response. Fictitious play and DRL combined with neural network approximation of the population distribution allowed us to solve a flocking model with continuous and high-dimensional space, see Perrin, Laurière, Pérolat, Geist, et al. (2021) or Chapter 6. Still in the γ-discounted setting, in Perrin, Perolat, et al. (2020) we provide a convergence rate for continuous-time fictitious play under monotonicity assumption (see Appendix of Chapter 4), while in Geist, Pérolat, et al. (2021) we establish a rate of convergence for discrete-time fictitious play in MFGs with a potential structure.

Finite horizon MFG setting. To learn finite horizon MFG solutions in a model-free way, we assume that a representative agent is encoded by a time-dependent policy $\boldsymbol{\pi} = (\boldsymbol{\pi}_n)_{n=0,\dots,N_T-1}$ and can interact with the environment to realize episodes. Each episode is done in the following way: the environment picks $x_0 \sim m_0$ and reveals it to the agent; then for $n = 0, \dots, N_T$, the agent observes x_n, chooses action $a_n \sim \boldsymbol{\pi}_n(\cdot|x_n)$, and the environment returns a realization of $x_{n+1} \sim p_n(\cdot|x_n, a_n, \boldsymbol{\mu}_n)$ as well as the value of $r_n(x_n, a_n, \boldsymbol{\mu}_n)$. Note that the agent does not need

to observe directly the mean field flow $(\boldsymbol{\mu}_n)_{n=0,\dots,N_T}$, which simply enters as a parameter of the transition and reward functions p_n and r_n.

Based on such episodes, the agent can for example estimate a policy $\boldsymbol{\pi}$ by approximately computing the state-action value function $\mathbf{Q}^{\boldsymbol{\pi},\boldsymbol{\mu}}$, or compute a best response by first approximating the optimal value function $\mathbf{Q}^{*,\boldsymbol{\mu}}$. The value functions can be estimated by backward induction as in (3.6) and (3.5), replacing the expectation by empirical averages over Monte Carlo samples.

In Perrin, Perolat, et al. (2020) (Chapter 4), we solve finite-horizon MFG by a fictitious play method in which the best responses are computing using tabular Q-learning. Mishra, Vasal, and Vishwanath, 2020 proposed a combination of RL and backward induction to solve finite-horizon MFGs by approximating the policy starting from the terminal time. Cui and Koeppl, 2021 applied best-response based and policy-evaluation based methods combined with DRL techniques and studied numerically the impact of entropy regularization on the convergence of these methods. Although DRL methods offer many promises in terms of scalability, it is in general hard to average or sum non-linear function approximators such as neural networks. Lauriere et al., 2022 proposed best-response based and policy-evaluation based methods (namely fictitious play and OMD) with DRL techniques in such ways that average or sum of neural networks can be approximated efficiently. This leads to scalable model-free methods for finite-horizon MFGs.

Some remarks about the distribution

Observing the mean field. In the above presentation, we assume that the agent does not observe the distribution, or at least does not exploit this information to learn the equilibrium policy. Although this is the most common approach in the RL and MFGs literature, the question of learning **population-dependent policies** arises quite naturally since one could expect that agents learn how to react to the current distribution they observe. This is usual in MARL, see Y. Yang, Luo, et al. (2018) who consider Q-functions depending on the actions of all the other players. In MFGs, we can expect that by learning a population-dependent policy, the agent will be able to *generalize*, *i.e.*, to behave (approximately) optimally even for population configurations that have not been encountered during training.

Such policies take as input a distribution, which is a high-dimensional object. As a consequence, they are much more challenging to approximate than population-independent policies. Mishra, Vasal, and Vishwanath (2020) considered a population-dependent value function and proposed an approach based on solving a fixed point at each time step for every possible distribution. Implementing this approach (at least in its current form) seems feasible only for very small state space. In Perrin, Laurière, Pérolat, Élie, et al. (2021), we introduce the concept of **master policies**, which are population-dependent policies allowing to recover an equilibrium

policy for any observed population distribution. They can be approximately computed by a combination of Fictitious play, DRL, and a suitable randomization of the initial distribution.

The concept of a value function depending on the population distribution is connected to the so-called **Master equation** in MFGs. Introduced by Lions (2012) in continuous MFGs (continuous time, continuous state and action spaces), this partial differential equation (PDE) corresponds to the limit of systems of Hamilton-Jacobi-Bellman PDEs characterizing Nash equilibria in symmetric N-player games. We refer the interested reader to Bensoussan, Frehse, and S. C. P. Yam (2015) and Cardaliaguet, Delarue, et al. (2019) for more details on this topic.

Distribution estimation. When the state space is finite but very large, storing the population distribution in a tabular way for every state and computing the evolution of this distribution in an exact way is prohibitive in terms of memory and computational time. Representing and updating the distribution is even more challenging in the continuous space setting, even if it is just for the purpose of implementing the RL environment. In this case, one needs to rely on approximations. As already mentioned above, a possible method consists in using an empirical distribution, whose evolution can be implemented by Monte Carlo samples of an interacting agent system. This amounts to using a finite population of agents to simulate the environment. For example, in linear-quadratic MFGs the interactions are only through the mean, which can be estimated even using a single agent, see Angiuli, Fouque, and Laurière (2022) in the stationary setting and Angiuli, Fouque, and Laurière (2021), Zaman et al. (2020), and Miehling and Başar (2022) in the finite-horizon setting. However, it should be noted that even if a finite number of agents is used in the environment, this approach does not directly reduce the problem to a MARL problem because the goal is still to learn the equilibrium policy for the MFG instead of the finite-agent equilibrium policy.

Another approach consists in representing efficiently the distribution using function approximation. This raises the questions of the choice of parameterization and of the training method for the parameters. This approach can be implemented in a model-free way using Monte Carlo samples, which is particularly suitable for spaces that are too large to be explored in an exhaustive fashion. For example, in Perrin, Laurière, Pérolat, Geist, et al. (2021) (Chapter 6), we use normalizing flows (Rezende and Mohamed, 2015; Papamakarios et al., 2021) to represent the distribution of agents in a flocking model.

3.2.1 Reinforcement learning for Mean Field Control and MFMDP

As discussed in Section 3.1.4, the problem of maximizing the social reward can be interpreted in the MDP framework through a mean-field MDP in which the state incorporates the whole mean field state. Adapting in a straightforward way the RL framework represented in Figure 2.1, we can consider the environment described in Figure 3.2 where the state is $\mu \in \overline{\mathcal{X}} = \Delta_{\mathcal{X}}$ instead of

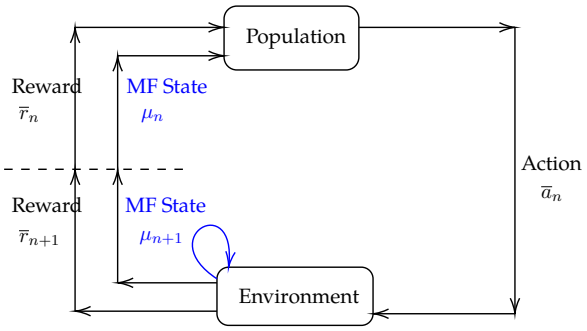


Figure 3.2 – Environment for MFC and MFMDP.

x, and the reward and the transition are given respectively by $\bar{r}$ and $\bar{p}$. An action, taken by the central planner or collectively by the population, is an element of $\overline{\mathcal{A}} = (\Delta_{\mathcal{A}})^{\mathcal{X}}$.

The problem can be interpreted as a situation in which all the agents in the population cooperate to learn a socially optimal behavior. Alternatively, we can adopt the point of view of a central planner trying to find an optimal policy that leads to a social optimum if it is followed by all the agents. In both cases, we assume here that the agent who is learning observes the whole population distribution (which is sometimes referred to as the centralized setting). The value function and the policy can thus depend on the state of the mean field, which is consistent with the dynamic programming equations presented in Section 3.1.4.

From here, standard RL techniques can be adapted to solve an MFMDP. In their implementation, the main challenge is the representation of the population distribution. A few noticeable cases are the following:

- In continuous space linear-quadratic case, the interaction is only through the mean so we do not need to give the full distribution as an input to the policy but only its first moment. In this case, policy gradient for the parameters of a suitable representation of the policy can be implemented and shown to converge, (Carmona, Laurière, and Tan, 2019a; W. Wang et al., 2021; Gu et al., 2020; Gu et al., 2021b). This approach can also be extended to more complex settings such as mean-field type games (Carmona, Hamidouche, et al., 2020; Carmona, Hamidouche, et al., 2021).

- When the state space $\mathcal{X}$ is finite, the mean field state can be represented as an element of the simplex, identified as a subset of $\mathbb{R}^{|\mathcal{X}|}$: $\{\mu \in [0,1]^{|\mathcal{X}|} : \sum_{i=1}^{|\mathcal{X}|} \mu_i = 1\}$. We can then use two different approaches.
  - First, this simplex can be discretized and replaced by a finite set $\tilde{\mathcal{X}} \subset \overline{\mathcal{X}}$. We can then approximate the MFMDP by an MDP with this finite state space. The action space is in principle $\Delta_{\tilde{\mathcal{X}}}$, which is continuous, but if we are also willing to discretize this space, then we obtain a finite state space, finite action space MDP for which tabular RL methods can be used. For example tabular Q-learning can be shown to

converge under suitable conditions (Carmona, Laurière, and Tan, 2019b; Gu et al., 2020).

- Alternatively, the original MFMDP can be tackled without space discretization by using RL techniques for continuous space MDPs. For example, Carmona, Laurière, and Tan (2019b) used DRL to learn optimal policies as functions of the population distribution viewed as an element of $\{\mu \in [0,1]^{|\mathcal{X}|} : \sum_{i=1}^{|\mathcal{X}|} \mu_i = 1\}$.

It can be argued that the environment described in Figure 3.2 is not very realistic because in general, we cannot assume that an agent observes the mean field distribution. Indeed, this distribution corresponds to the regime with an infinite number of agents while in practice, the number of agents is always finite. One can thus replace the "ideal" **McKean-Vlasov environment** by a more realistic **finite-population environment**. The former can be viewed as an approximation of the latter. The quality of the approximation gets better as the number of agents N in the environment increases. These two types of environments are discussed by Carmona, Laurière, and Tan (2019a), in which the finite-population environment analysis benefits from the analysis of the McKean-Vlasov environment. The connection between RL for MFC and finite-agent problems has also been analyzed by L. Wang, Z. Yang, and Z. Wang (2020), M. Chen et al. (2021), and Y. Li et al. (2021). Lastly, as in the MFG setting, for some MFC problems, it has been shows that observing the state of a single agent is sufficient to approximate the mean field distribution and learn the optimal behavior, see Angiuli, Fouque, and Laurière (2022) and Angiuli, Fouque, and Laurière (2021).

3.3 Metrics and Numerical Experiments

We now present numerical experiments to illustrate some of the techniques introduced in the previous sections. We first discuss metrics that can be used to assess convergence. We then present an MFG model in which the agents are encouraged to explore the spatial domain. Last, we present numerical results obtained using iterative methods.

3.3.1 Metrics

Here we discuss ways to measure convergence of the iterative methods discussed in Section 3.1. First, since many methods are based on fixed point iterations, we can measure distances between mean field terms or policies. Second, we can also measure convergence in terms of the exploitability of the current policy.

Wasserstein distance

Let us recall that the iterative methods described previously are based on the scheme described in (3.1). The pair (μ^ℓ, π^ℓ) computed at iteration ℓ is expected to converge to a fixed point. We can thus use the distance between μ^ℓ and $\mu^{\ell+1}$, and the distance between π^ℓ and $\pi^{\ell+1}$ to see whether the method has converged. Since both the mean field and the policy are distributions (respectively on the state space and the action space), we can use for instance the Wasserstein distance.

Let us focus on the mean field and look at the macroscopic behavior, at the scale of the whole population. We can proceed similarly with the policy. For simplicity, let us assume the state space $\mathcal{X}$ is a finite set endowed with a distance denoted by d. The Wasserstein distance $\mathcal{W}$ (or earth mover's distance) measures the minimum cost of turning one distribution into another and is defined as follows: for $\mu, \mu' \in \Delta_{\mathcal{X}}$,

$$\mathcal{W}(\mu, \mu') = \inf_{\nu \in \Gamma(\mu,\mu')} \sum_{(x,x') \in \mathcal{X}\times\mathcal{X}} d(x, x')\nu(x, x'),$$

where $\Gamma(\mu, \mu')$ is the set of probability distributions on $\mathcal{X} \times \mathcal{X}$ with marginals μ and μ'.

In a finite-horizon setting, the mean field term is a sequence of distributions, so we average the distances over the $N_T + 1$ time steps to get the following distance between mean field flows: for $\boldsymbol{\mu}, \boldsymbol{\mu}' \in \Delta_{\mathcal{X}}^{N_T}$,

$$\mathcal{W}_T(\boldsymbol{\mu}, \boldsymbol{\mu}') = \frac{1}{N_T + 1} \sum_{n=0}^{N_T} \mathcal{W}(\boldsymbol{\mu}_n, \boldsymbol{\mu}'_n).$$

This distance can be used in two ways to assess convergence in the context of the iterative scheme (3.1). First, in some cases the Nash equilibrium distribution (or an approximation of the equilibrium) $\hat{\mu}$ is known, so we can use $\mathcal{W}(\mu^\ell, \hat{\mu})$ to assess convergence. The MFG solution is typically unknown but in a few cases it admits an analytical solution, which can be convenient to check if a new numerical method works properly. Second, we can always measure the distance between two successive iterates, namely, $\mathcal{W}(\mu^\ell, \mu^{\ell+1})$. Although there is in general no guarantee that this distance should decrease monotonically, it goes to zero if the method converges to a fixed point. Similar ideas can be used for policies.

If $\mathcal{W}(\mu^\ell, \mu^{\ell+1})$ or $\mathcal{W}(\pi^\ell, \pi^{\ell+1})$ provide some information about the scale of the changes occurring between two iterations, these quantities do not directly say how close the pair (μ^ℓ, π^ℓ) is to a Nash equilibrium. One way to tackle this question is to measure the exploitability.

Exploitability

Instead of focusing directly on the quantities that are updated in the iterative procedure, namely the mean field and the policy, another way to assess the convergence of learning algorithms is

to consider the reward function. Indeed, the definition of an approximate Nash equilibrium can be formalized by measuring to what extent a representative player can improve their reward by deviating from the policy used by the rest of the population.

In the stationary setting for instance (similar ideas can be used in the other settings), the exploitability of a policy π is defined as (Perrin, Perolat, et al., 2020):

$$\phi(\pi) = \sup_{\pi'} J_{\text{statio}}(\pi'; \mu^\pi) - J_{\text{statio}}(\pi; \mu^\pi),$$

where $\mu^\pi = \mathbf{M}_{\text{statio}}(\pi)$ is the stationary mean field distribution induced by π as defined in (2.20), and J_{statio} is defined in (2.17). This notion is inspired by analogous concepts introduced in the context of computational game theory (Zinkevich et al., 2007; Lanctot, Waugh, et al., 2009).

Using this notion, we can rephrase the definition of mean field Nash equilibrium (see Definition 2) as: $\hat{\pi}$ is a stationary MFNE policy if:

$$\phi(\hat{\pi}) = 0.$$

Furthermore, ϕ is always non-negative, and for $\varepsilon > 0$,

$$\phi(\pi) \leq \varepsilon$$

corresponds to saying that the policy π is an ε**-stationary MFNE**, meaning that a representative player can improve its reward by at most ε by unilaterally deviating from the policy π used by the rest of the population. As such, the exploitability offers a different perspective than the Wasserstein distance discussed above to assess the convergence of learning methods. In the context of iterations described by (3.1), the quantity $\phi(\pi^\ell)$ measures convergence from the point of view of the potential reward improvement by deviating from π^ℓ. Note that, as it scales with rewards, the absolute value of the exploitability is not meaningful. What matters is its relative value compared with a reference point, such as the exploitability of the policy at initialization of the algorithm. In fact, the exploitability is game dependent and hard to re-scale without introducing other issues (dependence on the initial policy if we re-normalize with the initial exploitability for example).

In practice, the exploitability of a policy π can be computed only if we can compute $\sup_{\pi'} J_{\text{statio}}(\pi'; \mu^\pi)$. For many problems, there is not explicit formula for this value and if the environment is complex, exact methods cannot be used. However an approximation can be computed by learning an approximate best response to μ^π, for example using RL. If this step is too computationally expensive, then we can replace the supremum by a maximum over a finite set, say $\widetilde{\Pi}$. For example, we can take the set of policies computed in previous iterations.

We then obtain a notion of **approximate exploitability** (Perrin, Laurière, Pérolat, Geist, et al., 2021; Perrin, Laurière, Pérolat, Élie, et al., 2021):

$$\widetilde{\phi}(\pi) = \max_{\pi' \in \widetilde{\Pi}} J_{\text{statio}}(\pi'; \mu^{\pi}) - J_{\text{statio}}(\pi; \mu^{\pi}).$$

We conclude this section by mentioning that in the case of MFC, the convergence can be assessed through the social cost, see Section 2.2.6.

3.3.2 Experiments

In this section, we present a canonical example and we compare how the algorithms perform. The game and the algorithms are implemented in OpenSpiel (Lanctot, Lockhart, et al., 2019) and are publicly available, along with more examples and algorithms. OpenSpiel is a framework for many games besides MFGs, and it contains many reinforcement learning algorithms besides exact algorithms. See https://github.com/deepmind/open_spiel/.

Canonical example: Exploration via entropy maximization. We consider the following model, where the state space is a two dimensional grid world as in Figure 3.3. At each time step, the representative agent stay still or move by one step in the four directions provided there are no obstacles:

$$x_{n+1} = x_n + a_n + \varepsilon_n$$

with $a_n \in \{(-1,0),(0,-1),(0,0),(0,1),(1,0)\}$ and such that x_{n+1} belongs to the admissible states. Here ε_{n+1} is a perturbation which pushes the agents to a neighbor state with a small probability. In particular, in this simple model, the transitions probabilities do not depend on the mean field state. we define the reward as:

$$r(x,a,\mu) = r(x,a,\mu(x)) = -\log(\mu(x))$$

The goal for the representative agent is thus to avoid the crowd because the reward decreases as the density increases. Overall, we expect the population to spread as much as possible. In fact, at the macroscopic level of the population, the average one-step reward if the population distribution is μ is:

$$\mathbb{E}_{x\sim\mu}[-\log(\mu(x)] = -\sum_x \mu(x)\log(\mu(x)),$$

which is the entropy of μ. So maximizing the average reward amounts to maximizing the entropy. This model has been introduced and studied by Geist, Pérolat, et al. (2021), in which it is shown that it relates to an MFC problem.

For the numerical tests below, we assume that the initial distribution μ_0 is concentrated in the top-left corner as in Figure 3.3 (left). Since the reward is maximal when the distribution is uniform over the domain, one can wonder whether a uniform policy provides an approximately optimal solution. However, this is not the case, see Figure 3.3 (right), where we see that the distribution diffuses but remains mostly concentrated near the starting point. So learning a policy which induces a uniform distribution is not trivial.

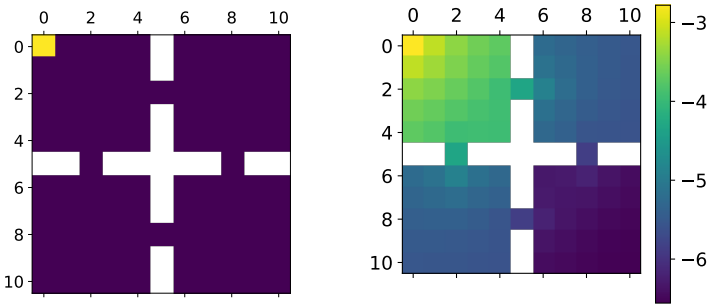


Figure 3.3 – Reading order: (**a**) the considered environment initial state in yellow, walls in white); (**b**) the log-density of a uniform policy (to illustrate entropy maximization).

Numerical experiments. For the sake of illustration, we now focus on the evolutive setting and compare the algorithms on the canonical example. We focus here on the exact iterative methods as described in Section 3.1, *i.e.* without RL approximations. In Figure 3.4, we use the notion of exploitability to check the performance of the following algorithms, which are based on the iterations described in (3.1):

- **Fixed point:** $\boldsymbol{\pi}^{\ell+1}$ is a best response against $\boldsymbol{\mu}^{\ell}$ and $\boldsymbol{\mu}^{\ell+1}$ is the mean field sequence induced by $\boldsymbol{\pi}^{\ell+1}$. We see that this method does not converge and the population concentrates on only a few states.

- **Fictitious play:** As described in (3.10), we update the mean field by averaging over past iterations. This method converges since the exploitability goes towards 0. Furthermore, the final distribution is close to uniform.

- **Online Mirror Descent:** As described in (3.12), the policy is updated by first computing a cumulative Q-function and then taking a softmax. This method converges even faster than Fictitious play, possibly because the size of each update in Fictitious play decreases with the iteration index whereas this is not the case for OMD. Accordingly, for the same number of iterations, the distribution is even closer to being uniform than with Fictitious play.

- **Damped Fixed Point:** As described in (3.9), the mean field term is updated by taking the average of the previous mean field and the mean field induced by the most recent

policy, with constant weights for the average. We see that the exploitability decreases but does not seem to converge, and the induced terminal distribution is not very close to uniform.

- **Softmax fixed point:** This method is like the fixed point iterations except that the policy is a softmax of the Q-function instead of being an argmax as in the pure best response case. We see that the method does not converge, even though the exploitability is a bit lower than in the pure fixed point method case. The induced terminal distribution is concentrated in one of the four rooms.

- **Softmax fictitious play:** This method is like the fictitious play iterations except that the policy is a softmax of the Q-function instead of being an argmax as in the pure fictitious play case. In this method, the exploitability goes down more quickly than the pure fictitious play case, as quickly as in the OMD case. However, it does not go towards 0. Instead, it remains roughly constant after a number of iterations. This is probably due to the regularization of the policy which prevents convergence towards the true Nash equilibrium policy. The induced terminal distribution is quite close to uniform but we see that the agents tend to avoid the walls, which is probably due to the extra regularization of the policy.

- **Boltzmann policy iteration:** This method corresponds to the policy iteration method described in (3.8) except that the policy is a softmax of the Q-function instead of being an argmax as in the pure fictitious play case. This method does not converge as the exploitability increases to a very high level. The induced terminal distribution is concentrated in one small part of the domain.

Based on these observations, we see that a few methods are failing to converge even when using exact updates. As discussed in Section 3.1.3, this is probably due to the lack of contraction property for the operator on which the iterations rely. In particular, we believe that in this example, only Fictitious Play and Online Mirror Descent converge to the true Nash Equilibrium. This is consistent with the results we will present in Chapter 4 and Chapter 5.

For other methods, it is worth investigating how they perform when combined with RL as described in Section 3.2. The model of exploration with four rooms considered above as well as a few other examples have been treated using DRL in Lauriere et al. (2022), where we observed that a DRL versions of pure fixed point iterations still fail to converge, while a DRL version of OMD still tends to perform better than a DRL version of fictitious play. This tends to show that testing methods with exact methods before implementing DRL versions can be helpful to compare the performance of learning methods.

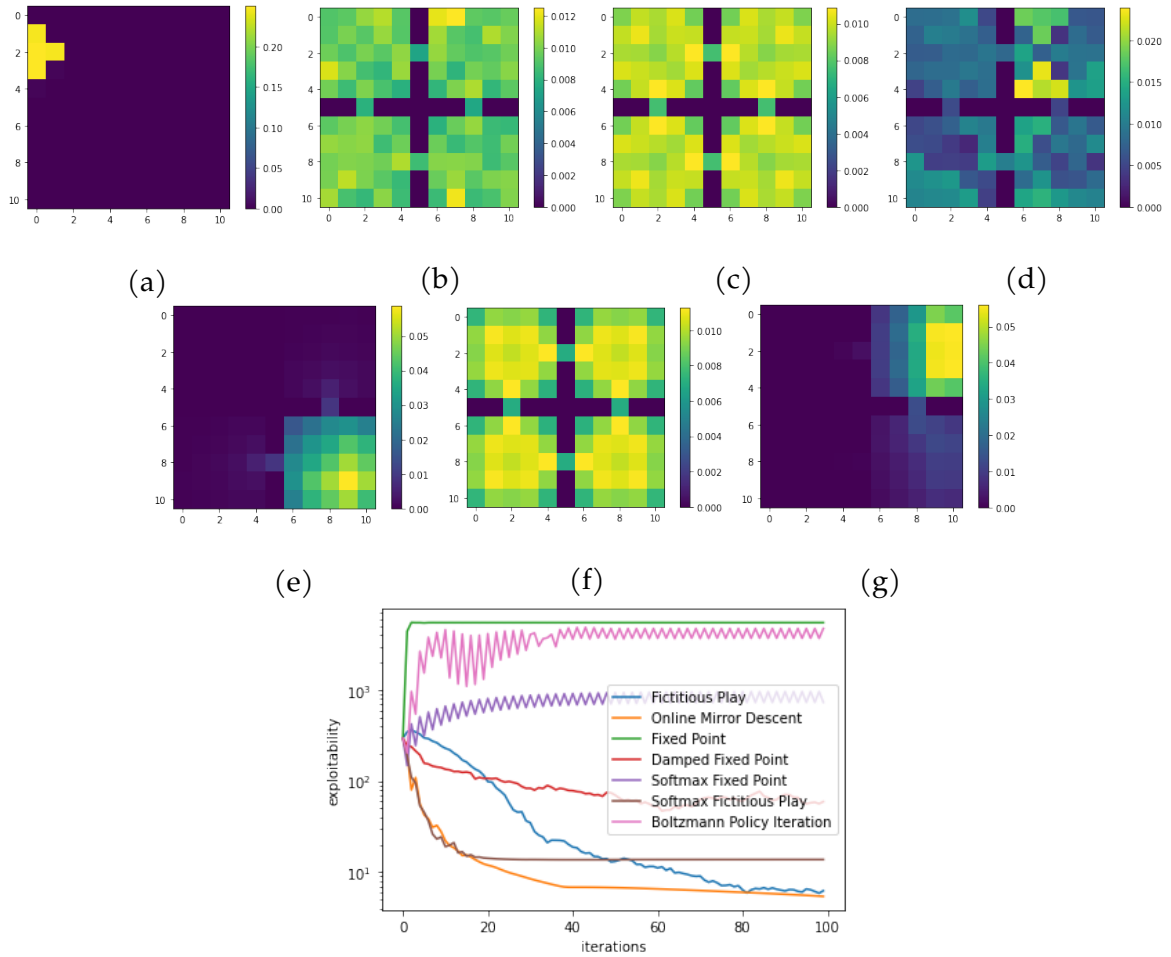


Figure 3.4 – Entropy maximization. From left to right: Terminal distribution induced by (**a**) Fixed Point; (**b**) Fictitious Play; (**c**) OMD; (**d**) Damped Fixed Point; (**e**) Softmax Fixed Point; (**f**) Softmax Fictitious Play; (**g**) Boltzmann Policy Iteration; (**h**) Exploitability curves for these methods.

Conclusion of the Chapter In this chapter, we have surveyed some of the main recent developments related to the question of learning MFGs and MFC solutions. We first clarified the definitions of several classical settings that have appeared in the literature. As far as we know, it is the first time that these settings are summarized and discussed in comparison with each other. Second, we proposed an overview of iterative methods to learn MFG and MFC solutions by updating the mean field and the policy. Starting from simple fixed-point iterations, we explained how these procedures can be enhanced by incorporating various smoothing methods. Along the way, we clarified the link with the framework of MDPs. Third, building on this connection with MDPs, we presented RL and DRL methods for MFGs and MFC. Finally, we provided some numerical results on a simple benchmark problem, highlighting that regu-

larization is often needed to ensure convergence. In the following chapters, we will zoom in two algorithms: Fictitious Play and Online Mirror Descent, and prove their convergence under the monotonicity condition.

Part II

Deep Dive to Iterative Methods: Fictitious Play and Online Mirror Descent

Chapter 4

Fictitious Play

In the previous chapter, we have discussed a general framework for iterative methods, which consists at a high level in alternatively updating the distribution and the policy. We now turn our attention to the Fictitious Play algorithm. We deepen the analysis of continuous time Fictitious Play learning algorithm to the consideration of various finite state Mean Field Game settings (finite horizon, γ-discounted), allowing in particular for the introduction of an additional common noise. We first present a theoretical convergence analysis of the continuous time Fictitious Play process and prove that the induced exploitability decreases at a rate $O(\frac{1}{t})$. Such analysis emphasizes the use of exploitability as a relevant metric for evaluating the convergence towards a Nash equilibrium in the context of Mean Field Games. These theoretical contributions are supported by numerical experiments provided in either model-based or model-free settings. We provide hereby for the first time converging learning dynamics for Mean Field Games in the presence of common noise. [1]

Contents

4.1 Motivation 74
4.2 Continuous Time Fictitious Play in Mean Field Games 78
4.3 Experiments on Fictitious Play in the Finite Horizon Case 79
4.4 Finite Horizon Mean Field Games with Common Noise 82
4.5 Experiments with Common Noise 83
4.6 Experiment at Scale 85
4.7 Conclusion of the Chapter 85

[1] This chapter is based on a preprint (Perrin, Perolat, et al., 2020) presented at the NeurIPS 2020 conference.

4.1 Motivation

We investigate a generic and scalable simulation-based learning algorithm for the computation of approximate Nash equilibria, building upon the Fictitious Play scheme (Robinson, 1951; Fudenberg and Levine, 1998b; Shapiro, 1958). We study the convergence of Fictitious Play for MFGs, using tools from the continuous learning time analysis (Harris, 1998; Ostrovski and Strien, 2013; Hofbauer and W. H. Sandholm, 2002). We then derive the convergence of the Fictitious Play process at a rate $O(\frac{1}{t})$ in finite horizon or over γ-discounted monotone MFGs (see Section B.5), thus extending previous convergence results restricted to simpler games (Harris, 1998). Besides, our approach covers games where the players share a common source of risk, which are widely studied in the MFG literature and crucial for applications. To the best of our knowledge, we derive for the first time convergence properties of a learning algorithm for these so-called MFGs with common noise (where a common source of randomness affects all players (Carmona and Delarue, 2018a)). Furthermore, our analysis emphasizes the role of *exploitability* as a relevant metric for characterizing the convergence towards a Nash equilibrium, whereas most approximation schemes in the MFG literature quantify the rate of convergence of the population empirical distribution. The contribution of this chapter is thus threefold: (1) we provide several theoretical results concerning the convergence of *continuous time* Fictitious Play in MFGs matching the $O(\frac{1}{t})$ rate existing in zero-sum two-player normal form game, (2) we generalize the notion of *exploitability* to MFGs and we show that it is a meaningful metric to evaluate the quality of a learned control in MFGs, and (3) we empirically illustrate the performance of the resulting algorithm on several MFG settings, including examples with *common noise*.

4.1.1 Related Work

Theoretical results in MFGs. Theoretical results in terms of uniqueness, existence and stability of Nash equilibrium in such games are numerous (Cardaliaguet, 2012; Bensoussan, Frehse, and S. C. P. Yam, 2013; Carmona and Delarue, 2018a). A key motivation is that the optimal control derived in an MFG provides an approximate Nash equilibrium in a game with a large but finite number of players. In general, most games are considered in a continuous setting while Gomes *et al.* D. A. Gomes, Mohr, and Souza (2010) proved existence results for finite state and action spaces MFGs and Saldi, Başar, and Raginsky (2018) considered finite state discounted cost MFGs. An important and challenging extension is the case of players sharing a common source of risk (such as several companies in the same economy market), giving rise to the so-called MFG with common noise (Carmona, Delarue, and Lacker, 2016b) or (Carmona and Delarue, 2018a, Volume II). These games are usually solved by numerical methods for partial differential equations (Achdou and Laurière, 2020) or probabilistic methods (Angiuli, Graves, et al., 2019; Carmona and Laurière, 2019; Fouque and Z. Zhang, 2020).

Learning in games and MFGs. The scaling limitations of traditional multi-agent learning methods with respect to the number of players remain quite hard to overcome as the complexity of independent learning methods (J. Foerster, R. Y. Chen, et al., 2018; Perolat, Piot, and Pietquin, 2018; Pérolat et al., 2021; Srinivasan et al., 2018; Omidshafiei et al., 2019; J. N. Foerster et al., 2018; J. Foerster, Nardelli, et al., 2017) scales at least linearly with the number of players and some methods may scale exponentially (*e.g.* Nash Q-learning (J. Hu and Wellman, 2003) or correlated Q-learning (Greenwald, Hall, and Serrano, 2003)). By approximating the discrete population by a continuous one, the MFG scheme made learning approaches more suitable and attracted a surge of interest. Model-based methods have been first considered (Yin et al., 2010) studied a MF oscillator game, (Cardaliaguet and Hadikhanloo, 2017) initiated the study of fictitious play in MFGs. Recently, several works have focused on model-free methods such as Q-learning (Guo, A. Hu, et al., 2019) but the convergence results rely on very strong hypotheses. Note that, although our method can make use of Q-learning to learn a best response, it does not rely on it. Also, our method can make use of both model-based and model-free algorithms. Finally, our method relies only on the Lasry-Lions monotonicity condition, which is much less restrictive than a potential or variational structure.

Fictitious Play (FP), which is also a classical method to learn in N-player games (Robinson, 1951; Ostrovski and Strien, 2013; Harris, 1998; Hofbauer and W. H. Sandholm, 2002; Heinrich, Lanctot, and Silver, 2015; Perolat, Piot, and Pietquin, 2018), combined with a model-free algorithm has been considered by Mguni, J. Jennings, and Munoz de Cote (2018) but with several inaccuracies, as already pointed out by J. Subramanian and Mahajan (2019), which focuses on policy gradient methods. However, they study a restricted stationary setting as opposed to the finite time horizon covered by our contribution and their convergence results hold under hardly verifiable assumptions.

Convergence of approximate FP has been proved by Elie, Perolat, et al. (2020) (based on the FP analysis of Hadikhanloo and Silva (2019)) but without common noise and their analysis is for discrete time FP and only for first-order MFGs (without noise in the dynamics). Our analysis, done in continuous time, is more transparent and works for MFGs with both idiosyncratic and common sources of randomness in the dynamics. Furthermore, their numerical example was stationary whereas we were also able to learn the solution of time-dependent MFGs, which covers a larger scope of meaningful applications. Finally, our analysis provides a rate of convergence ($O(\frac{1}{t})$) while previous FP work in MFG do not.

4.1.2 Background on Finite Horizon Mean Field Games

We recall the definition of the cumulative sum of rewards, in a finite horizon setting (with horizon N_T) where the representative player starts in $x_0 \sim m_0$ and follows the sequence of policies $\boldsymbol{\pi} = (\pi_n)_n$, while the rest of the population evolves following the sequence of

distributions $\boldsymbol{\mu} = (\mu_n)_n$

$$J(m_0, \pi, \mu) = \mathbb{E}\left[\sum_{n=0}^{N_T} r(x_n, a_n, \mu_n) \mid x_0 \sim m_0,\ x_{n+1} = p(.|x_n, a_n),\ a_n \sim \pi_n(.|x_n)\right]. \tag{4.1}$$

Q-functions and value functions. The Q-function is defined as the expected sum of rewards starting from state x and doing action a at time n:

$$Q_n^{\pi,\mu}(x, a) = \mathbb{E}\left[\sum_{k=n}^{N_T} r(x_k, a_k, \mu_k) \mid x_n = x,\ a_n = a,\ x_{k+1} = p(.|x_k, a_k),\ a_k \sim \pi_k(.|x_k)\right]. \tag{4.2}$$

By construction, it satisfies the recursive equation:

$$Q_{N_T}^{\pi,\mu}(x, a) = r(x, a, \mu_{N_T}), \quad Q_{n-1}^{\pi,\mu}(x, a) = r(x, a, \mu_{n-1}) + \sum_{x' \in \mathcal{X}} p(x'|x, a)\mathbb{E}_{b\sim\pi_n(.|x')}\left[Q_n^{\pi,\mu}(x', b)\right].$$

The value function is the expected sum of rewards for the player that starts from state x and can thus be defined as: $V_n^{\pi,\mu}(x) = \mathbb{E}_{a\sim\pi(.|x)}\left[Q_n^{\pi,\mu}(x, a)\right]$. Note that the objective function J of a representative player rewrites in particular as an average at time 0 of the value function V under the initial distribution m_0: $J(m_0, \pi, \mu) = \mathbb{E}_{x\sim m_0(.)}\left[V_0^{\pi,\mu}(x)\right]$.

Distribution induced by a policy. The state distribution induced by $\boldsymbol{\pi} = \{\pi_n\}_n$ is defined recursively by the forward equation starting from $\mu_0^\pi(x) = m_0(x)$ and:

$$\mu_{n+1}^\pi(x') = \sum_{x,a \in \mathcal{X}\times\mathcal{A}} \pi_n(a|x)p(x'|x, a)\mu_n^\pi(x)$$

Best Response. A best response policy π^{BR} is a policy that satisfies:

$$J(m_0, \pi^{BR}, \mu^\pi) = \max_{\pi'} J(m_0, \pi', \mu^\pi)$$

Intuitively, it is the optimal policy an agent could take if it was to deviate from the crowd's policy.

Exploitability. We recall that the exploitability $\phi(\pi)$ of policy π quantifies the average gain for a representative player to replace its policy by a best response, while the entire population plays with policy π: $\phi(\pi) := \max_{\pi'} J(m_0, \pi', \mu^\pi) - J(m_0, \pi, \mu^\pi)$. Please refer to Section 3.3.1 for further details.

Nash equilibrium. A Nash equilibrium is a policy satisfying $\phi(\pi) = 0$, while an approximate Nash equilibrium has a small level of exploitability.

The exploitability is an already well known metric within the computational game theory literature (Zinkevich et al., 2007; Bowling et al., 2015; Lanctot, Waugh, et al., 2009; Burch, Johanson, and Bowling, 2014), and one of the objectives of this chapter is to emphasize its important role in the context of MFGs. Classical ways of evaluating the performance of numerical methods in the MFG literature typically relate to distances between distribution μ or value function V, as for example in Achdou and Laurière (2020). A close version of the exploitability has been used in this context (Guo, A. Hu, et al., 2019), but being computed over all possible starting states at any time. Such formulation gives too much importance to each state, in particular those having a (possibly very) small probability of appearance. In comparison, the exploitability provides a well balanced average metrics over the trajectories of the state process.

Monotone games. A game is said monotone if the reward has the following structure: $r(x, a, \mu) = \tilde{r}(x, a) + \bar{r}(x, \mu)$ and $\forall \mu, \mu',\ \sum_{x\in\mathcal{X}}(\mu(x) - \mu'(x))(\bar{r}(x, \mu) - \bar{r}(x, \mu')) \leq 0$. This so-called Lasry-Lions monotonicity condition is classical to ensure the uniqueness of the Nash equilibrium (Lasry and Lions, 2007).

Learning in finite horizon problems. When the distribution μ of the population is given, the representative player faces a classical finite horizon Markov Decision problem. Several approaches can be used to solve this control problem such as model-based algorithms (*e.g.* backward induction: Algorithm B.3 in Section B.4, with update rule $\forall a, x \in \mathcal{A} \times \mathcal{X}\ Q^{\mu}_{n-1}(x, a) = r(x, a, \mu_{n-1}) + \sum_{x'\in\mathcal{X}} p(x'|x, a) \max_b Q^{\mu}_n(x', b)$) or model-free algorithms (*e.g.* Q-learning: Algorithm B.1 in Section B.4 with update rule $Q^{k+1}_n(x^k_n, a^k_n) = (1-\alpha)Q^{k+1}_n(x^k_n, a^k_n) + \alpha[r(x^k_n, a^k_n, \mu_{k-1}) + \max_b Q^k_{n+1}(x^k_{n+1}, b)]$).

Computing the population distribution. Once a candidate policy is identified, one needs to be able to compute (or estimate) the induced distribution of the population at each time step. It can either be computed exactly using a model-based method such as Algorithm B.4 in Section B.4, or alternatively be estimated with a model-free method like Algorithm B.2 in Section B.4.

Fictitious Play for MFGs. Consider available (1) a computation scheme for the population distribution given a policy, and (2) an approximation algorithm for an optimal policy of the representative player in response to a population distribution. Then, discrete time Fictitious Play presented in Algorithm 4.1 provides a robust approximation scheme for Nash equilibrium

by computing iteratively the best response against the distribution induced by the average of the past best responses. We will analyse this discrete time process in continuous time in Section 4.2. To differentiate the discrete time from the continuous time, we denote the discrete time with k and the continuous time with t. At a given step k of Fictitious Play, we have that:

$$\forall n,\ \bar{\mu}_n^k = \frac{k-1}{k}\bar{\mu}_n^{k-1} + \frac{1}{k}\mu_n^{\pi^k} \tag{4.3}$$

The policy generating this average distribution is:

$$\forall n,\ \bar{\pi}_n^k(a|x) = \frac{\sum_{i=0}^k \mu_n^{\pi^i}(x)\pi_n^i(a|x)}{\sum_{i=0}^k \mu_n^{\pi^i}(x)}. \tag{4.4}$$

Algorithm 4.1: Fictitious Play in Mean Field Games

input :Start with an initial policy π_0, an initial distribution m_0 and define $\bar{\pi}_0 = \pi_0$
for $k = 1, \dots, K$: **do**
  find π^k a best response against $\bar{\mu}^k$ (either with Q-learning or with backward induction);
  compute $\bar{\pi}^k$ the average of $(\pi^0, \dots, \pi^k)$;
  compute μ^{π^k} (either with a model-free or model-based method);
  compute $\bar{\mu}^k$ the average of $(\mu^0, \dots, \mu^{\pi^k})$
return $\bar{\pi}^K$, $\bar{\mu}^K$

4.2 Continuous Time Fictitious Play in Mean Field Games

In this section, we study a continuous time version of Algorithm 4.1. The continuous time Fictitious Play process is defined following the lines of Harris (1998) and Ostrovski and Strien (2013). First, we start for $t < 1$ with a fixed policy $\bar{\pi}^{t<1} = \{\bar{\pi}_n^{t<1}\}_n = \{\pi_n^{t<1}\}_n$ with induced distribution $\bar{\mu}^{t<1} = \mu^{t<1} = \mu^{\pi^{t<1}} = \{\mu_n^{\pi^{t<1}}\}_n$ (this arbitrary policy for $t \in [0,1]$ is necessary for the process to be defined at the starting point). Then, the Fictitious Play process is defined for all $t \geq 1$ and $n \in [1, \dots, N_T]$ as:

$$\frac{d}{dt}\bar{\mu}_n^t(x) = \frac{1}{t}\left(\mu_n^{\text{BR},t}(x) - \bar{\mu}_n^t(x)\right) \quad \text{or in integral form:} \quad \bar{\mu}_n^t(x) = \frac{1}{t}\int_{s=0}^{t} \mu_n^{\text{BR},s}(x)ds\,, \tag{4.5}$$

where $\mu_n^{\text{BR},t}$ denotes the distribution induced by a best response policy $\{\pi_n^{\text{BR},t}\}_n$ against $\bar{\mu}_n^t(x)$. Hence, the distribution $\mu_n^t(x)$ identifies to the population distribution induced by the averaged

policy $\{\pi_n^t\}_n$ defined as follows (proof in Section B.1):

$$\forall n,\ \bar{\mu}_n^t(x)\frac{d}{dt}\bar{\pi}_n^t(a|x) = \frac{1}{t}\mu_n^{\text{BR},t}(x)[\pi_n^{\text{BR},t}(a|x) - \bar{\pi}_n^t(a|x)] \tag{4.6}$$

$$\text{or in integral form: } \forall n,\ \bar{\pi}_n^t(a|x)\int_{s=0}^{t}\mu_n^{\text{BR},s}(x)ds = \int_{s=0}^{t}\mu_n^{\text{BR},s}(x)\pi_n^{\text{BR},s}(a|x)ds, \tag{4.7}$$

with $\pi_n^{\text{BR},s}$ being chosen arbitrarily for $t \leq 1$. We are now in position to provide the main result of the chapter, quantifying the convergence rate of the continuous Fictitious Play process.

Theorem 4.1. *If the MFG satisfies the monotony assumption, we can show that the exploitability is a strong Lyapunov function of the system,* $\forall t \geq 1$: $\frac{d}{dt}\phi(\bar{\pi}^t) \leq -\frac{1}{t}\phi(\bar{\pi}^t)$. *Hence* $\phi(\bar{\pi}^t) = O(\frac{1}{t})$.

The proof of the theorem is postponed to Section B.1. Furthermore, a similar property for γ discounted MFGs is provided in Section B.3. We chose to present an analysis in continuous time because it provides convenient mathematical tools allowing to exhibit state of the art convergence rate. In discrete time, similarly to normal form games (Karlin, 1959; Daskalakis and Pan, 2014), we conjecture that the convergence rate for monotone MFGs is $O(t^{-\frac{1}{2}})$, the same that in potential MFGs (Geist, Pérolat, et al., 2021).

4.3 Experiments on Fictitious Play in the Finite Horizon Case

In this section, we illustrate the theoretical convergence of continuous time Fictitious Play by looking at the discrete time implementation of the process. We focus on classical linear quadratic games which have been extensively studied (Bensoussan, Sung, et al., 2016; Graber, 2016; Duncan and Tembine, 2018) and for which a closed form solution is available. We then turn to a more difficult numerical setting for experiments[2]. We chose either a full model-based implementation or a full model-free approach of Algorithm 4.1. The model-based uses Backward Induction (Algorithm B.3) and an exact calculation of the population distribution (Algorithm B.2). The model-free approach uses Q-learning (Algorithm B.1) and a sampling-based estimate of the distribution (Algorithm B.4).

4.3.1 Linear Quadratic Mean Field Game

Environment. We consider a Markov Decision Process a finite action space $\mathcal{A} = \{-M, \dots, M\}$ together with a one dimensional finite state space domain $\mathcal{X} = \{-L, \dots, L\}$, which can be viewed as a truncated and discretized version of $\mathbb{R}$. The dynamics of a typical player picking

[2]In all experiments, we represent $\bar{\mu}$, but applying $\bar{\pi}$ to m_0 would give the same result as $\bar{\mu} = \mu^{\bar{\pi}}$.

action a_n at time n are governed by the following equation:

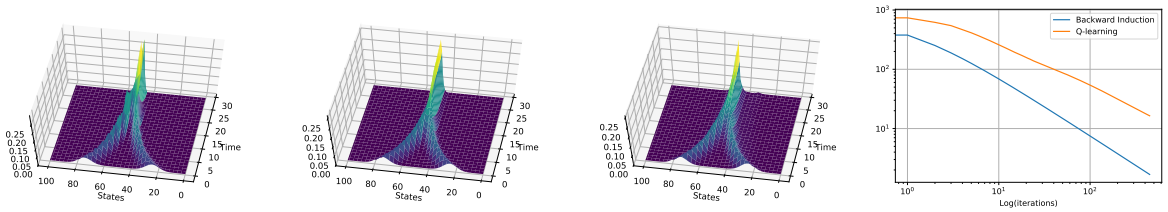
$$x_{n+1} = x_n + (K(m_n - x_n) + a_n)\Delta_n + \sigma\varepsilon_n\sqrt{\Delta_n}\ , \tag{4.8}$$

allowing the representative player to either stay still or move to the left or to the right. In order to make the model more complex, an additional discrete noise ε_n can also push the player to the left or to the right with a small probability: $\varepsilon_n \sim \mathcal{N}(0,1)$, which is in practice discretized over $\{-3\sigma, \dots, 3\sigma\}$. The resulting state x_{n+1} is rounded to the closest discrete state.

At each time step, the player can move up to M nodes and it receives the reward:

$$r(x_n, a_n, \mu_n) = [-\frac{1}{2}|a_n|^2 + qa_n(m_n - x_n) - \frac{\kappa}{2}(m_n - x_n)^2]\Delta_n$$

where $m_n = \sum_{x\in\mathcal{X}} x\mu_n(x)$ is the first moment of the state distribution μ_n. Δ_n is the time lapse between two successive steps, while q and κ are given non-negative constants. The first term quantifies the action cost, while the two last ones encourage the player to remain close to the average state of the population at any time. Hereby, the optimal policy pushes each player in the direction of the population average state. We set the terminal reward to $r(x_N, a_N, \mu_N) = -\frac{c_{\text{term}}}{2}(m_N - x_N)^2$.



(**a**) Exact Solution (**b**) Model-based (**c**) Model-free (**d**) Exploitability

Figure 4.1 – Evolution of the distribution in the linear quadratic MFG with finite horizon.

Experimental setup. We consider a Linear Quadratic MFG with 100 states and an horizon $N_T = 30$, which provides a closed-form solution for the continuous state and action version of the game (see Section B.3) and bounds the number of actions $M = 37$ required in the implementation. In practice, the variance σ of the idiosyncratic noise ε_n is adapted to the number of states. Here, we set $\sigma = 3$, $\Delta_n = 0.1$, $K = 1$, $q = 0.01$, $\kappa = 0.5$ and $c_{\text{term}} = 1$. In all the experiments, we set the learning rate α of Q-learning to 0.1 and the ε-greedy exploration parameter to 0.2.

Numerical results. Figure 4.1 illustrates the convergence of Fictitious Play model-based and model-free algorithm in such context. The initial distribution, which is set to two separated bell-shaped distributions, are both driven towards m and converge to a unique bell-shaped

distribution as expected. The parameter σ of the idiosyncratic noise influences the variance of the final normal distribution. We can observe that both Backward Induction and Q-learning provide policies that approximate this behaviour, and that the exploitability decreases with a rate close to $O(1/t)$ in the case of the model-based approach, while the model-free decreases more slowly.

4.3.2 The Beach Bar Process

As a second illustration, we now consider the beach bar process, a more involved monotone second order MFG with discrete state and action spaces, that does not offer a closed-form solution but can be analyzed intuitively. This example is a simplified version of the well known Santa Fe bar problem, which has received a strong interest in the MARL community (Arthur, 1994; Farago, Greenwald, and Hall, 2002).

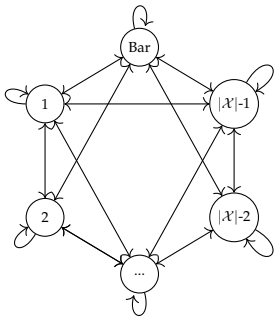


Figure 4.2 – The beach bar process.

Environment. The beach bar process (Figure 4.2) is a Markov Decision Process with $|\mathcal{X}|$ states disposed on a one dimensional torus ($\mathcal{X} = \{0, \dots, |\mathcal{X}| - 1\}$), which represents a beach. A bar is located in one of the states. As the weather is very hot, players want to be as close as possible to the bar, while keeping away from too crowded areas. Their dynamics is governed by the following equation:

$$x_{n+1} = x_n + b(x_n, a_n) + \varepsilon_n$$

where b is the drift, allowing the representative player to either stay still or move one node to the left or to the right. The additional noise ε_n can push the player one node away to the left or to the right with a small probability:

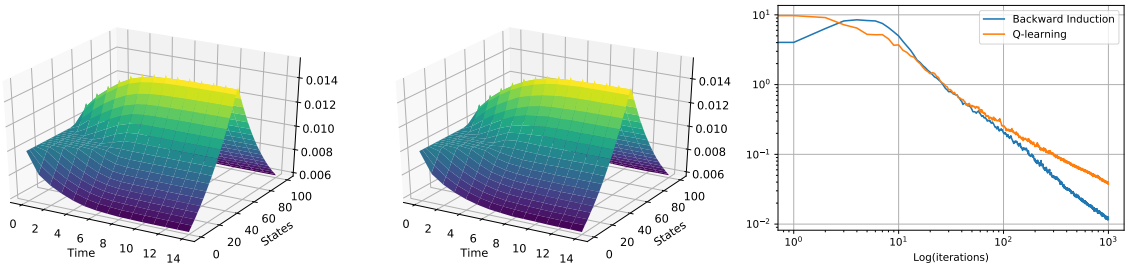
$$b(x_n, a_n) = \begin{cases} 1 & \text{if } a_n = \text{right} \\ 0 & \text{if } a_n = \text{still} \\ -1 & \text{if } a_n = \text{left} \end{cases} \qquad \varepsilon_n = \begin{cases} 1 & \text{with probability } \frac{1-p}{2} \\ 0 & \text{with probability } p \\ -1 & \text{with probability } \frac{1-p}{2} \end{cases} \tag{4.9}$$

Therefore, the player can go up to two nodes right or left and it receives, at each time step, the reward:

$$r(x_n, a_n, \mu_n) = \tilde{r}(x_n) - \frac{|a_n|}{|\mathcal{X}|} - \log(\mu_n(x_n)) \ ,$$

where $\tilde{r}(x_n)$ denotes the distance to the bar, whereas the last term represents the aversion of the player for crowded areas in the spirit of Almulla, Ferreira, and D. Gomes (2017).

Numerical results. We conduct an experiment with 100 states and an horizon $N_T = 15$. Starting from a uniform distribution, we can observe in Figure 4.3 that both backward induction and Q-learning algorithms converge quickly to a peaky distribution where the representative player intends to be as close as possible to the bar while moving away if the bar is already too crowded. The exploitability offers a nice way to measure how close we are from the Nash equilibrium and shows as expected that the model-based algorithm (backward induction) converges at a rate $O(1/t)$ and faster than the model-free algorithm (Q-learning).



(**a**) Model-based (**b**) Model-free (**c**) Exploitability

Figure 4.3 – Beach bar process in finite horizon: (a, b) evolution of the distribution, (c) exploitability.

4.4 Finite Horizon Mean Field Games with Common Noise

We now turn to the consideration of so-called MFG with common noise, that is including an additional discrete *and common* source of randomness in the dynamics. Players still sequentially take actions ($a \in \mathcal{A}$) in a state space $\mathcal{X}$, but the dynamics and the reward are affected by a common noise sequence $\{\xi_n\}_{0 \leq n \leq N}$. We denote $\Xi_n = \{\xi_k\}_{0 \leq k < n} = \Xi_{n-1}.\xi_{n-1}$ where $|\Xi_n|$ represents the total length of the sequence. The extra common source of randomness ξ affects both the reward $r(x, a, \mu, \xi)$ and the probability transition function $p(x'|x, a, \xi)$. We consider policies $\pi_n(a|x, \Xi)$ and population distribution $\mu_n(x|\Xi)$ which are both noise-dependent, and will simply be denoted $\pi_{n,\Xi}(a|x)$ and $\mu_{n|\Xi}(x)$. The Q function is defined as:

$$Q_N^{\pi,\mu}(x, a|\Xi_N) = r(x, a, \mu_{N|\Xi_N}, \xi_N), \quad Q_{n-1}^{\pi,\mu}(x, a|\Xi_{n-1}) = \sum_{\xi} P(\xi_{n-1} = \xi|\Xi_{n-1}) \Big[\tag{4.10}$$

$$r(x, a, \mu_{n-1,\Xi_{n-1}}, \xi) + \sum_{x' \in \mathcal{X}} p(x'|x, a, \xi) \mathbb{E}_{b \sim \pi_n(.|x', \Xi_{n-1}.\xi)} \left[Q_n^{\pi,\mu}(x', b|\Xi_{n-1}.\xi) \right] \Big], \tag{4.11}$$

while the value function is simply $V_n^{\pi,\mu}(x, \Xi_n) = \mathbb{E}_{a \sim \pi_{n,\Xi_n}(.|x)} \left[Q_n^{\pi,\mu}(x, a|\Xi_n) \right]$. Similarly, the distribution over states is conditioned on the sequence of noises and satisfies the balance equation: $\mu_0^{\pi}(x, \Xi_0) = m_0(x)$ (with Ξ_0 being the empty sequence $\{\}$) and $\mu_{n+1}^{\pi}(x'|\Xi.\xi) =$

$\sum_{x\in\mathcal{X}} p^{\pi_{n,\Xi},\xi}(x'|x,\xi)\mu_n^{\pi}(x|\Xi)$. The expected return for a representative player starting at m_0 is:

$$J(m_0,\pi,\mu)=\sum_{x\in\mathcal{X}} m_0(x)V_0^{\pi,\mu}(x,\Xi_0)=\sum_{n=0}^{N_T}\sum_{\Xi,\xi,|\Xi|=n} P(\Xi.\xi)\sum_{x\in\mathcal{X}}[\mu_n(x,\Xi)r(x,a,\mu_{n,\Xi},\xi)] \tag{4.12}$$

with $P(\Xi_0)=1$ and $P(\Xi.\xi)=P(\xi|\Xi)P(\Xi)$. Finally the **exploitability** is again defined as:

$$\phi(\pi)=\max_{\pi'} J(m_0,\pi',\mu^{\pi})-J(m_0,\pi,\mu^{\pi}). \tag{4.13}$$

Continuous time Fictitious Play for MFGs with common noise. The Fictitious play process on MFGs with common noise is as follows. For $t<1$, we start with an arbitrary policy $\bar{\pi}^{t<1}$ (by convention we will take $\bar{\pi}^t=\pi^{\text{BR},t}$ for $t<1$) whose distribution is $\bar{\mu}^{t<1}=\mu^{\pi^{t<1}}$ (with the convention that $\bar{\mu}^t=\mu^{\text{BR},t}$). Then, for all t and Ξ:

$$\bar{\mu}_n^t(x|\Xi)=\frac{1}{t}\int_{s=0}^{t}\mu_n^{\text{BR},s}(x|\Xi)ds, \tag{4.14}$$

where $\mu^{\text{BR},t}$ is the distribution of a best response policy $\pi^{\text{BR},t}$ against $\bar{\mu}^t$ when $t\geq 1$. The distribution μ^t is the distribution of a policy $\bar{\pi}^t$, which is defined as follows for $t\geq 1$:

$$\forall n,\Xi,\qquad \bar{\pi}_n^t(a|x,\Xi)\int_{s=0}^{t}\mu_n^{\text{BR},s}(x|\Xi)ds=\int_{s=0}^{t}\mu_n^{\text{BR},s}(x|\Xi)\pi_n^{\text{BR},s}(a|x,\Xi)ds. \tag{4.15}$$

Theorem 4.2. *Under the monotony assumption, the exploitability is a strong Lyapunov function of the system for $t\geq 1$: $\frac{d}{dt}\phi(\bar{\pi}^t)\leq-\frac{1}{t}\phi(\bar{\pi}^t)$. Therefore, $\phi(\bar{\pi}^t)=O(\frac{1}{t})$.*

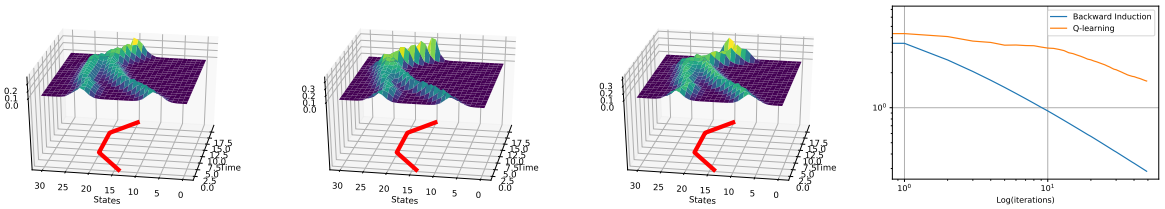
4.5 Experiments with Common Noise

4.5.1 Linear Quadratic Mean Field Game

Environment. We use a similar environment as the one described in the Linear Quadratic MFG. On top of the idiosyncratic noise ε_n, we add a common noise ξ_n, which is assumed to be stationary and i.i.d. We now consider the following dynamics:

$$x_{n+1}=x_n+(K(m_n-x_n)+a_n)\Delta_n+\sigma(\rho\xi_n+\sqrt{1-\rho^2}\varepsilon_n)\sqrt{\Delta_n}\,. \tag{4.16}$$

The reward remains unchanged, except that the first moment of the state distribution $\bar{\mu}_n$ now depends on the sequence of common noises Ξ_n: $m_n=\mathbb{E}[x_n|\Xi_n]$. We set $\rho=0.5$.



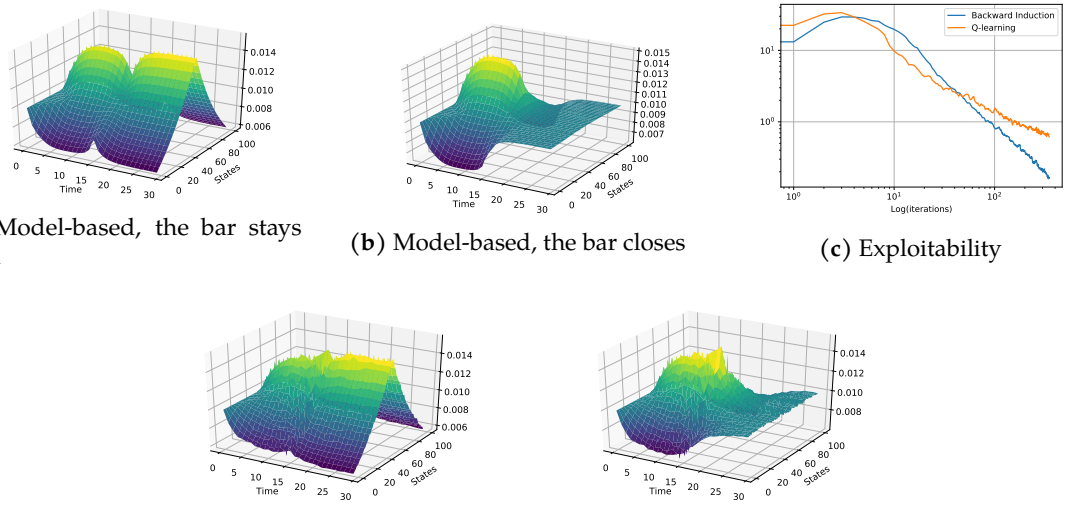
(a) Exact Solution (b) Model-based (c) Model-free (d) Exploitability

Figure 4.4 – Linear Quadratic with Common Noise.

Numerical results. On Figure 4.4, the two separated bell-shaped distributions reassemble and follow the sequence of common noises. Namely, the mean of the distribution moves with the successive common noises, which are represented by the red line below the distribution's evolution. This evolution can be interpreted as a school of fish which undergoes a water flow (*i.e.* the sequence of common noises). Both model-based and model-free approaches approximate the exact solution. The exploitability of model-based still decreases at a rate $O(1/t)$, while the one of model-free decreases more slowly.

4.5.2 The Beach Bar Process

Environment. We consider a setting where the bar can close at only one given time step. This gives two possible realizations of the common noise: (1) the bar stays open or (2) it closes at this time step. Here, the dynamics remain unchanged but the reward now depends on the common noise: r_{open} is the same reward as before, whereas $r_{closed}(x_n, a_n, \mu_n) = -\frac{|a_n|}{|\mathcal{X}|} - \log(\mu_n(x_n))$.



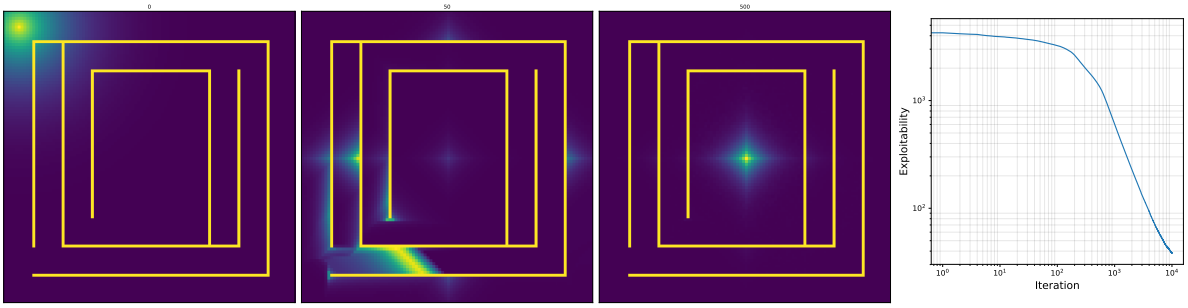
(a) Model-based, the bar stays open (b) Model-based, the bar closes (c) Exploitability

(d) Model-free, the bar stays open (e) Model-free, the bar closes

Figure 4.5 – First Common Noise setting, the bar has a probability 0.5 of closing at time step 15.

Numerical results. We set the time step of closure at $\frac{N_T}{2}$ where $N_T = 30$ is the horizon of the game and the number of states $|\mathcal{X}|$ to 100. We choose the probability of closure to be 0.5. Figure 4.5 shows that the players anticipate the possibility that the bar may close: the density of people next to the bar decreases before the time step of the common noise. After the common noise, the distribution becomes uniform if the bar has closed or people go back next to the bar if the bar stays open. Once again, the exploitability indicates that the model-based and model-free approaches both converge to the Nash equilibrium and that the model-based converges faster.

4.6 Experiment at Scale



(**a**) Start (**b**) Middle (**c**) End (**d**) Exploitability

Figure 4.6 – 2D crowd modeling example.

We finally present a crowd modeling experiment, motivated by swarm robotics (McGuire et al., 2019; Szymanski et al., 2006; Ducatelle et al., 2014), where a distribution of players is encouraged to move in a maze towards the center of a 100×100 grid. The reward at a state (i, j) is described as $r(s = (i, j), a, \mu) = 10 * (1 - \frac{\|(i,j)-(50,50)\|_1}{100}) - \frac{1}{2}\log(\mu(x))$, where the last term captures the aversion for crowded areas. The initial distribution is chosen proportional to $(1 - \frac{\|(i,j)-(5,5)\|_2}{\sqrt{2\times 95^2}})^{10}$ while being null on the maze obstacles (the yellow strait lines). The evolution of the distribution as well as the exploitability are represented in Figure 4.6.

4.7 Conclusion of the Chapter

In this chapter we have shown that Fictitious Play can serve as a basis for building practical algorithms to solve a wide variety of MFGs including finite horizon and γ-discounted MFGs as well as games perturbed by a common noise. We proved that, in all these settings, the resulting exploitability decreases at a rate of $O(\frac{1}{t})$ and that this metrics can be used to monitor the quality of the control throughout the learning. To illustrate our findings and the versatility of the method, we instantiated the Fictitious Play scheme using Backward Induction and Q-Learning to learn intermediate best responses. Application of these instances on different MFGs have

shown that the proposed algorithms consistently learned a near-optimal control and led to the desired behaviour for the population of players. This scheme has the potential to scale up dramatically by using advanced reinforcement learning algorithms combined with neural networks for the computation of the best response.

Chapter 5

Online Mirror Descent

We now turn our attention to Online Mirror Descent (OMD), another iterative method for computing Nash equilibria in Mean Field Games. We prove that continuous-time OMD converges to a Nash equilibrium under a natural and well-motivated set of monotonicity assumptions. A thorough experimental investigation on various single and multi-population MFGs highlights that OMD outperforms traditional algorithms such as Fictitious Play. We empirically show that OMD scales and converges significantly faster than Fictitious Play by solving, for the first time to our knowledge, examples of MFGs with hundreds of billions states. [1]

Contents

5.1 Motivation **88**
5.2 Preliminaries on Multi-Population Mean Field Games **89**
5.3 Online Mirror Descent: Algorithm and Convergence **93**
5.4 Numerical Experiments **97**
5.5 Conclusion of the Chapter **102**

[1] This chapter is based on a preprint (Perolat, Perrin, et al., 2021) presented at the AAMAS 2022 conference.

5.1 Motivation

In Chapter 4, we have proved that Fictitious Play converges to a Nash equilibrium under the monotonicity condition. We recall that Fictitious Play is a generic algorithm that alternates two steps starting from an arbitrary strategy for the representative player: i) computing the best response of this agent against the rest of the population, ii) compute the mixture of that best response with its previous strategy. Unfortunately, Fictitious Play seems hard to scale further for several reasons. Firstly, the computation of the best response remains a hard problem even if RL is promising, as it requires to solve a full best response at each iteration in order to converge, which is not convenient as it amounts to find a (near) optimal solution to a different MDP at each iteration. Furthermore, the regularization in Fictitious Play happens through the averaging of the distribution and policy. However, policies are pure (*i.e.* deterministic) as they are computed as the argmax of Q-functions. This means that Fictitious Play requires a high number of iterations in order for the averaging to be efficient (*i.e.* in order for the policies to be mixed enough). In contrast, we will see that Online Mirror Descent is mixing policies faster as it directly regularizes policies by computing a mixed policy (in practice, often a soft-max) with respect to the Q-function. Finally, Fictitious Play requires storing multiple quantities (e.g., averaged policies and induced distributions, etc.), which contributes to cap scalability.

Contributions. In this context, our main contribution in this chapter is the introduction of a new algorithm that can tackle a large number of agents as well as large state spaces. This algorithm, namely Online Mirror Descent (OMD) (Shalev-Shwartz, 2011), computes a NE in a large class of MFGs. Inspired by convex optimization and the Mirror Descent algorithm (Nemirovsky and Yudin, 1979), our method does not require the computation of a best response. It rather alternates a step of evaluation of the current strategy with a step of improvement of that strategy. The evaluation is done through the computation of the expected accumulated pay-offs of the strategy over time in the shape of a so-called Q-function. The improvement step reduces to computing the soft-max of the quantity obtained by integrating the Q-functions over iterations (like the Mirror Descent algorithm suggests). Quantities that need to be stored by OMD (the strategy and the integrated Q-function) are thus limited compared to Fictitious Play. As a second contribution, we provide a proof of convergence for continuous time OMD to a NE for MFGs under reasonable assumptions. These theoretical results naturally extend to multi-population MFGs as well as to settings where noise is commonly shared by all agents. Our third contribution is an extensive empirical evaluation of OMD on different tasks involving single or multiple populations, in the presence of common noise or not, with non trivial topologies. We highlight that the scale of the considered problems reaches 10^{11} states and trillions of state-action pairs, surpassing by four or five orders of magnitudes existing results. These experiments demonstrate that OMD's computational efficiency is much stronger

than Fictitious Play, which results in faster convergence. Furthermore, we provide a proof of convergence under a monotonicity assumption which improve over the more widely used contraction assumption used in the literature.

Related Work. OMD dynamics have been studied extensively within the field of multi-agent games (Cesa-Bianchi and Lugosi, 2006; Nisan et al., 2007). Leveraging the well known advantageous regret properties of such dynamics (Srebro, Sridharan, and Tewari, 2011), one can prove strong time-average convergence results both in zero-sum games (and network variants thereof) (Freund and Schapire, 1999; Cai et al., 2016) as well as in smooth-games (Roughgarden, 2009). Recently, there has been explicit focus on understanding their day-to-day behavior which has been shown to be non-equilibrating even in standard bilinear zero-sum games (Piliouras and Shamma, 2014; Mertikopoulos, Papadimitriou, and Piliouras, 2018). Moreover, even in simple games the behavior of such dynamics can become formally chaotic (Sato, Akiyama, and Farmer, 2002; Palaiopanos, Panageas, and Piliouras, 2017; Chotibut et al., 2019). Nevertheless, sufficient conditions have been established under which converge to NE is guaranteed even in the sense of the day-to-day behavior (Zhou et al., 2017; Bravo, Leslie, and Mertikopoulos, 2018). We find sufficient conditions for convergence in the more demanding setting of MP-MFG. Mirror Descent for MFGs has been introduced by Hadikhanloo (2017) for first-order, single-population MFG, while our results cover second order MP-MFG. As far as we know, our work is the first one to provide a well-suited monotonicity condition for MP-MFG. Traditional numerical methods for solving MFGs typically rely on a finite difference scheme introduced by Achdou and Capuzzo-Dolcetta (2010). This approach can be extended to solve MP-MFG (Achdou, Bardi, and Cirant, 2017). However, to the best of our knowledge, there is no general convergence guarantees, nor has it been tested on examples with as many states as we consider. Last, the question of learning with multiple infinite populations of agents has also been studied recently by J. Subramanian, Seraj, and Mahajan (2018). The authors consider several groups where the agents cooperate among each group, which differs from our setting where all the agents compete.

5.2 Preliminaries on Multi-Population Mean Field Games

We now present the theory under a slightly more general framework: Multi Population-Mean Field Games. Generalizations of the MFG framework to models with multiple populations have been introduced by M. Huang, R. P. Malhamé, and Caines (2006) and have attracted a growing interest (Bensoussan, Frehse, and S. C. P. Yam, 2013; Carmona and Delarue, 2018b; Bensoussan, T. Huang, and Laurière, 2018). Applications include urban settlements (Achdou, Bardi, and Cirant, 2017) and crowd motion (Lachapelle and Wolfram, 2011; Aurell and Djehiche, 2018). In this chapter, we denote the current iteration t in continuous time as a subscript instead of a

superscript, to alleviate the notations (because we added the number of the population i as a superscript).

In a Multi-Population Mean Field Game (MP-MFG), an infinite number of players from N_p different populations interact with each other in a temporally and spatially extended game (the case $N_p = 1$ corresponds to a standard MFG). MP-MFG are easily encompassed within MFGs on an extended state space (including the population type), but we use this setting for sake of clarity and completeness. Let $\mathcal{X}$ be the finite discrete state space and $\mathcal{A}$ be the finite discrete action space of the MP-MFG. We denote by $\Delta_{\mathcal{X}}$ and $\Delta_{\mathcal{A}}$ respectively the spaces of probability distributions over states and actions. In this sequential decision problem, a representative player of population $i \in \{1, \dots, N_p\}$ starts at a state $x_0^i \in \mathcal{X}$ according to a distribution $m_0^i \in \Delta_{\mathcal{X}}$. We consider a finite time horizon $N_T > 0$. At each time step $n \in \{0, \dots, N_T\}$, the representative player of population i is in state x_n^i and takes an action according to $\pi_n^i(.|x_n^i)$, where $\pi_n^i \in (\Delta_{\mathcal{A}})^{\mathcal{X}}$ is a policy. Given this action a_n^i, the representative player moves to a next state x_{n+1}^i with probability $p(.|x_n^i, a_n^i)$ and receives a reward $r^i(x_n^i, a_n^i, \mu_n^1, \dots, \mu_n^{N_p})$, where μ_n^j is the distribution of the population j at time n. Here $p \in (\Delta_{\mathcal{X}})^{\mathcal{X}\times\mathcal{A}}$ and $r^i : \mathcal{X} \times \mathcal{A} \times (\Delta_{\mathcal{X}})^{N_p} \to \mathbb{R}$. Observe that the transition kernel does not depend on the Multi-population distribution as in most classical MFG examples, see *e.g.*, the original work of Lasry and Lions (2007).

For the reader's convenience, we denote $\boldsymbol{\pi}^i = \{\pi_n^i\}_{n\in\{0,\dots,N_T\}}$, $\boldsymbol{\mu}^i = \{\mu_n^i\}_{n\in\{0,\dots,N_T\}}$, $\boldsymbol{\pi} = \{\boldsymbol{\pi}^i\}_{i\in\{1,\dots,N_p\}}$, $\boldsymbol{\mu} = \{\boldsymbol{\mu}^i\}_{i\in\{1,\dots,N_p\}}$, $\pi_n = \{\pi_n^i\}_{i\in\{1,\dots,N_p\}}$ and $\mu_n = \{\mu_n^i\}_{i\in\{1,\dots,N_p\}}$. Please note that accordingly to previous chapters, we keep the convention of bold symbols for time-dependant quantities.

During the game and given a fixed multi-population distributions sequence $\boldsymbol{\mu}$, a representative player of population i accumulates the following sum of rewards:

$$J^i(\boldsymbol{\pi}^i, \boldsymbol{\mu}) = \mathbb{E}\Big[\sum_{n=0}^{N_T} r^i(x_n^i, a_n^i, \mu_n) \;\Big|\; x_0^i \sim m_0^i, a_n^i \sim \pi_n^i(.|x_n^i), x_{n+1}^i \sim p(\cdot|x_n^i, a_n^i)\Big].$$

Backward Equation: Given a population i, a time n, a state x^i, an action a^i, a policy π^i and a multi-population distribution sequence μ, we define the Q**-function**:

$$Q_n^{i,\boldsymbol{\pi}^i,\boldsymbol{\mu}}(x^i, a^i) = \mathbb{E}\Big[\sum_{k=n}^{N_T} r^i(x_k^i, a_k^i, \mu_k)\Big| \; x_n^i = x^i, a_n^i = a^i, a_k^i \sim \pi_k^i(.|x_k^i), x_{k+1}^i \sim p(.|x_k^i, a_k^i)\Big]$$

and the **value function**:

$$V_n^{i,\boldsymbol{\pi}^i,\boldsymbol{\mu}}(x^i) = \mathbb{E}\Big[\sum_{k=n}^{N_T} r^i(x_k^i, a_k^i, \mu_k)\Big| \; x_n^i = x^i, a_k^i \sim \pi_k^i(.|x_k^i), x_{k+1}^i \sim p(.|x_k^i, a_k^i)\Big].$$

These two quantities can be computed recursively with the following backward equations:

$$
\begin{aligned}
&Q_{N_T}^{i,\boldsymbol{\pi}^i,\boldsymbol{\mu}}(x^i,a^i) = r^i(x^i,a^i,\mu_{N_T}) \\
&Q_{n-1}^{i,\boldsymbol{\pi}^i,\boldsymbol{\mu}}(x^i,a^i) = r^i(x^i,a^i,\mu_{n-1}) + \sum_{x'^i\in\mathcal{X}} p(x'^i|x^i,a^i)\mathbb{E}_{b^i\sim\pi_n^i(.|x'^i)}\Big[Q_n^{i,\boldsymbol{\pi}^i,\boldsymbol{\mu}}(x^i,b^i)\Big], \\
&V_n^{i,\boldsymbol{\pi}^i,\boldsymbol{\mu}}(x^i) = \mathbb{E}_{a^i\sim\pi_n^i(.|x'^i)}\Big[Q_n^{i,\boldsymbol{\pi}^i,\boldsymbol{\mu}}(x^i,a^i)\Big].
\end{aligned}
$$

Finally, the sum of rewards is $J^i(\boldsymbol{\pi}^i,\boldsymbol{\mu}) = \mathbb{E}_{x^i\sim m_0^i}[V_n^{i,\boldsymbol{\pi}^i,\boldsymbol{\mu}}(x^i)]$.

Forward Equation: If all the agents of a population i follow the policy $\boldsymbol{\pi}^i$, the induced population distribution defines recursively via the following forward equation: $\boldsymbol{\mu}_0^{i,\boldsymbol{\pi}^i} = \boldsymbol{\mu}_0^i$ and, for all $x'^i \in \mathcal{X}$,

$$
\mu_{n+1}^{i,\pi^i}(x'^i) = \sum_{(x^i,a^i)\in\mathcal{X}\times\mathcal{A}} \pi_n^i(a^i|x^i)p(x'^i|x^i,a^i)\mu_n^{i,\pi^i}(x^i)\ \text{, for } n \leq N-1\,. \tag{5.1}
$$

We denote $\boldsymbol{\mu}^{\pi} = (\boldsymbol{\mu}^{i,\boldsymbol{\pi}^i})_{i\in\{1,\dots,N_p\}}$ and emphasize the following property for the cumulative sum of rewards:

$$
J^i(\boldsymbol{\pi}^i,\boldsymbol{\mu}) = \sum_{n=0}^{N_T}\sum_{(x^i,a^i)\in\mathcal{X}\times\mathcal{A}} \mu_n^{i,\pi^i}(x^i)\pi_n^i(a^i|x^i)r^i(x^i,a^i,\mu_n)
$$

.

Best Response and Exploitability. A **best response policy** $\boldsymbol{\pi}^{i,BR,\mu}$ to a multi-population distribution sequence $\boldsymbol{\mu}$ verifies the following property $\max_{\boldsymbol{\pi}^i} J^i(\boldsymbol{\pi}^i,\boldsymbol{\mu}) = J^i(\boldsymbol{\pi}^{i,BR,\boldsymbol{\mu}},\boldsymbol{\mu})$. It can be computed recursively by finding the best responding Q-function $Q^{i,BR,\boldsymbol{\mu}}$:

$$
\begin{aligned}
&Q_{N_T}^{i,BR,\boldsymbol{\mu}}(x^i,a^i) = r^i(x^i,a^i,\boldsymbol{\mu}_{N_T}) \\
&Q_{n-1}^{i,BR,\boldsymbol{\mu}}(x^i,a^i) = r^i(x^i,a^i,\boldsymbol{\mu}_{n-1}) + \sum_{x'^i\in\mathcal{X}} p(x'^i|x^i,a^i)\max_{b^i}\Big[Q_n^{i,BR,\mu}(x^i,b^i)\Big].
\end{aligned}
$$

Finally, $\pi_n^{i,BR,\boldsymbol{\mu}}(.|x^i) \in \arg\max Q_n^{i,BR,\boldsymbol{\mu}}(x^i,.)$.

The **exploitability** measures the distance to an equilibrum and is defined as $\phi(\boldsymbol{\pi}) = \sum_{i=1}^{N_p}\phi^i(\boldsymbol{\pi})$ where, for each i, $\phi^i(\boldsymbol{\pi}) = \max_{\boldsymbol{\pi}'^i} J^i(\boldsymbol{\pi}'^i,\boldsymbol{\mu}^{\boldsymbol{\pi}}) - J^i(\boldsymbol{\pi}^i,\boldsymbol{\mu}^{\boldsymbol{\pi}})$.

Monotonicity. A multi-population game is said to be **weakly monotone** if, for any $\rho_n^i, {\rho'}_n^i \in \Delta(\mathcal{X} \times \mathcal{A})$ and $\mu_n^i, {\mu'}_n^i \in \Delta_{\mathcal{X}}$ satisfying

$$\mu_n^i = \sum_{a^i \in \mathcal{A}} \rho_n^i(\cdot, a^i) \text{ and } {\mu'}_n^i = \sum_{a^i \in \mathcal{A}} {\rho'}_n^i(\cdot, a^i)$$

for all i, n, we have

$$\sum_i \sum_{(x^i, a^i) \in \mathcal{X} \times \mathcal{A}} (\rho_n^i(x^i, a^i) - {\rho'}_n^i(x^i, a^i)) \times (r^i(x^i, a^i, \mu_n) - r^i(x^i, a^i, {\mu'}_n)) \leq 0\,.$$

It is **strictly weakly monotone** if the inequality is strict whenever $\rho_n \neq \rho'_n$. This condition means that the players are discouraged from taking similar state-action pairs as the rest of the population. Intuitively, it can be interpreted as an aversion to crowded areas.

We have the following property, whose proof is postponed to Section C.5.

Lemma 1. *The weak monotonicity property implies that for any $\boldsymbol{\pi}, \boldsymbol{\pi}'$ with $\boldsymbol{\pi} \neq \boldsymbol{\pi}'$,*

$$\tilde{\mathcal{M}}(\boldsymbol{\pi}, \boldsymbol{\pi}') := \sum_{i=1}^{N_p} \left[J^i(\boldsymbol{\pi}^i, \boldsymbol{\mu}^{\boldsymbol{\pi}}) + J^i({\boldsymbol{\pi}'}^i, \boldsymbol{\mu}^{\boldsymbol{\pi}'}) - J^i(\boldsymbol{\pi}^i, \boldsymbol{\mu}^{\boldsymbol{\pi}'}) - J^i({\boldsymbol{\pi}'}^i, \boldsymbol{\mu}^{\boldsymbol{\pi}}) \right] \leq 0. \tag{5.2}$$

Strictly weak monotonicity implies a strict inequality above.

Moreover, the weak monotonicity condition is met in the following classical setting (see Section C.1).

Lemma 2. *Assume the reward is* **separable**, *i.e. $r^i(x^i, a^i, \mu) = \bar{r}^i(x^i, a^i) + \tilde{r}^i(x^i, \mu)$ and the following* **monotonicity condition** *holds: for all $\mu \neq \mu'$, $\sum_i \sum_{x \in \mathcal{X}} (\mu^i(x^i) - {\mu'}^i(x^i))(\tilde{r}^i(x^i, \mu) - \tilde{r}^i(x^i, \mu')) \leq 0$ (resp. < 0). Then the game is weakly monotone (resp. strictly weakly monotone).*

An example of such a separable and monotone reward can be found in multi-population predator prey models where the reward can be expressed as a network zero-sum game:

$$r^i(x^i, a^i, \mu) = \bar{r}^i(x^i, a^i) + \underbrace{\hat{r}^i(x^i, \mu^i) + \sum_{j \neq i} \mu^j(x^i) \hat{r}^{i,j}(x^i)}_{=\tilde{r}^i(x^i, \mu)} \tag{5.3}$$

if $\hat{r}^{i,j} = -\hat{r}^{j,i}$ and $\hat{r}$ satisfies the previous monotonicity condition.

Nash Equilibrium (NE). A NE is a vector of policies for all populations that has 0 exploitability. The existence of a NE in MFGs has been studied in many settings (Cardaliaguet, 2012;

Bensoussan, Frehse, and S. C. P. Yam, 2013; Carmona and Delarue, 2018a). In our framework, it is a consequence of the convergence of the Fictitious Play dynamics in monotone games, which is detailed in Section C.3.

Proposition 1 (Existence and uniqueness of Nash)**.** *Any weakly monotone MP-MFG admits a NE. Besides, if the weak monotonicity is strict, the NE is unique.*

Proof. The existence result follows from Theorem C.1 and uniqueness is proven in Section C.6. □

5.3 Online Mirror Descent: Algorithm and Convergence

We now turn to the Online Mirror Descent Algorithm and introduce a regularizer $h : \Delta_{\mathcal{A}} \to \mathbb{R}$, that is assumed to be ρ-strongly convex for some constant $\rho > 0$. Furthermore, we will assume from this point forward that the regularizer h is *steep*, i.e., $\|\nabla h(\pi)\| \to \infty$ whenever π approaches the border of $\Delta_{\mathcal{A}}$; The classic negentropy regularizer, which results to replicator dynamics is the prototypical example of this class. Denote by $h^* : \mathbb{R}^{|\mathcal{A}|} \to \mathbb{R}$ its convex conjugate defined by $h^*(y) = \max_{\pi \in \Delta\mathcal{A}} [\langle y, \pi \rangle - h(\pi)]$. Since h is differentiable almost everywhere, we have, for almost every y,

$$\Gamma(y) := \nabla h^*(y) = \arg\max_{\pi \in \Delta\mathcal{A}} [\langle y, \pi \rangle - h(\pi)]. \tag{5.4}$$

Discrete Time Online Mirror Descent. The OMD algorithm is implemented as described in Algorithm 5.1. At each iteration, the first step consists in computing, for each population, the evolution of the population's distribution by using the current policy, see Equation (5.1). In the second step, each population's policy is updated with learning rate α. This update is done by first updating the corresponding y variable and then obtaining the policy thanks to the function Γ. As in Chapter 4, we denote the discrete iterations $k \in \{1, \dots, K\}$ and not t than we keep for continuous-time updates. We have for all $k \in \mathbb{N}^*, i \in \{1, \dots, N_p\}, n \in \{0, \dots, N_T\}$,

$$\begin{aligned} y_n^{i,k+1}(x^i, a^i) &= \sum_{l=0}^{k} \alpha Q_n^{i,\pi^{i,l},\mu^{\pi^l}}(x^i, a^i), \\ \pi_n^{i,k+1}(.|x^i) &= \Gamma(y_n^{i,k+1}(x^i, .)). \end{aligned}$$

Algorithm 5.1: Online Mirror Descent (OMD)

input : Learning rate α, $y_n^{i,0} = 0$ for all i, n; number of iterations K

for $k = 1, \dots, K$: **do**

  Forward Update: Compute for all i, μ^{i,π^k};

  Backward Update: Compute for all i, $Q^{i,\pi^{i,k},\mu^{i,\pi^k}}$;

  Update for all i, n, x, a,

$$y_n^{i,k+1}(x,a) = y_n^{i,k}(x,a) + \alpha Q_n^{i,\pi^{i,k},\mu^{\pi^k}}(x,a)$$
$$\pi_n^{i,k+1}(.|x) = \Gamma(y_n^{i,k+1}(x,.))$$

output : Last policy $\boldsymbol{\pi}^K$ and mean field flow $\boldsymbol{\mu}^K$

Continuous Time Online Mirror Descent. We study the theoretical convergence of the continuous time version of Algorithm 5.1. Namely, the Continuous Time Online Mirror Descent (CTOMD) algorithm (Mertikopoulos, Papadimitriou, and Piliouras, 2018) is defined as: for all $i \in \{1, \dots, N_p\}, n \in \{0, \dots, N_T\}$, $y_{n,0}^i = 0$, and for all $t \in \mathbb{R}_+$,

$$y_{n,t}^i(x^i, a^i) = \int_0^t Q_n^{i,\pi_s^i,\mu^{\pi_s}}(x^i, a^i) ds, \tag{5.5}$$

$$\pi_{n,t}^i(.|x^i) = \Gamma(y_{n,t}^i(x^i, .)). \tag{5.6}$$

From here on, unless otherwise specified, we assume that the weak monotonicity condition holds and denote by $\boldsymbol{\pi}^*$ the policy of a NE, whose existence follows from Proposition 1. We let $y^{i,*} : (x^i, a^i) \mapsto y^{i,*}(x^i, a^i)$ be the corresponding dual variable such that $\boldsymbol{\pi}^{i,*}(.|x^i) = \Gamma(y^{i,*}(x^i, .))$ for every i.

Measure of similarity with the NE $\boldsymbol{\pi}^*$. Based on the regularizer h, we define in the dual space the following measure of similarity $H : \mathbb{R}^{|\mathcal{A}|} \to \mathbb{R}$ with the NE $\boldsymbol{\pi}^*$:

$$H(y) := \sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{x^i \in \mathcal{X}} \mu_n^{i,\pi^*}(x^i) \Big[h^*(y_{n,t}^i(x^i, .)) - h^*(y^{i,*}(x^i, .)) - \langle \pi_{n,t}^{i,*}, y_{n,t}^i(x^i, .) - y_{n,t}^{i,*}(x^i, .) \rangle \Big].$$

As detailed below, this quantity will be decreasing through the iterations of CTOMD. Observe that since the regularizer is steep and thus always maps in the interior of the simplex,

it can also be expressed in terms of Bregman divergence as:

$$H(y) = \sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{x^i \in \mathcal{X}} \mu_n^{i,\boldsymbol{\pi}^*}(x^i)[D_h(\pi_n^{i,*}(x^i,\cdot), \pi_n^i(x^i,\cdot))].$$

which is always non-negative. Here D_F denotes the Bregman divergence associated with a map F and defined as :

$$D_F(p,q) := F(p) - F(q) - \langle \nabla F(q), p - q \rangle.$$

In this derivation we have used known relations between Fenchel couplings and Bregman divergences (Mertikopoulos and W. H. Sandholm, 2016) and denoted $\pi_n^i := \Gamma(y_n^i)$. Thus, the similarity measure H can also be expressed in terms of proximity between policies.

We are now in position to characterize the dynamics of the similarity to the Nash mapping via the following lemma, whose proof is provided in Section C.4.

Lemma 3 (Similarity dynamics). *In CTOMD, the measure of similarity H to the Nash $\boldsymbol{\pi}^*$ satisfies*

$$\frac{d}{dt} H(y_t) = \Delta J(\boldsymbol{\pi}_t, \boldsymbol{\pi}^*) + \tilde{\mathcal{M}}(\boldsymbol{\pi}_t, \boldsymbol{\pi}^*) \tag{5.7}$$

where $\Delta J(\boldsymbol{\pi}_t, \boldsymbol{\pi}^*) := \sum_{i=1}^{N_p} J^i(\boldsymbol{\pi}_t^i, \boldsymbol{\mu}^{\boldsymbol{\pi}^*}) - J^i(\boldsymbol{\pi}^{i,*}, \boldsymbol{\mu}^{\boldsymbol{\pi}^*})$ is always non-positive, and the weak monotonicity metric $\tilde{\mathcal{M}}$ is defined in (5.2).

Convergence to the Nash for MP-MFGs We now turn to the main theoretical contribution of the paper, by deriving the convergence of CTOMD to the set of NE for MP-MFGs (proof in Section C.7).

Environment	$\lvert\mathcal{X}\rvert$	$\lvert\mathcal{X}\rvert \times \lvert\mathcal{A}\rvert$	OMD	Fictitious Play
Garnet	$2 \times 10^3 - 2 \times 10^4$	$2 \times 10^4 - 4 \times 10^5$	84 – 229KB	168 – 458KB
Building	8×10^9	5.6×10^{10}	0.21TB	0.42TB
Common noise	2.73×10^{11}	1.092×10^{12}	5.0TB	10TB
Multi-Pop. medium	5×10^7	2×10^8	0.93GB	1.9GB
Multi-Pop. large	8×10^8	3.2×10^9	73GB	146GB

Table 5.1 – Number of states, action-states pairs & RAM memory required for the experiments. $|\mathcal{X}|$ = positions × timesteps × common noise × number of populations (KB stands for Kilo Byte, G stands for Giga and T stands for Tera).

Theorem 5.1 (Convergence of CTOMD). *If a MP-MFG satisfies $\tilde{\mathcal{M}}(\pi, \pi') < 0$ if $\mu^\pi \neq \mu^{\pi'}$ and 0 otherwise, then $(\pi_t)_{t \geq 0}$ generated by CTOMD given in (5.6) converges to the set of Nash equilibria of the game as $t \to +\infty$.*

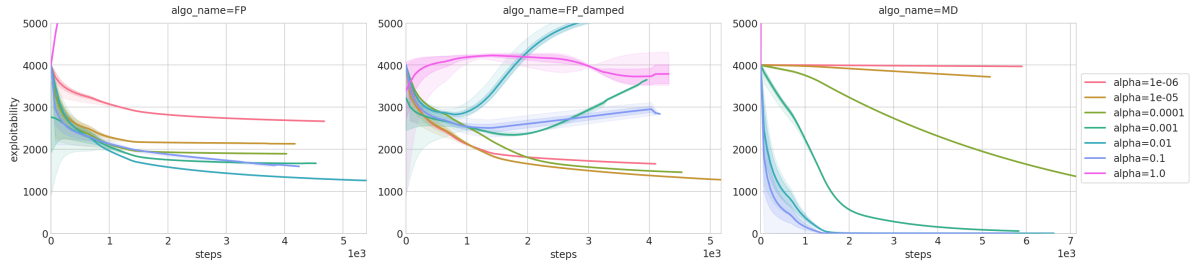


Figure 5.1 – 5 Garnet sampled with param $n_x = 20000$, $n_a = 10$, $t = 2000$, $s_f = 10$

Proof. The assumption $\frac{dH(y_t)}{dt} < 0$ is enough to guarantee convergence of $H(y_t)$ to 0. This relies on the so-called strict Lyapunov condition, which is classical in Lyapunov theory. It can be found in non-linear system books such as Khalil (2002) or more recently in discrete time in Pérolat et al. (2021). Let's briefly sketch the main argumentation that relies on a contradiction argument and divides in the following steps in the context of our problem:

- First, in order to have a one to one mapping between π and y, one can rewrite an equivalent dynamical system on the policy

$$y^i_{n,t}(x^i, a^i) = \int_{s=0}^{t} Q_n^{i,\pi^i_s,\mu^{\pi_s}}(x^i, a^i)ds - \int_{s=0}^{t} Q_n^{i,\pi^i_s,\mu^{\pi_s}}(x^i, a^i_{x^i})ds$$

  where $a^i_{x^i}$ is a fixed action for state x^i without changing the trajectory of the policy.

- Second, if $\frac{d}{dt}H(y_t) = \Delta J(\boldsymbol{\pi}_t, \boldsymbol{\pi}^*) + \tilde{\mathcal{M}}(\boldsymbol{\pi}_t, \boldsymbol{\pi}^*) = 0$, we have $\tilde{\mathcal{M}}(\boldsymbol{\pi}_t, \boldsymbol{\pi}^*) = 0$ as $\Delta J(\boldsymbol{\pi}_t, \boldsymbol{\pi}^*) \leq 0$ which is only true if $\boldsymbol{\pi}_t$ is a Nash (under Theorem 5.1 conditions).

- Then, assume that $H(y_t)$ is bounded from below by $c > 0$. Given the sign of the derivative, it is also bounded from above by $C = H(y_0)$.

- The set $\{y | H(y) \leq C\}$ must be bounded which is true in our case as h is *steep* (H goes to infinity as π gets close to the boundary) and :

$$H(y) = \sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{x^i \in \mathcal{X}} \mu_n^{i,\boldsymbol{\pi}^*}(x^i)[D_h(\pi_n^{i,*}(x^i, \cdot), \pi_n^i(x^i, \cdot))]$$

- Hence, the set $A_{Cc} = \{y | c \leq H(y) \leq C\}$ is compact as a closed bounded set (recall that H is continuous in y).

- As
$$\frac{dH(y_t)}{dt} = \Delta J(\Gamma(y_t), \boldsymbol{\pi}^*) + \tilde{\mathcal{M}}(\Gamma(y_t), \boldsymbol{\pi}^*)$$
while $\Delta J(\Gamma(y), \pi^*) + \tilde{\mathcal{M}}(\Gamma(y), \boldsymbol{\pi}^*) < 0$ for all y in the compact set A_{Cc}, we deduce the existence of a constant k_{max} such that $\Delta J(\Gamma(y), \boldsymbol{\pi}^*) + \tilde{\mathcal{M}}(\Gamma(y), \boldsymbol{\pi}^*) \leq k_{max} < 0$ for $y \in A_{Cc}$ (the image of a compact through a continuous function is a compact).

- As $H(y_t) = H(y_0) + \int_0^t \frac{dH(y_\tau)}{d\tau} d\tau$, this implies that $H(y_t) \leq C + t \times k_{max}$, so there will be a time t when $H(y_t) < c$. This provides a contradiction and implies that $H(y_t)$ must converge to 0.

This concludes the sketch of the proof. □

Thanks to Lemma 1 together with Proposition 1, we easily deduce the convergence to the unique NE in some more stringent classes of MP-MFGs. It is worth noticing that our line of argument differs from the usual approaches on regret minimization arguments as e.g. considered by Zinkevich et al. (2007).

Corollary 1 (Convergence of CTOMD for weakly monotone MFG)**.** *For any strictly weakly monotone MP-MFG, $(\boldsymbol{\pi}_t)_{t\geq 0}$ generated by CTOMD given in (5.6) converges to the unique NE, as $t \to +\infty$.*

Restriction to single population MFG. Finally, considering the number of populations N_p equal to 1, the convergence of CTOMD to the NE for single population strictly weakly monotone MFG follows.

Corollary 2 (Convergence of CTOMD for Single Population MFG)**.** *For any single population MFG satisfying the strictly weak monotonicity assumption, $(\pi^{(t)})_{t\geq 0}$ generated by CTOMD given in (5.6) converges to the unique NE of the game, as $t \to +\infty$.*

Remark 16. *Our proofs only give an asymptotic result and we don't think anything better is achievable in general. However if one can upper bound the monotony coefficient $\tilde{\mathcal{M}}(\boldsymbol{\pi}_t, \boldsymbol{\pi}^*)$ by the Lyapunov function $H(y_t)$ for example, the Gronwall inequality would give an exponential convergence rate.*

5.4 Numerical Experiments

We illustrate the theoretical convergence of CTOMD with an extensive empirical evaluation of OMD described in Algorithm 5.1 within various settings involving single or multiple populations as well as non trivial topologies (videos available here). These settings are typically hardly tractable using classical numerical approximation schemes for partial differential equations.

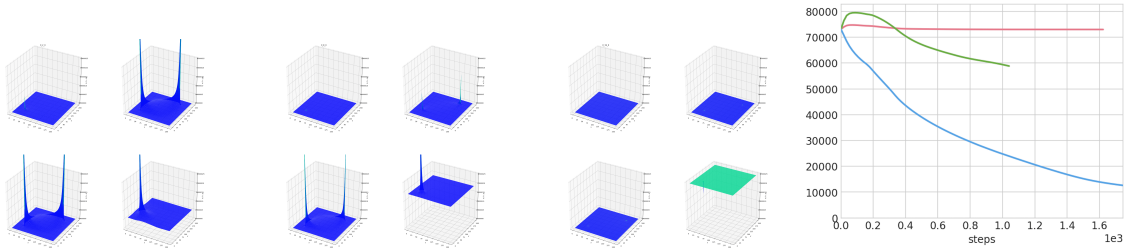


Figure 5.2 – Population distribution at consecutive dates (three first figures on the left). Each plot of a subfigure is a different floor, the bottom floor is the bottom-right plot, the top floor is the top-left plot. The figure on the right displays the exploitability of: Fictitious Play (red, $\alpha = 10^{-5}$), Fictitious Play damped (green, $\alpha = 10^{-3}$) and OMD (blue, $\alpha = 10^{-4}$).

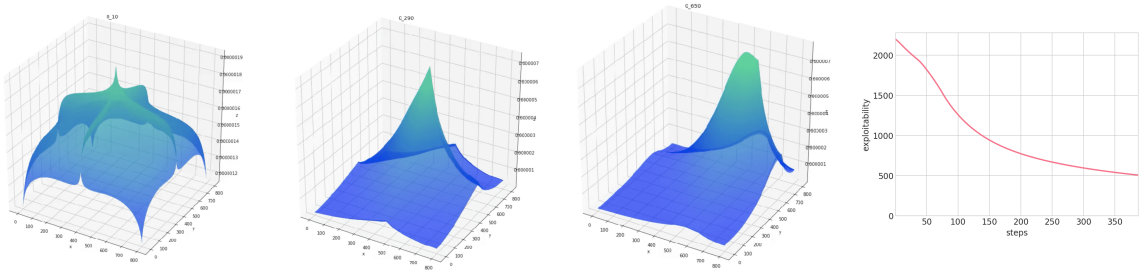


Figure 5.3 – Crowd position at different consecutive dates when the point of interest is randomly shifted to the right by a common noise. The fourth graph is displaying the exploitability of MD.

Besides, the scale of the numerical experiments grows up to 10^{12} states, establishing a new scalability benchmark in the MFG literature. We emphasize the diversity of tractable environments by considering (randomized MDP) Garnet settings, a twenty-storey high building evacuation, a crowd movement example in the presence of common noise and finally an essentially zero sum multi-population chasing game.

Experimental setup. We compare OMD and Fictitious Play with different learning rates α. In discrete-time OMD, α appears in the backward update of y:

$$y_n^{i,k+1}(x,a) = y_n^{i,k}(x,a) + \alpha Q_n^{i,\pi^{i,k},\mu^{\pi^k}}(x,a)$$

whereas in discrete-time Fictitious Play, it corresponds to the weight for updating the average policy with the new best response $\pi_n^{i,k+1}(x^i,a^i)$ given by

$$\frac{(1-\alpha_k)\mu_n^{i,\pi^k}(x^i)\pi_n^{i,k+1}(x^i,a^i) + \alpha_k \mu_n^{i,k,br}(x^i)\pi_n^{i,k,br}(x^i,a^i)}{(1-\alpha_k)\mu_n^{i,\pi^k}(x^i) + \alpha_k \mu_n^{i,k,br}(x^i)}.$$

Fictitious Play is experimented with decreasing $\alpha_k = \alpha/(2+k)$ or constant $\alpha_t = \alpha$ learning rate. This latter is referred to hereafter as *Fictitious Play damped*, while $\alpha = 1$ corresponds to the fixed point iteration algorithm, *i.e.* the population applies the last best response policy. The theoretical proof of convergence relies on restrictive conditions which only hold for a small class of games. We provide a thorough evaluation in Table 5.1 of the complexity of the environments along with the memory required to compute our results. For OMD, we only need to store y of size $|\mathcal{X}| \times |\mathcal{A}|$ and the distributions, of size $|\mathcal{X}|$. For Fictitious Play, we need to store the last best response, the average policy, the last distribution and the average distribution, requiring a total of $2 \times (|\mathcal{X}| \times |\mathcal{A}|) + 2 \times |\mathcal{X}|$. In all the experiments, h is the entropy: $h = -\sum_{a \in \mathcal{A}} \pi(a) \log(\pi(a))$. This implies that $h^*(y) = \log(\sum_a \exp(y(a)))$, and we find that Γ is a softmax if we take the gradient of h^*.

5.4.1 Garnet

We first evaluate our algorithm Algorithm 5.1 on a set of randomly generated problems (repeatability of our results for varying sizes).

Environment. A garnet is an abstract and randomly generated MDP (Archibald, McKinnon, and Thomas, 1995). We adapt this concept to single-population MFGs by modifying the reward. In our case, a Garnet is built from the set of parameters $(n_x, n_a, n_b, s_f, \eta)$, with n_x and n_a respectively the numbers of states and actions. The term n_b is a branching factor, and the transition kernel (independent of μ) is built as follows: n_b transiting states are drawn randomly without replacement, and the associated transition probabilities are obtained by partitioning the unit interval with $n_x - 1$ uniformly sampled random points. The reward term $\tilde{r}(x, u)$ is set to 0 for s_f states sampled randomly without replacement, for each of the remaining states it is set for all actions to a random value sampled uniformly in the unit interval. We set $\bar{r}(s, \mu) = -\eta \log(\mu(x))$. This reward encourages the agents to spread out accross the MDP states and can model social distancing. This process generates a monotone MFG.

Numerical results. Figure 5.1 and Figure C.4 (Section C.8.1) shows various Garnet experiments. We fix $s_f = 10$, $N_T = 2000$, $\eta = 1$ and $n_b = 1$ (deterministic dynamics) and vary $n_x \in \{2.10^3; 2.10^4\}$ and $n_a \in \{10, 20\}$. In each case, results are averaged over 5 randomly generated Garnets. We compare OMD to Fictitious Play, damped or not. We observe that OMD consistently converges faster for the right choice of α. $\alpha = 1$ might lead to unstable results while $\alpha = 0.1$ consistently provides fast convergence to the Nash. In all cases, the number of states influences the convergence rate, but much less for OMD.

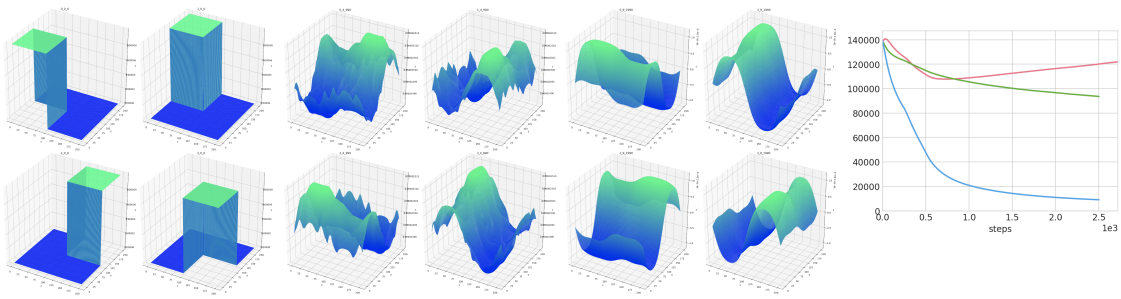


Figure 5.4 – 4-population chasing. Right figure : Fictitious Play (red, $\alpha = 10^{-3}$), Fictitious Play damped (green, $\alpha = 10^{-5}$) and OMD (blue, $\alpha = 10^{-5}$). From left to right, 3 picture chowing the distribution evolving through time and a fourth one displaying the exploitability.

5.4.2 Building evacuation

Environment. We now turn to a single-population crowd modeling problem, namely a building evacuation. This kind of problem has been considered in several studies on MFG (see e.g. Achdou and Laurière (2015) and Achdou and Lasry (2019) for a single room and Djehiche, Tcheukam, and Tembine (2017) for a multilevel building). The building consists of 20 floors, each of dimension 200×200. At each floor, two staircases are located at two opposite corners, such as the crowd has to cross the whole floor to take the next staircase. Each agent can remain in place, move in the 4 directions (up, down, right, left) as well as go up or down when on a staircase location. The initial distribution is uniform over all the floors. Each agent of the crowd wants to go downstairs as quickly as possible - as it gets a reward of 10 at the bottom floor - while favoring social distancing:

$$r(x, a, \mu) = -\eta \log(\mu(x)) + 10 \times 1_{floor=0}$$

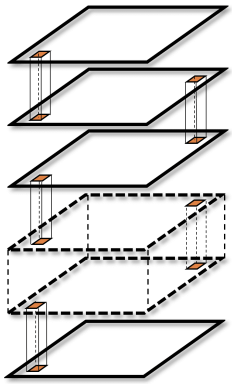


Figure 5.5 – Building environment.

Numerical results. We compute this problem with a horizon of 10000, so $|\mathcal{X}| = 8^{10}$. We take $\eta = 1$. To ensure that the reward stays bounded, we clip the first part $-\eta \log(\mu(x))$ to -40. As expected, we observe in Figure 5.2 that the agents go downstairs and do not concentrate on the shortest path but rather spread mildly. OMD converges faster than both Fictitious Play and damped Fictitious Play.

5.4.3 Crowd motion with randomly shifted point of interest

Environment. We consider a second crowd modeling MFG, extending the Beach Bar problem (Perrin, Perolat, et al., 2020) in two dimensions. The environment is a 2D torus of dimensions 1000×1000, with a point of interest initially located at the center of the square. After 200 timesteps, the point of interest changes location, moving randomly in the direction of one of the corner. This process repeats itself 5 times. This random location change adds common noise to the environment and increases exponentially the number of states. Considering MFG with common noise can be encompassed in our previous study by simply increasing the state space with the common noise and adding time to the reward and the transition kernel. For every random movement, four possible directions are possible, making the total number of states $|\mathcal{X}| = 2 \times 10^8 \times \sum_{k=0}^{4} 4^k = 2.73 \times 10^{11}$ states. The reward is: $r(x_n, a_n, \mu_n) = C \times (1 - \frac{\|bar - (i,j)\|_1}{2 \times N_{side}}) - \log(\mu_n(x_n))$.

Numerical results. We set $C = 10$. We observe in Figure 5.3 that the population is organizing itself with respect to the point of interest and follows it closely as it randomly moves within the dedicated square region. In the common noise setting, we get more than a trillion states, making it hard for Fictitious Play to scale. More plots with a smaller state space are available in Section C.8 for a comparison of OMD and Fictitious Play.

5.4.4 Multi-population chasing

Environment. We finally look at MP-MFGs, where the populations are chasing each other in a cyclic manner. For the sake of clarity, we explain the reward structure with 3 populations, but more populations are considered in the experiments. With three populations, the game closely relates to the well known Hens-Foxes-Snakes outdoor game for kids. Hens are trying to catch snakes, while snakes are chasing foxes, who are willing to eat hens. It can also be interpreted as a control version of the spatially extended Rock-Paper-Scissors, where patterns of travelling waves appear under certain conditions (Postlethwaite and Rucklidge, 2017). The interplay between nontransitive interactions and biodiversity has been the subject of extensive, mostly experimental, research showing that the setting details critically affect the emergent

behavior (Szolnoki, Oliveira, and Bazeia, 2020). To ensure $\bar{r}^{i,j} = -\bar{r}^{j,i}$ we implement MP-MFGs with the reward structure defined in Figure 5.6 (ex. with 3 populations).

	R	P	S
R	0	-1	1
P	1	0	-1
S	-1	1	0

Figure 5.6 – $\bar{r}^{i,j}$ for three-population.

The reward of population i is monotone (*cf.* Section C.8.4) and follows the definition (5.3):

$$r^i(x, a, \mu^1, \dots, \mu^N) = -\log(\mu^i(x)) + \sum_{j \neq i} \mu^j(x)\bar{r}^{i,j}(x)$$

.

The distributions are initialized either randomly or in different corners. The number of agents of each population is fixed, but the reward encourages the agent to chase the population that it dominates. For example, if an agent is Rock, the second term of the reward is proportional to the amount μ^S of Scissors agents where the Rock agent is located, and inversely proportional to the proportion μ^P of Paper agents, making the Rock agent to flee from places populated by Paper agents.

Numerical results. We present a four-population example, each is initially located at a corner of the environment. We observe that the populations are chasing each other in a cyclic fashion. Figure 5.4 highlights that OMD algorithm outperforms Fictitious Play in terms of exploitability minimization (full comparison with different values of α in Section C.8.4). It demonstrates the robustness of the OMD algorithm within the different topologies considered. Topologies of the environment are a torus, a basic square or the "donut" topology (an environment where the agent gets a negative reward if it goes inside a large zone at the center of the square).

5.5 Conclusion of the Chapter

We proposed Online Mirror Descent for MP-MFGs and proved that, under appropriate monotonicity assumptions, OMD converges to a NE. Moreover, we considered multiple experimental benchmarks, some with hundreds of billions states, and have extensively compared OMD to Fictitious Play. OMD scales empirically remarkably well, and consistently converges significantly faster than Fictitious Play. An interesting direction of future work would be to study the rate of convergence of OMD. Fictitious Play benefits from a $O(1/t)$ rate of convergence (see Section C.3) but the corresponding line of argument does not extend to OMD. Empirically,

we envision to extend this approach to a model-free setting with function approximation and address even larger problems.

Conclusion of Part I and II

To this point, we have introduced the different settings that the reader may encounter in the literature (Chapter 2) and explain how to successfully combine Mean Field Games with Reinforcement Learning (Chapter 3). We have zoomed in two algorithms, Fictitious Play (Chapter 4) and Online Mirror Descent (Chapter 5), and proved that they both converge to a Nash equilibrium under the monotonicity condition. Thus, we have brought answers to the first question, *i.e. How to design algorithms to find Nash equilibria in Mean Field Games?* We also demonstrated through a wide variety of numerical examples than Mean Field Games can scale multi-agent games up to an infinite number of agents, answering positively to the title of the thesis "Scaling up Multi-agent Reinforcement Learning with Mean Field Games". What about the "Vice-versa"? Although we have not yet tackled the resolution with model-free methods (except through the use of Q-learning in Chapter 4), our methods can tackle complex domains, with any type of conditions at borders. In this sense, they are thus already more applicable to real-world problems than the ones solving forward-backward systems. However, as our methods still require to know the full model, *i.e.* transition dynamics and reward, and scale linearly with the size of the state and action spaces, they remain limited to relatively small domains, except when we distribute the code and have access to a gigantic amount of computational power as in Chapter 5.

Thus, the third part is dedicated to the last question, *How to adapt these algorithms to a model-free setting, using deep reinforcement learning?* In particular, we will study how Deep Reinforcement Learning can be efficiently combined to Fictitious Play in the flocking example (Chapter 6). However, we restrict ourselves to the γ-stationary setting and our resulting policy is only applicable to the initial distribution considered during training. This is why we then question one of the basic hypothesis of MFGs, namely that agents are always starting from the same distribution, and propose a new approach that allows policies to both generalize to many initial distributions and react to the current distribution of agents (Chapter 7). However, we were still bound to use Fictitious Play, which has proven quite slow compared to Online Mirror Descent. Thus, lastly, we propose two deep RL adaptations of Fictitious Play and Online Mirror Descent that are both scalable and we demonstrate their applicability in a wide variety

of problems, all implemented in the OpenSpiel library. We find that in line with Chapter 5, the deep version of OMD surpasses Fictitious Play.

Part III

Deep Reinforcement Learning for Mean Field Games

Chapter 6

Flocking

We start this part with a focus on the application of flocking, that we address in a total model-free way. This problem is particularly challenging as it has many continuous dimensions, both in the state and action spaces, making it impossible to track the exact distribution of agents with dynamic programming as we did before. We present a model-free method enabling a large number of agents to learn how to flock, which is a natural behavior observed in large populations of animals. We phrase this problem as a Mean Field Game, where each individual chooses its acceleration depending on the population behavior. Combining Deep Reinforcement Learning and Normalizing Flow, we obtain a tractable solution requiring only very weak assumptions. Our algorithm finds a Nash Equilibrium and the agents adapt their velocity to match the neighboring flock's average one. We use Fictitious Play and alternate: (1) computing an approximate best response with Deep RL, and (2) estimating the next population distribution with NF. We show numerically that our algorithm learn multi-group or high-dimensional flocking with obstacles. [1]

Contents

6.1 Motivation **110**
6.2 The Model of Flocking **111**
6.3 Our Approach **112**
6.4 Experiments **116**
6.5 Conclusion of the Chapter **119**

[1] This chapter is based on a preprint (Perrin, Laurière, Pérolat, Geist, et al., 2021) presented at the AAMAS 2022 conference.

6.1 Motivation

The term flocking describes the behavior of large populations of birds that fly in compact groups with similar velocities, often exhibiting elegant motion patterns in the sky. Such behavior (*e.g.* herding, schooling, swarming), is pervasive in the animal realm, from fish to birds, bees or ants. This intriguing property has been widely studied in the scientific literature (Shaw, 1975; Partridge, 1982; Okubo, 1986; Reynolds, 1987; Toner and Tu, 1998; Olfati-Saber, 2006) and its modeling finds applications in psychology, (*e.g.* to understand animal behaviors), animation, (*e.g.* to simulate natural crowd motions), social science, (*e.g.* for consensus formation) or swarm robotics. One of the most popular approaches to model flocking was proposed by Cucker and Smale (2007) and allows predicting the evolution of each agent's velocity from the speeds of its neighbors.

To go beyond pure description of population behaviours and emphasize on the decentralized aspect of the underlying decision making process, this model has been revisited to integrate an optimal control perspective (Caponigro et al., 2013; Bailo et al., 2018). Under this point of view, each agent controls its velocity and hence its position by dynamically adapting its acceleration so as to maximize a reward that depends on the others' behavior. An important question is a proper understanding of the nature of the equilibrium reached by the population of agents, emphasizing how a consensus can be reached in a group without centralized decisions. Such question is often studied using the notion of Nash equilibrium and becomes extremely complex when the number of agents grows.

A way to approximate Nash equilibria in large games is to study the limit case of an continuum of identical agents, in which the local effect of each agent becomes negligible. This is the basis of the Mean Field Games (MFGs) paradigm introduced by Lasry and Lions (2007). However, finding an equilibrium in MFGs is computationally intractable when the state space exceeds a few dimensions. In traditional flocking models, each agent's state is described by a position and a velocity, while the control is the acceleration. In terms of computational cost, this typically rules out state-of-the-art numerical techniques for MFGs based on finite difference schemes for partial differential equations (PDEs) (Achdou and Capuzzo-Dolcetta, 2010). In addition, PDEs are in general hard to solve when the geometry is complex and require full knowledge of the model.

For these reasons, Reinforcement Learning (RL) to learn control strategies for MFGs has recently gained in popularity (Guo, A. Hu, et al., 2019; Elie, Perolat, et al., 2020; Perrin, Perolat, et al., 2020). Combined with deep neural nets, RL has been used successfully to tackle problems which are too complex to be solved by exact methods Silver, T. Hubert, et al., 2018 or to address learning in multi-agent systems (Lanctot, Zambaldi, et al., 2017). Particularly relevant in our context, are works providing techniques to compute an optimal policy (Haarnoja et al.,

2018; Lillicrap et al., 2016) and methods to approximate probability distributions in high dimension (Rezende and Mohamed, 2015; Kobyzev, Prince, and Brubaker, 2020).

Contributions. Our main contributions are: **(1)** we cast the flocking problem into a MFG and propose variations which allow multi-group flocking as well as flocking in high dimension with complex topologies, **(2)** we introduce the Flock'n RL algorithm that builds upon the Fictitious Play paradigm and involves deep neural networks and RL to solve the model-free flocking MFG, and **(3)** we illustrate our approach on several numerical examples and evaluate the solution with approximate performance matrix and exploitability.

Related Work Most work using flocking models focus on the dynamical aspect without optimization. To the best of our knowledge, the only existing numerical approach to tackle a MFG with flocking effects is in Carmona and Delarue (2018b, Section 4.7.3), but it is restricted to a very special and simpler type of rewards. The idea of using FP in MFGs has been introduced by Cardaliaguet and Hadikhanloo (2017), assuming the agent can compute perfectly the best response. Elie, Perolat, et al. (2020) and Perrin, Perolat, et al. (2020) combined FP with RL methods. However, the numerical techniques used therein do not scale to higher dimensions as needed for flocking MFGs.

6.2 The Model of Flocking

To model a flocking behaviour, we consider the following system of N agents derived in a discrete time setting by Nourian, Caines, and R. P. Malhamé (2011) from Cucker-Smale flocking modeling. Each agent i has a position and a velocity, each in dimension d and denoted respectively by x^i and v^i. We assume that it can control its velocity by choosing the acceleration, denoted by u^i. The dynamics of agent $i \in \{1, \dots, N\}$ is:

$$x^i_{n+1} = x^i_n + v^i_n \Delta n, \qquad v^i_{n+1} = v^i_n + u^i_n \Delta n + \varepsilon^i_{n+1},$$

where Δn is the time step and ε^i_n is a random variable playing the role of a random disturbance. We assume that each agent is optimizing for a *flocking criterion* f that is underlying to the flocking behaviour. For agent i at time t, f is of the form:

$$f^i_n = f(x^i_n, v^i_n, u^i_n, \mu^N_n), \tag{6.1}$$

where the interactions with other agents are only through the empirical distribution of states and velocities denoted by: $\mu^N_n = \frac{1}{N} \sum_{j=1}^N \delta_{(x^j_n, v^j_n)}$. This model is inspired by MFG setting derived

from Cucker-Smale type flocking (Nourian, Caines, and R. P. Malhamé, 2011), in a discrete time setting.

We focus on criteria incorporating a term of the form:

$$f_\beta^{\text{flock}}(x, v, u, \mu) = -\left\| \int_{\mathbb{R}^{2d}} \frac{(v - v')}{(1 + \|x - x'\|^2)^\beta} \, d\mu(x', v') \right\|^2, \tag{6.2}$$

where $\beta \geq 0$ is a parameter and $(x, v, u, \mu) \in \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \times \Delta_{\mathbb{R}^d \times \mathbb{R}^d}$, with Δ_E denoting the set of probability measures on a set E. This criterion incentivises agents to align their velocities, especially if they are close to each other. Note that β parameterizes the level of interactions between agents and strongly impacts the flocking behavior: if $\beta = 0$, each agent tries to align its velocity with all the other agents of the population irrespective of their positions, whereas the larger $\beta > 0$, the more importance is given to its closest neighbors (in terms of position).

In the N-agent case, for agent i, it becomes:

$$f_{\beta,n}^{\text{flock},i} = -\left\| \frac{1}{N} \sum_{j=1}^{N} \frac{(v_n^i - v_n^j)}{(1 + \|x_n^i - x_n^j\|^2)^\beta} \right\|^2. \tag{6.3}$$

The actual criterion will typically include other terms, for instance to discourage agents from using a very large acceleration, or to encourage them to be close to a specific position. We provide such examples in Section 6.4.

Since the agents may be considered as selfish (they try to maximize their own criterion) and may have conflicting goals (*e.g.* different desired velocities), we consider Nash equilibrium as a notion of solution to this problem and the individual criterion can be seen as the payoff for each agent. The total payoff of agent i given the other agents' strategies $u^{-i} = (u^1, \ldots, u^{i-1}, u^{i+1}, \ldots, u^N)$ is: $F_{u^{-i}}^i(u^i) = \mathbb{E}_{x_n^i, v_n^i}\left[\sum_{t \geq 0} \gamma^t f_n^i\right]$, with f_n^i defined Equation (6.1). In this context, a *Nash equilibrium* is a strategy profile $(\hat{u}^1, \hat{u}^2, \ldots, \hat{u}^N)$ such that there's no profitable unilateral deviation, *i.e.*, for every $i = 1, \ldots, N$, for every control u^i, $F_{\hat{u}^{-i}}^i(\hat{u}^i) \geq F_{\hat{u}^{-i}}^i(u^i)$.

6.3 Our Approach

In this section, we put together the pieces of the puzzle to numerically solve the flocking model. Based on a mean-field approximation, we first recast the flocking model as an MFG with decentralized decision making, for which we propose a numerical method relying on RL and deep neural networks.

6.3.1 Flocking as an MFG

Mean field limit. We go back to the model of flocking introduced in Section 6.2. When the number of agents grows to infinity, the empirical distribution μ_n^N is expected to converge towards the law μ_n of (x_n, v_n), which represents the position and velocity of an infinitesimal agent and have dynamics:

$$x_{n+1} = x_n + v_n \Delta n, \qquad v_{n+1} = v_n + u_n \Delta n + \varepsilon_{n+1}.$$

This problem can be viewed as an instance of the MFG framework discussed before, by taking the state to be the position-velocity pair and the action to be the acceleration, *i.e.*, the dimensions are $\ell = 2d$, $k = d$, and $\xi = (x, v)$, $\alpha = u$. To accommodate for the degeneracy of the noise as only the velocities are disturbed, we take $\sigma = \begin{pmatrix} 0_d & 0_d \\ 0_d & 1_d \end{pmatrix}$ where 1_d is the d-dimensional identity matrix.

The counterpart of the notion of N-agent Nash equilibrium is an MFG equilibrium. We focus here on equilibria which are stationary in time, in the γ-stationary setting described in Section 2.2.5. In other words, the goal is to find a pair $(\hat{\mu}, \hat{u})$ where $\hat{\mu} \in \Delta_{\mathbb{R}^\ell \times \mathbb{R}^\ell}$ is a position-velocity distribution and $\hat{u} : \mathbb{R}^\ell \times \mathbb{R}^\ell \mapsto \mathbb{R}^\ell$ is a feedback function to determine the acceleration given the position and velocity, such that: **(1)** $\hat{\mu}$ is an invariant position-velocity distribution if the whole population uses the acceleration given by $\hat{u}$, and **(2)** $\hat{u}$ maximizes the rewards when the agent's initial position-velocity distribution is $\hat{\mu}$ and the population distribution is $\hat{\mu}$ at every time step. In mathematical terms, $\hat{u}$ maximizes $J_{\hat{\mu}}(u) = \mathbb{E}_{x_n, v_n, u_n}\left[\sum_{n\geq 0} \gamma^n \varphi(x_n, v_n, u_n, \hat{\mu})\right]$, where $(x_n, v_n)_{n\geq 0}$ is the trajectory of an infinitesimal agent who starts with distribution $\hat{\mu}$ at time $n = 0$ and is controlled by the acceleration $(u_n)_{n\geq 0}$. As the payoff function φ we use f_β^{flock} from Equation (6.2). Moreover, the consistency condition rewrites as: $\hat{\mu}$ is the γ-stationary distribution of $(x_n, v_n)_{n\geq 0}$ if controlled by $(\hat{u}_n)_{n\geq 0}$.

Theoretical analysis. The analysis of MFG with flocking effects is challenging due to the unusual structure of the dynamics and the payoff, which encourages gathering of the population. This is running counter to the classical Lasry-Lions monotonicity condition (Lasry and Lions, 2007), which typically penalizes the agents for being too close to each other. However, existence and uniqueness have been proved in some cases. If $\beta = 0$, every agent has the same influence over the representative agent and it is possible to show that the problem reduces to a Linear-Quadratic setting. Th 2 in Nourian, Caines, and R. P. Malhamé (2011) shows that a mean-consensus in velocity is reached asymptotically with individual asymptotic variance $\frac{\sigma^2}{2}$. If $\beta > 0$, Nourian, Caines, and R. P. Malhamé (2011) shows that if the MF problem admits a unique solution, then there exists an ε_N Nash equilibrium for the N_a agents problem and $\lim_{N\to+\infty} \varepsilon_N = 0$. Existence has also been proved when $\beta > 0$ in Carmona and Delarue (2018b,

Algorithm 6.1: Flock'n RL

input : MFG = $\{(x, v), f_\beta^{\text{flock}}, m_0\}$; # of iterations K
Define $\bar{\mu}_0 = m_0$ **for** $k = 1, \dots, K$ **do**
  1. Set best response $\pi^k = \arg\max_\pi J_{\bar{\mu}^{k-1}}(\pi)$ with **SAC** and let $\bar{\pi}^k$ be the average of $(\pi^0, \dots, \pi^k)$
  2. Using a **Normalizing Flow**, compute $\mu^k = \gamma$-stationary distribution induced by π^k
  3. Using a **Normalizing Flow** and samples from $(m^0, \mu^1, \dots, \mu^{k-1})$, estimate $\bar{\mu}^k$
return $\bar{\pi}^K, \bar{\mu}^K$

Section 4.7.3), with a slight modification of the payoff, namely considering, with φ a bounded function, $r_n = -\varphi\Big(\Big\| \int_{\mathbb{R}^{2d}} \frac{(v_n - v')}{\left(1+\|x_n - x'\|^2\right)^\beta} \mu_n(dx', dv') \Big\|^2\Big)$.

6.3.2 The Flock'n RL Algorithm

We propose Flock'n RL, a deep RL version of the Fictitious Play algorithm for MFGs (Elie, Perolat, et al., 2020) adapted to flocking. We consider a γ-discounted setting with continuous state and action spaces and we adapt Algorithm 6.1 from its original tabular formulation (Perrin, Perolat, et al., 2020). It alternates 3 steps: (1) estimation of a best response (using DRL) against the mean distribution $\bar{\mu}^{k-1}$, which is fixed during the process, (2) estimation (with Normalizing Flows (Kobyzev, Prince, and Brubaker, 2020)) of the new induced distribution from trajectories generated by the previous policy, and (3) update of the mean distribution $\bar{\mu}^k$.

Computing the Best Response with SAC

The first step in the loop of Algorithm 6.1 is the computation of a best response against $\bar{\mu}^k$. In fact, the problem boils down to solving an MDP in which $\bar{\mu}^k$ enters as a parameter. We take the state and action spaces to be respectively $\mathcal{X} = \mathbb{R}^{2d}$ (for position-velocity pairs) and $\mathcal{A} = \mathbb{R}^d$ (for accelerations). Letting $s = (x, v)$ and $a = u$, the reward is: $(x, v, u) = (s, a) \mapsto r(s, a) = f(x, v, u, \bar{\mu}^k)$, which depends on the given distribution $\bar{\mu}^k$. Remember that r is the reward function of the MDP while f is the optimization criterion in the flocking model.

As we set ourselves in continuous state and action spaces and in possibly high dimensions, we need an algorithm that scales. We choose to use Soft Actor Critic (SAC) (Haarnoja et al., 2018), an off-policy actor-critic deep RL algorithm using entropy regularization. SAC is trained to maximize a trade-off between expected return and entropy, which allows to keep enough exploration during the training. It is designed to work on continuous action spaces, which makes it suited for acceleration controlled problems such as flocking.

The best response is computed against $\bar{\mu}^k$, the fixed average distribution at step k of Flock'n RL. SAC maximizes the reward which is a variant of $f_{\beta,n}^{\text{flock},i}$ from Equation (6.3). It needs samples from $\bar{\mu}^k$ in order to compute the positions and velocities of the fixed population. Note that, in order to measure more easily the progress during the learning at step j, we sample N agents from $\bar{\mu}^k$ at the beginning of step 1 (*i.e.* we do not sample new agents from $\bar{\mu}^k$ every time we need to compute the reward). During the learning, at the beginning of each episode, we sample a starting state $s_0 \sim \bar{\mu}^k$.

In the experiments, we will not need $\bar{\pi}^k$ but only the associated reward (see the exploitability metric in Section 6.4). To this end, it is enough to keep in memory the past policies $(\pi_0, \dots, \pi^k)$ and simply average the induced rewards.

Normalizing Flow for Distribution Embedding

We choose to represent the different distributions using a generative model because the continuous state space prevents us from using a tabular representation. Furthermore, even if we could choose to discretize the state space, we would need a huge amount of data points to estimate the distribution using methods such as kernel density estimators. In dimension 6 (which is the dimension of our state space with 3-dimensional positions and velocities), such methods already suffer from the curse of dimensionality.

Thus, we choose to estimate the second step of Algorithm 6.1 using a Normalizing Flow (NF) (Rezende and Mohamed, 2015; Kobyzev, Prince, and Brubaker, 2020), which is a type of generative model, different from Generative Adversarial Networks (GAN) (Goodfellow, Pouget-Abadie, et al., 2014) or Variational Autoencoders (VAE). (Kingma and Welling, 2014). A flow-based generative model is constructed by a sequence of invertible transformations and allows efficient sampling and distribution approximation. Unlike GANs and VAEs, the model explicitly learns the data distribution and therefore the loss function simply identifies to the negative log-likelihood. An NF transforms a simple distribution (*e.g.* Gaussian) into a complex one by applying a sequence of invertible transformations. In particular, a single transformation function f of noise z can be written as $x = f(z)$ where $z \sim h(z)$. Here, $h(z)$ is the noise distribution and will often be in practice a normal distribution.

Using the change of variable theorem, the probability density of x under the flow can be written as: $p(x) = h(f^{-1}(x)) \left| \det \left(\frac{\partial f^{-1}}{\partial x} \right) \right|$. We thus obtain the probability distribution of the final target variable. In practice, the transformations f and f^{-1} can be approximated by neural networks. Thus, given a dataset of observations (in our case rollouts from the current best response), the flow is trained by maximizing the total log likelihood $\sum_n \log p(x^{(n)})$.

Computation of $\bar{\mu}^k$

Due to the above discussion on the difficulty to represent the distribution in continuous space and high dimension, the third step (Line 5 of Algorithm 6.1) can not be implemented easily. We represent every μ^k as a generative model, so we can not "average" the normalizing flows corresponding to $(\mu^i)_{i=1,\dots,k}$ in a straightforward way but we can sample data points $x \sim \mu^i$ for each $i = 1, \dots, k$. To have access to $\bar{\mu}^k$, we keep in memory every model μ^k, $k \in \{1, \dots, K\}$ and, in order to sample points according to $\bar{\mu}^k$ for a fixed k, we sample points from $\mu^i, i \in \{1, \dots, k\}$, with probability $1/k$. These points are then used to learn the distribution $\bar{\mu}^k$ with an NF, as it is needed both for the reward and to sample the starting state of an agent during the process of learning a best response policy.

6.4 Experiments

Environment. We implemented the environment as a custom OpenAI gym environment (Brockman et al., 2016) to benefit from the powerful gym framework and use the algorithms available in stable baselines (Hill et al., 2018). We define a state $s \in S$ as $s = (x, v)$ where x and v are respectively the vectors of positions and velocities. Each coordinate x_i of the position can take any continuous value in the d-dimensional box $x_i \in [-100, +100]$, while the velocities are also continuous and clipped $v_i \in [-1, 1]$. The state space for the positions is a torus, meaning that an agent reaching the box limit reappears at the other side of the box. We chose this setting to allow the agents to perfectly align their velocities (except for the effect of the noise), as we look for a stationary solution.

At the beginning of each iteration k of Fictitious Play, we initialize a new gym environment with the current mean distribution $\bar{\mu}^k$, in order to compute the best response.

Model - Normalizing Flows. To model distributions, we use Neural Spline Flows (NSF) with a coupling layer (Durkan et al., 2019). More details about how coupling layers and NSF work can be found in Section D.2.

Model - SAC. To compute the best response at each Flock'n RL iteration, we use Soft Actor Critic (SAC) (Haarnoja et al., 2018) (but other Policy Gradient algorithms would work). SAC is an off-policy algorithm which, as mentioned above, uses the key idea of regularization: instead of considering the objective to simply be the sum of rewards, an entropy term is added to encourage sufficient randomization of the policy and thus address the exploration-exploitation trade-off. To be specific, in our setting, given a population distribution μ, the objective is to

maximize: $J_\mu(\pi) = \mathbb{E}_{(s_n,u_n)}\left[\sum_{n=0}^{+\infty}\gamma^n r(x_n,v_n,u_n,\mu_n) + \delta\mathcal{H}(\pi(\cdot|s_n))\right]$, where $\mathcal{H}$ denotes the entropy and $\delta \geq 0$ is a weight.

To implement the optimization, the SAC algorithm follows the philosophy of actor-critic by training parameterized Q-function and policy. To help convergence, the authors of SAC also train a parameterized value function V. In practice, the three functions are often approximated by neural networks.

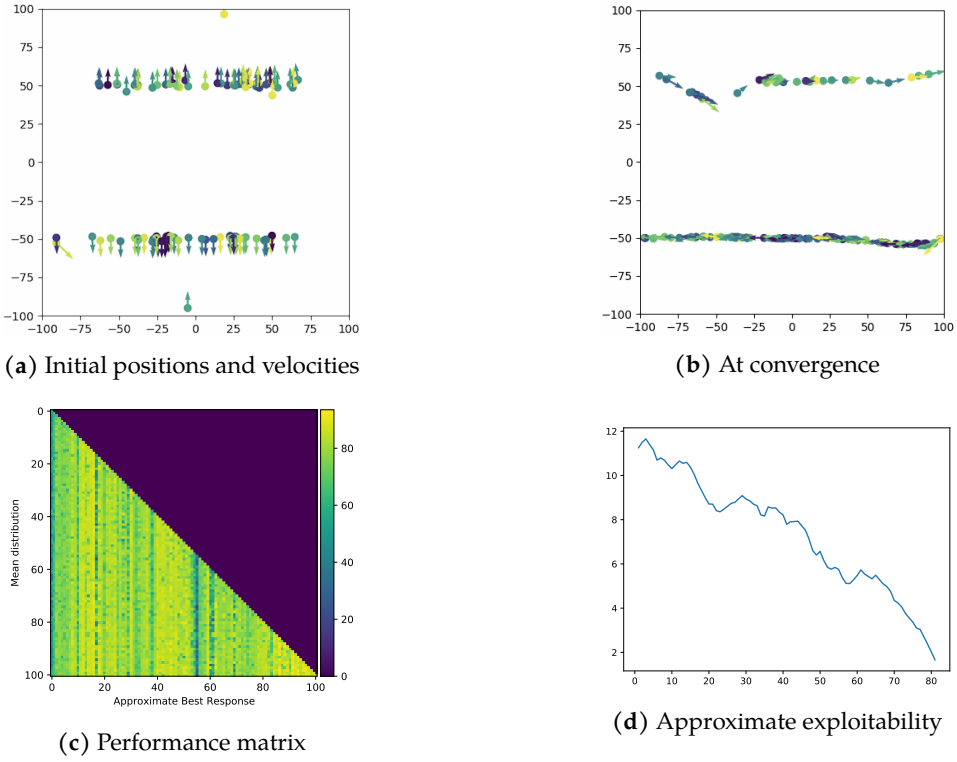
In comparison to other successful methods such as Trust Region Policy Optimization (TRPO) (Schulman et al., 2015) or Asynchronous Actor-Critic Agents (A3C), SAC is expected to be more efficient in terms of number of samples required to learn the policy thanks to the use of a replay buffer in the spirit of methods such as Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016).

Metrics. An issue with studying our flocking model is the absence of a gold standard. Especially, we can not compute the exact exploitability (Perrin, Perolat, et al., 2020) of a policy against a given distribution since we can not compute the exact best response. The exploitability measures how much an agent can gain by replacing its policy π with a best response π', when the rest of the population plays π: $\phi(\pi) := \max_{\pi'} J(m_0,\pi',\mu^\pi) - J(m_0,\pi,\mu^\pi)$. If $\phi(\bar{\pi}^k) \to 0$ as the number of iterations k of the algorithm increases, FP approaches a Nash equilibrium. To cope with these issues, we introduce the following ways to measure progress of the algorithm:

- *Performance matrix:* we build the matrix $\mathcal{M}$ of performance of learned policies versus estimated distributions. The entry $\mathcal{M}_{i,j}$ on the i-th row and the j-th column is the total γ-discounted sum of rewards: $\mathcal{M}_{i,j} = \mathbb{E}\left[\sum_{n=0}^{N_T}\gamma^n r_{n,i} \mid s_0 \sim \bar{\mu}^{i-1},\ u_n \sim \pi^k(.|s_n)\right]$, where $r_{n,i} = r(s_n,u_n,\bar{\mu}^{i-1})$, obtained with π^k against $\bar{\mu}^{i-1}$. The diagonal term $\mathcal{M}_{j,j}$ corresponds to the value of the best response computed at iteration j.

- *Approximate exploitability:* We do not have access to the exact best response due to the model-free approach and the continuous spaces. However, we can approximate the first term of $\phi(\bar{\pi})$ directly in the Flock'n RL algorithm with SAC. The second term, $J(m_0,\bar{\pi},\mu^{\bar{\pi}})$, can be approximated by replacing $\bar{\pi}$ with the average over past policies, *i.e.*, the policy sampled uniformly from the set $\{\pi_0,\dots,\pi^k\}$. At step j, the approximate exploitability is $e_j = \mathcal{M}_{j,j} - \frac{1}{j-1}\sum_{k=1}^{j-1}\mathcal{M}_{j,k}$. To smoothen the exploitability, we take the best response over the last 5 policies and use a moving average over 10 points. Please note that only relative values are important as it depends on the scale of the reward.

A 4-Dimensional Example. We illustrate in a four dimensional setting (*i.e.* two-dimensional positions and velocities) how the agents learn to adopt similar velocities by controlling their acceleration. We focus on the role of β in the flocking effect. We consider noise $\varepsilon_n^i \sim \mathcal{N}(0,\Delta n)$ and the following reward: $r_n^i = f_{\beta,n}^{\text{flock},i} - \|u_n^i\|_2^2 + \|v_n^i\|_\infty - \min\{\|x_{2,n}^i \pm 50\|\}$, where $x_{2,n}^i$ stands



(**a**) Initial positions and velocities

(**b**) At convergence

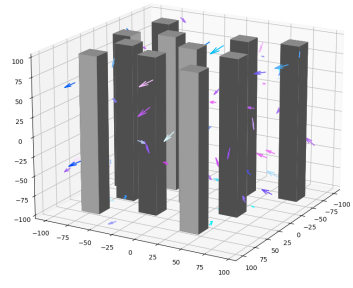
(**c**) Performance matrix

(**d**) Approximate exploitability

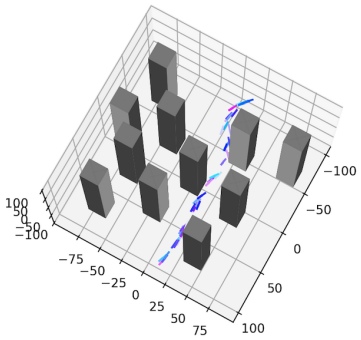
Figure 6.1 – Multi-group flocking with noise and $\beta = 100$.

for the second coordinate of the i-th agent's position at time t. The last term attracts the agents' positions towards one of two lines corresponding to the second coordinate of x being either -50 or $+50$. We added a term regarding the norm of the velocity to prevent agents from stopping. Here we take $\|v_n^i\|_\infty = \max\{|v_{1,n}^i|, |v_{2,n}^i|\}$. Hence, a possible equilibrium is with two groups of agents, one for each line. When $\beta = 0$, the term $f_{\beta,n}^{\text{flock},i}$ encourages agent i to have the same velocity vector as the rest of the whole population. At equilibrium, the agents in the two groups should thus move in the same direction (to the left or to the right, in order to stay on the two lines of x's). On the other hand, when $\beta > 0$ is large enough (*e.g.* $\beta = 100$), agent i gives more importance to its neighbors when choosing its control and it tries to have a velocity similar to the agents that are position-wise close to. This allows the emergence of two groups moving in different directions: one group moves towards the left (overall negative velocity) and the other group moves towards the right (overall positive velocity).

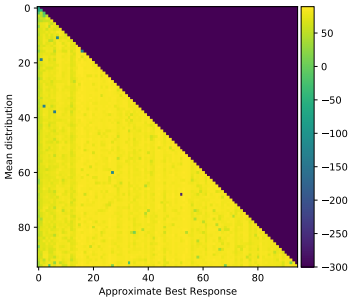
This is confirmed by Figure 6.1. In the experiment, we set the initial velocities perpendicular to the desired ones to illustrate the robustness of the algorithm. We observe that the approximate exploitability globally decreases. In the case $\beta = 0$, we experimentally verified that there is always a global consensus, *i.e.*, only one line or two lines but moving in the same direction.



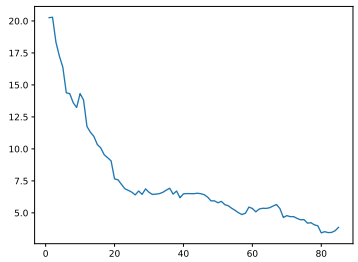
(**a**) Initial positions and velocities



(**b**) At convergence



(**c**) Performance matrix



(**d**) Approximate exploitability

Figure 6.2 – Flocking with noise and many obstacles.

Scaling to 6 Dimensions and non-smooth topology. We now present an example with arbitrary obstacles (and thus non-smooth topology) in dimension 6 (position and velocity in dimension 3) which would be very hard to address with classical numerical methods. In this setting, we have multiple columns that the agents are trying to avoid. The reward has the following form: $r_n^i = f_{\beta,n}^{\text{flock},i} - \|u_n^i\|_2^2 + \|v_n^i\|_\infty - \min\{\|x_{2,n}^i\|\} - c * \mathbf{1}_{obs}$. If an agent hits an obstacle, it gets a negative reward and bounces on it like a snooker ball. After a few iterations, the agents finally find their way through the obstacles. This situation can model birds trying to fly in a city with tall buildings. In our experiments, we noticed that different random seeds lead to different solutions. This is not surprising as there are a lot of paths that the agents can take to avoid the obstacles and still maximizing the reward function. The exploitability decreases quicker than in the previous experiment. We believe that this is because agents find a way through the obstacles in the first iterations.

6.5 Conclusion of the Chapter

In this work we introduced Flock'n RL, a new numerical approach which allows solving MFGs with flocking effects where the agents reach a consensus in a decentralized fashion. Flock'n RL combines Fictitious Play with deep neural networks and reinforcement learning

techniques (normalizing flows and soft actor-critic). We illustrated the method on challenging examples, for which no solution was previously known. In the absence of existing benchmark, we demonstrated the success of the method using a new kind of approximate exploitability. Thanks to the efficient representation of the distribution and to the model-free computation of a best response, the techniques developed here could be used to solve other acceleration controlled MFGs (Achdou, Mannucci, et al., 2020) or, more generally, other high-dimensional MFGs. Last, the flexibility of RL, which does not require a perfect knowledge of the model, allow us to tackle MFGs with complex topologies (such as boundary conditions or obstacles), which is a difficult problem for traditional methods based on partial differential equations. However, we restricted ourselves to the γ-stationary setting, which as we have seen has limited applicability. Furthermore, our resulting policy is only applicable to the initial distribution considered during training, which is counter-intuitive from the reinforcement learning perspective.

Chapter 7

Generalization in Mean Field Games

In this chapter, we make a first step towards generalization in Mean Field Games. Mean Field Games (MFGs) can potentially scale multi-agent systems to extremely large populations of agents. Yet, most of the literature assumes a single initial distribution for the agents, which limits the practical applications of MFGs. Machine Learning has the potential to solve a wider diversity of MFG problems thanks to generalizations capacities. Thus, we study how to leverage these generalization properties to learn policies enabling a typical agent to behave optimally against any population distribution. In reference to the Master equation in MFGs, we coin the term "Master policies" to describe them and we prove that a single Master policy provides a Nash equilibrium, whatever the initial distribution. We propose a method to learn such Master policies. Our approach relies on three ingredients: adding the current population distribution as part of the observation, approximating Master policies with neural networks, and training via Reinforcement Learning and Fictitious Play. We illustrate on numerical examples not only the efficiency of the learned Master policy but also its generalization capabilities beyond the distributions used for training. This chapter directly extends the notion of *population-dependent policies* discussed in section 2.2.8. [1]

7.1 Motivation

As in many multi-player games, solving an MFG boils down to finding a Nash equilibrium. Intuitively, it corresponds to a situation where no player can increase their reward (or decrease their cost) by changing their strategy, given that other players keep their current behavior. MFGs are classically described with a forward-backward system of partial differential equations

[1] This chapter is based on a preprint (Perrin, Laurière, Pérolat, Élie, et al., 2021) presented at the AAAI 2022 conference.

(PDEs) or stochastic differential equations (SDEs) and can only be solved analytically in some specific cases. When an analytical solution is not available, numerical methods such as finite differences can be called to solve the PDE system. However, these techniques do not scale well with the dimensions of the state and action spaces. Another issue with PDE methods is their sensitivity to initial conditions. Especially, the policy obtained is only valid for a single initial distribution m_0 for the population over the state space. This is a strong limitation for practical applications. For example, in an evacuation or traffic-flow scenario, the solution found by a PDE solver could potentially lead to an unforeseen congestion if the agents are not initially distributed as the model expected. This could have dramatic consequences. On the other hand, solving for every possible initial distribution is of course infeasible. Following the traditional trend in the literature, even solutions to MFGs that use most recent Machine Learning methods consider that the initial distribution is fixed and thus compute policies that are agnostic to the current population. A sensible idea to alleviate the sensitivity issue is to incorporate the population as part of the observation for the representative agent, such that it can behave optimally against the population, and not only with respect to its current state. Yet, using such a modification of the observation cannot be done seamlessly as the uniqueness of the initial distribution is a core assumption of existing methods, including very recent ones based on Machine Learning.

Here we do a first crucial step in this direction using Deep Reinforcement Learning (DRL), which sounds particularly well fitted to overcome the aforementioned difficulty. Our core contribution is to propose the first DRL algorithm that calculates an optimal policy independently of the initial population distribution.

Main contributions. First, we extend the basic framework of MFGs by introducing a class of *population-dependent policies* enabling agents to react to any population distribution. Within this class, we identify a *Master policy* and establish its connection with standard population-agnostic policies arising in MFG Nash equilibria (Theorem 7.1). Second, we propose an algorithm, based on Fictitious Play and DRL, to learn a Master policy. We analyze a continuous time version of Fictitious Play and prove convergence at a linear rate (Theorem 7.2). Last, we provide empirical evidence that not only this method learns the Master policy on a training set of distributions, but that the learned policy *generalizes* to unseen distributions. Our approach is the first to tackle this question in the literature on MFGs.

7.2 Background and Related Works

In this section, we introduce the key concepts needed to explain our main contributions. We will use the same formalism than in the previous chapters but we write explicitly the dependency

in the initial distribution m_0. This will be important for introducing the concept of population-dependent policies and Master policies in the sequel.

Although there is no prior work tackling explicitly the question of generalization in MFG, we review along the way several related studies.

7.2.1 Mean Field Games

In the usual MFG setup (Lasry and Lions, 2007; M. Huang, R. P. Malhamé, and Caines, 2006), a stationary policy is a function $\pi : \mathcal{X} \to \Delta_{\mathcal{A}}$ and a non-stationary policy $\boldsymbol{\pi}$ is an infinite sequence of stationary policies. Let Π and $\mathbf{\Pi} = \Pi^{\mathbb{N}}$ be the sets of stationary and non-stationary policies respectively. Unless otherwise specified, by policy we mean a non-stationary policy. A mean-field (MF) state is a $\mu \in \Delta_{\mathcal{X}}$. It represents the state of the population at one time step. An MF flow $\boldsymbol{\mu}$ is an infinite sequence of MF states. We denote by $\mathcal{M} = \Delta_{\mathcal{X}}$ and $\mathbf{M} = \mathcal{M}^{\mathbb{N}}$ the sets of MF states and MF flows. For $\mu \in \mathcal{M}$, $\pi \in \Pi$, let

$$\psi(\mu, \pi) : x \mapsto \sum_{x' \in \mathcal{X}} p(x|x', \pi(x'), \mu)\mu(x')$$

denote the next MF state. The MF flow starting from m_0 and controlled by $\boldsymbol{\pi} \in \mathbf{\Pi}$ is denoted by $\Phi(m_0, \boldsymbol{\pi}) \in \mathbf{M}$:

$$\Phi(m_0, \boldsymbol{\pi})_0 = m_0, \;\; \Phi(m_0, \boldsymbol{\pi})_{n+1} = \psi(\Phi(m_0, \boldsymbol{\pi})_n, \boldsymbol{\pi}_n), n \geq 0.$$

Facing such a population behavior, an infinitesimal agent seeks to solve the following Markov Decision Process (MDP). Given an initial m_0 and a flow $\boldsymbol{\mu}$, maximize:

$$\boldsymbol{\pi} \mapsto J(m_0, \boldsymbol{\pi}; \boldsymbol{\mu}) = \mathbb{E}\Big[\sum_{n=0}^{+\infty} \gamma^n r(x_n, a_n, \boldsymbol{\mu}_n)\Big],$$

subject to: $x_0 \sim m_0$, $x_{n+1} \sim p(.|x_n, a_n, \boldsymbol{\mu}_n)$, $a_n \sim \boldsymbol{\pi}_n(.|x_n)$. Note that, at time n, the reward and transition depend on the current MF state $\boldsymbol{\mu}_n$. So this MDP is non-stationary but since the MF flow $\boldsymbol{\mu}$ is fixed and given, it is an MDP in the classical sense. In an MFG, we look for an equilibrium situation, in which the population follows a policy from which no individual player is interested in deviating.

Definition 6 (MFG Nash equilibrium)**.** *Given $m_0 \in \mathcal{M}$, $(\hat{\boldsymbol{\pi}}^{m_0}, \hat{\boldsymbol{\mu}}^{m_0}) \in \mathbf{\Pi} \times \mathbf{M}$ is an MFG Nash equilibrium (MFG-NE) consistent with m_0 if: (1) $\hat{\boldsymbol{\pi}}^{m_0}$ maximizes $J(m_0, \cdot; \hat{\boldsymbol{\mu}}^{m_0})$, and (2) $\hat{\boldsymbol{\mu}}^{m_0} = \Phi(m_0, \hat{\boldsymbol{\pi}}^{m_0})$.*

Being an MFG-NE amounts to say that the *exploitability* $\phi(m_0, \hat{\boldsymbol{\pi}}^{m_0})$ is 0, where the exploitability of a policy $\boldsymbol{\pi} \in \boldsymbol{\Pi}$ given the initial MF state m_0 is defined as:

$$\phi(m_0, \boldsymbol{\pi}) = \max_{\boldsymbol{\pi}'} J(m_0, \boldsymbol{\pi}'; \Phi(m_0, \boldsymbol{\pi})) - J(m_0, \boldsymbol{\pi}; \Phi(m_0, \boldsymbol{\pi})).$$

It quantifies how much a representative player can be better off by deciding to play another policy than $\boldsymbol{\pi}$ when the rest of the population uses $\boldsymbol{\pi}$ and the initial distribution is m_0 for both the player and the population. Similar notions are widely used in computational game theory (Zinkevich et al., 2007; Lanctot, Waugh, et al., 2009).

In general, $\hat{\boldsymbol{\pi}}^{m_0}$ is not an MFG-NE policy consistent with $m_0' \neq m_0$. Imagine for example a game in which the agents need to spread uniformly throughout a one-dimensional domain (see the experimental section). Intuitively, the movement of an agent at the center depends on where the bulk of the population is. If m_0 is concentrated on the left (resp. right) side, this agent should move towards the right (resp. left). Hence the optimal policy depends on the whole population distribution.

Equilibria in MFG are traditionally characterized by a forward-backward system of equations (Lasry and Lions, 2007; Carmona and Delarue, 2018b). Indeed, the value function of an individual player facing an MF flow $\boldsymbol{\mu}$ is:

$$V_n(x; \boldsymbol{\mu}) = \sup_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} \mathbb{E}_{x, \boldsymbol{\pi}} \Big[\sum_{n'=n}^{+\infty} \gamma^{n'-n} r(x_{n'}, a_{n'}, \boldsymbol{\mu}_{n'}) \Big],$$

where $x_n = x$ and $a_{n'} \sim \boldsymbol{\pi}_{n'}(\cdot | x_{n'})$, $n' \geq n$. Dynamic programming yields:

$$V_n(x; \boldsymbol{\mu}) = \sup_{\pi \in \Pi} \mathbb{E}_{x, \pi} \Big[r(x_n, a_n, \boldsymbol{\mu}_n) + \gamma V_{n+1}(x'; \boldsymbol{\mu}) \Big],$$

where $x_n = x$, $a_n \sim \pi(\cdot | x)$ and $x' \sim p(\cdot | x, a, \boldsymbol{\mu}_n)$. Taking the maximizer gives an optimal policy for a player facing $\boldsymbol{\mu}$. To find an equilibrium policy, we replace $\boldsymbol{\mu}$ by the equilibrium MF flow $\hat{\boldsymbol{\mu}}$: $\hat{V}_n(\cdot) = V_n(\cdot; \hat{\boldsymbol{\mu}})$. But $\hat{\boldsymbol{\mu}}$ is found by using the corresponding equilibrium policy. This induces a coupling between the backward equation for the representative player and the forward population dynamics.

The starting point of our Master policy approach is to notice that $V_n(\cdot; \boldsymbol{\mu})$ depends on n and $\boldsymbol{\mu}$ only through $(\boldsymbol{\mu}_{n'})_{n' \geq n}$ hence V_n depends on n only through $(\boldsymbol{\mu}_{n'})_{n' \geq n}$:

$$V_n(x; \boldsymbol{\mu}) = V(x; (\boldsymbol{\mu}_{n'})_{n' \geq n})$$

where, for $\boldsymbol{\mu} \in \mathbf{M}, x \in \mathcal{X}$,

$$V(x; \boldsymbol{\mu}) = \sup_{\pi \in \Pi} \mathbb{E}_{x, \pi} \Big[r(x, a, m_0) + \gamma V(x'; (\boldsymbol{\mu}_n)_{n \geq 1}) \Big], \tag{7.1}$$

where $a \sim \pi(\cdot|x)$ and $x' \sim p(\cdot|x, a, m_0)$.

From here, we will express the equilibrium policy $\hat{\boldsymbol{\pi}}_n$ as a stationary policy (independent of n) which takes $\hat{\boldsymbol{\mu}}_n$ as an extra input. Replacing n by $\hat{\boldsymbol{\mu}}_n$ increases the input size but it opens new possibilities in terms of *generalization in MFGs*.

7.2.2 Learning and Generalization in MFGs

For background on iterative methods for MFGs, we refer the reader to previous chapters. We review here the other parts where learning can occur when solving MFGs with RL.

Reinforcement learning subroutine. For a given population distribution, to update the representative player's policy or value function, we can rely on RL techniques. For instance Guo, A. Hu, et al. (2019) and Anahtarcı, Karıksız, and Saldi (2020a) rely on Q-learning to approximate the Q-function in a tabular setting, Fu et al. (2019) study an actor-critic method in a linear-quadratic setting, and Elie, Perolat, et al. (2020) and Perrin, Laurière, Pérolat, Geist, et al. (2021) solve continuous spaces problems by relying respectively on deep deterministic policy gradient (Lillicrap et al., 2016) or soft actor-critic (Haarnoja et al., 2018). Two time-scales combined with policy gradient has been studied by J. Subramanian and Mahajan (2019) for stationary MFGs. Policy iterations together with sequential decomposition has been proposed by Mishra, Vasal, and Vishwanath (2020) while Guo, A. Hu, et al. (2020) proposes a method relying on Trust Region Policy Optimization (TRPO, Schulman et al. (2015)).

Distribution embedding. Another layer of complexity in MFGs is to take into consideration population distributions for large spaces or even continuous spaces. To compute MFG solutions through a PDE approach, Al-Aradi et al. (2018) and Carmona and Laurière (2021) used deep neural networks to approximate the population density in high dimension. In the context of RL for MFGs, recently, Perrin, Laurière, Pérolat, Geist, et al. (2021) have used Normalizing Flows (Rezende and Mohamed, 2015) to approximate probability measures over continuous state space in complex environments.

So far, learning approaches for MFGs have considered only two aspects: optimization algorithms (*e.g.*, Fictitious Play or Online Mirror Descent), or model-free learning of a representative player's best response based on samples (*e.g.*, Q-learning or actor-critic methods). Here, we build upon the aforementioned notions and add to this picture another dimension of learning: *generalization* over population distributions. We develop an approach to learn the representative player's best response as a function of any current population distribution and not only the ones corresponding to a fixed MFG-NE. This is tightly connected with the so-called Master equation in MFGs (Lions, 2012; Bensoussan, Frehse, and S. C. P. Yam, 2015;

Cardaliaguet, Delarue, et al., 2019). Introduced in the continuous setting (continuous time, state and action), this equation is a partial differential equation (PDE) which corresponds to the limit of systems of Hamilton-Jacobi-Bellman PDEs characterizing Nash equilibria in symmetric N-player games. In our discrete context, we introduce a notion of Master Bellman equation and associated Master policy, which we then aim to compute with a new learning algorithm based on Fictitious Play. To the best of our knowledge, the literature focuses on Master equations on a finite horizon. Here, we consider infinite horizon discounted problems, which allows us to look for stationary solutions.

7.3 Master Policies for MFGs

We introduce the notion of Master policy and connect it to standard population-agnostic policies arising in MFG-NE.

Consider an MFG-NE $(\hat{\boldsymbol{\pi}}^{m_0}, \hat{\boldsymbol{\mu}}^{m_0})$ consistent with some m_0. Let $\hat{V}(\cdot; m_0) = V(\cdot; \hat{\boldsymbol{\mu}}^{m_0})$, *i.e.*,

$$\hat{V}(x; m_0) = \sup_{\pi \in \Pi} \mathbb{E}_\pi \Big[r(x, a, m_0) + \gamma V(x'; (\hat{\boldsymbol{\mu}}_n^{m_0})_{n \geq 1}) \Big],$$

where $a \sim \pi(\cdot|x, m_0)$ and $x' \sim p(\cdot|x, a, m_0)$. By definition, $\hat{\boldsymbol{\pi}}_0^{m_0}$ is a maximizer in the sup above. Moreover, in the right-hand side,

$$V(x'; (\hat{\boldsymbol{\mu}}_n^{m_0})_{n \geq 1}) = \hat{V}(x'; \hat{\boldsymbol{\mu}}_1^{m_0}), \qquad \hat{\boldsymbol{\mu}}_1^{m_0} = \psi(m_0, \hat{\boldsymbol{\pi}}_0^{m_0}).$$

By induction, the equilibrium can be characterized as:

$$\begin{cases} \hat{\boldsymbol{\pi}}_n^{m_0} \in \arg\max_{\pi \in \Pi} \mathbb{E}_\pi \Big[r(x, a, \hat{\boldsymbol{\mu}}_n^{m_0}) + \gamma \hat{V}(x'; \hat{\boldsymbol{\mu}}_{n+1}^{m_0}) \Big] \\ \hat{V}(x; \hat{\boldsymbol{\mu}}_n^{m_0}) = \mathbb{E}_{\hat{\boldsymbol{\pi}}_n^{m_0}} \Big[r(x, a, \hat{\boldsymbol{\mu}}_n^{m_0}) + \gamma \hat{V}(x'; \hat{\boldsymbol{\mu}}_{n+1}^{m_0}) \Big] \\ \hat{\boldsymbol{\mu}}_{n+1}^{m_0} = \psi(\hat{\boldsymbol{\mu}}_n^{m_0}, \hat{\boldsymbol{\pi}}_n^{m_0}). \end{cases}$$

Note that $\hat{\boldsymbol{\mu}}_{n+1}^{m_0}$ and $\hat{\boldsymbol{\pi}}_n^{m_0}$ depend on each other (and also on m_0), which creates a forward-backward structure.

In the sequel, we will refer to this function V as the *Master value function*. Computing the value function $(x, \mu) \mapsto \hat{V}(x; \mu)$ would allow us to know the value of any individual state x facing an MFG-NE starting from any MF state μ. However, it would not allow to easily find the corresponding equilibrium policy, which still depends implicitly on the equilibrium MF flow. For this reason, we introduce the notion of population-dependent policy. The set of population-dependent policies $\tilde{\pi} : \mathcal{X} \times \Delta_{\mathcal{X}} \to \Delta_{\mathcal{A}}$ is denoted by $\widetilde{\Pi}$.

Definition 7. *A population-dependent $\widetilde{\pi}^* \in \widetilde{\Pi}$ is a Master policy if for every m_0, $(\boldsymbol{\pi}^{m_0,\widetilde{\pi}^*}, \boldsymbol{\mu}^{m_0,\widetilde{\pi}^*})$ is an MFG-NE, where: $\boldsymbol{\mu}_0^{m_0,\widetilde{\pi}^*} = m_0$ and for $n \geq 0$,*

$$\begin{cases} \boldsymbol{\pi}_n^{m_0,\widetilde{\pi}^*}(x) = \widetilde{\pi}^*(x, \boldsymbol{\mu}_n^{m_0,\widetilde{\pi}^*}) \\ \boldsymbol{\mu}_{n+1}^{m_0,\widetilde{\pi}^*} = \psi(\boldsymbol{\mu}_n^{m_0,\widetilde{\pi}^*}, \boldsymbol{\pi}_n^{m_0,\widetilde{\pi}^*}). \end{cases} \tag{7.2}$$

A Master policy allows recovering the MFG-NE starting from any initial MF state. A core question is the existence of such a policy, which we prove in Theorem 7.1 below. Hence, if there is a unique Nash equilibrium MF flow (*e.g.*, thanks to monotonicity), the MF flow $\boldsymbol{\mu}^{m_0,\widetilde{\pi}^*}$ obtained with the Master policy $\widetilde{\pi}^*(a|x, \boldsymbol{\mu}_n^{m_0,\widetilde{\pi}^*})$ is the same as the one obtained with a best response policy $\hat{\boldsymbol{\pi}}_n^{m_0}(a|x)$ starting from m_0.

Theorem 7.1. *Assume that, for all $m_0 \in \mathcal{M}$, the MFG admits an equilibrium consistent with m_0 and that the equilibrium MF flow is unique. Then there exists a Master policy $\widetilde{\pi}^*$.*

Existence and uniqueness of the MFG-NE for a given m_0 can be proved under a mild monotonicity condition (Perrin, Perolat, et al., 2020). Theorem 7.1 is proved by checking, step by step, that the MF flow generated by $\widetilde{\pi}^*$ and the associated population-agnostic policy as defined in (7.2) form a MFG-NE. The key idea is to use dynamic programming relying on the Master value function V and the uniqueness of the associated equilibrium MF flow. We omit the details for brevity.

7.4 Algorithm

We have demonstrated above that the Master policy is well-defined and allows to recover Nash equilibria. We now propose a method to compute such a policy.

7.4.1 Fictitious Play

We introduce an adaptation of the Fictitious Play algorithm to learn a Master policy. This extends to the case of population-dependent policies the algorithm introduced by Cardaliaguet and Hadikhanloo (2017). In the same fashion, at every iteration k, it alternates three steps: (1) computing a best response policy $\widetilde{\pi}_k$ against the current averaged MF flows $\bar{\mathcal{M}}_k$, (2) computing $(\boldsymbol{\mu}^{m_0,\widetilde{\pi}_k})_{m_0 \in \mathcal{M}}$, the MF flows induced by $\widetilde{\pi}_k$, and (3) updating $\bar{\mathcal{M}}_{k+1}$ with $(\boldsymbol{\mu}^{m_0,\widetilde{\pi}_k})_{m_0 \in \mathcal{M}}$.

We choose Fictitious Play rather than a simpler fixed-point approach because it is generally easier to check that an MFG model satisfies the assumptions used to prove convergence (monotonicity condition rather than contraction properties, as *e.g.* in M. Huang, R. P. Malhamé, and Caines (2006) and Guo, A. Hu, et al. (2019)).

Ideally, we would like to train the population-dependent policy on every possible distributions, but this is not feasible. Thus, we take a finite training set $\mathcal{M}$ of initial distributions. Each training distribution is used at each iteration of Fictitious Play. Another possibility would have been to swap these two loops, but we chose not to do this because of *catastrophic forgetting* (French, 1999; Goodfellow, Mirza, et al., 2014), a well-know phenomenon in cognitive science that also occurs in neural networks, describing the tendency to forget previous information when learning new information. Our proposed algorithm is summarized in Algorithm 7.1 and we refer to it as *Master Fictitious Play*.

Algorithm 7.1: Master Fictitious Play

input : Initial $\widetilde{\pi}_0 \in \widetilde{\Pi}$, training set of initial distributions $\mathcal{M}$, number of Fictitious Play steps K
Let $\bar{\pi} = \widetilde{\pi}_0$; let $\bar{\boldsymbol{\mu}}_{0,n}^{m_0} = m_0$ for all $m_0 \in \mathcal{M}$, all $n \geq 0$
Let $\bar{\mathcal{M}}_0 = (\bar{\boldsymbol{\mu}}_0^{m_0})_{m_0 \in \mathcal{M}}$
for $k = 1, \ldots, K$ **do**
  Train $\widetilde{\pi}_k$ against $\bar{\mathcal{M}}_k = (\bar{\boldsymbol{\mu}}_k^{m_0})_{m_0 \in \mathcal{M}}$, to maximize Eq. (7.4)
  for $m_0 \in \mathcal{M}$ **do**
    Compute $\boldsymbol{\mu}_k^{m_0}$, the MF flow starting from m_0 induced by $\widetilde{\pi}_k$ against $\bar{\boldsymbol{\mu}}_k^{m_0}$
    Let $\bar{\boldsymbol{\mu}}_k^{m_0} = \frac{k}{k+1}\bar{\boldsymbol{\mu}}_{k-1}^{m_0} + \frac{1}{k+1}\boldsymbol{\mu}_k^{m_0}$
  end
  Update $\bar{\widetilde{\pi}}_k = \text{UNIFORM}(\widetilde{\pi}_0, \ldots, \widetilde{\pi}_k)$
end
return $\bar{\widetilde{\pi}}_K = \text{UNIFORM}(\widetilde{\pi}_0, \ldots, \widetilde{\pi}_K)$

Algorithm 7.1 returns $\bar{\widetilde{\pi}}_K$, which is the uniform distribution over past policies. We use it as follows. First, let: $\boldsymbol{\mu}_{k,0}^{m_0} = m_0$, $k = 1, \ldots, K$, $\bar{\boldsymbol{\mu}}_{K,0}^{m_0} = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{\mu}_{k,0}^{m_0}$, and then, for $n \geq 0$,

$$\begin{cases} \boldsymbol{\mu}_{k,n+1}^{m_0} = \psi(\boldsymbol{\mu}_{k,n}^{m_0}, \widetilde{\pi}_k(\cdot|\cdot, \bar{\boldsymbol{\mu}}_{K,n}^{m_0})), & k = 1, \ldots, K \\ \bar{\boldsymbol{\mu}}_{K,n+1}^{m_0} = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{\mu}_{k,n+1}^{m_0}. \end{cases}$$

Note that $\bar{\widetilde{\pi}}_K$ is used in the same way for every m_0. We will show numerically that this average distribution and the associated average reward are close to the equilibrium ones.

Define the average exploitability as:

$$\bar{\phi}_{\mathcal{M}}(\bar{\widetilde{\pi}}_K) = \mathbb{E}_{m_0 \sim \text{UNIFORM}(\mathcal{M})}\left[\bar{\psi}(m_0, \bar{\widetilde{\pi}}_K)\right], \tag{7.3}$$

where

$$\bar{\phi}(m_0, \bar{\widetilde{\pi}}_K) = \max_{\boldsymbol{\pi}'} J(m_0, \boldsymbol{\pi}'; \bar{\boldsymbol{\mu}}_K^{m_0}) - \frac{1}{K}\sum_{k=1}^{K} J(m_0, \widetilde{\pi}_k; \bar{\boldsymbol{\mu}}_K^{m_0}).$$

We expect $\bar{\phi}(m_0, \bar{\widetilde{\pi}}_K) \to 0$ as $K \to +\infty$. We show that this indeed holds under suitable conditions in the idealized setting with continuous time updates, where $\bar{\widetilde{\pi}}_k$, $k = 0, 1, 2, \dots$, is replaced by $\bar{\widetilde{\pi}}_t$, $t \in [0, +\infty)$.

Theorem 7.2. *Assume the reward is separable and monotone, i.e., $r(x, a, \mu) = r_A(x, a) + r_M(x, \mu)$ and $\sum_{x \in \mathcal{X}} (r_M(x, \mu) - r_M(x, \mu'))(\mu - \mu')(x) < 0$ for every $\mu \neq \mu'$. Assume the transition depends only on x and a: $p(\cdot|x, a, \mu) = p(\cdot|x, a)$. Then $\bar{\phi}_{\mathcal{M}}(\bar{\widetilde{\pi}}_t) = O(1/t)$, where $\bar{\widetilde{\pi}}_t$ is the average policy at time t in the continuous time version of Master Fictitious Play.*

The proof follows the lines of Perrin, Perolat, et al. (2020) adapted to our setting and is omitted for the sake of brevity. Studying continuous time updates instead of discrete ones enables us to use calculus, which leads to a simple proof. To the best of our knowledge, there is no rate of convergence for discrete time Fictitious Play in the context of MFG except for potential or linear-quadratic structures (Geist, Pérolat, et al., 2021; Delarue and Vasileiadis, 2021).

7.4.2 DRL to Learn a Population-dependent Policy

In Algorithm 7.1, a crucial step is to learn a population-dependent best response against the current averaged MF flows $\bar{\mathcal{M}}_k = (\bar{\boldsymbol{\mu}}_k^{m_0})_{m_0 \in \mathcal{M}}$, *i.e.*, $\widetilde{\pi}_k^*$ maximizing

$$\widetilde{\pi} \mapsto \tfrac{1}{|\mathcal{M}|} \textstyle\sum_{m_0 \in \mathcal{M}} J(m_0, \widetilde{\pi}; \bar{\boldsymbol{\mu}}_k^{m_0}). \tag{7.4}$$

Solving the optimization problem (7.4) can be reduced to solving a standard but non-stationary MDP. Since we aim at optimizing over population-dependent policies, the corresponding Q-function is a function of not only an agent's state-action pair (x, a) but also of the population distribution: $\widetilde{Q}(x, \mu, a)$. Adding the current mean field state μ to the Q-function allows us to recover a stationary MDP. As we know that the optimal policy is stationary, we now have a classical RL problem with state (x, μ) (instead of x only), and we can use DRL methods such as DQN (Mnih et al., 2013) to compute $\widetilde{Q}_k$. The policy $\widetilde{\pi}_k$ can then be recovered easily by applying the $\arg\max$ operator to the Q-function.

Various algorithms could be used, but we choose DQN to solve our problem because it is sample-efficient. For the numerical results presented below, we used the default implementation of RLlib (Liang et al., 2017).

The neural network representing the Q-function takes as inputs the state x of the representative player and the current distribution μ of the population, which can simply be represented as a histogram (the proportion of agents in each state). In practice, μ is a mean-field state coming from one of the averaged MF flows $\bar{\boldsymbol{\mu}}_k^{m_0}$ and is computed in steps 7 and 8 of Algorithm 7.1 with a Monte-Carlo method, *i.e.* by sampling a large number of agents that follow the last population-dependent best response $\widetilde{\pi}_k$ and averaging it with $\bar{\boldsymbol{\mu}}_{k-1}^{m_0}$. Then, the Q-function can

be approximated by a feedforward fully connected neural network with these inputs. In the examples considered below, the finite state space comes from the discretization of a continuous state space in dimension 1 or 2. The aforementioned simple approximation gives good results in $1D$. However, in $2D$, the neural network did not manage to learn a good population-dependent policy. This is probably because passing a histogram as a flat vector ignores the geometric structure of the problem. We thus resort to a more sophisticated representation. We first create an *embedding* of the distribution by passing the histogram to a convolutional neural network (ConvNet). The output of this embedding network is then passed to a fully connected network which outputs probabilities for each action (see Figure 7.1). The use of a ConvNet is motivated by the fact that the state space in our examples has a clear geometric interpretation and that the population can be represented as an image.

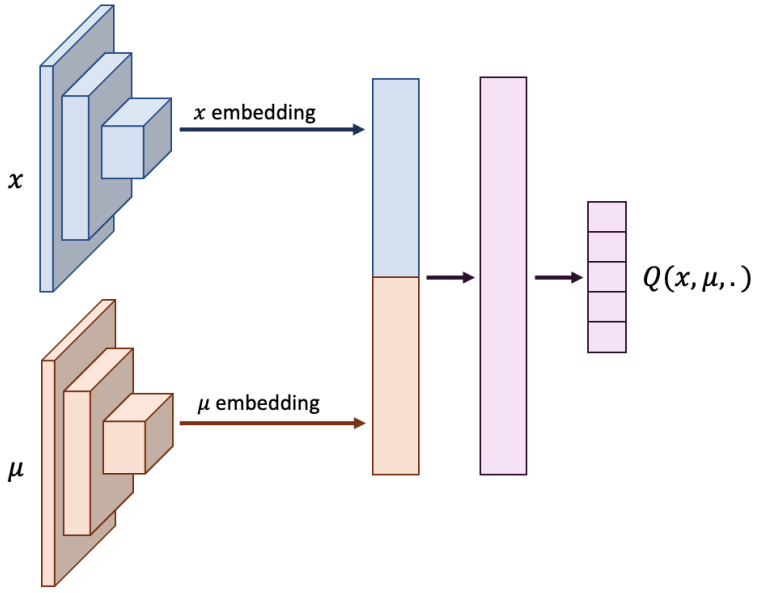


Figure 7.1 – Neural network architecture of the Q-function for the 2D beach bar experience.

7.4.3 On the Theoretical vs. Experimental Settings

Theoretically, we expect the algorithm Algorithm 7.1 to converge perfectly to a Master policy. This intuition is supported by Theorem 7.2 and comes from the fact that Fictitious Play has been proved to converge to population-agnostic equilibrium policies when the initial distribution is fixed (Cardaliaguet and Hadikhanloo, 2017; Perrin, Perolat, et al., 2020). However, from a practical viewpoint, here we need to make several approximations. The main one is related to the challenges of conditioning on a MF state. Even though the state space $\mathcal{X}$ is finite, the space of MF states $\mathcal{M} = \Delta_{\mathcal{X}}$ is infinite and of dimension equal to the number of states, which is potentially very large. This is why we need to rely on function approximation (*e.g.*, by neural networks as in our implementation) to learn an optimal population-dependent policy.

Furthermore, the training procedure uses only a finite (and relatively small) set of training distributions. On top of this, other more standard approximations are to be taken into account, in particular due to the use of a DRL subroutine.

7.5 Numerical Experiments

7.5.1 Experimental Setup

We now illustrate the efficiency and generalization capabilities of the Master policy learned with our proposed method.

Procedure. To demonstrate experimentally the performance of the learned Master policy trained by Algorithm 7.1, we consider: several initial distributions, several benchmark policies and several metrics. For each metric, we illustrate the performance of each policy on each initial distribution. The initial distributions come from two sets: the training set $\mathcal{M}$ used in Algorithm 7.1 and a testing set. For the benchmark policies, in the absence of a population-dependent baseline of reference (since, to the best of our knowledge, our work is the first to deal with Master policies), we focus on natural candidates that are population-agnostic. The metrics are chosen to give different perspectives: the population distribution and the policy performance in terms of reward.

Training set of initial distributions. In our experiments, we consider a training set $\mathcal{M}$ composed of Gaussian distributions such that the union of all these distributions sufficiently covers the whole state space. This ensures that the policy learns to behave on any state $x \in \mathcal{X}$. Furthermore, although we call "training set" the set of initial distributions, the policy actually sees more distributions during the training episodes. Each distribution visited could be considered as an initial distribution. Note however that it is very different from training the policy on all possible *population distributions* (which is a simplex with dimension equal to the number of states, *i.e.*, 32 or $16^2 = 256$ in our examples).

Testing set of initial distributions. The testing set is composed of two types of distributions. First, random distributions generated by sampling uniformly a number in $[0, 1]$ for each state independently, and then normalizing the distribution. Second, Gaussian distributions with means located between the means of the training set, and various variances.

Benchmark type 1: Specialized policies. For a given initial distribution m_0^i with $i \in \{1, \dots, |\mathcal{M}|\}$, we consider a Nash equilibrium starting from this MF state, *i.e.*, a population-agnostic policy $\hat{\boldsymbol{\pi}}^i$

and a MF flow $\hat{\boldsymbol{\mu}}^i$ satisfying Definition 6 with m_0 replaced by m_0^i. In the absence of analytical formula, we compute such an equilibrium using Fictitious Play algorithm with backward induction (Perrin, Perolat, et al., 2020). We then compare our learned Master policy with each $\hat{\boldsymbol{\pi}}^i$, either on m_0^i or on another m_0^j. In the first case, it allows us to check the correctness of the learned Master policy, and in the second case, to show that it generalizes better than $\hat{\boldsymbol{\pi}}^i$.

Benchmark type 2: Mixture-reward policy. Each (population-agnostic) policy discussed above is specialized for a given m_0^i but our Algorithm 7.1 trains a (population-dependent) policy on various initial distributions. It is thus natural to see how the learned Master policy fares in comparison with a population-agnostic policy trained on various initial distributions. We thus consider another benchmark, called *mixture-reward policy*, which is a population-agnostic policy trained to optimize an average reward. It is computed as the specialized policies described above but we replace the reward definition with an average over the training distributions. For $1 \leq i \leq |\mathcal{M}|$, recall $\hat{\boldsymbol{\mu}}^i$ is a Nash equilibrium MF flow starting with MF state m_0^i. We consider the average reward: $\bar{r}_n(x,a) = \frac{1}{|\mathcal{M}|}\sum_{i=1}^{|\mathcal{M}|} r(x,a,\hat{\boldsymbol{\mu}}_n^i)$. The mixture-reward policy is an optimal policy for the MDP with this reward function. In our experiments, we compute it as for the specialized policies described above.

Benchmark type 3: Unconditioned policy. Another meaningful comparison is to use the same algorithm while removing the population input. This amounts to running Algorithm 7.1 where, in the DQN subroutine, the Q-function neural network is a function of x and a only. So in Figure 7.1, we replace the μ input embedding by zeros. We call the resulting policy *unconditioned policy* because it illustrates the performance when removing the conditioning on the MF term. This benchmark will be used to illustrate that the success of our approach is not only due to combining DRL with training on various m_0: conditioning the Q-function and the policy on the MF term plays a key role.

Metric 1: Wasserstein distance between MF flows. We first measure how similar the policies are in terms of induced behavior at the scale of the population. Based on the Wasserstein distance W between two distributions, we compute the following distance between MF flows truncated at some horizon N_T:

$$W_{i,j} := \frac{1}{N_T+1}\sum_{n=0}^{N_T} W(\boldsymbol{\mu}_n^{\boldsymbol{\pi}^i,m_0^j}, \boldsymbol{\mu}_n^{\boldsymbol{\pi}^j,m_0^j}).$$

Note that $W_{i,i} = 0$. The term $\boldsymbol{\mu}^{\boldsymbol{\pi}^j,m_0^j} = \hat{\boldsymbol{\mu}}^j$ is the equilibrium MF flow starting from m_0^j, while $\boldsymbol{\mu}^{\boldsymbol{\pi}^i,m_0^j}$ is the MF flow generated by starting from m_0^j and using policy $\boldsymbol{\pi}^i$.

Metric 2: Exploitability. We also assess the performance of a given policy by measuring how far from being a Nash it is. To this end, we use the exploitability. We compute for each i, j: $E_{i,j} = \phi(m_0^j, \hat{\boldsymbol{\pi}}^i)$. When $i = j$, $E_{i,i} = 0$ because $(\hat{\boldsymbol{\pi}}^i, \hat{\boldsymbol{\mu}}^i)$ is a Nash equilibrium starting from m_0^i. When $i \neq j$, $E_{i,j}$ measures how far from being optimal $\hat{\boldsymbol{\pi}}^i$ is when the population also uses $\hat{\boldsymbol{\pi}}^i$, but both the representative player and the population start with m_0^j. If $E_{i,j} = 0$, then $\hat{\boldsymbol{\pi}}^i$ is a Nash equilibrium policy even when starting from m_0^j.

7.5.2 Experiment 1: Pure Exploration in 1D

We consider a discrete 1D environment inspired by Geist, Pérolat, et al. (2021). Transitions are deterministic, the state space is $\mathcal{X} = \{1, \dots, |\mathcal{X}| = 32\}$. The action space is $\mathcal{A} = \{-1, 0, 1\}$: agents can go left, stay still or go right (as long as they stay in the state space). The reward penalizes the agent with the amount of people at their location, while discouraging them from moving too much: $r(x, a, \mu) = -\log(\mu(x)) - \frac{1}{|X|}|a|$. The training set of initial distributions $\mathcal{M}$ consists of four Gaussian distributions with the same variance but different means. The testing set is composed of random and Gaussian distributions with various variances. We can see that the Master policy is still performing well on these distributions, which highlights its generalization capacities. The diagonal is white since the Wasserstein distance and exploitability are zero for specialized baselines evaluated on their corresponding m_0. We also observe that the random policy is performing well on random distributions, and that exact solutions trained on a randomly generated distribution seem to perform quite well on other randomly generated distributions. We believe this is due to this specific environment, because a policy that keeps enough entropy performs well.

7.5.3 Experiment 2: Beach Bar in 2D

We now consider the 2 dimensional beach bar problem, introduced by Perrin, Perolat, et al. (2020), to highlight that the method can scale to larger environments. The state space is a discretization of a 2-dimensional square. The agents can move by one state in the four directions: up, down, left, right, but there are walls on the boundaries. The instantaneous reward is: $r(x, a, \mu) = d_{\text{bar}}(x) - \log(\mu(x)) - \frac{1}{|X|}\|a\|_1$, where d_{bar} is the distance to the bar, located at the center of the domain. Here again, the second term discourages the agent from being in a crowded state, while the last term discourages them from moving if it is not necessary. Starting from an initial distribution, we expect the agents to move towards the bar while spreading a bit to avoid suffering from congestion.

We use the aforementioned architecture (Figure 7.1) with one fully connected network following two ConvNets: one for the agent's state, represented as a one-hot matrix, and one for the MF state, represented as a histogram. Having the same dimension (equal to the number $|\mathcal{X}|$ of states) and architecture for the position and the distribution makes it easier for the deep

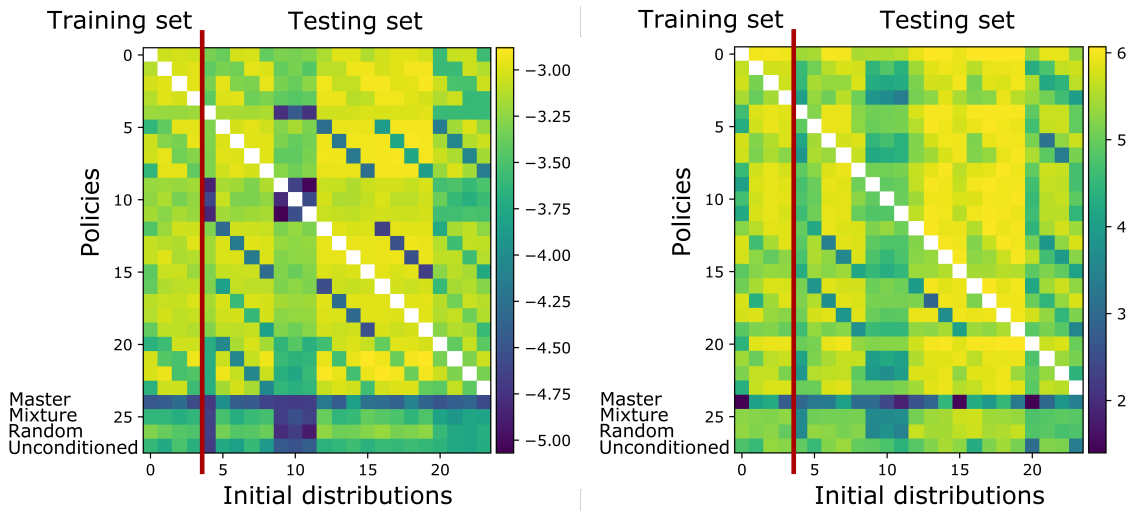


Figure 7.2 – Exploration 1D: Performance matrices when the training set is made of Gaussian distributions. From left to right: **(a)** Log of Wasserstein distances to the exact solution (time average); **(b)** Log of exploitabilities. The x-axis is the initial distribution index: on the left (resp. right) of the vertical red line are the training (resp. testing) distributions.

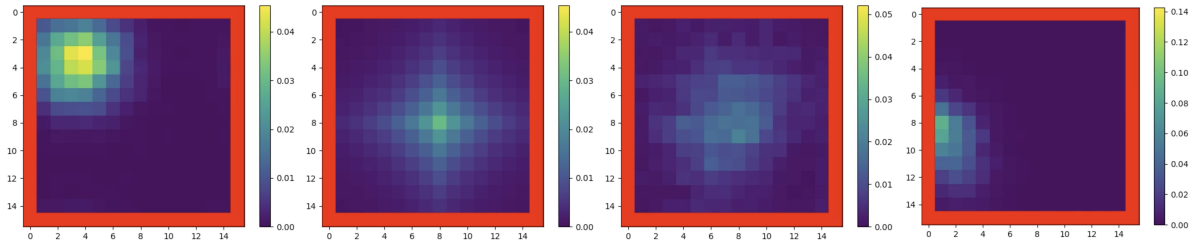


Figure 7.3 – Beach bar 2D: Environment. From left to right: **(a)** an initial distribution $m_0 \in \mathcal{M}$; **(b)** MF state at equilibrium (specialized policy); **(c)** MF state at equilibrium (learned Master policy); **(d)** MF state at equilibrium (specialized policy of another initial distribution). Note that the scale is very different for the last figure.

neural network to give an equal importance to both of these features. DRL is crucial to cope with the high dimensionality of the input. Here $|\mathcal{X}| = 16^2 = 256$.

Figure 7.4 illustrates the performance of the learned Master policy. Once again, it outperforms the specialized policies as well as the random, mixture-reward, and unconditioned policies. An illustration of the environment and of the different policies involved is available in Figure 7.3.

7.6 Conclusion of the Chapter

Motivated by the question of generalization in MFGs, we extended the notion of policies to let them depend explicitly on the population distribution. This allowed us to introduce the concept of Master policy, from which a representative player is able to play an optimal

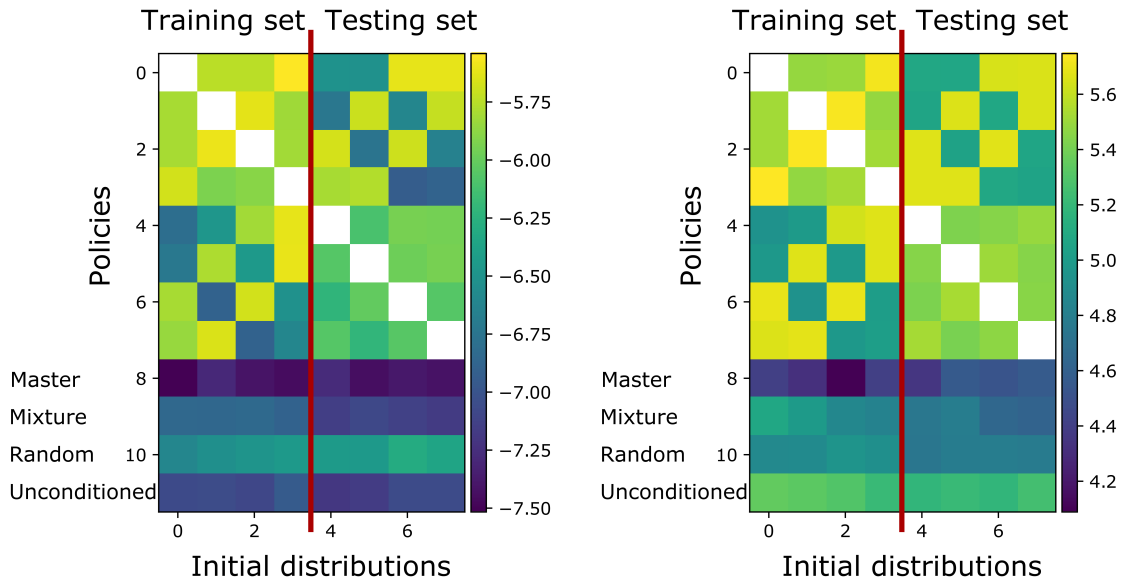


Figure 7.4 – Beach bar 2D: Performance matrices with Gaussian distributions. From left to right: **(a)** Log of Wasserstein distances to the exact solution (average over time steps); **(b)** Log of exploitabilities. Each row is a policy. Top part: a row j gives the performance of the equilibrium policy for the j-th initial distribution. Bottom part: policies given in the text).

policy against any population distribution, as we proved in Theorem 7.1. We then proved that a continuous time adaptation of Fictitious Play can approximate the Master policy at a linear rate (Theorem 7.2). However, implementing this method is not straightforward because policies and value functions are now functions of the population distribution and, hence, out of reach for traditional computational methods. We thus proposed a DRL-based algorithm to compute an approximate Master policy. Although this algorithm trains the Master policy using a small training set of distributions, we demonstrated numerically that the learned policy is competitive on a variety of unknown distributions. In other words, for the first time in the RL for MFG literature, our approach allows the agents to generalize and react to many population distributions. This is in stark contrast with the existing literature, which focuses on learning population-agnostic policies (Guo, A. Hu, et al., 2019; Anahtarcı, Karıksız, and Saldi, 2020a; Fu et al., 2019; Elie, Perolat, et al., 2020; Perrin, Laurière, Pérolat, Geist, et al., 2021). To the best of our knowledge, the only work considering policies that depend on the population is Mishra, Vasal, and Vishwanath (2020), but their approach relies on solving a fixed point at each time step for every possible distribution, which is infeasible except for very small state space.

Our approach opens many directions for future work. First, the algorithm we proposed should be seen as a proof of concept and other methods could be investigated, such as Online Mirror Descent (Hadikhanloo, 2017; Perolat, Perrin, et al., 2021). For high-dimensional examples, the question of distribution embedding deserves a special attention. Second, the generalization capabilities of the learned Master policy offers many new possibilities for ap-

plications. We plan to investigate how it can be used when the agent can only access a partial observation of the population. Last, the theoretical properties (such as the approximation and generalization theory) are also left for future work. An interesting question, from the point of view of learning is choosing the training set so as to optimize generalization capabilities of the learned Master policy.

Chapter 8

Scalable Algorithms

This last chapter presents two scalable Deep Reinforcement Learning algorithms for Mean Field Games. We have learned in Chapter 3 that one limiting factor to further scale up MFGs using RL is that existing algorithms such as Fictitious Play or Online Mirror Descent require the mixing of approximated quantities such as strategies or Q-values. This is far from being trivial in the case of non-linear function approximation that enjoy good generalization properties, such as neural networks. We propose two methods to address this shortcoming. The first one learns a mixed strategy from distillation of historical data into a neural network and is applied to the Fictitious Play algorithm. The second one is an online mixing method based on regularization that does not require memorizing historical data or previous estimates. It is used to extend Online Mirror Descent. We demonstrate numerically that these methods efficiently enable the use of DRL algorithms to solve various MFGs. In addition, we show that these methods outperform SotA baselines from the literature.[1]

Contents

8.1 Motivation **138**
8.2 Background **139**
8.3 Deep Reinforcement Learning for MFGs **142**
8.4 Experiments **146**
8.5 Conclusion of the Chapter **152**

[1]This chapter is based on a preprint (Lauriere et al., 2022) presented at the ICML 2022 conference.

8.1 Motivation

Many recent works have combined MFGs with RL to leverage their mutual potential – albeit mostly without deep neural nets thus far. The intertwinement of MFGs and RL happens through an *optimization* (or learning) procedure. The simplest algorithm of this type is the (Banach-Picard) fixed-point approach, consisting in alternating a best response computation against a given population distribution with an update of this distribution (M. Huang, R. P. Malhamé, and Caines, 2006). However, this method fails in many cases by lack of contractivity as proved by Cui and Koeppl (2021). Several other procedures have thus been introduced, often inspired by game theory or optimization algorithms. Fictitious Play (FP) and its variants average either distributions or policies (or both) to stabilize convergence (Cardaliaguet and Hadikhanloo, 2017; Elie, Perolat, et al., 2020; Perrin, Perolat, et al., 2020; Xie et al., 2021; Perrin, Laurière, Pérolat, Élie, et al., 2021; Perrin, Laurière, Pérolat, Geist, et al., 2021), whereas Online Mirror Descent (OMD) (Hadikhanloo, 2017; Perolat, Perrin, et al., 2021) relies on policy evaluation. Other works have leveraged regularization (Anahtarcı, Karıksız, and Saldi, 2020a; Cui and Koeppl, 2021; Guo, Xu, and Zariphopoulou, 2020) to ensure convergence, at the cost of biasing the Nash equilibrium. These methods require to sum or average some key quantities: FP needs to average the distributions, while OMD needs to sum Q-functions. This is a key component of most (if not all) smoothing methods and needs to be tackled efficiently. These operations are simple when the state space is finite and small, and the underlying objects can be represented with tables or linear functions. However, there is no easy and efficient way to sum non-linear approximations such as neural networks, which raises a major challenge when trying to combine learning methods (such as FP or OMD) with deep RL.

Contributions. The main contribution of the chapter is to solve this important question in dynamic MFGs. We propose two algorithms. The first one, that we name Deep Average-network Fictitious Play (D-AFP), builds on FP and uses the Neural Fictitious Self Play (NFSP) approach (Heinrich and Silver, 2016) to compute a neural network approximating an average over past policies. The second one is Deep Munchausen Online Mirror Descent (D-MOMD), inspired by the Munchausen reparameterization of Vieillard, Pietquin, and Geist (2020). We prove that in the exact case, Munchausen OMD is equivalent to OMD. Finally, we conduct numerical experiments and compare D-AFP and D-MOMD with SotA baselines adapted to dynamic MFGs. We find that D-MOMD converges faster than D-AFP on all tested games from the literature, which is consistent with the results obtained for exact algorithms (without RL) in (Perolat, Perrin, et al., 2021; Geist, Pérolat, et al., 2021).

8.2 Background

8.2.1 Mean Field Games

We recall briefly the main quantities of interest needed in this chapter. We stress that we consider a *finite horizon* setting as it encompasses a broader class of games, which needs time-dependant policies and distributions. The two main quantities of interest are the policy of the representative player $\pi = (\pi_n)_n \in (\Delta_{\mathcal{A}}^{\mathcal{X}})^{N_T+1}$ and the distribution flow (*i.e.* sequence) of agents $\mu = (\mu_n)_n \in \Delta_{\mathcal{X}}^{N_T+1}$. Given a population mean field flow μ, the goal for a representative agent is to maximize over π the total reward:

$$J(\pi, \mu) = \mathbb{E}_\pi\Big[\sum_{n=0}^{N_T} r_n(x_n, a_n, \mu_n)\Big| x_0 \sim m_0\Big]$$
$$\text{s.t.: } a_n \sim \pi_n(\cdot|x_n),\ x_{n+1} \sim p_n(\cdot|x_n, a_n, \mu_n),\ n \geq 0.$$

A policy π is called a best response (BR) against a mean field flow μ if it is a maximizer of $J(\cdot, \mu)$. We denote by $BR(\mu)$ the set of best responses to the mean field flow μ.

Given a policy π, a mean field flow μ is said to be induced by π if: $\mu_0 = m_0$ and for $n = 0, \dots, N_T - 1$,

$$\mu_{n+1}(x) = \sum_{x',a'} \mu_n(x')\pi_n(a'|x')p_n(x|x', a', \mu_n),$$

which we can simply write $\mu_{n+1} = P_n^{\mu_n,\pi_n}\mu_n$, where $P_n^{\mu_n,\pi_n}$ is the transition matrix of x_n. We denote by μ^π or $\Phi(\pi) \in \Delta_{\mathcal{X}}^{N_T+1}$ the mean-field flow induced by π.

Definition 8. *A pair $(\hat{\pi}, \hat{\mu})$ is a (finite horizon) Mean Field Nash Equilibrium (MFNE) if (1) $\hat{\pi}$ is a BR against $\hat{\mu}$, and (2) $\hat{\mu}$ is induced by $\hat{\pi}$.*

Equivalently, $\hat{\pi}$ is a fixed point of the map $BR \circ \Phi$. Given a mean field flow μ, a representative player faces a traditional MDP, which can be studied using classical tools. The value of a policy can be characterized through the Q-function defined as: $Q_{N_T+1}^{\pi,\mu}(x, a) = 0$ and for $n \leq N_T$,

$$Q_n^{\pi,\mu}(x, a) = \mathbb{E}\Big[\sum_{n' \geq n} r_{n'}(x_{n'}, a_{n'}, \mu_{n'})\Big|(x_n, a_n) = (x, a)\Big].$$

It satisfies the Bellman equation: for $n \leq N_T$,

$$Q_n^{\pi,\mu}(x, a) = r_n(x, a, \mu_n) + \mathbb{E}_{x',a'}[Q_{n+1}^{\pi,\mu}(x', a')], \tag{8.1}$$

where $x' \sim p(\cdot|x, a, \mu_n)$ and $a' \sim \pi_n(\cdot|x, a, \mu_n)$, with the convention $Q_{N_T+1}^{\pi,\mu}(\cdot, \cdot) = 0$. The optimal Q-function $Q^{*,\mu}$ is the value function of any best response π^* against μ. It is defined as $Q_n^{*,\mu}(x, a) = \max_\pi Q_n^{\pi,\mu}(x, a)$ for every n, x, a, and it satisfies the optimal Bellman equation: for

$n \leq N_T$,

$$Q_n^{*,\mu}(x,a) = r_n(x,a,\mu_n) + \mathbb{E}_{x',a'}[\max_{a'} Q_{n+1}^{*,\mu}(x',a')], \tag{8.2}$$

where $Q_{N_T+1}^{*,\mu}(\cdot,\cdot) = 0$.

8.2.2 Fictitious Play

The most straightforward method to compute a MFNE is to iteratively update in turn the policy π and the distribution μ, by respectively computing a BR and the induced mean field flow. The BR can be computed with the backward induction of (8.2), if the model is completely known. We refer to this method as *Banach-Picard (BP) fixed point iterations*. See Algorithm F.4 in appendix for completeness. The convergence is ensured as soon as the composition $BR \circ \Phi$ is a strict contraction (M. Huang, R. P. Malhamé, and Caines, 2006). However, this condition holds only for a restricted class of games and, beyond that, simple fixed point iterations typically fail to converge and oscillations appear (Cui and Koeppl, 2021).

To address this issue, a memory of past plays can be added. The *Fictitious Play (FP) algorithm*, introduced by G. W. Brown (1951) computes the new distribution at each iteration by taking the average over all past distributions instead of the latest one. This stabilizes the learning process so that convergence can be proved for a broader class of games under suitable assumptions on the structure of the game such as potential structure (Cardaliaguet and Hadikhanloo, 2017; Geist, Pérolat, et al., 2021) or monotonicity (Perrin, Perolat, et al., 2020). The method can be summarized as follows: after initializing Q_n^0 and π_n^0 for $n = 0, \ldots, N_T$, repeat at each iteration k:

$$\begin{cases} 1.\ \text{Distribution update: } \mu^k = \mu^{\pi^k},\ \bar{\mu}^k = \frac{1}{k-1}\sum_{i=1}^{k-1} \mu^i \\ 2.\ Q\text{-function update: } Q^k = Q^{*,\bar{\mu}^k} \\ 3.\ \text{Policy update: } \pi_n^{k+1}(.|x) = \arg\max_a Q_n^k(x,a). \end{cases}$$

In the distribution update, $\bar{\mu}^k$ corresponds to the population mean field flow obtained if, for each $i = 1, \ldots, k-1$, a fraction $1/(k-1)$ of the population follows the policy π^i obtained as a BR at iteration i. At the end, the algorithm returns the latest mean field flow μ^k as well as a policy that generates this mean field flow. This can be achieved either through a single policy or by returning the vector of all past BR, $(\pi^i)_{i=1,\ldots,k-1}$, from which $\bar{\mu}^k$ can be recovered. See Algorithm F.5 in appendix for completeness.

8.2.3 Online Mirror Descent

The aforementioned methods are based on computing a BR at each iteration. Alternatively, we can follow a policy iteration based approach and simply *evaluate* a policy at each iteration. In

finite horizon, this operation is less computationally expensive than computing a BR because it avoids a loop over the actions to find the optimal one.

The *Policy Iteration* (*PI*) *algorithm* for MFG (Cacace, Simone, Camilli, Fabio, and Goffi, Alessandro, 2021) consists in repeating, from an initial guess π^0, μ^0, the update: at iteration k, first evaluate the current policy π^k by computing $Q^{k+1} = Q^{\pi^k, \mu^k}$, then let π^{k+1} be the greedy policy such that $\pi^{k+1}(\cdot|x)$ is a maximizer of $Q^k(x, \cdot)$. The evaluation step can be done with the backward induction (8.1), provided the model is known. See Algorithm F.6 in appendix for completeness.

Here again, to stabilize the learning process, one can rely on information from past iterations. Using a weighted sum over past Q-functions yields the so-called *Online Mirror Descent* (*OMD*) *algorithm* for MFG, which can be summarized as follows: after initializing q_n^0 and π_n^0 for $n = 0, \dots, N_T$, repeat at each iteration k:

$$\begin{cases} 1.\ \text{Distribution update: } \mu^k = \mu^{\pi^k} \\ 2.\ Q\text{-function update: } Q^k = Q^{\pi^k, \mu^k} \\ 3.\ \text{Regularized } Q\text{-function update: } \bar{q}^{k+1} = \bar{q}^k + \frac{1}{\tau} Q^k \\ 4.\ \text{Policy update: } \pi_n^{k+1}(\cdot|x) = \mathrm{softmax}(\bar{q}_n^{k+1}(x, \cdot)). \end{cases}$$

For more details, see Algorithm F.7 in appendix. Although we focus on softmax policies in the sequel, other conjugate functions of steep regularizers could be used in OMD, see Perolat, Perrin, et al. (2021).

This method is known to be empirically faster than FP, as illustrated by Perolat, Perrin, et al. (2021). Intuitively, this can be explained by the fact that the learning rate in FP is of the order $1/k$ so this algorithm is slower and slower as the number of iterations increases.

8.2.4 Deep Reinforcement Learning

Reinforcement learning aims to solve optimal control problems when the agent does not know the model (*i.e.*, p and r) and must learn through trial and error by interacting with an environment. In a finite horizon MFG setting, we assume that a representative agent is encoded by a policy π, either explicitly or implicitly (through a Q-function) and can realize an episode, in the following sense: for $n = 0, \dots, N_T$, the agent observes x_n (with $x_0 \sim m_0$), chooses action $a_n \sim \pi_n(\cdot|x_n)$, and the environment returns a realization of $x_{n+1} \sim p_n(\cdot|x_n, a_n, \mu_n)$ and $r_n(x_n, a_n, \mu_n)$. Note that the agent does not need to observe directly the mean field flow μ_n, which simply enters as a parameter of the transition and cost functions p_n and r_n.

Based on such samples, the agent can approximately compute the Q-functions $Q^{\pi,\mu}$ and $Q^{*,\mu}$ following (8.1) and (8.2) respectively where the expectation is replaced by Monte-Carlo samples. In practice, we often use trajectories starting from time 0 and state $x_0 \sim m_0$ instead of

starting from any pair (x, a). Vanilla RL considers infinite horizon, discounted problems and looks for a stationary policy, whereas we consider a finite-horizon setting with non-stationary policies. To simplify the implementation, we treat time as part of the state by considering (n, x_n) as the state. We can then use standard Q-learning. However, it is important to keep in mind that the Bellman equations are not fixed-point equation for some stationary Bellman operators.

When the state space is large, it becomes impossible to evaluate precisely every pair (x, a). Motivated by both memory efficiency and generalization, we can approximate the Q-functions by non linear functions such as neural networks, say $Q_\theta^{\pi,\mu}$ and $Q_\theta^{*,\mu}$, parameterized by θ. Then, the quantities in (8.1) and (8.2) are replaced by the minimization of a loss to train the neural network parameters θ. Namely, treating time as an input, one minimizes over θ the quantities

$$\hat{\mathbb{E}}\left[\left|Q_{\theta,n}^{\pi,\mu}(x,a) - r_n(x,a,\mu_n) - Q_{\theta_t,n+1}^{\pi,\mu}(x',a')\right|^2\right]$$
$$\hat{\mathbb{E}}\left[\left|Q_{\theta,n}^{*,\mu}(x,a) - r_n(x,a,\mu_n) - \max_{a'} Q_{\theta_t,n+1}^{*,\mu}(x',a')\right|^2\right],$$

where $\hat{\mathbb{E}}$ is an empirical expectation based on Monte Carlo samples and θ_t is the parameter of a target network.

8.3 Deep Reinforcement Learning for MFGs

To develop scalable methods for solving MFGs, a natural idea consists in combining the above optimization methods (FP and OMD) with deep RL. This requires summing or averaging policies or distributions, and induces hereby a major challenge as they are approximated by non linear operators, such as neural networks. In this section, we develop innovative and scalable solutions to cope with this. In the sequel, we denote $Q_\theta((n, x), a)$ with the time in the input when we refer to the neural network Q-function.

8.3.1 Deep Average-network Fictitious Play

To develop a model-free version of FP, one first needs to compute a BR at each iteration, which can be done using standard deep RL methods, such as DQN (Mnih et al., 2013). A policy that generates the average distribution over past iterations can be obtained by simply keeping in memory all the BRs from past iterations. This approach has already been used successfully *e.g.* by Perrin, Perolat, et al. (2020) and Perrin, Laurière, Pérolat, Geist, et al. (2021). However, it requires a memory that is linear in the number of iterations and each element is potentially large (*e.g.*, a deep neural network), which does not scale well. Indeed, as the complexity of the

Algorithm 8.1: D-AFP

input : Initialize an empty reservoir buffer $\mathcal{M}_{SL}$ for supervised learning of average policy, the mean field distribution flow $\bar{\mu}^0 = m_0$ and the parameters $\bar{\theta}^0$

for $k = 1, \ldots, K$: **do**

1. Distribution: Generate $\bar{\mu}^k$ with $\bar{\pi}_{\bar{\theta}^{k-1}}$;

2. BR: Train $\hat{\pi}_{\theta^k}$ against $\bar{\mu}^{k-1}$ using DQN;

Collect $N_{samples}$ state-action using $\hat{\pi}_{\theta^k}$ and add them to $\mathcal{M}_{SL}$;

3. Average policy: Update $\bar{\pi}_{\bar{\theta}^k}$ by adjusting $\bar{\theta}^k$ (through gradient descent) to minimize:

$$\mathcal{L}(\bar{\theta}) = \mathbb{E}_{(s,a)\sim\mathcal{M}_{SL}}\left[-\log\left(\bar{\pi}_{\bar{\theta}}(a|s)\right)\right],$$

where $\bar{\pi}_{\bar{\theta}}$ is the neural net policy with parameters $\bar{\theta}$

output : $\bar{\mu}^K, \bar{\pi}_{\bar{\theta}^K}$

environment grows, we expect FP to need more and more iterations to converge, and hence the need to keep track of a larger and larger number of policies.

An alternative approach is to learn along the way the policy generating the average distribution. We propose to do so by keeping a buffer of state-action pairs generated by past BRs and learning the average policy by minimizing a categorical loss. To tackle potentially complex environments, we rely on a neural network representation of the policy. This approach is inspired by the Neural Fictitious Self Play (NFSP) method (Heinrich and Silver, 2016), developed initially for imperfect information games with a finite number of players, and adapted here to the MFG setting. The proposed algorithm, that we call D-AFP because it learns an average policy, is summarized in Algorithm 8.1. Details are in Section F.2.

This allows us to learn an approximation of the MFNE policy with a single neural network instead of having it indirectly through a collection of neural networks for past BRs. After training, we can use this neural average policy in a straightforward way. Although the buffer is not needed after training, a drawback of this method is that during the training it requires to keep a buffer whose size increases linearly with the number of iterations. This motivates us to investigate a modification of OMD which is not only empirically faster, but also less memory consuming.

8.3.2 Deep Munchausen Online Mirror Descent

We now turn our attention to the combination of OMD and deep RL. One could simply use RL for the policy evaluation step by estimating the Q-function using equation (8.1). However, it is not straightforward to train a neural network to approximate the cumulative Q-function. To that end, we propose a reparameterization allowing us to compute the cumulative Q-function

in an implicit way, building on the Munchausen trick from Vieillard, Pietquin, and Geist (2020) for classical RL (with a single agent and no mean-field interactions).

Reparameterization in the exact case. OMD requires summing up Q-functions to compute the regularized Q-function $\bar{q}$. However, this quantity $\bar{q}$ is hard to approximate as there exists no straightforward way to sum up neural networks. We note that this summation is done by Pérolat et al. (2021) via the use of the NeuRD loss. However, this approach relies on two types of approximations, as one must learn the correct Q-function, but also the correct sum. This is why we instead transform the OMD formulation into Munchausen OMD, which only relies on one type of approximation. We start by describing this reparameterization in the exact case, *i.e.*, without function approximation. We show that, in this case, the two formulations are equivalent.

We consider the following modified Bellman equation:

$$\begin{cases} \tilde{Q}^{k+1}_{N_T+1}(x,a) = 0 \\ \tilde{Q}^{k+1}_{n-1}(x,a) = r(x,a,\mu^k_{n-1}) + \tau \ln \pi^k_{n-1}(a|x) + \mathbb{E}_{x',a'}\left[\tilde{Q}^{k+1}_n(x',a') - \tau \ln \pi^k_n(a'|x')\right], \end{cases} \tag{8.3}$$

where $x' \sim p_n(\cdot|x,a,\mu^k_{n-1})$ and $a' \sim \pi^k_n(\cdot|x')$. The red term penalizes the policy for deviating from the one in the previous iteration, π^k_{n-1}, while the blue term compensates for this change in the backward induction, as we will explain in the proof of Theorem 8.1 below.

The Munchausen OMD (MOMD) algorithm for MFG is as follows: after initializing π^0, repeat for $k \geq 0$:

$$\begin{cases} \text{Distribution update: } \mu^k = \mu^{\tilde{\pi}^k} \\ \text{Regularized } Q\text{-function update: } \tilde{Q}^{k+1}_n \text{ as in (8.3)} \\ \text{Policy update: } \tilde{\pi}^{k+1}_n(\cdot|x) = \mathrm{softmax}(\frac{1}{\tau}\tilde{Q}^{k+1}_n(x,\cdot)). \end{cases}$$

Theorem 8.1. *MOMD is equivalent to OMD in the sense that $\tilde{\pi}^k = \pi^k$ for every k.*

As a consequence, despite seemingly artificial log terms, this method does not bias the Nash equilibrium (in contrast with, *e.g.*, Cui and Koeppl (2021) and Xie et al. (2021)). Thanks to this result, MOMD enjoys the same convergence guarantees as OMD, see (Hadikhanloo, 2017; Perolat, Perrin, et al., 2021).

Proof. **Step 1: Softmax transform.** We first replace this projection by an equivalent viewpoint based on the Kullback-Leibler (KL) divergence, denoted by $KL(\cdot\|\cdot)$. We will write $Q^{k+1}_n = Q^{\pi^k,\mu^{\pi^k}}_n$ for short and $Q^0_n = \bar{q}^0_n$. We have: $\bar{q}^{k+1}_n = \frac{1}{\tau}\sum_{\ell=0}^{k+1} Q^\ell_n$. We take π^0_n as the uniform policy over actions, in order to have a precise equivalence with the following for $\bar{q}^0_n = 0$. We could

consider any π_n^0 with full support, up to a change of initialization $\bar{q}_n^0$. We have:

$$\begin{aligned}\pi_n^{k+1}(\cdot|x) &= \text{softmax}\left(\frac{1}{\tau}\sum_{\ell=0}^{k+1} Q_n^\ell(\cdot|x)\right)\\ &= \arg\max_{\pi\in\Delta_A}\Big(\langle\pi, Q_n^{k+1}(x,\cdot)\rangle - \tau KL(\pi\|\pi_n^k(\cdot|x))\Big),\end{aligned} \tag{8.4}$$

where $\langle\cdot,\cdot\rangle$ denotes the dot product.

Indeed, this can be checked by induction, using the Legendre-Fenchel transform: omitting n and x for brevity,

$$\begin{aligned}\pi^{k+1} &\propto \pi^k e^{\frac{1}{\tau}q^{k+1}}\\ &\propto \pi^{k-1}e^{\frac{1}{\tau}Q^k}e^{\frac{1}{\tau}Q^{k+1}} = \pi^{k-1}e^{\frac{1}{\tau}(Q^k+Q^{k+1})}\\ &\propto \ldots \propto e^{\bar{q}^{k+1}}.\end{aligned}$$

Step 2: Munchausen trick. Simplifying a bit notations,

$$\begin{aligned}\pi^{k+1} &= \arg\max(\langle\pi, Q^{k+1}\rangle - \tau KL(\pi\|\pi^k))\\ &= \arg\max(\langle\pi, \underbrace{Q^{k+1}+\tau\ln\pi^k}_{\tilde{Q}^{k+1}}\rangle \underbrace{-\tau\langle\pi,\ln\pi\rangle}_{+\tau\mathcal{H}(\pi)})\\ &= \text{softmax}\Big(\frac{1}{\tau}\tilde{Q}^{k+1}\Big)\end{aligned}$$

where $\mathcal{H}$ denotes the entropy and we defined $\tilde{Q}_n^{k+1} = Q_n^{k+1} + \tau\ln\pi_n^k$. Since Q_n^{k+1} satisfies the Bellman equation (8.1) with π replaced by π^k and μ replaced by μ^k, we deduce that $\tilde{Q}_n^{k+1}$ satisfies (8.3). □

Remark: In OMD, $\frac{1}{\tau}$ (denoted α in the original paper (Perolat, Perrin, et al., 2021)), is homogeneous to a learning rate. In the MOMD formulation, τ can be seen as a temperature.

Stabilizing trick. We have shown that MOMD is equivalent to OMD. However, the above version of Munchausen sometimes exhibits numerical instabilities. This is because, if an action a is suboptimal in a state x, then $\pi_n^k(a|x)\to 0$ as $k\to+\infty$, so $\tilde{Q}^k(x,a)$ diverges to $-\infty$ due to the relation $\tilde{Q}_n^{k+1} = Q_n^{k+1} + \tau\ln\pi_n^k$. This causes issues even in the tabular setting when we get close to a Nash equilibrium, due to numerical errors on very large numbers. To avoid this issue, we introduce another parameter, denoted by $\alpha\in[0,1]$ and we consider the following modified Munchausen equation:

$$\check{Q}_{n-1}^{k+1}(x,a) = r_{n-1}(x,a,\mu_{n-1}^k) + \alpha\tau\log(\pi_{n-1}^{k-1}(a|x)) + \mathbb{E}_{x',a'}\Big[\tilde{Q}_n^{k+1}(x',a') - \tau\ln\pi_n^k(a'|x')\Big], \tag{8.5}$$

where $x' \sim p_n(\cdot|x, a, \mu^k_{n-1})$ and $a' \sim \pi^k_n(\cdot|x')$. In fact, such iterations have a natural interpretation as they can be obtained by applying OMD to a regularized problem in which, when using policy π, a penalty $-(1-\alpha)\tau \log(\pi_n(\cdot|x_n))$ is added to the reward $r_n(x_n, a_n, \mu_n)$. Details are provided in Section F.3.

Deep RL version. Motivated by problems with large spaces, we then replace the Munchausen Q-function at iteration k, namely $\check{Q}^k$, by a neural network whose parameters θ^k are trained to minimize a loss function representing (8.5). Since we want to learn a function of time, we consider (n, x) to be the state. To be specific, given samples of transitions

$$\left\{\Big((n_i, x_i), a_i, r_{n_i}(x_i, a_i, \mu^k_{n_i}), (n_i+1, x'_i)\Big)\right\}_{i=1}^{N_B},$$

with $x'_i \sim p_{n_i}(x_i, a_i, \mu^k_{n_i})$, the parameter θ^k is trained using stochastic gradient descent to minimize the empirical loss:

$$\frac{1}{N_B}\sum_i \left|\check{Q}_\theta((n_i, x_i), a_i) - T_i\right|^2,$$

where the target T_i is:

$$\begin{aligned} T_i = &-r_{n_i}(x_i, a_i, \mu^k_{n_i}) - \alpha\tau \log(\pi^{k-1}(a_i|(n_i, x_i))) \\ &- \sum_{a'} \pi^{k-1}(a'|(n_i+1, x'_i))\Big[\check{Q}_{\theta^{k-1}}((n_i+1, x'_i), a') \\ &- \tau \log(\pi^{k-1}(a'|(n_i+1, x'_i)))\Big]. \end{aligned} \tag{8.6}$$

Here the time n is passed as an input to the Q-network along with x, hence our change of notation. This way of learning the Munchausen Q-function is similar to DQN, except for two changes in the target: (1) it incorporates the penalization for deviating from the previous policy, and (2) we do not take the argmax over the next action but an average according to the previous policy. A simplified version of the algorithm is presented in Algorithm 8.2 and more details are provided in Section F.2.

8.4 Experiments

In this section, we first discuss the metric used to assess quality of learning, detail baselines to which we compare our algorithms, and finally present numerical results on diverse and numer-

Algorithm 8.2: D-MOMD

input : Munchausen parameters τ and α; numbers of OMD iterations K and DQN estimation iterations L

Initialize the parameters θ^0 ;

Set $\pi^0(a|(n,x)) = \text{softmax}\Big(\frac{1}{\tau}\check{Q}_{\theta^0}((n,x),\cdot)\Big)(a)$;

for $k = 1, \dots, K$: **do**

  1. Distribution: Generate μ^k with π^{k-1};

  2. Value function: Initialize θ^k ;

  for $\ell = 1, \dots, L$: **do**

    Sample a minibatch of N_B transitions:

    $\Big\{\Big((n_i,x_i),a_i,r_{n_i}(x_i,a_i,\mu^k_{n_i}),(n_i+1,x'_i)\Big)\Big\}_{i=1}^{N_B}$ with $n_i \le N_T$,

    $x'_i \sim p_{n_i}(\cdot|x_i,a_i,\mu^k_{n_i})$ and a_i is chosen by an $\varepsilon-$greedy policy based on $\check{Q}_{\theta^k}$;

    Update θ^k with one gradient step of:

    $\theta \mapsto \frac{1}{N_B}\sum_{i=1}^{N_B}\Big|\check{Q}_\theta((n_i,x_i),a_i) - T_i\Big|^2$

    where T_i is defined in (8.6)

  3. Policy: for all n, x, a, let

    $\pi^k(a|(n,x)) = \text{softmax}\Big(\frac{1}{\tau}\check{Q}_{\theta^k}((n,x),\cdot)\Big)(a)$

output : Cumulated Q value function $\check{Q}_{\theta^K}$, policy π^K

ous environments. The code for Deep Munchausen OMD is available in OpenSpiel (Lanctot, Lockhart, et al., 2019).[2]

8.4.1 Exploitability

To assess the quality of a learnt equilibrium, we check whether, in response to the reward generated by the population MF flow, a typical player can improve their reward by deviating from the policy used by the rest of the population. This is formalized through the notion of exploitability.

The exploitability of a policy π is defined as:

$$\phi(\pi) = \max_{\pi'} J(\pi';\mu^\pi) - J(\pi;\mu^\pi),$$

where μ^π is the mean field flow generated from m_0 when using policy π. Intuitively a large exploitability means that, when the population plays π, any individual player can be much better off by deviating and choosing a different strategy, so π is far from being a Nash equilibrium

[2] See https://github.com/deepmind/open_spiel/blob/master/open_spiel/python/mfg/algorithms/munchausen_deep_mirror_descent.py.

policy. Conversely, an exploitability of 0 means that π is an MFNE policy. Similar notions are widely used in computational game theory (Zinkevich et al., 2007; Lanctot, Waugh, et al., 2009).

In the sequel, we consider problems for which a BR can be computed exactly given a mean-field flow. Otherwise an approximate exploitability could be used as a good proxy to assess convergence, see *e.g.* Perrin, Laurière, Pérolat, Geist, et al. (2021).

8.4.2 Baselines

To assess the quality of the proposed algorithms, we consider three baselines from the literature: Banach-Picard (BP) fixed point iterations, policy iterations (PI), and Boltzmann iterations (BI). FP can be viewed as a modification of the first one, while OMD as a modification of the second one. They have been discussed at the beginning of Section 8.2.2 and Section 8.2.3 respectively, in the exact case. Adapting them to the model-free setting with deep networks can be done in a similar way as discussed above for D-AFP and D-MOMD. See Section F.2 for more details. The third baseline has been introduced recently by Cui and Koeppl (2021). It consists in updating in turn the population distribution and the policy, but here the policy is computed as a weighted softmax of the optimal Q-values (and hence requires the resolution of an MDP at each iteration). More precisely, given a reference policy π_B, a parameter $\eta > 0$, and the Q-function Q^k computed at iteration k, the new policy is defined as:

$$\pi_n^k(a|x) = \frac{\pi_{B,n}(a|x) \exp\left(Q_n^k(x,a)/\eta\right)}{\sum_{a'} \pi_{B,n}(a'|x) \exp\left(Q_n^k(x,a')/\eta\right)}.$$

In the plots, D-BP, D-AFP, D-PI, D-BI and D-MOMD refer respectively to Deep Banach-Picard iterations, Deep Average-network Fictitious Play, Deep Policy Iteration, Deep Boltzmann Iteration, and Deep Munchausen OMD.

8.4.3 Numerical results

Epidemics model. We first consider the SIS model of Cui and Koeppl (2021), which is a toy model for epidemics. There are two states: susceptible (S) and infected (I). Two actions can be used: social distancing (D) or going out (U). The probability of getting infected increases if more people are infected, and is smaller when using D instead of U. The transitions are: $p(S|I,D,\mu) = p(S|I,U,\mu) = 0.3$, $p(I|S,U,\mu) = 0.9^2 \cdot \mu(I)$, $p(I|S,D,\mu) = 0$, the reward is: $r(s,a,\mu) = -1_I(s) - 0.5 \cdot 1_D(s)$, and the horizon is $N_T = 50$. Note that, although this model has only two states, the state dynamics is impacted by the distribution, which is generally challenging when computing MFG solutions. As shown in Figure 8.1, both D-MOMD and D-AFP generate an exploitability diminishing with the learning steps, whereas the other baselines are not able to do so. Besides, D-MOMD generates smooth state trajectories, as opposed to the

one observed in Cui and Koeppl (2021), that contained many oscillations. For this example and the following ones, we display the best exploitability curves obtained for each method after running sweeps over hyperparameters. See Section F.4 for some instances of sweeps for D-MOMD.

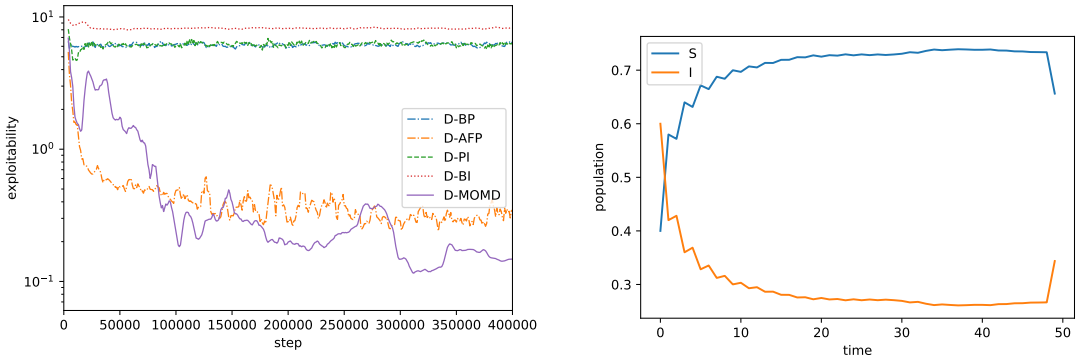


Figure 8.1 – Left: exploitability. Right: evolution of the distribution obtained by the policy learnt with D-MOMD.

Linear-Quadratic MFG. We then consider an example with more states, in 1D: the classical linear-quadratic environment of Carmona, Fouque, and L.-H. Sun (2015a), which admits an explicit closed form solution in a continuous space-time domain. We focus on a discretized approximation (Perrin, Perolat, et al., 2020) of the time grid $\{0, \dots, N_T\}$, where the dynamics of the underlying state process controlled by action a_n is given by $x_{n+1} = x_n + a_n\Delta_n + \sigma\varepsilon_n\sqrt{\Delta_n}$, with $(\bar{m}_n)_n$ the average of the population states, Δ_n the time step and $(\varepsilon_n)_n$ i.i.d. noises on $\{-3, -2, -1, 0, 1, 2, 3\}$, truncated approximations of $\mathcal{N}(0, 1)$ random variables. Given a set of actions $(a_n)_n$ in state trajectory $(x_n)_n$ and a mean field flow $(\mu_n)_n$, the reward $r(x_n, a_n, \mu_n)$ of a representative player is given for $n < N_T$ by

$$\Big[-\frac{1}{2}|a_n|^2 + qa_n(\bar{m}_n - x_n) - \frac{\kappa}{2}|\bar{m}_n - x_n|^2\Big]\Delta_n \,,$$

together with the terminal reward $-\frac{c_{term}}{2}|\bar{m}_{N_T} - x_{N_T}|^2$. The reward penalizes high actions, while providing incentives to remain close to the average state despite the noise $(\varepsilon_n)_n$. For the experiments, we used $N_T = 10$, $\sigma = 1$, $\Delta_n = 1$, $q = 0.01$, $\kappa = 0.5$, $c_{term} = 1$ and $|\mathcal{X}| = 100$. The action space is $\{-3, -2, -1, 0, 1, 2, 3\}$.

In Figure 8.2 (top), we see that the distribution estimated by D-MOMD concentrates, as is expected from the reward encouraging a mean-reverting behavior: the population gathers as expected into a bell-shaped distribution. The analytical solution (Appx. E in (Perrin, Perolat, et al., 2020)) is exact in a continuous setting (*i.e.*, when the step sizes in time, state and action go to 0) but only approximate in the discrete one considered here. Hence, we choose instead to use the distribution estimated by exact tabular OMD as a benchmark, as it reaches an exploitability

of 10^{-12} in our experiments. Figure 8.2 (bottom right) shows that the Wasserstein distance between the learnt distribution by D-MOMD and the benchmark decreases as the learning occurs. In Figure 8.2 (bottom left), we see that D-MOMD and D-AFP outperform other methods in minimizing exploitability.

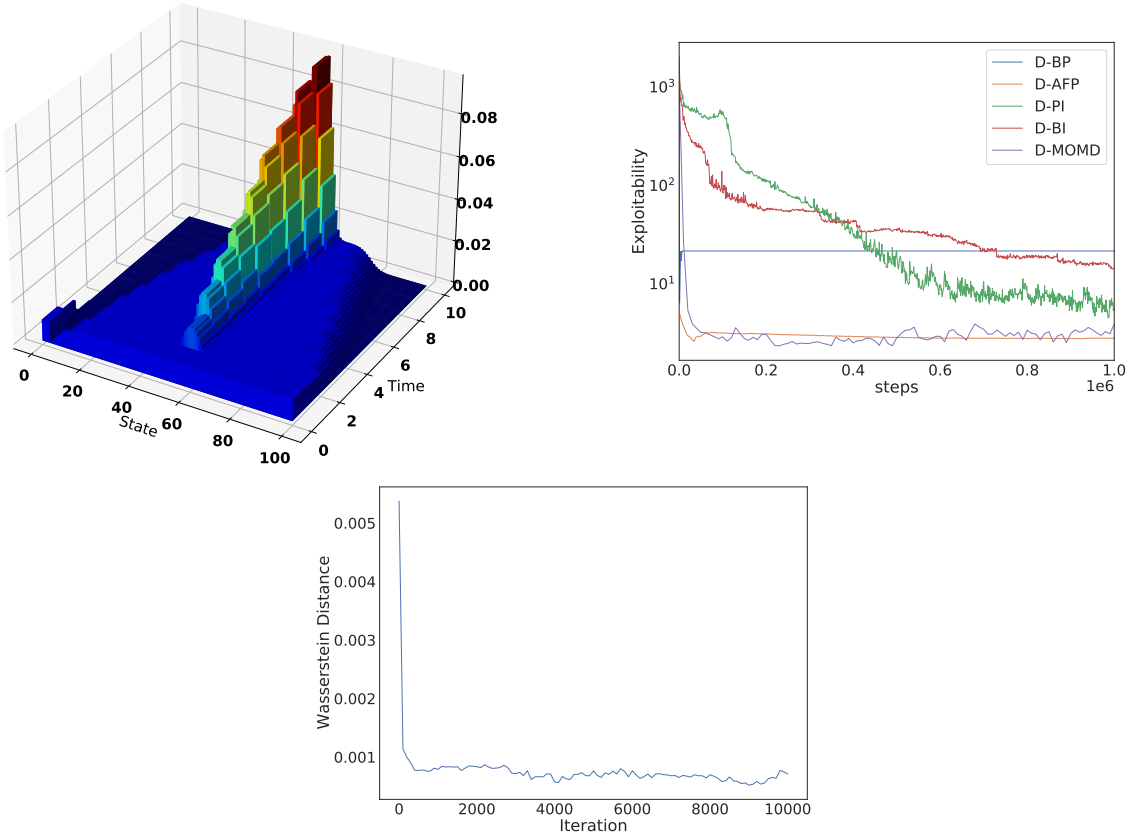


Figure 8.2 – Top: Evolution of the distribution generated by the policy learnt by D-MOMD. Bottom left: Exploitability of different algorithms on the Linear Quadratic environment. Bottom right: Wasserstein distance between the solution learnt by D-MOMD and the benchmark one, over its iterations.

Exploration.

We now increase the state dimension and turn our attention to a 2-dimensional grid world example. The state is the position. An action is a move, and valid moves are: left, right, up, down, or stay, as long as the agent does not hit a wall. In the experiments, we consider 10×10 states and a time horizon of $N_T = 40$ time steps. The reward is: $r(x, a, \mu) = r_{\text{pop}}(\mu(x))$, where $r_{\text{pop}}(\mu(x)) = -\log(\mu(x))$ discourages being in a crowded area – which is referred to as crowd aversion. Note that $\mathbb{E}_{x\sim\mu}(r_{\text{pop}}(\mu(x))) = \mathcal{H}(\mu)$, *i.e.*, the last term of the reward provides, in expectation, the entropy of the distribution. This setting is inspired by the one considered by Geist, Pérolat, et al. (2021). The results are shown in Figure 8.3. D-MOMD and D-AFP

outperform all the baselines. The induced distribution matches our intuition: it spreads symmetrically until filling almost uniformly the four rooms.

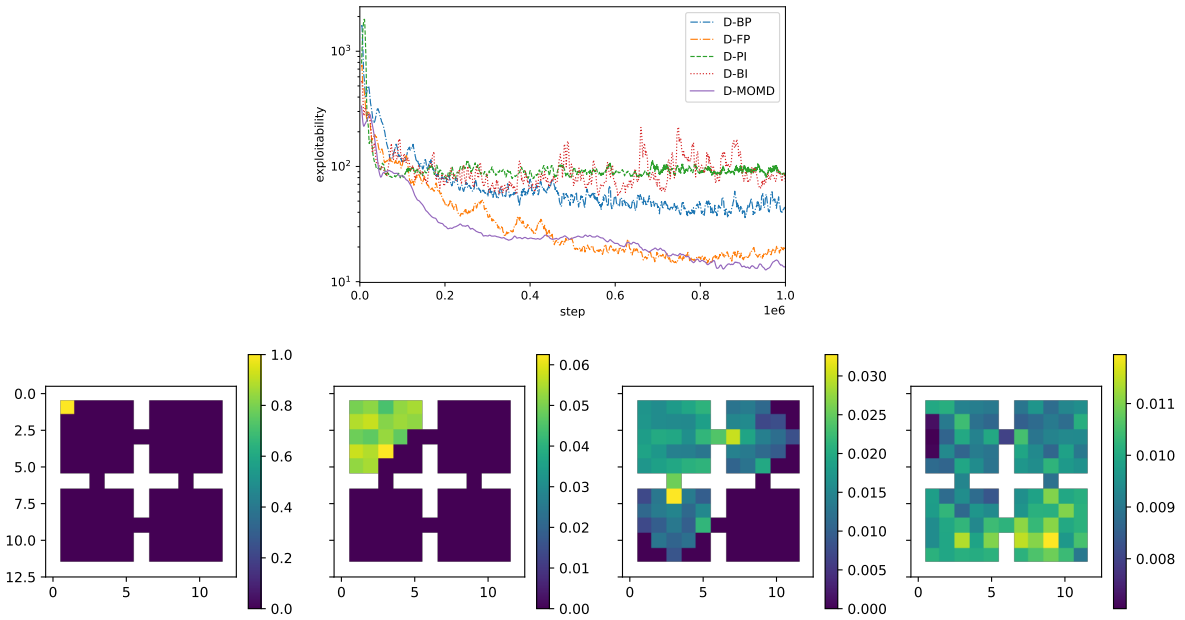


Figure 8.3 – Top: exploitability. Bottom: evolution of the distribution obtained by the policy learnt with D-MOMD.

Crowd modeling with congestion. We consider the same environment but with a maze, and a more complex reward function:

$$r(x, a, \mu) = r_{\text{pos}}(x) + r_{\text{move}}(a, \mu(x)) + r_{\text{pop}}(\mu(x)),$$

where $r_{\text{pos}}(x) = -\text{dist}(x, x_{\text{ref}})$ is the distance to a target position x_{ref}, $r_{\text{move}}(a, \mu(x)) = -\mu(x)\|a\|$ is a penalty for moving ($\|a\| = 1$) which increases with the density $\mu(x)$ at x – which is called congestion effect in the literature. The state space has 20×20 states, and the time horizon is $N_T = 100$. We see in Figure 8.4 that D-MOMD outperforms the other methods.

Multi-population chasing. We finally turn to an extension of the MFG framework, where agents are heterogeneous: each type of agent has its own dynamics and reward function. The environment can be extended to model multiple populations by simply extending the state space to include the population index on top of the agent's position. Following Perolat, Perrin, et al. (2021), we consider three populations and rewards of the form: for population $i = 1, 2, 3$,

$$r^i(x, a, \mu^1, \mu^2, \mu^3) = -\log(\mu^i(x)) + \sum_{j \neq i} \mu^j(x) \bar{r}^{i,j}(x).$$

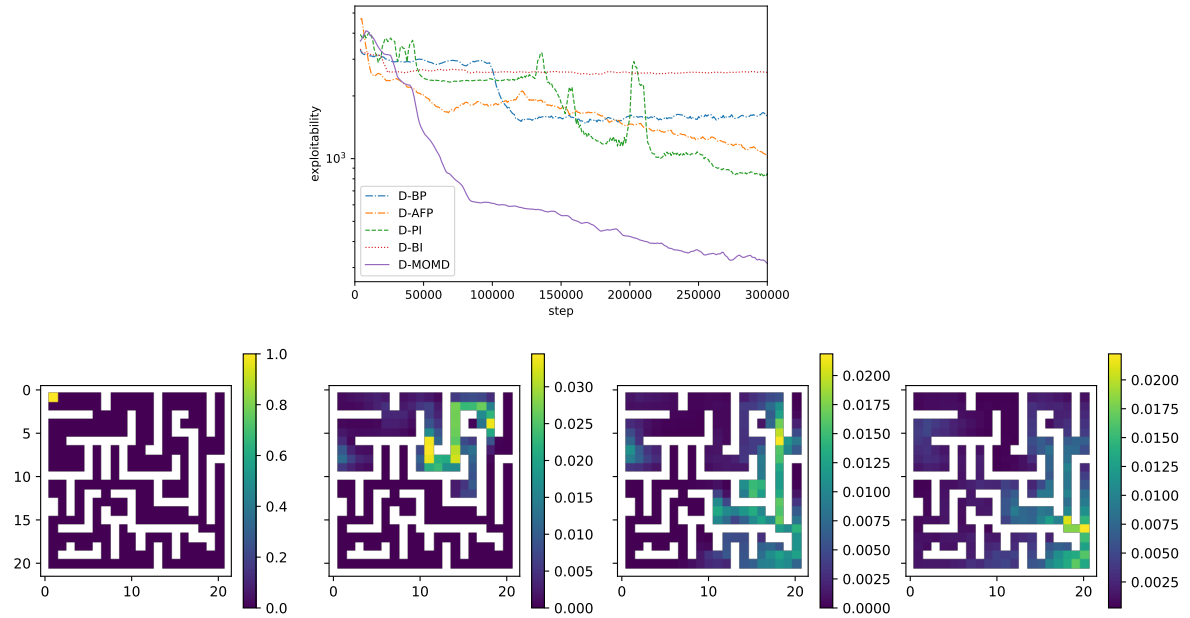


Figure 8.4 – Maze example. Top: exploitability. Bottom: evolution of the distribution obtained by the policy learnt with D-MOMD.

where $\bar{r}^{i,j} = -\bar{r}^{j,i}$, with $\bar{r}^{1,2} = -1$, $\bar{r}^{1,3} = 1$, $\bar{r}^{2,3} = -1$. In the experiments, three are 5×5 states and the time horizon is $N_T = 10$ time steps. The initial distributions are in the corners, the number of agents of each population is fixed, and the reward encourages the agent to chase the population it dominates and flee the dominating one. We see in Figure 8.5 that D-AFP outperform the baselines and D-MOMD performs even better.

8.5 Conclusion of the Chapter

We proposed two scalable algorithms that can compute Nash equilibria in various MFGs in the finite horizon setting. The first one, D-AFP, is the first implementation of Fictitious Play for MFGs that does not need to keep all previous best responses in memory and that learns an average policy with a single neural network. The second one, D-MOMD, takes advantage of a subtle reparameterization to learn implicitly a sum of Q-functions usually required in the Online Mirror Descent algorithm. We demonstrated numerically that they both perform well on five benchmark problems and that D-MOMD consistently performs better than D-AFP as well as three baselines from the literature.

In our D-OMD algorithm the policy is computed using a softmax, which is reminiscent of maximum entropy RL methods (Todorov, 2008; Toussaint, 2009; Rawlik, Toussaint, and Vijayakumar, 2012) which led to efficient deep RL methods such as soft actor critic (Haarnoja

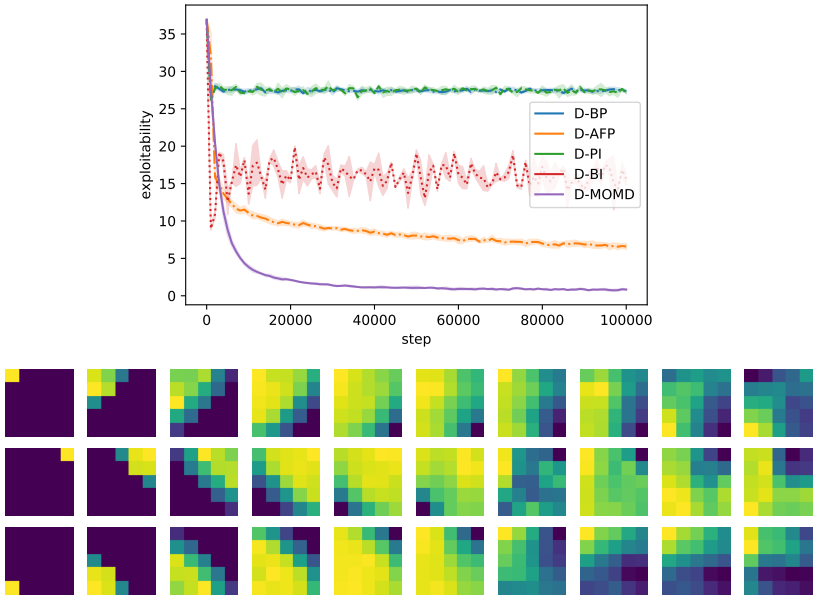


Figure 8.5 – Multi-population chasing example. Top: Exploitability. Bottom: evolution of the distributions for the three populations.

et al., 2018). However there is a crucial difference: here we use a KL divergence with respect to the previous policy, as can be seen in (8.4).

Future work. We would like to include more complex examples, with larger state spaces or even continuous spaces. Continuous state spaces should be relatively easy to address, as neural networks can handle continuous inputs, while continuous actions would require some adjustments particularly to compute argmax or softmax of Q-functions. Furthermore continuous spaces require manipulating continuous population distributions, raising additional questions related to how to represent and estimate them efficiently.

Conclusion of Part III

In Part III, we examined the last question: *How to adapt the algorithms to a model-free setting, using deep reinforcement learning?* While Part I and Part II were dedicated to answering the first question, mostly taking into consideration what we refer to as the first meaning of learning, *i.e.* the game-theoretic meaning, in Part III, we tackle the second meaning, *i.e.* from the machine learning perspective.

Chapter 6 is a proof of work that fictitious play can be efficiently combined with deep (reinforcement) learning. In fact, we demonstrate that replacing the computation of the best response with a DRL algorithm (here soft actor-critic) and the estimation of the distribution with a generative model (here normalizing flows) still allows to learn a Nash equilibrium despite the various approximations introduced. The overall approach is completely model-free and solve up to a six-dimensional continuous state space, three-dimensional action space, with complex geometry. This problem could not have been solved with standard methods of mean field games. However, it remains limited to a (γ-)stationary setting; because estimating one distribution per time step as required in a finite-horizon setting seems out-of-reach without access to powerful resources. The question of applicability then arises: what is the point of using this method in a setting where the agents are only optimal against the (γ)-stationary distribution? Real-life systems are more often than not evolutive, and possible deviations from a Nash equilibrium are likely to happen.

Therefore, in Chapter 7, we introduced the new paradigm of population-dependant policies. This chapter mostly studies a third variation of what learning means, under the prism of generalization. Population-dependant policies allow to take into consideration the current state of the population, by incorporating directly into the policy the distribution of agents. We demonstrate that under the monotonicity condition, and assuming that the representative agent can start from every possible initial distribution, there exists a population-dependant policy, that we name the Master policy, able to be optimal against any distribution of agents. In practice, as it is not possible to start from infinitely many distributions, we build a set of initial distributions and demonstrate experimentally that agents learning with the Master fictitious play algorithm learn an approximate Nash equilibrium in many configurations, even in some not observed during training. Although the approach is the first to tackle the question of generalization,

the overall method is computationally expensive; converging to a solution requires days of training even in simple examples with finite state and action spaces. In particular, the deep reinforcement learning adaptation of fictitious play is not scalable, in the sense that it needs to keep in memory all best responses computed along the iterations of the algorithm.

Our last contribution thus proposed a scalable version of fictitious play and online mirror descent. In the two previous contributions, although we were aware that online mirror descent could probably converge faster to a Nash equilibrium than fictitious play, adapting the algorithm to a deep reinforcement learning setting was not straightforward. In Chapter 8, we introduced two algorithms: Deep Average Fictitious Play (D-AFP) and Deep Munchausen Online Mirror Descent (D-MOMD) that do not require to keep in memory all previous policies or Q-functions. D-AFP approximates the average policy by keeping transitions of all iterations in a buffer and optimizing a categorical loss in a supervised learning fashion. D-MOMD leverages the Legendre-Fenchel transform and regularizes directly the Q-function, allowing to get rid of the previous sum. Consistently with results in the exact case, D-MOMD converges faster than D-AFP and other baselines, suggesting strongly that it should be used from now on in applications.

The contributions in this last part clearly validate that Reinforcement Learning can scale up Mean Field Games, both in terms of model complexity (Chapter 6), generalization (Chapter 7) and algorithms (Chapter 8).

Chapter 9

General Conclusion and Perspectives

9.1 Conclusion on our Contributions

In this thesis, we have built brick by brick all the ingredients to solve complex multi-agent problems in real-world settings. Our first question was *How to design algorithms to find Nash equilibria in mean field games?*, let us see what elements of answer we brought.

In Chapter 2, we have introduced the necessary background to the understanding of the dissertation. We started by introducing Markov Decision Processes in both stationary and finite horizon, followed by Reinforcement Learning and Deep Reinforcement Learning basics. We also defined properly the different settings that one may encounter when using mean-field approximations with reinforcement learning. This chapter, along with Chapter 3 are based on Laurière, Perrin, et al. (2022).

Then, Chapter 3 spelled out iterative methods, why fixed point iterations easily fail and proposed a unified approach that summarizes the different types of regularization one can introduce to stabilize convergence. We concluded Part I with a numerical comparison of several iterative methods and showed that fictitious play and online mirror descent converge to the Nash equilibrium in the considered example. Consequently, Part II was dedicated to a more detailed study of these two algorithms.

Chapter 4, based on Perrin, Perolat, et al. (2020), introduced a continuous-time version of fictitious play in order to prove the convergence to the Nash equilibrium under the monotonicity condition and even exhibited a convergence rate thanks to Lyapunov theory. It studied the finite horizon setting as well as the γ-discounted one, with and without common noise. We also introduced a discrete time version which we implemented, and verified experimentally that the exploitability of the algorithm went towards zero, which ensures the convergence to the equilibrium.

Chapter 5 was dedicated to the study of Online Mirror Descent and proved its convergence with continuous-time updates to the Nash equilibrium under the monotonicity condition, in the finite horizon setting. We proved that the algorithm also works with common noise and with multiple populations. This chapter drew on Perolat, Perrin, et al. (2021) which was published at AAMAS 2022. Therefore, Part I and Part II brought clear answers to the first question, as using either FP or OMD allows to find Nash equilibrium in mean field games.

The last part of the thesis was dedicated to the third question: *How to adapt these algorithms to a model-free setting, using deep reinforcement learning?* Based on Perrin, Laurière, Pérolat, Geist, et al. (2021), in Chapter 6 we demonstrated that a deep reinforcement learning adaptation of Fictitious Play was able to converge in the complex continuous multi-dimensional example of flocking, even without the monotonicity condition. We believe that using neural networks, *i.e.* replacing the computation of the best response by the SAC algorithm and learning the population's distribution with a Normalizing Flow naturally leads to the selection of a single equilibrium, even in games with potentially an infinite amount of equilibria. However, our application was limited to the (γ-)stationary setting and scaling this method to a finite horizon where we need a different policy and mean field state at every time step is not feasible. Furthermore, it is restricted to a single initial distribution which limits its possible applications to real-world problems.

It justified the introduction of the concept of master policy in Chapter 7, which allows to be optimal against many different initial distributions by letting the policy depending on the current distribution of agents. Based on Perrin, Laurière, Pérolat, Élie, et al. (2021), this work is yet still using fictitious play in a non-scalable way, *i.e.* requiring to keep in memory all past best responses. Although we demonstrated numerically that the approach worked in some simple examples, the computational cost of the algorithm prevented us to use this approach in more complex games.

Our last contribution is Lauriere et al. (2022) in Chapter 8. It proposed two deep reinforcement learning adaptations that finally allowed to use fictitious play and online mirror descent in a scalable way, *i.e.* without requiring to store every past policy of Q-function. Deep Averaged Fictitious Play (D-AFP) scaled up fictitious play by keeping a replay buffer containing transitions for all past best responses. Minimizing a categorical over this dataset in a supervised learning way allowed to directly approximate the average policy that we needed. Deep Munchausen Online Mirror Descent (D-MOMD) is a new state-of-the-art for Mean Field Games with Reinforcement Learning. It took advantages of a clever reparameterization based upon Legendre-Fenschel transform that regularizes directly the Q-function instead of having to sum them up. We compared these algorithms with three baselines in a variety of games and open-sourced everything in OpenSpiel (Lanctot, Lockhart, et al., 2019).[1], hoping that it

[1] https://github.com/deepmind/open_spiel

will incite other researchers to contribute, which would be very beneficial for the community as numerical results are not always easy to compare. I personally hope that this thesis will contribute to harmonizing the research on this topic although many aspects remain to be unified, and that it will inspire other people to try to extend even further the field.

9.2 Future Work and Perspectives

I would like to finish this dissertation with a more personal point of view about what remains to be done and how our work could be applied to real-world scenarios. Although our research is mostly theoretical, mean field games can model a wide variety of situations and the ultimate goal remains to use these tools for practical applications. Reinforcement learning, on its hand, aims at developing algorithms for real-world problems; this was indeed the very first reason why we decided to solve mean field games with (model-free) reinforcement learning. Cabannes et al. (2021), which studies traffic routing with the lens of mean field games, is a first step towards applying MFGs to such scenarios. However, it supposes a perfect knowledge of the model and it is not clear that it would work better in practice than methods based on heuristics. Therefore, I believe there are several points that need to be addressed before being able to deploy our methods to other applications.

Approximation of the distribution. Approximating the distribution has received relatively less interest than the question of approximating the policy and the value function. Efficiently representing and learning the distribution is important for mean field problems, particularly for large or continuous environments for which exact tabular representations are not suitable. In Chapter 6, we have used normalizing flows to learn the continuous distribution of agents over their positions and velocities in a γ-discounted setting. However, it does not seem scalable to use this method for evolutive settings, as it would require either to learn a different normalizing flow at every time step or to condition the normalizing flow on time. We did some preliminary tests following both of these approaches, but they did not give satisfying results. But approximating distributions is a central problem in Machine Learning and there are numerous methods for doing it, which makes me believe that other methods could efficiently solve this problem. Although exploring this question for mean field games remains largely open, I believe the next priority is to tackle this question before hoping to apply our algorithms to very large or continuous domains.

Types of equilibria. The second question of importance concerns the type of solutions we are looking for. On one hand, Nash equilibria model purely competitive games, in which the players are solely interested in maximizing their own reward. On the other hand, Pareto

optimum can be reached in collaborative settings with perfect communication, or when a central controller can decide exactly on the behavior of all players. However, there is evidence that equilibria in real-world problems are often in-between these two notions. In a wide variety of problems, people do not act selfishly, while even in completely collaborative games, a perfect communication and coordination are not always possible between players. We recently contributed to extend the notion of (coarse) correlated equilibria from Aumann (1987) to Mean Field Games (Muller, Elie, et al., 2022). The general idea of correlated equilibria is to have a correlation device that recommends an action to each player, in order to coordinate the population's action. The player can decide to follow or not this recommendation: it may have a partial misalignment with correlation device, but taking its decision without the mediator's knowledge can be dangerous. The classical example is the one of traffic lights, where it is obviously very dangerous not to respect the lights but deviating can sometimes save you time. This work, along with Muller, Rowland, et al. (2021) and X. Wang et al. (2022), are a first step to find more subtle notions of solutions that could be more adapted to real applications.

What is the right model? In the dissertation, we took as granted a model of a game and focused mainly on finding a Nash equilibrium given this fixed model. We proved that when a reward function and an environment are provided, our methods can find such equilibrium. We can now take a step back and wonder how models are designed, and if they always lead to desirable outcomes. If designing meaningful reward functions is easy in games such as Go or video games wherein there is an intrinsic notion of what *winning* or *losing* means, it is less straightforward to know what is the good reward function in economics or societal problems. Furthermore, when there exists several equilibria in such games, there is no guarantee that the algorithm will converge to a good one in terms of social benefit. To circumvent this difficulty, a possible approach is to use *Mechanism design*, which provides ways of influencing the behavior of players. For example, Balaguer et al. (2022) propose a method where an agent or a mechanism allows to shape other agents behavior in order to reach a better equilibrium. This method could maybe be extended to a mean field setting. Another idea to shape more realistic models would be to use data-oriented methods and infer directly the reward and transition probabilities from the data.

Gamification. Lastly, another orthogonal research question I would like to explore concerns gamification of machine learning problem. The general principle of gamification is to rewrite or *gamify* a problem as a game, which would be here a mean field game. Once the problem has been casted in as an MFG, it is straightforward to apply all existing results of the MFG theory to this newly formulated problem. This is exactly what we did in Geist, Pérolat, et al. (2021) with concave utility reinforcement learning, allowing to draw new connections between fields and prove new theoretical results. I would like to try to do the same in other domains such as

generative modeling, and explore how and if diffusion models, which are based on stochastic equations, could be cast in as an MFG or MFC problem.

Appendix A

Complements on Chapter 2

A.1 Some applications

Mean Field Games have found applications in various domains such as population dynamics (Guéant, Lasry, and Lions, 2011; Achdou, Bardi, and Cirant, 2017; Cardaliaguet, Porretta, and Tonon, 2016), crowd motion modeling (Achdou and Lasry, 2019; Burger et al., 2013; Djehiche, Tcheukam, and Tembine, 2017; Aurell and Djehiche, 2019; Achdou and Laurière, 2016; Chevalier, Le Ny, and R. Malhamé, 2015), flocking (Nourian, Caines, and R. P. Malhamé, 2010; Nourian, Caines, and R. P. Malhamé, 2011; Grover, Bakshi, and Theodorou, 2018; Perrin, Laurière, Pérolat, Geist, et al., 2021), opinion dynamics and consensus formation (Stella et al., 2013; Bauso, Tembine, and Basar, 2016; Parise et al., 2015), autonomous vehicles (K. Huang et al., 2017; Shiri, Park, and Bennis, 2019), epidemics control (Laguzet and Turinici, 2015; E. Hubert and Turinici, 2018; Elie, E. Hubert, and Turinici, 2020; Lee et al., 2021; Aurell, Carmona, et al., 2022; Doncel, Gast, and Gaujal, 2022). But MFGs have also naturally found applications in banking, finance and economics including banking systemic risk (Carmona, Fouque, and L.-H. Sun, 2015b; Elie, Ichiba, and Laurière, 2020), high frequency trading (Lachapelle, Lasry, et al., 2016; Cardaliaguet and Lehalle, 2018), income and wealth distribution (Achdou, Han, et al., 2017), economic contract design (Elie, Mastrolia, and Possamai, 2019), economics in general (Achdou, Han, et al., 2017; Achdou, Buera, et al., 2014; Chan and Sircar, 2015; D. Gomes, Velho, and Wolfram, 2014; Djehiche, Tcheukam Siwe, and Tembine, 2017), price formation (Lasry and Lions, 2007; Lachapelle, Lasry, et al., 2016; D. A. Gomes and Saúde, 2020), finance in general (Cardaliaguet and Lehalle, 2018; Lasry and Lions, 2007; Carmona, 2020), energy production and management (Alasseur, Taher, and Matoussi, 2020; Couillet et al., 2012; Elie, E. Hubert, Mastrolia, et al., 2019; Bagagiolo and Bauso, 2014; Kizilkale, Salhab, and R. P. Malhamé, 2019; F. Li, R. P. Malhamé, and Le Ny, 2016; Guéant, Lasry, and Lions, 2011; Achdou, Giraud, et al., 2016; Chan and Sircar, 2017; Graber and Bensoussan, 2018), security and communication (Mériaux, Varma, and Lasaulce, 2012; Samarakoon et al., 2015; Hamidouche et al., 2016; C.

Yang et al., 2017; Kolokoltsov and Bensoussan, 2016; Kolokoltsov and Malafeyev, 2018), traffic modeling (Bauso, X. Zhang, and Papachristodoulou, 2016; Salhab, Le Ny, and R. P. Malhamé, 2018; K. Huang et al., 2019; Tanaka et al., 2020; Cabannes et al., 2021) or engineering (Djehiche, Tcheukam Siwe, and Tembine, 2017).

A.2 An introduction to MFGs in OpenSpiel

We distinguish **models** (games or environments in the sense of RL) and **algorithms**. Intuitively, a game contains everything that is needed to define the model, whereas an algorithm is dedicated to computing a solution to the problem.

For now, OpenSpiel focuses primarily on the evolutive setting. The stationary setting is indirectly supported through transforming stationary games into evolutive ones, and other settings could also be implemented. In the sequel, we restrict our attention to evolutive MFGs.

A.2.1 Models

Games can be implemented in C++ or in python. We discuss here the implementation in python, but implementation in C++ follows the same lines. The games are located in open_spiel/tree/master/open_spiel/python/mfg/games. To compute the evolution of one player's state, the dynamics can be viewed as sequence of nodes. One round consists in updating the representative player's state and the mean field state. For the sake of consistency with other games implemented in OpenSpiel, we represent this evolution using the notion of player. The idea is that each player influences one transition between two nodes, in turn. For MFGs, the representative player and the population are encoded as two different players, respectively called `DEFAULT_PLAYER_ID` and `MEAN_FIELD`. Furthermore, the randomness appearing in the dynamics is also encoded as a player, called `CHANCE`. This is the main difference with the way MFGs are presented in this survey (and more generally in the literature). Randomness at initial time is also encoded as an action of the chance player.

Another difference is that the end of the game (*e.g.*, the finite horizon) is encoded through the notion of terminal state. The game starts from an initial state and runs until a terminal state is reached. This can be used to represent finite-horizon MFGs by considering that the state of the game is not only the state of the representative player but also the time index. In this way, we can define the terminal states of the game as all the states for which the time index is equal to the time horizon.

From an implementation viewpoint, we stress the following points. We can use the `crowd_modeling`[1] game as typical example.

- The two main building blocks are:
  - One class for the game, which inherits from `pyspiel.Game`.
  - One class for the state, which inherits from `pyspiel.State`
- The state's evolution is implemented in the state's class.
- The basic order of nodes is:

  $$\texttt{CHANCE} \xrightarrow{\texttt{\_apply\_action}} \texttt{MEAN\_FIELD} \xrightarrow{\texttt{update\_distribution}} \texttt{DEFAULT\_PLAYER\_ID} \xrightarrow{\texttt{\_apply\_action}} \texttt{CHANCE}\ldots$$

  where we write over the arrow the method that is used for the update. These two methods are:
  - `_apply_action`: it takes an action as an input and uses it to update the state; notice that not only the representative player but also the can player can take actions, the actions of the chance player representing the randomness from the environment that influences the evolution of the representative player's state.
  - `update_distribution`: it updates the distribution using the one that is passed as an input (and which needs to be computed externally)
- Other functions of the state class include:
  - `chance_outcomes`: it returns the probabilities of all the actions when at a chance node; this can be viewed as the (fixed) policy of the chance player
  - `_legal_actions`: it returns the actions are valid in the current state; this can be used to forbid some actions; for example, in a grid world, some movements can be forbidden due to the presence of obstacles
  - `_rewards`: it computes the reward of the representative player in the current state

A.2.2 Algorithms

The algorithms for MFGs are located in:
open_spiel/tree/master/open_spiel/python/mfg/algorithms.

To implement algorithms, some important auxiliary files are:

- `greedy_policy.py`: allows to compute the greedy policy with respect to a Q-function.

[1] https://github.com/deepmind/open_spiel/blob/master/open_spiel/python/mfg/games/crowd_modelling.py

- `distribution.py`: the class `DistributionPolicy` contains a tabular representation of a distribution associated to a given policy.
- `nash_conv.py`: computes the exploitability of a policy.
- `policy_value.py`: the class `PolicyValue` allows to compute the value of a policy.

We discuss here the fictitious play algorithm as an example.

At the moment, existing algorithms for MFGs in OpenSpiel are:

- Fictitious play as introduced in Elie, Perolat, et al., 2020 (based on the continuous MFGs version of Cardaliaguet and Hadikhanloo, 2017): see `fictitious_play.py`; it also covers fixed point (damped fixed point) with `learning_rate` equal to 1.0 (resp. in $(0, 1)$) in the `iteration` method
- Online mirror descent as introduced in Perolat, Perrin, et al., 2021: see `mirror_descent.py`
- Deep RL fictitious play with average neural network as introduced in Lauriere et al., 2022: see `average_network_fictitious_play.py`
- Deep Munchausen Online Mirror Descent as introduced in Lauriere et al., 2022: see `munchausen_deep_mirror_descent.py`

Appendix B

Complements on Chapter 4

B.1 Continuous Time Fictitious Play in Finite Horizon

In this section, we prove the Fictitious Play convergence result in the absence of common noise. For the sake of clarity, we will write:

$$r^{\pi}(x,\mu) = \mathbb{E}_{a\sim\pi(.|x)}\left[r(x,a,\mu)\right] \quad \text{and} \quad p^{\pi}(x'|x) = \mathbb{E}_{a\sim\pi(.|x)}\left[p(x'|x,a)\right]$$

for the rest of this section.

First, we prove the following property, which stems from monotonicity.

Property 1. *Let f be a smooth enough function and let assume that the ODE $\dot{\mu} = f(\mu)$ (with $\dot{\mu} = \frac{d}{dt}\mu$) has a solution $(\mu^t)_{t\geq 0} = (\mu^t_n(x))_{t\geq 0, x\in\mathcal{X}}$. If the game is monotone, then:*

$$\sum_{x\in\mathcal{X}} < \nabla_{\mu}\bar{r}(x,\mu), \dot{\mu} > \dot{\mu}(x) \leq 0.$$

Proof. The monotonicity condition implies that, for all $\tau \geq 0$, we have:

$$\sum_{x\in\mathcal{X}} (\mu^t(x) - \mu^{t+\tau}(x))(\bar{r}(x,\mu^t) - \bar{r}(x,\mu^{t+\tau})) \leq 0.$$

Thus:

$$\sum_{x\in\mathcal{X}} \frac{\mu^t(x) - \mu^{t+\tau}(x)}{\tau} \frac{\bar{r}(x,\mu^t) - \bar{r}(x,\mu^{t+\tau})}{\tau} \leq 0.$$

The result follows when $\tau \to 0$. □

Property 2. *Let $\hat{\boldsymbol{\pi}}^t = (\hat{\pi}^t_n)_{n=0,\dots,N}$ be a sequence of time-dependent policies and let*

$$\boldsymbol{\mu}^{\hat{\boldsymbol{\pi}}^t} = (\mu^{\hat{\pi}^t}_n(x))_{n=0,\dots,N, x\in\mathcal{X}}$$

be the sequence of their distributions over states. Let us denote, for all t, n, x, $\bar{\mu}_n^t(x) = \frac{1}{t}\int_{s=0}^{t} \mu_n^{\hat{\pi}^s}(x)ds$. *Then, the policy generating this average distribution is:*

$$\bar{\pi}_n^t(a|x) = \frac{\int_{s=0}^{t} \mu_n^{\hat{\pi}^s}(x)\hat{\pi}_n^s(a|x)ds}{\int_{s=0}^{t} \mu_n^{\hat{\pi}^s}(x)ds}. \tag{B.1}$$

Note that $\int_0^t \mu_n^{\hat{\pi}^s}(x)ds$ *can be chosen to be strictly positive as one can choose an arbitrary policy on the time interval* $[0, 1]$ *(for example, the uniform policy).*

Or, more simply, one can write:

$$\bar{\mu}_n^t(x)\bar{\pi}_n^t(a|x) = \frac{1}{t}\int_{s=0}^{t} \mu_n^{\hat{\pi}^s}(x)\hat{\pi}_n^s(a|x)ds. \tag{B.2}$$

Moreover, we have:

$$\dot{\bar{\mu}}_n^t(x)\bar{\pi}_n^t(a|x) + \bar{\mu}_n^t(x)\dot{\bar{\pi}}_n^t(a|x) = \frac{1}{t}\left[\mu_n^{\hat{\pi}^t}(x)\hat{\pi}^t(a|x) - \bar{\mu}_n^{\hat{\pi}^t}(x)\bar{\pi}_n^t(a|x)\right]. \tag{B.3}$$

Proof. Let us start with the following equality, which holds by definition of the dynamics:

$$\mu_{n+1}^{\hat{\pi}^s}(x') = \sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} p(x'|x,a)\hat{\pi}_n^s(a|x)\mu_n^{\hat{\pi}^s}(x). \tag{B.4}$$

Then, taking on both sides the average over the Fictitious Play time yields:

$$\frac{1}{t}\int_{s=0}^{t} \mu_{n+1}^{\hat{\pi}^s}(x')ds = \sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} p(x'|x,a)\frac{1}{t}\int_{s=0}^{t} \hat{\pi}_n^s(a|x)\mu_n^{\hat{\pi}^s}(x)ds. \tag{B.5}$$

The left hand side is $\bar{\mu}_{n+1}^t(x')$ by definition, and the time average in the right hand side can be written as:

$$\frac{1}{t}\int_{s=0}^{t} \hat{\pi}_n^s(a|x)\mu_n^{\hat{\pi}^s}(x)ds = \frac{\int_{s=0}^{t} \hat{\pi}_n^s(a|x)\mu_n^{\hat{\pi}^s}(x)ds}{\int_{s=0}^{t} \mu_n^{\hat{\pi}^s}(x)ds}\frac{1}{t}\int_{s=0}^{t} \mu_n^{\hat{\pi}^s}(x)ds = \bar{\mu}_n^{\hat{\pi}^t}(x)\bar{\pi}_n^t(a|x).$$

Combining the terms, we obtain:

$$\bar{\mu}^t_{n+1}(x') = \sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} p(x'|x,a)\bar{\mu}^{\hat{\pi}^t}_n(x)\bar{\pi}^t_n(a|x),$$

which proves that the policy $\bar{\pi}^t_n$ defined in (B.1) indeed generates $\bar{\mu}^t_n$. The other equalities in the statement can be deduced from here readily. □

Based on the above properties, we now proceed to the proof of the convergence of Fictitious Play (Theorem 4.1) in the finite horizon case.

Proof of Theorem 4.1. To alleviate the notation, given a policy π, we denote:

$$r^\pi(x,\mu) = \sum_a \pi(a|x)r(x,a,\mu).$$

We start by noticing that, thanks to the structure of the reward coming from the monotonicity assumption,

$$\nabla_\mu r^{\pi^{\text{BR},t}_n}(x,\bar{\mu}^t_n) = \nabla_\mu \bar{r}(x,\bar{\mu}^t_n) \text{ and } \nabla_\mu r^{\pi_n}(x,\bar{\mu}^t_n) = \nabla_\mu \bar{r}(x,\bar{\mu}^t_n). \tag{B.6}$$

Moreover, from Property 2 with π replaced by $\hat{\pi}^{\text{BR}}$ and $\mu^{\hat{\pi}^t}$ replaced by $\mu^{\text{BR},t}$, we obtain (B.2). Dropping the overlines to alleviate the presentation (so μ^t and π^t denote respectively the average sequence of distributions and the average sequence of policies), it implies:

$$\mu^t_n(x)\frac{d}{dt}\pi^t_n(a|x) = \frac{1}{t}\mu^{\text{BR},t}_n(x)[\pi^{\text{BR},t}_n(a|x) - \pi^t_n(a|x)]. \tag{B.7}$$

Moreover, recall that:

$$\frac{d}{dt}\mu^t_n(x) = \frac{1}{t}\left[\mu^{\text{BR},t}_n(x) - \mu^t_n(x)\right]. \tag{B.8}$$

From the above observations, we deduce successively:

$$\begin{aligned}
\frac{d}{dt}\phi(\pi^t) &= \frac{d}{dt}\left[\max_{\pi'} J(m_0,\pi',\mu^t) - J(m_0,\pi^t,\mu^t)\right] \\
&= \sum_{n=0}^{N}\sum_{x\in\mathcal{X}}\Big[< \nabla_\mu r^{\pi^{\text{BR}}_n}(x,\mu^t_n), \frac{d}{dt}\mu^t_n > \mu^{\text{BR},t}_n(x) - < \nabla_\mu r^{\pi_n}(x,\mu^t_n), \frac{d}{dt}\mu^t_n > \mu^t_n(x) \\
&\qquad\qquad - < \frac{d}{dt}\pi^t_n(.|x), r(x,.,\mu^t_n) > \mu^t_n(x) - r^{\pi_n}(x,\mu^t_n)\frac{d}{dt}\mu^t_n(x)\Big] \\
&= \sum_{n=0}^{N}\sum_{x\in\mathcal{X}}\left[t < \nabla_\mu \bar{r}(x,\mu^t_n), \frac{d}{dt}\mu^t_n > \frac{1}{t}\left(\mu^{\text{BR},t}_n(x) - \mu^t_n(x)\right)\right] \\
&\quad + \sum_{n=0}^{N}\sum_{x\in\mathcal{X}}\left[\frac{1}{t}r^{\pi_n}(x,\mu^t_n)\mu^t_n(x) - \frac{1}{t}r^{\pi^{\text{BR},t}_n}(x,\mu^t_n)\mu^{\text{BR},t}_n(x)\right]
\end{aligned}$$

$$= -\frac{1}{t}\phi(\pi^t) + \sum_{n=0}^{N}\sum_{x\in\mathcal{X}}\left[t < \nabla_\mu \bar{r}(x,\mu_n^t), \frac{d}{dt}\mu_n^t > \frac{d}{dt}\mu_n^t(x)\right],$$

where the third equality holds by (B.6), (B.7) and (B.8). Note that the product $< \nabla_\mu \bar{r}(x,\mu_n^t), \frac{d}{dt}\mu_n^t >$ in the last sum above is non-positive thanks to Property 1 (*i.e.*, thanks to the monotonicity assumption). Hence, the conclusion holds. □

B.2 Continuous Time Fictitious Play in Finite Horizon with Common Noise

In this section, we prove the convergence result of continuous time fictitious play in finite horizon MFGs with common noise (Theorem 4.2). The reasoning is similar as in the finite horizon case without common noise (Section B.1). The only difference comes from the conditioning with the common noise.

Proof of Theorem 4.2. For any policy, recall that we write $\pi^t_{n,\Xi}(a|x) = \pi^t_n(a|x,\Xi)$.

We first note that, by the structure of the reward function, we have,

$$\nabla_\mu r^{\pi^{\text{BR},t}_{n,\Xi.\xi}}(x,\mu^t_{n|\Xi}) = \nabla_\mu \bar{r}(x,\mu^t_{n|\Xi}) \text{ and } \nabla_\mu r^{\pi_{n,\Xi.\xi}}(x,\mu^t_{n|\Xi}) = \nabla_\mu \bar{r}(x,\mu^t_{n|\Xi}).$$

Moreover,

$$-<\frac{d}{dt}\pi^t_{n,\Xi.\xi}(.|x), r(x,.,\mu^t_{n|\Xi})>\mu^t_{n|\Xi}(x) = -\frac{1}{t}r^{\pi^{\text{BR},t}_{n,\Xi.\xi}}(x,\mu^t_{n|\Xi})\mu^{\text{BR},t}_{n|\Xi}(x) + \frac{1}{t}r^{\pi_{n,\Xi.\xi}}(x,\mu^t_{n|\Xi})\mu^{\text{BR},t}_{n|\Xi}(x)$$

and

$$-r^{\pi_{n,\Xi.\xi}}(x,\mu^t_{n|\Xi})\frac{d}{dt}\mu^t_{n|\Xi}(x) = \frac{1}{t}r^{\pi_{n,\Xi.\xi}}(x,\mu^t_{n|\Xi})\mu^t_{n|\Xi}(x) - \frac{1}{t}r^{\pi_{n,\Xi.\xi}}(x,\mu^t_{n|\Xi})\mu^{\text{BR},t}_n(x)$$

Using the definition of exploitability together with the above remarks, we deduce:

$$\frac{d}{dt}\phi(\boldsymbol{\pi}^t) = \frac{d}{dt}\left[\max_{\boldsymbol{\pi}'} J(m_0,\boldsymbol{\pi}',\boldsymbol{\mu}^{\boldsymbol{\pi}}) - J(m_0,\boldsymbol{\pi},\boldsymbol{\mu}^{\boldsymbol{\pi}})\right] \tag{B.9}$$

$$= \sum_{n=0}^{N_T}\sum_{\Xi,|\Xi|=n}\sum_{\xi}P(\Xi.\xi)\sum_{x\in\mathcal{X}}\Big[< \nabla_\mu r^{\pi^{\text{BR}}_{n,\Xi.\xi},\xi}(x,\mu^t_{n|\Xi}), \frac{d}{dt}\mu^t_{n|\Xi} > \mu^{\text{BR},t}_{n,\Xi.\xi}(x) \tag{B.10}$$

$$- < \nabla_\mu r^{\pi_{n,\Xi.\xi}}(x,\mu^t_{n|\Xi},\xi), \frac{d}{dt}\mu^t_{n|\Xi} > \mu^t_{n|\Xi}(x) \tag{B.11}$$

$$- <\frac{d}{dt}\pi^t_{n,\Xi.\xi}(.|x), r(x,.,\mu^t_{n|\Xi})>\mu^t_{n|\Xi}(x) - r^{\pi_{n,\Xi.\xi}}(x,\mu^t_{n|\Xi})\frac{d}{dt}\mu^t_{n|\Xi}(x)\Big] \tag{B.12}$$

$$= \sum_{n=0}^{N_T} \sum_{\Xi,|\Xi|=n} \sum_{\xi} P(\Xi.\xi) \sum_{x\in\mathcal{X}} \Big[t < \nabla_\mu \bar{r}(x, \mu^t_{n|\Xi})), \frac{d}{dt}\mu^t_{n|\Xi} > \frac{1}{t}\left(\mu^{\text{BR},t}_{n|\Xi}(x) - \mu^t_{n|\Xi}(x)\right) \Big] \tag{B.13}$$

$$+ \sum_{n=0}^{N_T} \sum_{\Xi,|\Xi|=n} \sum_{\xi} P(\Xi.\xi) \sum_{x\in\mathcal{X}} \big[\frac{1}{t} r^{\pi_{n,\Xi.\xi}}(x, \mu^t_{n|\Xi})\mu^t_{n|\Xi}(x) - \frac{1}{t} r^{\pi^{\text{BR},t}_{n,\Xi.\xi}}(x, \mu^t_{n|\Xi})\mu^{\text{BR},t}_{n|\Xi}(x) \big] \tag{B.14}$$

$$= -\frac{1}{t}\phi(\boldsymbol{\pi}^t) + \sum_{n=0}^{N_T} \sum_{\Xi,|\Xi|=n} \sum_{\xi} P(\Xi.\xi) \sum_{x\in\mathcal{X}} \big[t < \nabla_\mu \bar{r}(x, \mu^t_{n|\Xi}), \frac{d}{dt}\mu^t_{n|\Xi} > \frac{d}{dt}\mu^t_{n|\Xi}(x) \big], \tag{B.15}$$

where the last term is non-positive by Property 1 (*i.e.*, thanks to the monotonicity assumption). □

Experiments: A More Complex Setting for the Beach Bar Process with common noise

Environment. Following the first setting of the paper where the bar could only close at one given time step, we now introduce a second more complex setting, bringing also of common noise in the beach bar process. Namely, the bar has a probability p to close at every time step up to a point (in practice, this point is half of the horizon: $\frac{N_T}{2}$). Once the bar is closed, it does not open again. This setting gives $\frac{N_T}{2} + 1$ possible realizations of the common noise: (1) the case where the bar never closes and (2) the $\frac{N_T}{2}$ cases where it closes at any of the first $\frac{N_T}{2}$ time steps. For the sake of clarity, we only present the evolution of the distributions when the bar finally remains open after $\frac{N_T}{2}$ time steps, and when it closes at the $\frac{N_T}{2}^{\text{th}}$ time step.

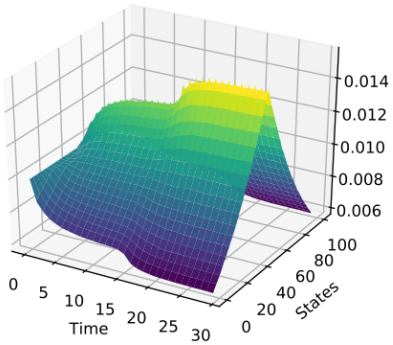
Numerical results. Similarly to the first setting, we take $|\mathcal{X}| = 100$ states and $N_T = 30$ time steps. As the bar has a probability $p = 0.5$ to close at every time step until $\frac{N_T}{2}$, the distribution is flatter to anticipate the fact that people might need to spread. We can see that both model-based and model-free approaches converge to a Nash equilibrium and that model-based converges faster than model-free.

B.3 Continuous Time Fictitious Play: the γ-discounted case

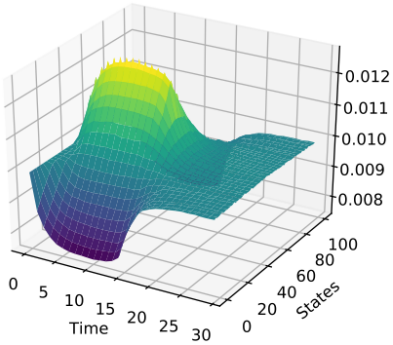
Surprisingly, the analysis also holds in the γ-discounted case with again the same style of reasoning. However, the distribution considered will be the γ-weighted occupancy measure instead of the distribution over states. In this section, we reintroduce the notations and we prove similar continuous time FP convergence results.

Consider, given the following:

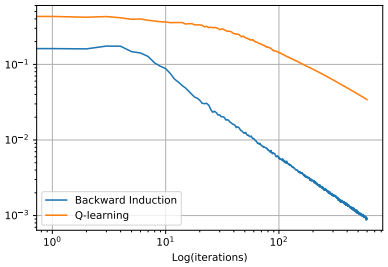
- a finite state space $\mathcal{X}$ ($x \in \mathcal{X}$),



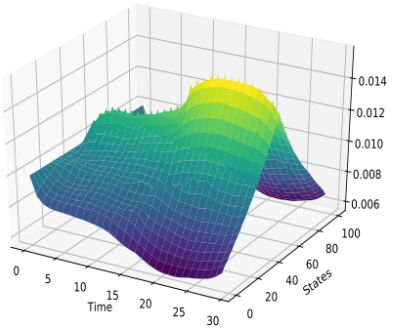
(**a**) Model-based, the bar stays open



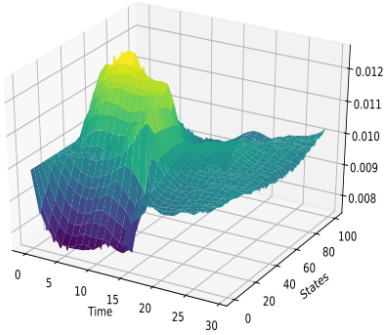
(**b**) Model-based, the bar closes



(**c**) Exploitability



(**d**) Model-free, the bar stays open



(**e**) Model-free, the bar closes

Figure B.1 – 2nd common noise setting, the bar has a probability $p = 0.5$ to close at every time step before $\frac{N}{2}$.

- a finite action space $\mathcal{A}$ ($a \in \mathcal{A}$),
- the set of distributions over state is $\Delta_{\mathcal{X}}$ ($\mu \in \Delta_{\mathcal{X}}$),
- a reward function $r(x, a, \mu)$,
- the transition function $p(x'|x, a)$,
- a policy: $\pi(a|x)$.

We will write:

- $p^\pi(x'|x) = \mathbb{E}_{a\sim\pi(.|x)}[p(x'|x, a)]$,
- $r^\pi(x, \mu) = \mathbb{E}_{a\sim\pi(.|x)}[r(x, a, \mu)]$,

The cumulative γ-discounted reward is defined as:

$$J_\gamma(x_0, \pi, \mu) = \mathbb{E}\left[\sum_{n=0}^{+\infty} \gamma^n r(x_n, a_n, \mu) \mid x_{n+1} \sim p(.|x_n, a_n),\ a_n \sim \pi(.|x_n)\right]$$

Useful properties. We have $\mu_\gamma^\pi(x') = m_0(x') + \gamma \sum_{x\in\mathcal{X}} p^\pi(x'|x)\mu_\gamma^\pi(x)$ (in vectorial notations $\mu_\gamma^{\pi\top} = m_0^\top(I - \gamma P^\pi)^{-1}$).
The γ-discounted reward can be written as: $J_\gamma(x_0, \pi, \mu) = \sum_{x\in\mathcal{X}} \mu_\gamma^\pi(x) r^\pi(x, \mu)$.
We then have a similar formula for the policy generating the average distribution $\bar{\mu}_\gamma^\pi(x, t) = \frac{1}{t}\int_{s=0}^{t} \mu_\gamma^\pi(x, s)ds$ can be written $\bar{\pi}_\gamma(a|x, t) = \frac{\int_{s=0}^{t} \mu_\gamma^\pi(x,s)\pi(a|x,s)ds}{\int_{s=0}^{t} \mu_\gamma^\pi(x,s)ds}$.

Finally, we can write:

$$\bar{\mu}_\gamma^\pi(x, t)\bar{\pi}_\gamma(a|x, t) = \frac{1}{t}\int_{s=0}^{t} \mu_\gamma^\pi(x, s)\pi(a|x, s)ds \tag{B.16}$$

And:

$$\dot{\bar{\mu}}_\gamma^\pi(x, t)\bar{\pi}_\gamma(a|x, t) + \bar{\mu}_\gamma^\pi(x, t)\dot{\bar{\pi}}_\gamma(a|x, t) = \frac{1}{t}\left[\mu_\gamma^\pi(x, t)\pi(a|x, t) - \bar{\mu}_\gamma^\pi(x, t)\bar{\pi}_\gamma(a|x, t)\right]. \tag{B.17}$$

Fictitious Play in MFGs. In the γ-discounted case, Fictitious Play can be written as (for $t \geq 1$):

$$\dot{\mu}(x, t) = \frac{1}{t}(\mu_\gamma^{\mathtt{BR}}(x, t) - \mu(x, t))$$

where $\mu_\gamma^{\mathtt{BR}}(x, t)$ is the distribution of a best response against $\mu(x, t)$ of policy $\pi^{\mathtt{BR}}(a|x, t)$. In this section, we will write $\pi(a|x, t)$ the policy of the distribution $\mu(x, t)$. From Eq.(B.17), we can deduce the following property:

Property 3.

$$\forall n,\ \dot{\pi}(a|x,t)\mu(x,t) = \frac{1}{t}\mu_\gamma^{\text{BR}}(x,t)[\pi^{\text{BR}}(a|x,t) - \pi(a|x,t)]$$

Proof. Such representation directly follows from Eq.(B.17). □

We are now in position to turn to the Lyapounov congerging property of the Fictitious process.

Property 4. *Under the monotony assumption, we can show that the exploitability* ($\phi(t) = \max_{\pi'} J_\gamma(x_0, \pi', \mu^\pi) - J_\gamma(x_0, \pi, \mu^\pi)$) *is a strong Lyapunov function of the system:*

$$\dot{\phi}(t) \leq -\frac{1}{t}\phi(t)$$

Proof.

$$\begin{aligned}
&\dot{\phi}(t) \\
&= \sum_{x\in\mathcal{X}} [\overbrace{< \nabla_\mu r^{\pi^{\text{BR}}}(x,\mu(t)), \dot{\mu}(t) > \mu_\gamma^{\text{BR}}(x,t) - < \nabla_\mu r^{\pi}(x,\mu(t)), \dot{\mu}(t) > \mu(x,t)}^{\text{With } \nabla_\mu r^{\pi^{\text{BR}}}(x,\mu(t))=\nabla_\mu \bar{r}(x,\mu(t)) \text{ and } \nabla_\mu r^{\pi}(x,\mu(t))=\nabla_\mu \bar{r}(x,\mu(t))} \\
&\underbrace{- < \dot{\pi}(.|x,t), r(x,.,\mu(t)) > \mu(x,t)}_{=-\frac{1}{t} r^{\pi^{\text{BR}}}(x,\mu(t))\mu_\gamma^{\text{BR}}(x,t) + \frac{1}{t} r^{\pi}(x,\mu(t))\mu_\gamma^{\text{BR}}(x,t)} \quad \underbrace{-r^\pi(x,\mu(t))\dot{\mu}(x,t)}_{= \frac{1}{t} r^{\pi}(x,\mu(t))\mu(x,t) - \frac{1}{t} r^{\pi}(x,\mu(t))\mu_\gamma^{\text{BR}}(x,t)} \quad] \\
&= \sum_{x\in\mathcal{X}} [t < \nabla_\mu \bar{r}(x,\mu(t))), \dot{\mu}(t) > [\frac{1}{t}(\mu_\gamma^{\text{BR}}(x,t) - \mu(x,t))]] \\
&\quad + \sum_{x\in\mathcal{X}} [\frac{1}{t} r^{\pi}(x,\mu(t))\mu(x,t) - \frac{1}{t} r^{\pi^{\text{BR}}}(x,\mu(t))\mu_\gamma^{\text{BR}}(x,t)] \\
&= -\frac{1}{t}\phi(t) + \underbrace{\sum_{x\in\mathcal{X}} [t < \nabla_\mu \bar{r}(x,\mu(t))), \dot{\mu}(t) > \dot{\mu}(x,t))]}_{\leq 0 \text{ by monotony}} \qquad \text{(B.18)} \\
&\leq -\frac{1}{t}\phi(t) \qquad \text{(B.19)}
\end{aligned}$$

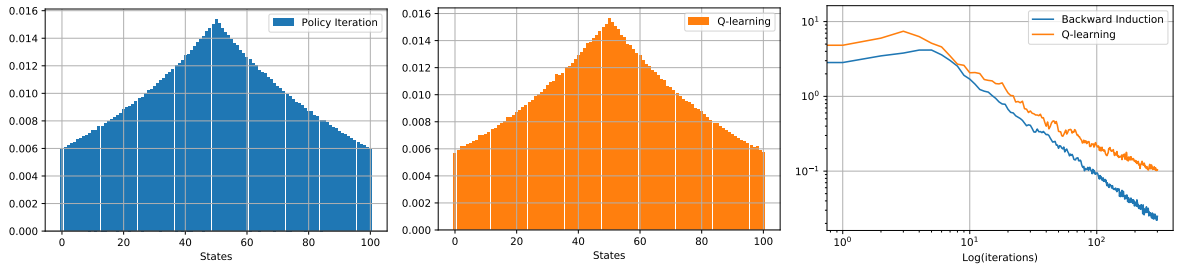
□

Experiment: the Beach Bar Process with γ-discounted reward.

Environment. We implement the beach bar process in the γ-discounted setting.

Numerical results. We set $\gamma = 0.9$. The algorithm estimating the best response to a fixed distribution μ is Policy Iteration in the case of the model based approach and Q-learning in the model-free. As the flow of distributions converges towards the stationary distribution which

is not time-dependant, we only plot the final distribution obtained after 300 time steps (and not the evolution throughout time as before). In particular, we notice that model-based and model-free approaches converge towards the same distribution. We can also observe that the convergence rate of exploitability is $O(1/t)$ for the model-based and slower for the model-free approach.



(**a**) Model-based (**b**) Model-free (**c**) Exploitability

Figure B.2 – Final distributions and exploitability in the γ-discounted case

B.4 Algorithms

Algorithm B.1: Q-Learning in Mean Field Games

input : Start with a fixed distribution $\mu = (\mu_k)_k$ and $Q^k = 0$ and ε and the learning rate α.
for $k = 0, \dots, K$: **do**
  sample $x_0^k \sim m_0$;
  for $n = 0, \dots, N_T$: **do**
    a_n^k is ε-greedy with respect to $Q^k(x_n^k, .)$.;
    if not terminal sample x_{n+1}^k according to $p(.|x_n^k a_n^k)$.;
    $Q_n^{k+1}(x_n^k, a_n^k) = (1-\alpha)Q_n^{k+1}(x_n^k, a_n^k) + \alpha[r(x_n^k, a_n^k, \mu_{k-1}) + \max_b Q_{n+1}^k(x_{n+1}^k, b)]$;
return π^* *a greedy policy with respect to* Q^K

Algorithm B.2: Empirical Density Estimation

input : Start with a fixed policy π and an initial distribution $m_0 = \mu_0^\pi$
for $k = 0, \dots, K$: **do**
  sample $x_0^k \sim m_0$;
  for $n = 0, \dots, N_T$: **do**
    a_n^k with respect to $Q^k(x_n^k, .)$.;
    if not terminal sample x_{n+1}^k according to $p(.|x_n^k a_n^k)$.
Finally $\forall x, n \in \mathcal{X} \times \{0, \dots, N\}$ $\hat{\mu}_n^\pi(x) = \frac{1}{K+1} \sum_{k=0}^K 1_{x_n^k = x}$;
return $\hat{\mu}_n^\pi$

Algorithm B.3: Backward Induction in Mean Field Games

input : Start with a fixed distribution $\mu = (\mu_k)_k$ and a terminal Q-function $Q_{N_T}^\mu(x, a) = r(x, a, \mu_{N_T})$
for $n = N_T, \dots, 0$: **do**
  π_k^* is greedy with respect to $Q_k^\mu(x, a)$. ;
  $\forall a, x \in \mathcal{A} \times \mathcal{X}$ $Q_{n-1}^\mu(x, a) = r(x, a, \mu_{n-1}) + \sum_{x' \in \mathcal{X}} p(x'|x, a) \max_b Q_n^\mu(x', b)$;
return π^*

Algorithm B.4: Density Estimation

input : Start with a fixed policy π and an initial distribution $m_0 = \mu_0^\pi$
for $n = 1, \dots, N_T$: **do**
  $\forall x \in \mathcal{X}$ $\mu_n^\pi(x') = \sum_{x, a \in \mathcal{X} \times \mathcal{A}} \pi_{n-1}(a|x) p(x'|x, a) \mu_{n-1}^\pi(x)$;
return μ^π

B.5 Linear Quadratic Model

B.5.1 Description

For the sake of completeness, we explain here how we obtained the benchmark solution for the LQ problem. The original model has been introduced by Carmona, Fouque, and L.-H. Sun (2015b) and corresponds to the continuous time and continuous spaces version of the LQ problem implemented in Section 4.3. Each player can influence their speed with a control denoted by α_t. The dynamics of the players is linear in their state, their control and the mean position, denoted by $\bar{m}_t$. It is affected by an idiosyncratic source of randomness $\mathbf{W} = (W_t)_{t\geq 0}$ as well as a common noise in the form of a Brownian motion $\mathbf{W}^0 = (W_t^0)_{t\geq 0}$. Given a flow of *conditional* mean positions $\bar{\boldsymbol{\mu}} = (\bar{\mu}_t)_{t\in[0,T]}$ adapted to the filtration generated by $\mathbf{W}^0$, the cost function of a representative player is defined as:

$$J(a;\bar{\boldsymbol{\mu}}) = \mathbb{E}\left[\int_{t=0}^{T}\left(\frac{1}{2}a_t^2 - qa_t(\bar{\mu}_t - X_t) + \frac{\kappa}{2}(\bar{\mu}_t - X_t)^2\right)dt + \frac{c_{\text{term}}}{2}(\bar{\mu}_T - X_T)^2\right] \tag{B.20}$$

Subject to the dynamics:

$$dX_t = [K(\bar{\mu}_t - X_t) + a_t]dt + \sigma\left(\rho\, dW_t^0 + \sqrt{1-\rho^2}dW_t\right).$$

At equilibrium, we must have $\bar{\mu}_t = \mathbb{E}[X_t|(W_s^0)_{s\leq t}]$ for every $t \in [0,T]$.

Here, $\rho \in [0,1]$ is a constant parameterizing the correlation between the noises, and q, κ, c, a, σ are positive constants. We assume that $q \leq \kappa^2$ so that the running cost is jointly convex in the state and the control variables.

The terms $(\bar{\mu}_t - X_t)$ in the dynamics and the cost function attract the process towards the mean $\bar{\mu}_t$. For the interpretation of this model in terms of systemic risk, the reader is referred to Carmona, Fouque, and L.-H. Sun (2015b). The model is of linear-quadratic type and has an explicit solution through a Riccati equation, which we use as a benchmark. The optimal control at time t is a linear combination of X_t and $\bar{\mu}_t$, whose coefficients depend on time. More precisely, it is given by:

$$a_t = (q + \eta_t)(\bar{\mu}_t - X_t),$$

where η solves the following Riccati ODE:

$$\dot{\eta}_t = 2(a+q)\eta_t + \eta_t^2 - (\kappa - q^2), \qquad \eta_T = c_{\text{term}},$$

whose solution is explicitly given by:

$$\eta_t = \frac{-(\kappa - q^2)\left(e^{(\delta^+ - \delta^-)(T-t)} - 1\right) - c\left(\delta^+ e^{(\delta^+ - \delta^-)(T-t)} - \delta^-\right)}{\left(\delta^- e^{(\delta^+ - \delta^-)(T-t)} - \delta^+\right) - c\left(e^{(\delta^+ - \delta^-)(T-t)} - 1\right)}$$

where $\delta^{\pm} = -(a+q) \pm \sqrt{R}$ with $R = (a+q)^2 + (\kappa - q^2) > 0$.

B.6 Common Success Metrics in Mean Field Games

The optimal value function satisfies the recursive equation:

$$V_{N_T}^{*,\mu}(x) = r(x, \mu_{N_T}), \qquad V_{n-1}^{*,\mu}(x) = \max_a \left\{ r(x, a, \mu_{n-1}) + \sum_{x' \in \mathcal{X}} p(x'|x, a) V_n^{*,\mu}(x') \right\}.$$

In particular, by definition:

$$\max_{\pi'} J(m_0, \pi', \mu^\pi) = \mathbb{E}_{x \sim m_0}[V_0^{*,\mu^\pi}(x)]$$

And:

$$J(m_0, \pi, \mu^\pi) = \mathbb{E}_{x \sim m_0}[V_0^{\pi,\mu^\pi}(x)].$$

Let $(x, \mu) \mapsto a^*(x, \mu)$ be such that for every n and (reasonable?) μ:

$$V_{n-1}^{*,\mu}(x) = r(x, a^*(x, \mu_{n-1}), \mu_{n-1}) + \sum_{x' \in \mathcal{X}} p(x'|x, a^*(x, \mu_{n-1})) V_n^{*,\mu}(x'), \tag{B.21}$$

i.e. a^* is an optimal control. Then, one way to check whether the value function we learned (*e.g.*, deduced from the Q-table) is a good approximate solution, is to compute the residual in the fixed point equation (B.21). In other words, if the learned value function is $\tilde{V}$ and the policy is π with associated distribution μ^π, then, we compute:

$$\tilde{V}_{n-1}(x) - \left[r(x, a^*(x, \mu_{n-1}^\pi), \mu_{n-1}^\pi) + \sum_{x' \in \mathcal{X}} p(x'|x, a^*(x, \mu_{n-1}^\pi)) \tilde{V}_n(x') \right]$$

for every n, x. Taking the norm over $(n, x) \in \{1, \dots, N\} \times \mathcal{X}$ provides a metric to assess the convergence of the value function.

Link with fixed-point iterations. One of the most basic methods to compute a MFG equilibrium is to iteratively solve the forward equation for the distribution and the backward equation for the value function. A typical stopping criterion is that the distribution and the value function do not change too much between two successive iterations. We argue that this property implies an upper bound on the exploitability. To be specific, say that at iteration k, given a value function V^k and its associated optimal control π^k, we compute the induced flow of distributions $\mu^k = \mu^{\pi_k}$, and then we compute the value function V^{k+1} and the best response π^{k+1} of an infinitesimal player against this flow of distributions. Note that:

$$\max_{\pi'} J(m_0, \pi', \mu^k) = \max_{\pi'} J(m_0, \pi', \mu^{\pi^k}) = J(m_0, \pi^{k+1}, \mu^{\pi^k}) = \sum_x V_0^{k+1}(x) m_0(x)$$

And:

$$J(m_0, \pi^k, \mu^k) = \sum_x V_0^k(x) m_0(x).$$

Hence, if we know that $\|V^{k+1} - V^k\|_\infty := \sup_{x,n} |V_n^{k+1}(x) V_n^k(x)| < \varepsilon$, then, in particular, $|V_0^{k+1}(x) - V_0^k(x)| < \varepsilon$ for all x and hence the exploitability is at most ε too. Conversely, under suitable regularity assumptions, we can expect that a small exploitability implies $V^{k+1} \approx V^k$ not only at time 0 but at every time.

Appendix C

Complements on Chapter 5

C.1 Separability and Monotonicity Imply Weak Monotonicity

Proof of Lemma 2. Let us assume that the reward is separable $r^i(x^i, a^i, \mu) = \bar{r}^i(x^i, a^i) + \tilde{r}^i(x^i, \mu)$ and that it follows the monotonicity condition:

$$\forall \mu \neq \mu', \ \sum_i \sum_{x \in \mathcal{X}} (\mu^i(x^i) - \mu'^i(x^i))(\tilde{r}^i(x^i, \mu) - \tilde{r}^i(x^i, \mu')) \leq 0$$

. Then, we have:

$$\begin{aligned}
&\sum_{i=1}^{N_p} \left[J^i(\pi, \mu^\pi) - J^i(\pi', \mu^\pi) - J^i(\pi, \mu^{\pi'}) + J^i(\pi', \mu^{\pi'}) \right] \\
&= \sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{(x^i, a^i) \in \mathcal{X} \times \mathcal{A}} \left[\mu_n^{i,\pi^i}(x^i) \pi_n^i(a^i|x^i) r^i(x^i, a^i, \mu_n^\pi) - \mu_n^{i,\pi'^i}(x^i) \pi'^i{}_n(a^i|x^i) r^i(x^i, a^i, \mu_n^\pi) \right. \\
&\qquad \left. - \mu_n^{i,\pi^i}(x^i) \pi_n^i(a^i|x^i) r^i(x^i, a^i, \mu_n^{\pi'}) + \mu_n^{i,\pi'^i}(x^i) \pi'^i{}_n(a^i|x^i) r^i(x^i, a^i, \mu_n^{\pi'}) \right] \\
&= \sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{(x^i, a^i) \in \mathcal{X} \times \mathcal{A}} \left(\mu_n^{i,\pi^i}(x^i) \pi_n^i(a^i|x^i) - \mu_n^{i,\pi'^i}(x^i) \pi'^i{}_n(a^i|x^i) \right) \left(r^i(x^i, a^i, \mu_n^\pi) - r^i(x^i, a^i, \mu_n^{\pi'}) \right) \\
&= \sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{(x^i, a^i) \in \mathcal{X} \times \mathcal{A}} \left(\mu_n^{i,\pi^i}(x^i) \pi_n^i(a^i|x^i) - \mu_n^{i,\pi'^i}(x^i) \pi'^i{}_n(a^i|x^i) \right) \left(\tilde{r}^i(x^i, \mu_n^\pi) - \tilde{r}^i(x^i, \mu_n^{\pi'}) \right) \\
&= \sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{x^i \in \mathcal{X}} \left(\mu_n^{i,\pi^i}(x^i) - \mu_n^{i,\pi'^i}(x^i) \right) \left(\tilde{r}^i(x^i, \mu_n^\pi) - \tilde{r}^i(x^i, \mu_n^{\pi'}) \right) \leq 0.
\end{aligned}$$

With a similar proof, we obtain the corresponding property with strict inequality. □

C.2 Multi-Population Reward

Let us suppose:

$$r^i(x^i, a^i, \mu) = \bar{r}^i(x^i, a^i) + \underbrace{\hat{r}^i(x^i, \mu^i) + \sum_{j \neq i} \mu^j(x^i)\hat{r}^{i,j}(x^i)}_{=\tilde{r}^i(x^i, \mu)}$$

With $\forall x \in \mathcal{X}, \hat{r}^{i,j}(x) = -\hat{r}^{j,i}(x)$ and if $\forall \mu \neq \mu', \forall i, \sum_{x \in \mathcal{X}} \Big(\mu^i(x^i) - \mu'^i(x^i)\Big)\Big(\hat{r}^i(x^i, \mu^i) - \hat{r}^i(x^i, \mu'^i)\Big) \leq 0$.

$$\begin{aligned}
&\sum_i \sum_{x \in \mathcal{X}} (\mu^i(x^i) - \mu'^i(x^i))(\tilde{r}^i(x^i, \mu) - \tilde{r}^i(x^i, \mu')) \\
&= \sum_i \sum_{x \in \mathcal{X}} (\mu^i(x^i) - \mu'^i(x^i))(\hat{r}^i(x^i, \mu^i) + \sum_{j \neq i} \mu^j(x^i)\hat{r}^{i,j}(x^i) - \hat{r}^i(x^i, \mu'^i) - \sum_{j \neq i} \mu'^j(x^i)\hat{r}^{i,j}(x^i)) \\
&= \sum_i \underbrace{\sum_{x \in \mathcal{X}} (\mu^i(x^i) - \mu'^i(x^i))(\hat{r}^i(x^i, \mu^i) - \hat{r}^i(x^i, \mu'^i))}_{\leq 0} + \underbrace{\sum_i \sum_{j \neq i} \sum_{x \in \mathcal{X}} (\mu^i(x^i) - \mu'^i(x^i))(\mu^j(x^i) - \mu'^j(x^i))\hat{r}^{i,j}(x^i)}_{=0 \text{ since } \hat{r}^{i,j}(x) = -\hat{r}^{j,i}(x)} \\
&\leq 0
\end{aligned}$$

C.3 Fictitious Play

We can extend the fictitious play algorithm of Perrin, Perolat, et al. (2020) to a multi-population setting. In the multi-population case, the fictitious play process is defined as follows. Let first picking 1 as an arbitrary but classical reference time. For $t < 1$, we consider a fixed uniform policy for all representative player i at all time-step n denoted $\pi^{i,br}_{n,t<1}$ and inducing a distribution $\mu^{i,br}_{n,t}$. We define $\forall t \geq 1$ the distribution $\mu^i_{n,t}$ as:

$$\forall i, n,\ \mu^i_{n,t}(x^i) = \frac{1}{t}\int_{s=0}^{t} \mu^{i,br}_{n,s}(x^i)ds\,,$$

where, for all $t \geq 1$, $\mu^{i,br}_{n,t}$ is the distribution of a best response policy $\pi^{i,br}_{n,t}$ to $\mu^i_{n,t}(x^i)$. The policy $\pi^i_{n,t}$ of the distribution $\mu^i_{n,t}$ verifies the following equation (see Perrin, Perolat, et al. (2020)): for all i, n, x^i, a^i,

$$\pi^i_{n,t}(a^i|x^i)\int_{s=0}^{t} \mu^{i,br}_{n,s}(x^i)ds = \int_{s=0}^{t} \pi^{i,br}_{n,s}(a^i|x^i)\mu^{i,br}_{n,s}(x^i)ds$$

Theorem C.1. *If a MP-MFG satisfies the weak monotony assumption, the exploitability is a strong Lyapunov function of the Fictitious Play dynamical system, $\forall t \geq 1$: $\frac{d}{dt}\phi(\pi^t) \leq -\frac{1}{t}\phi(\pi^t)$. Hence $\phi(\pi^t) = O(\frac{1}{t})$.*

In the following, we prove that under the weak monotonicity condition, the Fictitious Play process converges to a NE.

First, we prove the following property, which stems from the weak monotonicity.

Property 5. *Let f be a smooth enough function and let assume that the ODE $\dot{\rho} = f(\rho)$ (with $\dot{\rho} = \frac{d}{dt}\rho$) has a solution $(\rho^t)_{t\geq 0} = (\rho^t_n(x))_{t\geq 0, x\in\mathcal{X}}$. If the game is weakly monotone, then:*

$$\sum_{i=1}^{N_p}\sum_{x^i,a^i\in\mathcal{X}\times\mathcal{A}} \langle \nabla_\rho r^i(x^i, a^i, \rho), \dot{\rho}\rangle \dot{\rho}^i(x^i, a^i) \leq 0.$$

Proof. The monotonicity condition implies that, for all $\tau \geq 0$, we have:

$$\sum_{i=1}^{N_p}\sum_{x^i,a^i\in\mathcal{X}\times\mathcal{A}} (\rho^i_t(x^i, a^i) - \rho^i_{t+\tau}(x^i, a^i))(r^i(x^i, a^i, \rho_t) - r^i(x^i, a^i, \rho_{t+\tau})) \leq 0.$$

Thus:

$$\sum_{i=1}^{N_p}\sum_{x^i,a^i\in\mathcal{X}\times\mathcal{A}} \frac{\rho^i_t(x^i, a^i) - \rho^i_{t+\tau}(x^i, a^i)}{\tau}\frac{r^i(x^i, a^i, \rho_t) - r^i(x^i, a^i, \rho_{t+\tau})}{\tau} \leq 0.$$

The result follows when $\tau \to 0$. □

In the space of distributions over state actions, the fictitious play process can be expressed as follows. First, we start with a distribution $\rho^i_{n,t}$ following the balance equation on the state action distributions:

$$\sum_{a'^i \in \mathcal{A}} \rho^i_{n-1,t}(x'^i, a'^i) = \sum_{x^i, a^i \in \mathcal{X} \times \mathcal{A}} p(x'^i | x^i, a^i) \rho^i_{n,t}(x^i, a^i).$$

And for $t < 1$, the policy $\pi^i_{n,t}(a^i|x^i) = \frac{\rho^i_{n,t}(x^i,a^i)}{\sum_{a^i \in \mathcal{A}} \rho^i_{n,t}(x^i,a^i)}$ is the uniform policy whenever $\sum_{a^i \in \mathcal{A}} \rho^i_{n,t}(x^i, a^i) > 0$.

A best response state action distribution to ρ is written $\rho^{i,br}_{n,t}(x^i, a^i)$ (which will be assumed to be equal to ρ_t for $t < 1$) and finally the Fictitious Play process on the state action distribution is written as for all $t \geq 1$:

$$\rho^i_{n,t}(x^i, a^i) = \frac{1}{t} \int_0^t \rho^{i,br}_{n,s}(x^i, a^i) ds.$$

The exploitability can then be written as:

$$\phi(t) = \max_\rho \Big[\sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{x^i, a^i \in \mathcal{X} \times \mathcal{A}} \rho^i_n(x^i, a^i) r^i(x^i, a^i, \rho_{n,t}) \Big] - \Big[\sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{x^i, a^i \in \mathcal{X} \times \mathcal{A}} \rho^i_{n,t}(x^i, a^i) r^i(x^i, a^i, \rho_{n,t}) \Big]$$

Property 6. *We have that* $\frac{d}{dt}\rho^i_{n,t}(x^i, a^i) = \frac{1}{t}\Big[\rho^{i,br}_{n,s}(x^i, a^i) - \rho^i_{n,t}(x^i, a^i)\Big]$ *by taking the derivative of* $\rho^i_{n,t}(x^i, a^i) = \frac{1}{t} \int_0^t \rho^{i,br}_{n,s}(x^i, a^i) ds$ *on both sides.*

Finally, we take the derivative of the exploitability and get:

$$\begin{aligned}
\frac{d}{dt}\phi(t) &= \frac{d}{dt} \max_\rho \Big[\sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{x^i, a^i \in \mathcal{X} \times \mathcal{A}} \rho^i_n(x^i, a^i) r^i(x^i, a^i, \rho_{n,t}) \Big] - \frac{d}{dt} \Big[\sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{x^i, a^i \in \mathcal{X} \times \mathcal{A}} \rho^i_{n,t}(x^i, a^i) r^i(x^i, a^i, \rho_{n,t}) \Big] \\
&= \Big[\sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{x^i, a^i \in \mathcal{X} \times \mathcal{A}} \rho^{i,br}_{n,t}(x^i, a^i) \frac{d}{dt} \Big(r^i(x^i, a^i, \rho_{n,t}) \Big) \Big] \\
&\quad - \Big[\sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{x^i, a^i \in \mathcal{X} \times \mathcal{A}} \Big(\rho^i_{n,t}(x^i, a^i) \frac{d}{dt} \Big(r^i(x^i, a^i, \rho_{n,t}) \Big) + r^i(x^i, a^i, \rho_{n,t}) \frac{d}{dt} \Big(\rho^i_{n,t}(x^i, a^i) \Big) \Big) \Big]
\end{aligned}$$

$$= \Big[\sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{x^i, a^i \in \mathcal{X} \times \mathcal{A}} \underbrace{[\rho_{n,t}^{i,br}(x^i, a^i) - \rho_{n,t}^i(x^i, a^i)]}_{= t \frac{d}{dt}\left(\rho_{n,t}^i(x^i, a^i)\right)} \underbrace{\frac{d}{dt}\Big(r^i(x^i, a^i, \rho_{n,t})\Big)}_{= \langle \nabla_\rho r^i(x^i, a^i, \rho_{n,t}), \dot{\rho}_{n,t} \rangle} \Big]$$

$$- \Big[\sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{x^i, a^i \in \mathcal{X} \times \mathcal{A}} r^i(x^i, a^i, \rho_{n,t}) \quad \underbrace{\frac{d}{dt}\Big(\rho_{n,t}^i(x^i, a^i)\Big)}_{= \frac{1}{t}\left[\rho_{n,t}^{i,br}(x^i, a^i) - \rho_{n,t}^i(x^i, a^i)\right]} \Big]$$

$$= t \sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{x^i, a^i \in \mathcal{X} \times \mathcal{A}} \underbrace{\Big[\frac{d}{dt}\Big(\rho_{n,t}^i(x^i, a^i)\Big) \langle \nabla_\rho r^i(x^i, a^i, \rho_{n,t}), \dot{\rho}_{n,t} \rangle\Big]}_{\leq 0}$$

$$- \frac{1}{t} \underbrace{\Big[\sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{x^i, a^i \in \mathcal{X} \times \mathcal{A}} \Big[\rho_{n,t}^{i,br}(x^i, a^i) - \rho_{n,t}^i(x^i, a^i)\Big] r^i(x^i, a^i, \rho_{n,t}) \Big]}_{= \phi(t)}$$

$$\leq -\frac{1}{t}\phi(t).$$

C.4 Online Mirror Descent Dynamics

Proof of Lemma 3. The Continuous Time Online Mirror Descent (CTOMD) algorithm is defined as: for all $t > 0, i \in \{1, \dots, N_p\}, n \in \{0, \dots, N_T\}$,

$$
\begin{aligned}
y^i_{n,t}(x^i, a^i) &= \int_{s=0}^{t} Q_n^{i,\pi^i_s,\mu^{\pi_s}}(x^i, a^i) ds, \\
\pi^i_{n,t}(.|x^i) &= \Gamma(y^i_{n,t}(x^i, .)).
\end{aligned}
$$

$$
\begin{aligned}
\frac{d}{dt}H(y_t) &= \frac{d}{dt}\sum_{i=1}^{N_p}\sum_{n=0}^{N_T}\sum_{x^i\in\mathcal{X}} \mu_n^{i,\pi^*}(x^i)\Big[h^*(y^i_{n,t}(x^i,.)) - h^*(y^{i,*}(x^i,.)) - \langle \pi^{i,*}_{n,t}, y^i_{n,t}(x^i,.) - y^{i,*}_{n,t}(x^i,.)\rangle\Big] \\
&= \sum_{i=1}^{N_p}\sum_{n=0}^{N_T}\sum_{x^i\in\mathcal{X}} \mu_n^{i,\pi^*}(x^i)\frac{d}{dt}\Big[h^*(y^i_{n,t}(x^i,.)) - h^*(y^{i,*}(x^i,.)) - \langle \pi^{i,*}_{n,t}, y^i_{n,t}(x^i,.) - y^{i,*}_{n,t}(x^i,.)\rangle\Big] \\
&= \sum_{i=1}^{N_p}\sum_{n=0}^{N_T}\sum_{x^i\in\mathcal{X}} \mu_n^{i,\pi^*}(x^i)\Big[\frac{d}{dt}h^*(y^i_{n,t}(x^i,.)) - \langle \pi^{i,*}_{n,t}, \frac{d}{dt}y^i_{n,t}(x^i,.)\rangle\Big] \\
&= \sum_{i=1}^{N_p}\sum_{n=0}^{N_T}\sum_{x^i\in\mathcal{X}} \mu_n^{i,\pi^*}(x^i)\Big[\langle \pi^i_{n,t}(.|x^i) - \pi^{i,*}_{n,t}(.|x^i), Q_n^{i,\pi^i_t,\mu^{\pi_t}}(x^i,.)\rangle\Big] \\
&= \sum_{i=1}^{N_p}\sum_{n=0}^{N_T}\sum_{x^i\in\mathcal{X}} \mu_n^{i,\pi^*}(x^i)\Big[V_n^{i,\pi^i_t,\mu^{\pi_t}}(x^i) - \langle \pi^{i,*}_{n,t}(.|x^i), Q_n^{i,\pi^i_t,\mu^{\pi_t}}(x^i,.)\rangle\Big] \\
&= \sum_{i=1}^{N_p}\sum_{n=0}^{N_T}\sum_{x^i\in\mathcal{X}} \mu_n^{i,\pi^*}(x^i)\Big[V_n^{i,\pi^i_t,\mu^{\pi_t}}(x^i) - \langle \pi^{i,*}_{n,t}(.|x^i), r^i(x^i,.,\mu^{\pi_t}) + \sum_{x'^i\in\mathcal{X}} p(x'^i|x^i,a^i)V_{n+1}^{i,\pi^i_t,\mu^{\pi_t}}(x'^i,.)\rangle\Big] \\
&= \sum_{i=1}^{N_p}\sum_{n=0}^{N_T}\Big[\sum_{x^i\in\mathcal{X}} \mu_n^{i,\pi^*}(x^i)V_n^{i,\pi^i_t,\mu^{\pi_t}}(x^i)\Big] - \Big[\sum_{x^i\in\mathcal{X}} \mu_n^{i,\pi^*}(x^i)\langle \pi^{i,*}_{n,t}(.|x^i), r^i(x^i,.,\mu^{\pi_t})\Big] \\
&\qquad - \Big[\sum_{x'^i\in\mathcal{X}} V_{n+1}^{i,\pi^i_t,\mu^{\pi_t}}(x'^i) \underbrace{\sum_{x^i,a^i\in\mathcal{X}\times\mathcal{A}} \mu_n^{i,\pi^*}(x^i)\pi^{i,*}_{n,t}(a^i|x^i)p(x'^i|x^i,a^i)}_{=\mu^{i,\pi^*}_{n+1}(x'^i)}\Big] \\
&= \underbrace{\sum_{i=1}^{N_p}\sum_{n=0}^{N_T}\Big[\sum_{x^i\in\mathcal{X}} \mu_n^{i,\pi^*}(x^i)V_n^{i,\pi^i_t,\mu^{\pi_t}}(x^i)\Big] - \sum_{n=0}^{N_T}\Big[\sum_{x'^i\in\mathcal{X}} V_{n+1}^{i,\pi^i_t,\mu^{\pi_t}}(x'^i)\mu^{i,\pi^*}_{n+1}(x'^i)\Big]}_{=J^i(\pi^i_t,\mu^{\pi_t})} \\
&\qquad - \underbrace{\sum_{i=1}^{N_p}\sum_{n=0}^{N_T}\Big[\sum_{x^i\in\mathcal{X}} \mu_n^{i,\pi^*}(x^i)\langle \pi^{i,*}_{n,t}(.|x^i), r^i(x^i,.,\mu^{\pi_t})\Big]}_{=J^i(\pi^{i,*},\mu^{\pi_t})}
\end{aligned}
$$

$$= \sum_{i=1}^{N_p} \left[J^i(\pi_t^i, \mu^{\pi_t}) - J^i(\pi^{i,*}, \mu^{\pi_t}) \right] = \Delta J(\pi_t, \pi^*) + \tilde{\mathcal{M}}(\pi_t, \pi^*).$$

□

C.5 Weak monotonicity

Proof of Lemma 1. Consider two policies π, π'. Denote by $\mu = \mu^{\pi}, \mu' = \mu^{\pi'}$ respectively the induced distribution sequences. Let ρ, ρ' be the associated joint distribution sequences:

$$\rho_n^i(x^i, a^i) = \mu_n^i(x^i)\pi_n^i(a^i|x^i)$$

and likewise for ρ'. By the weak monotonicity, we have:

$$0 \geq \sum_i \sum_{(x^i,a^i)\in\mathcal{X}\times\mathcal{A}} (\rho_n^i(x^i, a^i) - {\rho'}_n^i(x^i, a^i))(r^i(x^i, a^i, \mu_n) - r^i(x^i, a^i, {\mu'}_n)) = \Delta J(\pi, \pi') + \Delta J(\pi', \pi), \tag{C.1}$$

with

$$\Delta J(\pi, \pi') = \sum_i \sum_{(x^i,a^i)\in\mathcal{X}\times\mathcal{A}} (\rho_n^i(x^i, a^i) - {\rho'}_n^i(x^i, a^i))r^i(x^i, a^i, \mu_n),$$

and

$$\Delta J(\pi', \pi) = \sum_i \sum_{(x^i,a^i)\in\mathcal{X}\times\mathcal{A}} ({\rho'}_n^i(x^i, a^i) - \rho_n^i(x^i, a^i))r^i(x^i, a^i, {\mu'}_n).$$

From here, we deduce (5.2). Similarly, the strictly weak monotonicity implies a strict inequality in (5.2).

□

C.6 Strictly weak monotonicity implies uniqueness

Proof of Proposition 1. Consider a strictly weakly monotone game. For the sake of contradiction, assume that there exist two different Nash equilibria, say π, π'.

Proceeding as in the proof of Lemma 1, we obtain (C.1) with a strict inequality.

Note that $\Delta J(\pi, \pi')$ corresponds to the difference between the reward of a typical player following π when the population follows π and the reward of a typical player following π' when the population still follows π, and vice versa for $\Delta J(\pi', \pi)$. Moreover, π, π' are Nash equilibria, so we deduce that these two terms are non-negative, which yields a contradiction with (C.1).

□

C.7 Online Mirror Descent Convergence

Proof of Theorem 5.1. Let Ξ be defined as :

$$\Xi(\pi^*, \pi) := \sum_{i=1}^{N_p} \sum_{n=0}^{N_T} \sum_{x^i \in \mathcal{X}} \mu_n^{i,\pi^*}(x^i)[D_h(\pi_n^{i,*}(x^i, \cdot), \pi_n^i(x^i, \cdot))].$$

We pick $\pi \in \Delta\mathcal{A}$. If $\Delta J(\pi, \pi^*) + \tilde{\mathcal{M}}(\pi, \pi^*) = 0$ then, $\tilde{\mathcal{M}}(\pi, \pi^*) = 0$ and we can deduce that $\mu^\pi = \mu^{\pi^*}$. This implies that π is a Nash as π and π^* share the same distribution and thus the reward of a best response against π or π^* will be the same.

Let us suppose now that π is a Nash and $\Delta J(\pi, \pi^*) + \tilde{\mathcal{M}}(\pi, \pi^*) < 0$, then $\sum_{i=1}^{N_p} J^i(\pi^i, \mu^\pi) - J^i(\pi^{i,*}, \mu^\pi) < 0$ meaning that there exists an i such that $J^i(\pi^i, \mu^\pi) - J^i(\pi^{i,*}, \mu^\pi) < 0$. But as π is a Nash, for all π', i, we have $J^i(\pi^i, \mu^\pi) - J^i(\pi'^i, \mu^\pi) \geq 0$ which is a contradiction.

Hence, if $\Delta J(\pi, \pi^*) + \tilde{\mathcal{M}}(\pi, \pi^*) < 0$ then π is not a Nash.

This proves that the Bregman divergence $\min_{\pi^* \in \text{Nash}} \Xi(\pi^*, .)$ is a strict Lyapunov function of the CTOMD system. Hereby, π_t converges to the set of Nash equilibria. □

Related to the hypothesis in Theorem 5.1, we can show the following:

Lemma 4. *If a MP-MFG satisfies $\tilde{\mathcal{M}}(\pi, \pi') < 0$ if $\mu^\pi \neq \mu^{\pi'}$ and 0 otherwise, then there is at most one Nash equilibrium distribution.*

Note that uniqueness of the equilibrium distribution does not imply uniqueness of the equilibrium policy. This implication holds however under extra assumptions (e.g., some kind of strict convexity of the cost function).

Proof. Consider a MP-MFG satisfying the assumption. Consider two Nash equilibria, say π, π'. For the sake of contradiction, assume that they generate two different distributions $\mu^\pi, \mu^{\pi'}$. We have:

$$\begin{aligned}
0 &> \tilde{\mathcal{M}}(\pi, \pi') \\
&= \sum_{i=1}^{N_p} \left[J^i(\pi^i, \mu^\pi) + J^i(\pi'^i, \mu^{\pi'}) - J^i(\pi^i, \mu^{\pi'}) - J^i(\pi'^i, \mu^\pi)\right] \\
&= \sum_{i=1}^{N_p} \left[J^i(\pi^i, \mu^\pi) - J^i(\pi'^i, \mu^\pi)\right] + \sum_{i=1}^{N_p} \left[J^i(\pi'^i, \mu^{\pi'}) - J^i(\pi^i, \mu^{\pi'})\right]
\end{aligned}$$

where both terms are non-negative because π and π' are Nash equilibria. Hence we must have $\mu^\pi = \mu^{\pi'}$. □

C.8 Numerical Experiments

C.8.1 Garnet

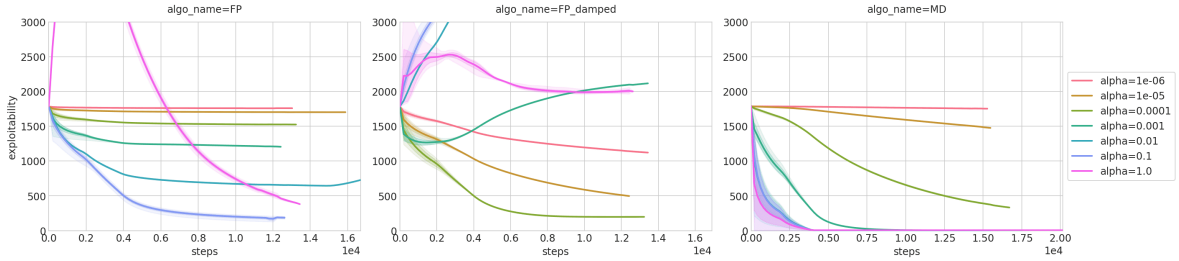


Figure C.1 – 5 garnet sampled with param $n_x = 2000$, $n_a = 20$, $N_T = 2000$, $s_f = 10$

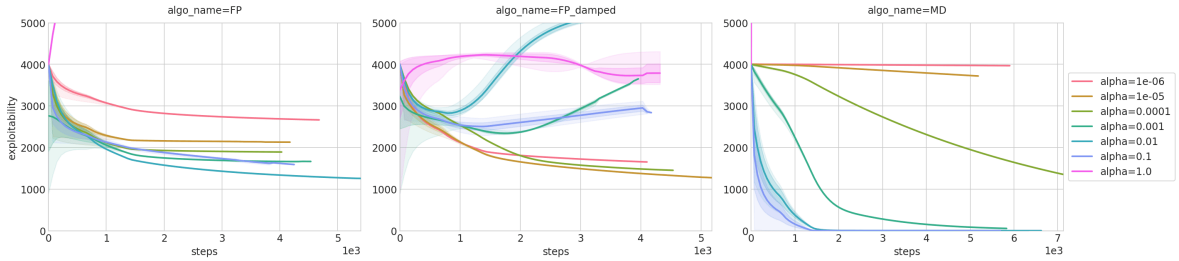


Figure C.2 – 5 garnet sampled with param $n_x = 20000$, $n_a = 10$, $N_T = 2000$, $s_f = 10$

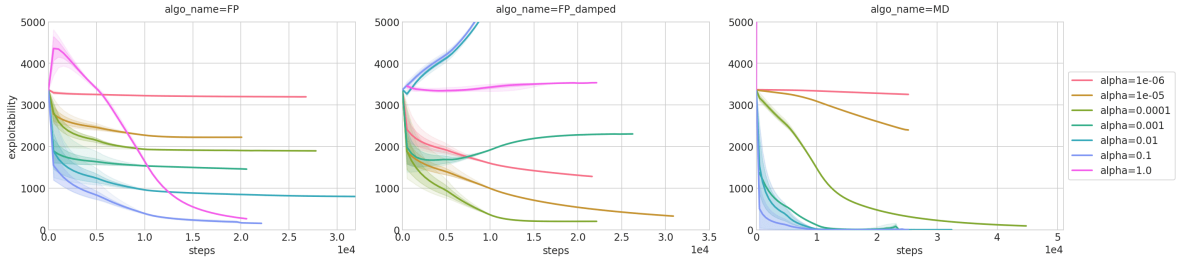


Figure C.3 – 5 garnet sampled with param $n_x = 2000$, $n_a = 10$, $N_T = 2000$, $s_f = 10$

Figure C.4 – Garnet Experiments performances

C.8.2 Building experiment

Building experiment performances

Building experiment solution

The full building evacuation dynamics over the 20 floors is presented in Figure C.9 below.

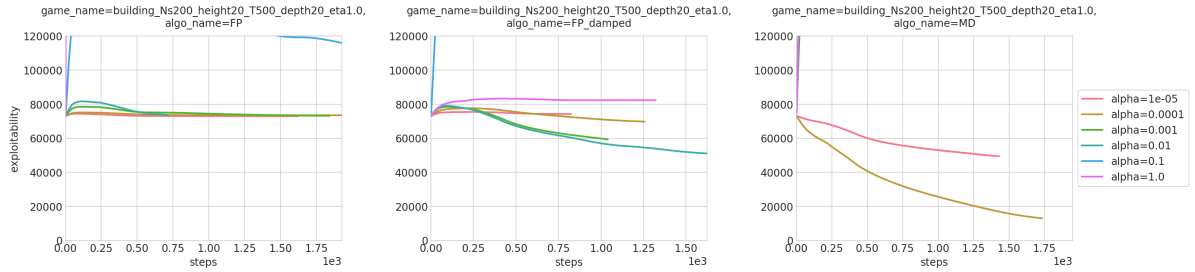


Figure C.5 – Building Experiment performances

C.8.3 Crowd motion with randomly shifted point of interest

In this section, we discuss how to extend our results to the case of multi-population MFGs with common noise. In the example of Section 5.4.3, the common noise corresponds to the geographical shifts of the point of interest.

The action space and the state space are the same but the dynamics and the reward are affected by a common noise sequence $\{\xi_n\}_{0\leq n\leq N}$. We denote $\Xi_n := \{\xi_k\}_{0\leq k<n} = \Xi_{n-1}.\xi_{n-1}$ the concatenation of the sequence Ξ_{n-1} and the new noise ξ_{n-1}. By convention, we denote by Ξ_0 the empty sequence $\{\}$. $|\Xi_n| = n$ represents the total length of the sequence. The distribution of ξ_n given the past sequence Ξ_n is denoted by $P(.|\Xi_n)$. Here, ξ plays the role of a source or randomness which affects both the reward $r(x, a, \mu, \xi)$ and the probability transition function $p(x'|x, a, \xi)$. It appears on top of the idiosyncratic randomness affecting each player. Policies and population distributions are now functions of the common noise and denoted respectively by $\pi^i_n(a|x, \Xi)$ and $\mu^i_n(x|\Xi)$ for population i. We will sometimes simply write $\pi^i_{n|\Xi}(a|x)$ and $\mu^i_{n|\Xi}(x)$. Notice that the common noise is shared by all populations (we could also, with a slight modification, consider noises which are common to players of a given population and not shared with other populations). The Q function of the i-th population now satisfies the following backward equation:

$$
\begin{aligned}
Q_N^{i,\pi^i,\mu}(x^i, a^i|\Xi_N) &= r^i(x^i, a^i, \mu_{N|\Xi_N}, \xi_N) \\
Q_{n-1}^{i,\pi^i,\mu}(x^i, a^i|\Xi_{n-1}) &= \sum_{\xi} P(\xi_{n-1} = \xi|\Xi_{n-1}) \Big[r^i(x^i, a^i, \mu_{n-1|\Xi_{n-1}}, \xi) \\
&\quad + \sum_{x'^i \in \mathcal{X}} p(x'^i|x^i, a^i, \xi) \mathbb{E}_{b^i \sim \pi^i_n(.|x'^i, \Xi_{n-1}.\xi)} \Big[Q_n^{i,\pi^i,\mu}(x^i, b^i|\Xi_{n-1}.\xi) \Big] \Big].
\end{aligned}
$$

For each population, the evolution of the distribution is conditioned on the realization of the common noise. It satisfies the forward equation: for all $x^i \in \mathcal{X}, \mu^{i,\pi^i}_{0|\Xi_0}(x) = \mu^i_0(x)$ and for all $x'^i \in \mathcal{X}$,

$$
\mu^{i,\pi^i}_{n+1|\Xi_n.\xi_n}(x'^i) = \sum_{(x^i,a^i)\in\mathcal{X}\times\mathcal{A}} \pi^i_n(a^i|x^i, \Xi_n) p(x'^i|x^i, a^i, \xi_n) \mu^{i,\pi^i}_{n|\Xi_n}(x^i)
$$

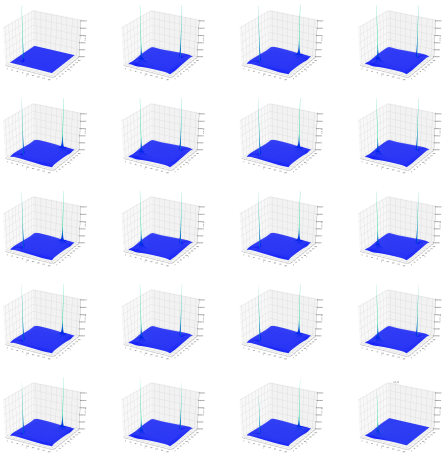


Figure C.6 – After a few timesteps

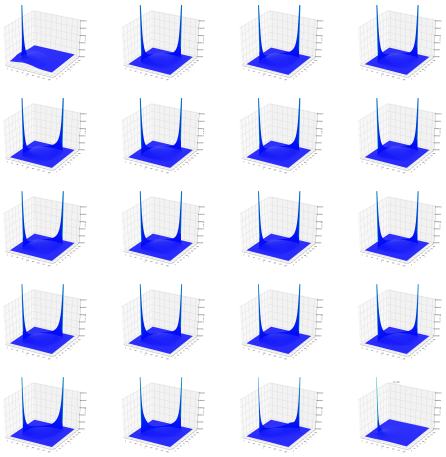


Figure C.7 – Intermediate time

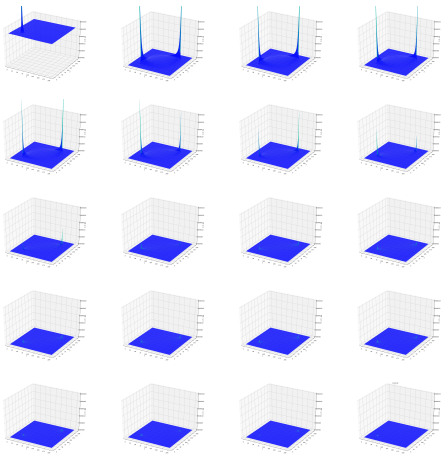


Figure C.8 – Almost at arrival timestep

Figure C.9 – Building Experiment solution (ground floor on the upper left corner)

for $n \leq N-1$. We denote $\mu^{\pi} = (\mu^{i,\pi^i})_{i\in\{1,\dots,N_p\}}$.

The expected total reward for a representative player of population i using policy π^i and facing the crowd behavior given by μ is:

$$J^i(\pi^i,\mu) = \mathbb{E}\Big[\sum_{n=0}^{N_T} r^i(x_n^i, a_n^i, \mu_{n|\Xi_n}, \xi_n) \;\Big|\; x_0^i \sim \mu_0^i, a_n^i \sim \pi_n^i(.|x_n^i,\xi_n), x_{n+1}^i \sim p(.|x_n^i,a_n^i,\xi_n), \xi_n \sim P(.|\Xi_n)\Big].$$

Continuous time Online Mirror Descent for MP-MFGs with common noise:

In this setting, the Continuous Time Online Mirror Descent (CTOMD) algorithm is defined as: for all $i \in \{1, \dots, N_p\}, n \in \{0, \dots, N_T\}$, $y^i_{n,0} = 0$, and for all $t \in \mathbb{R}_+$,

$$y^i_{n,t}(x^i, a^i|\Xi_n) = \int_0^t Q_n^{i,\pi^i_s,\mu^{\pi_s}}(x^i, a^i|\Xi_n)ds, \tag{C.2}$$

$$\pi^i_{n,t}(.|x^i, \Xi_n) = \Gamma(y^i_{n,t}(x^i, .|\Xi_n)). \tag{C.3}$$

Our theoretical results naturally extend to this setting by following similar arguments as the ones in Perrin, Perolat, et al. (2020).

C.8.4 Multi-population

Monotony of the multi-population reward

We prove rigorously that the MP-MFG reward is monotone. As $\tilde{r}(x, a) = 0$, the separability condition is trivially verified. Furthermore, we have:

$$\begin{aligned}\sum_i \sum_{x\in\mathcal{X}} (\mu^i(x) - \mu'^i(x))(\hat{r}^i(x, \mu) - \hat{r}^i(x, \mu')) =& \sum_i \sum_{x\in\mathcal{X}} (\mu^i(x) - \mu'^i(x))(-\log(\mu^i(x)) + \\ & \sum_{j\neq i} \mu^j(x)\bar{r}^{i,j}(x) + \log(\mu^i(x)) - \sum_{j\neq i} \mu^j(x)\bar{r}^{i,j}(x)) \\ =& \underbrace{\sum_i \sum_{x\in\mathcal{X}} (\mu^i(x) - \mu'^i(x))(-\log(\mu^i(x)) + \log(\mu'^i(x)))}_{(1)} + \\ & \underbrace{\sum_i \sum_{x\in\mathcal{X}} (\mu^i(x) - \mu'^i(x))(\sum_{j\neq i} \mu^j(x)\bar{r}^{i,j}(x) - \sum_{j\neq i} \mu^j(x)\bar{r}^{i,j}(x))}_{(2)};\end{aligned}$$

where we have:

- $(1) \leq 0$ because $\forall x, \forall i$, $(\mu^i(x) - \mu'^i(x))(-\log(\mu^i(x)) + \log(\mu'^i(x))) \leq 0$ as log is an increasing function;
- $(2) = 0$ because $\forall i, \forall j, i \neq j, \bar{r}^{i,j}(x) = -\bar{r}^{j,i}(x)$.

Thus,

$$\sum_i \sum_{x\in\mathcal{X}} (\mu^i(x) - \mu'^i(x))(\hat{r}^i(x, \mu) - \hat{r}^i(x, \mu')) \leq 0.$$

Multi-population performances

The performances of Fictitious Play and OMD for the multi-population chasing Mean Field Game with different field topologies and initial distribution are presented in Figure C.16.

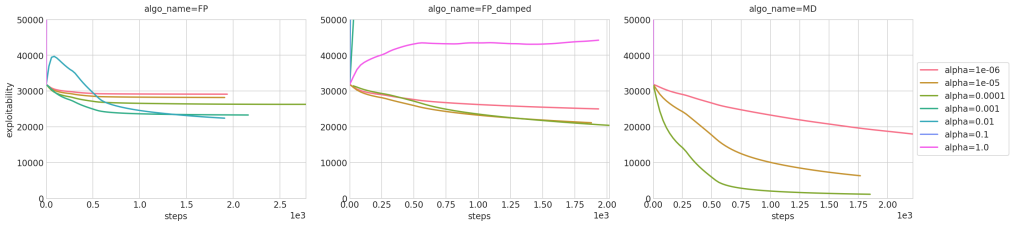


Figure C.10 – Torus topology and corner initialization

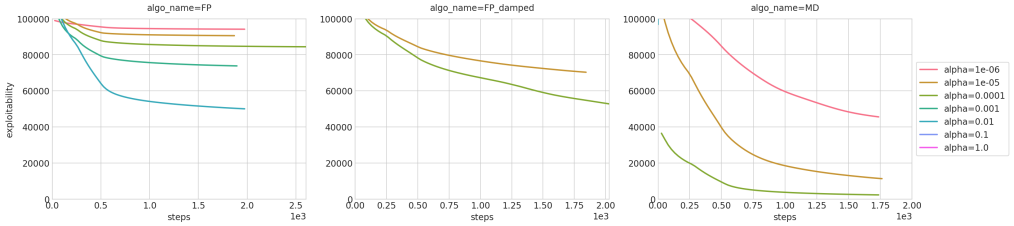


Figure C.11 – Square topology and corner initialization

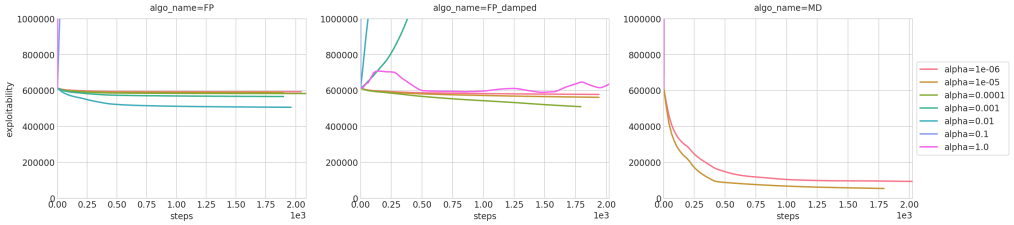


Figure C.12 – Donut topology and corner initialization

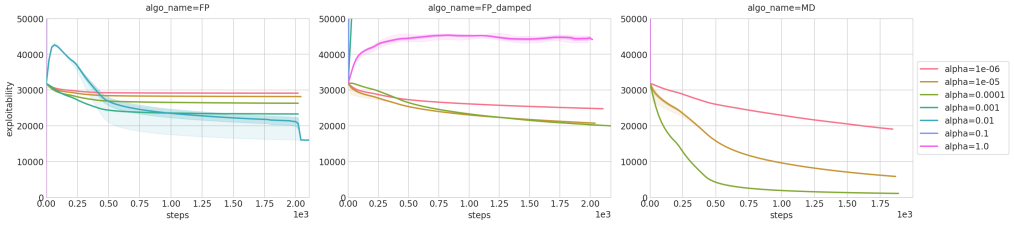


Figure C.13 – Torus topology and random initialization

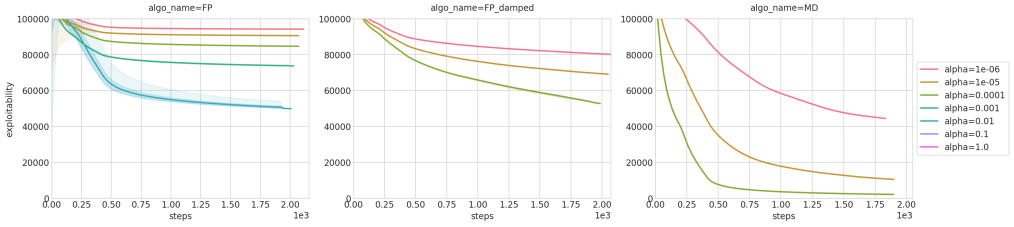


Figure C.14 – Square topology and random initialization

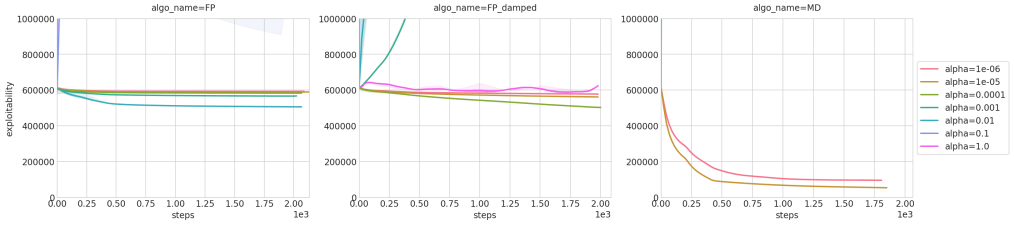


Figure C.15 – Donut topology and random initialization

Figure C.16 – Multi-population experiments, performances with different topologies

Appendix D

Complements on Chapter 6

D.1 More numerical tests

D.1.1 A Simple Example in Four dimensions

We illustrate in a simple four dimensional setting (*i.e.* two-dimensional positions and velocities hence the total dimension is 4) how the agents learn to adopt similar velocities by controlling their acceleration.

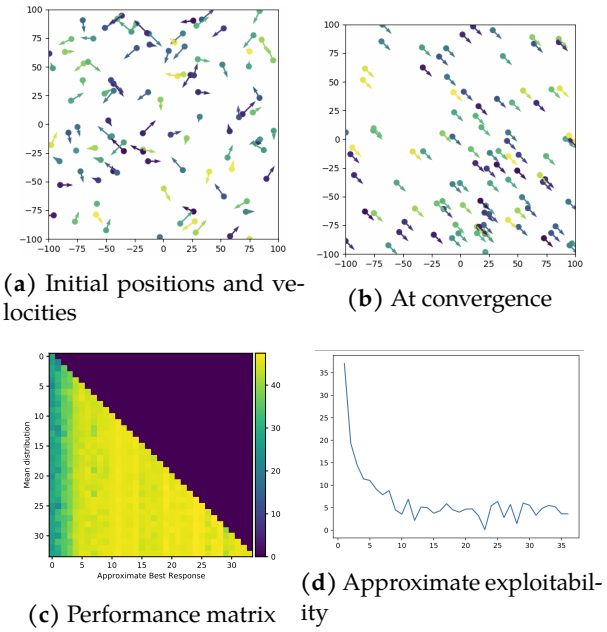
Here, we take $\varepsilon_n^i \equiv 0$ (no noise), and we define the reward as:

$$r_n^i = f_{\beta=0,n}^{\text{flock},i} - \|u_n^i\|_2^2 + \|v_n^i\|_2^2. \tag{D.1}$$

We set $\beta = 0$ to have a intuitive example. The first term encourages the agent to adapt its velocity to the crowd's one, giving equal importance to all the agents irrespective of their distance. The second term penalizes a strong acceleration and we added the last term (not present in the original flocking model) to reduce the number of Nash equilibria and prevent the agent to converge to a degenerate solution which consists in putting the whole crowd at a common position with a null velocity.

As the velocity is bounded by 1 in our experiments, there are at least four obvious Nash equilibria in terms of velocity that remain from reward (D.1): $\hat{v}_n^i \equiv (-1,-1)$, $(-1,1)$, $(1,-1)$ and $(1,1)$, while the position is not important anymore and any distribution for the positions is valid. Experimentally, we observe that if we start from a normal initial distribution $v_0^i \sim \mathcal{N}(0,1)$, then the equilibrium found by Flock'n RL is randomly one of the four previous velocities. However, if we set the initial velocities with a positive or negative bias, then we observe experimentally that the corresponding equilibrium is reached. Thus, as we could expect, the Nash equilibria found by the algorithm depends on the initial distribution.

In Figure D.1, we can see that the agents have adopted the same velocities and the performance matrix indicates that the policy learned at each step of Flock'n RL tends to perform better than previous policies. The exploitablity decreases quickly because Flock'n RL algorithm learns during the first iterations an approximation of a Nash equilibrium.



(**a**) Initial positions and velocities

(**b**) At convergence

(**c**) Performance matrix

(**d**) Approximate exploitability

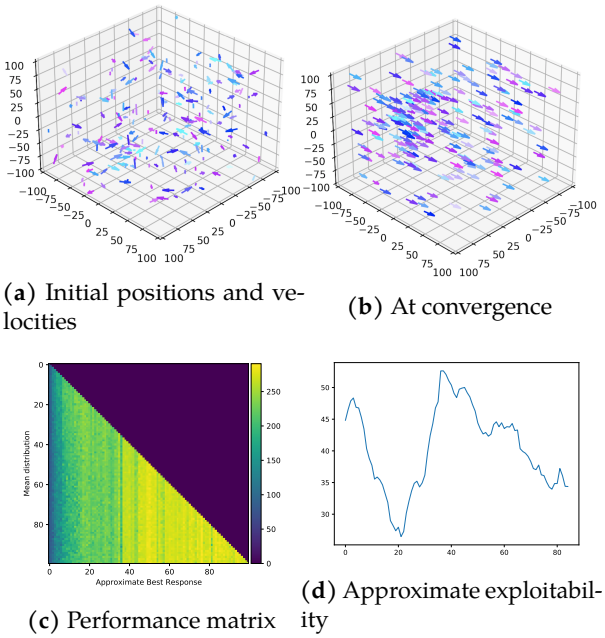
Figure D.1 – Flocking with an intuitive example

D.1.2 A Simple Example in Six Dimensions

We consider the simple example defined with reward from Eq. (D.1) but now we add an additional dimension for the position and the velocity, making the problem six-dimensional. As before, the agents are still encouraged to maximize their velocities. We set $\beta = 0$ and we do not put any noise on the velocity dynamics. We can notice that experimentally a consensus is reached by the agents as they learn to adopt the same velocities (Figure D.2b), even if the agents start from a random distribution in terms of positions and velocities (Figure D.2a). The performance matrix (Figure D.2c) highlights that the best response improves until iteration 40, which explains why there is a bump in performance in the exploitability before the 40th iteration.

D.1.3 Examples with an obstacle

We present an example with a single obstacle and noise in the dynamics, both in dimension 4 and 6. In dimension 4, the obstacle is a square located at the middle of the environment, whereas in dimension 6 it is a cube. In dimension 4, we can see in Figure D.3 that the agents



(**a**) Initial positions and velocities

(**b**) At convergence

(**c**) Performance matrix

(**d**) Approximate exploitability

Figure D.2 – Flocking for a simple example in six dimensions.

that are initially spawned in the environment with random positions and velocities manage to learn to adopt the same velocities while avoiding the obstacle (Figure D.3b). The same behavior is observed in dimension 6. We also notice that the exploitability is slower to decrease in dimension 6, similarly to what we observed with the simple example without the obstacle.

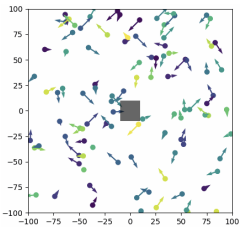
D.1.4 Many obstacles in 4D

Finally, we present an example in 4D with many obstacles, in the same fashion than the 6-dimensional example with the columns located in the main part of the article.

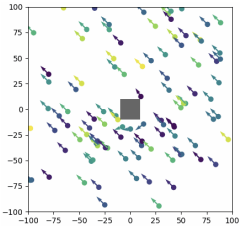
D.2 Normalizing Flows

In this section we provide some background on Normalizing Flows, which are an important part of our approach.

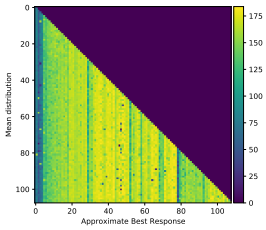
Coupling layers. A coupling layer applies a coupling transform, which maps an input x to an output y by first splitting x into two parts $x = [x_{1:d-1}, x_{d:D}]$, computing parameters $\theta = NN(x_{1:d-1})$ of an arbitrary neural network (in our case a fully connected neural network with 8 hidden units), applying g to the last coordinates $y_i = g_{\theta_i}(x_i)$ where $i \in [d, \dots, D]$ and g_{θ_i} is an invertible function parameterized by θ_i. Finally, we set $y_{1:d-1} = x_{1:d-1}$. Coupling



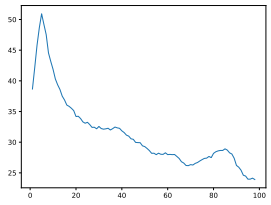
(**a**) Initial positions and velocities



(**b**) At convergence

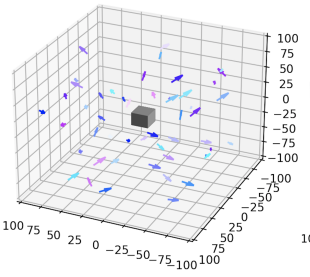


(**c**) Performance matrix

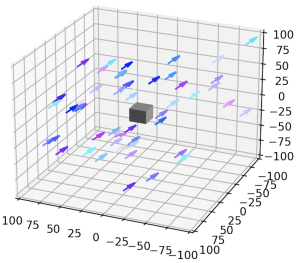


(**d**) Approximate exploitability

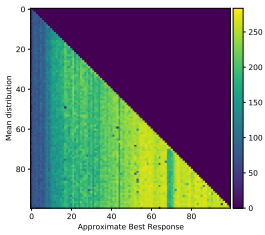
Figure D.3 – Flocking in 4D with one obstacle.



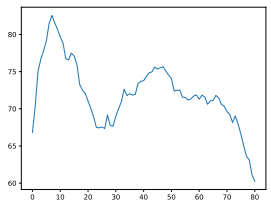
(**a**) Initial positions and velocities



(**b**) At convergence

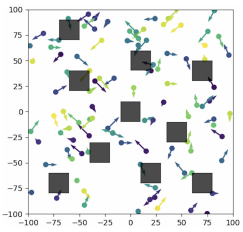


(**c**) Performance matrix

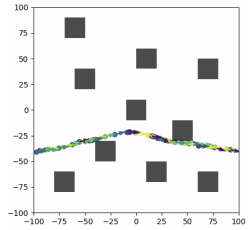


(**d**) Approximate exploitability

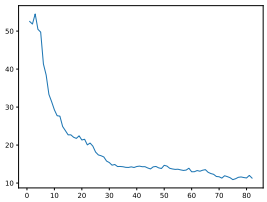
Figure D.4 – Flocking in 6D with one obstacle.



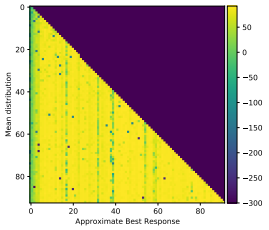
(**a**) Initial positions and velocities



(**b**) At convergence



(**c**) Performance matrix



(**d**) Approximate exploitability

Figure D.5 – Flocking in 4D with many obstacles.

transforms offer the benefit of having a tractable Jacobian determinant, and they can be inverted exactly in a single pass.

Neural Spline Flows. NSFs are based on monotonic rational-quadratic splines. These splines are used to model the function g_{θ_i}. A rational-quadratic function takes the form of a quotient of two quadratic polynomials, and a spline uses K different rational-quadratic functions.

Following the implementation described in Durkan et al., 2019, we detail how a NSF is computed.

1. A neural network NN takes $x_{1:d-1}$ as inputs and outputs θ_i of length $3K-1$ for each $i \in [1, \dots, D]$.

2. θ_i is partitioned as $\theta_i = [\theta_i^w, \theta_i^h, \theta_i^d]$, of respective of sizes K, K, and $K-1$.

3. θ_i^w and θ_i^h are passed through a softmax and multiplied by $2B$, the outputs are interpreted as the widths and heights of the K bins. Cumulative sums of the K bin widths and heights yields the $K+1$ knots $(x^k, y^k)_{k=0}^K$.

4. θ_i^d is passed through a softplus function and is interpreted as the values of the derivatives of the internal knots.

D.3 Visual Rendering with Unity

Once we have trained the policy with the Flock'n RL algorithm, we generate trajectories of many agents and stock them in a csv file. We have coded an integration in Unity, making it possible to load these trajectories and visualize the flock in movement, interacting with its environment. We can then easily load prefab models of fishes, birds, or any animal that we want our agents to be. We can also load the obstacles and assign them any texture we want. Examples of rendering are available in Figure D.6.

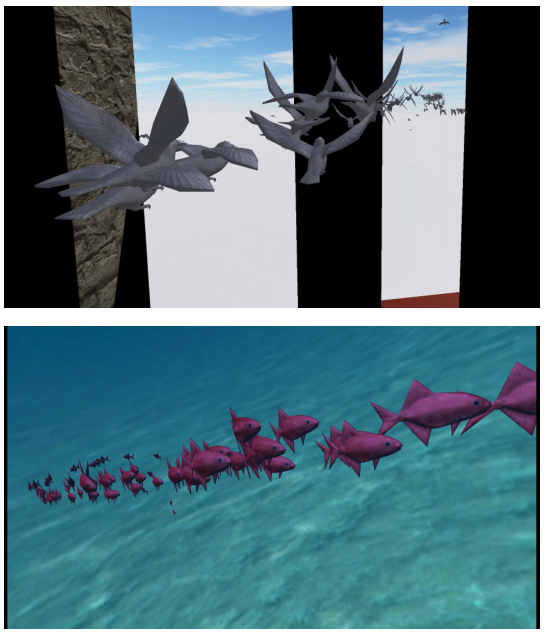


Figure D.6 – Visual rendering with Unity

Appendix E

Complements on Chapter 7

E.1 Notations

The main notations used in the text are summarized in the following table. Please note that $\boldsymbol{\pi}$, $\overline{\pi}$ and $\hat{\pi}$ are population-agnostic policies, while $\widetilde{\pi}$, $\overline{\widetilde{\pi}}$ and $\widetilde{\pi}^*$ are population-dependent policies.

Policy	$\boldsymbol{\pi} \in \boldsymbol{\Pi}$
Average policy	$\overline{\pi} \in \boldsymbol{\Pi}$
Equilibrium policy	$\hat{\pi} \in \boldsymbol{\Pi}$
Population-dependent policy	$\widetilde{\pi} \in \widetilde{\Pi}$
Average population-dependent policy	$\overline{\widetilde{\pi}} \in \widetilde{\Pi}$
Master policy	$\widetilde{\pi}^* \in \widetilde{\Pi}$
Mean field state	$\mu \in \mathcal{M}$
Mean field flow	$\boldsymbol{\mu} \in \mathbf{M}$
Training set of initial distributions	$\mathcal{M} \subset \mathcal{M}$

E.2 Experimental Details

Wasserstein distance. The Wasserstein distance W (or earth mover's distance) measures the minimum cost of turning one distribution into another: for $\mu, \mu' \in \mathcal{M} = \Delta_{\mathcal{X}}$,

$$W(\mu, \mu') = \inf_{\nu \in \Gamma(\mu, \mu')} \sum_{(x,x') \in \mathcal{X} \times \mathcal{X}} d(x, x') \nu(x, x'),$$

where $\Gamma(\mu, \mu')$ is the set of probability distributions on $\mathcal{X} \times \mathcal{X}$ with marginals μ and μ'. This notion is well defined if the state space has a natural notion of distance d, which is the case

in our numerical examples because they come from the discretization of 1D or 2D Euclidean domains.

Initial distributions. We provide here a representation of the initial distributions used in the experiments.

For the pure exploration model in 1D, the training and testing sets are represented in Figure E.1 and Figure E.2 respectively.

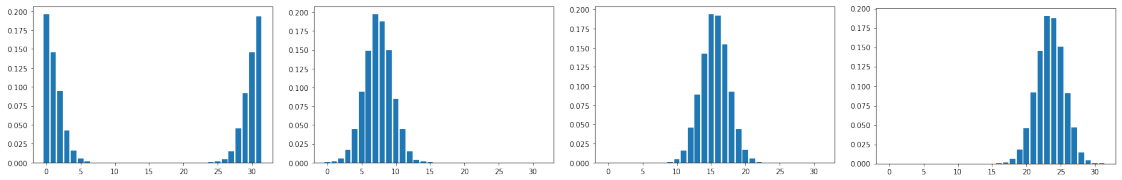


Figure E.1 – Pure exploration 1D: Training set

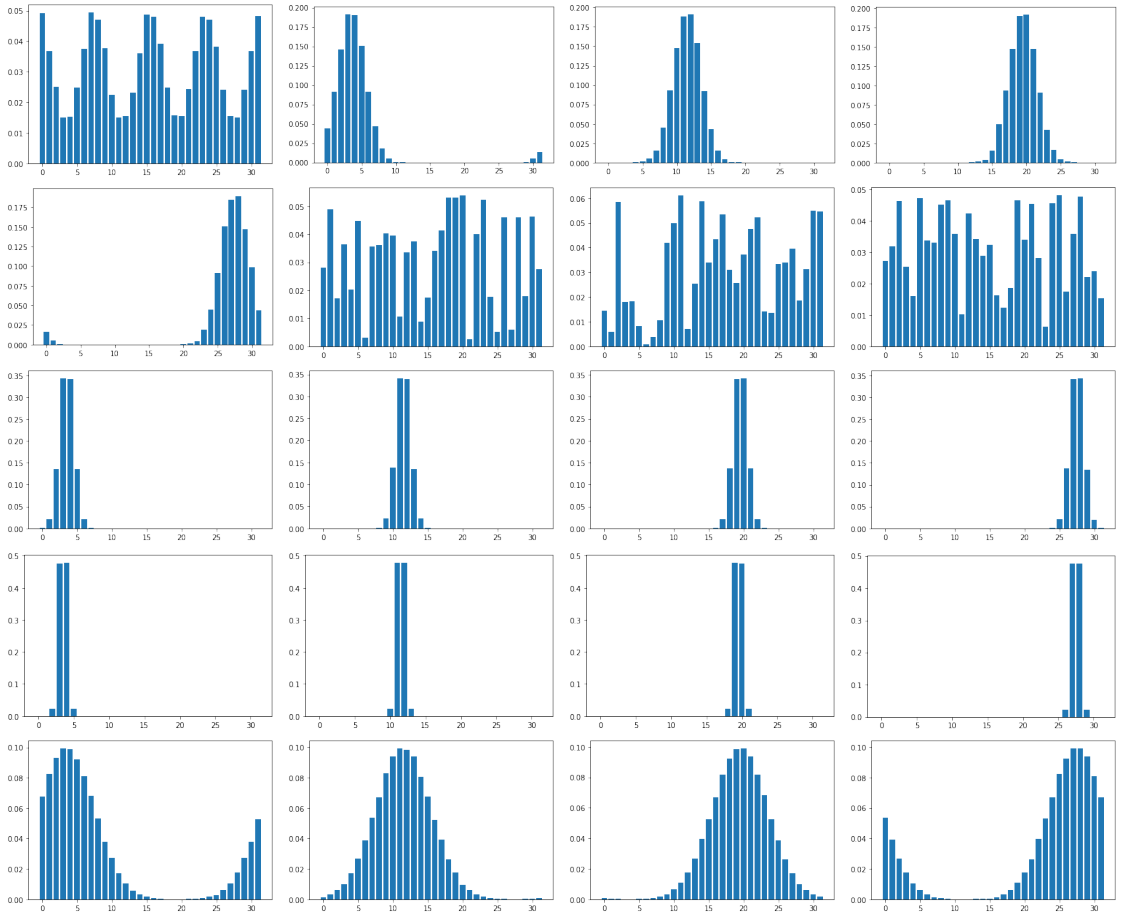


Figure E.2 – Pure exploration 1D: Testing set

For the beach bar model in 2D, the training and testing sets are represented in Figure E.3 and Figure E.4 respectively.

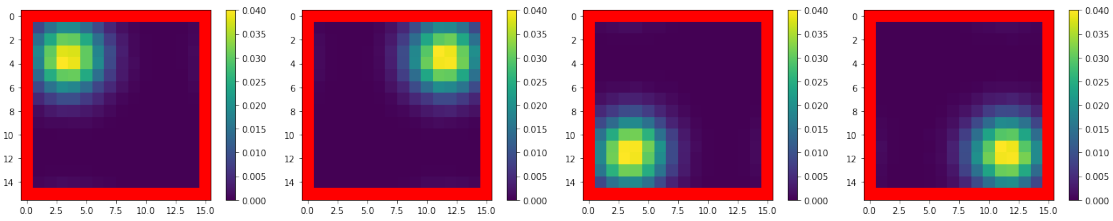


Figure E.3 – Beach bar 2D: Training set

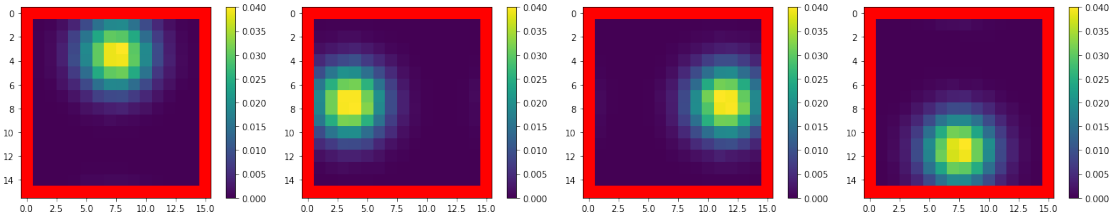


Figure E.4 – Beach bar 2D: Testing set

E.3 Learning a Population-dependent Policy with Deep RL

Recall that in line 4 of Algorithm 7.1, we want solve an MDP which is stationary because we have put the distribution μ as an input together with the agent's state x. To this end, we use DQN and, as described in Algorithm E.1, we use a finite horizon approximation N_T. This approximation is common in the literature and is not problematic as we set the horizon high enough so that the stationary population distribution can be (approximately) reached.

E.4 Proof of Theorem 7.1

Proof of Theorem 7.1. By assumption, for every m_0, there is a unique equilibrium MF flow $\hat{\boldsymbol{\mu}}^{m_0}$. We also consider an associated equilibrium (population-agnostic) policy $\hat{\boldsymbol{\pi}}^{m_0}$ (if there are multiple choices of such policies, we take one of them). The superscript is used to stress the dependence on the initial MF state. Let us define the following population dependent policy:

$$\widetilde{\pi}(x, m_0) := \hat{\boldsymbol{\pi}}_0^{m_0}(x). \tag{E.1}$$

We prove that any population-dependent policy defined in the above way is a master policy, *i.e.*, for each m_0 it gives an equilibrium policy not only at initial time but at all time steps.

Algorithm E.1: DQN for a population-dependent Best Response

input : Initial weights θ_k and θ'_k for network $\widetilde{Q}_{\theta_k}$ and target network $\widetilde{Q}_{\theta'_k}$; training set $\mathcal{M}$ of initial distributions; set $\bar{\mathcal{M}}_k$ of average MF flows; number of episodes N_{episodes}; number of inner steps N; horizon N_T for estimation; number of steps C between synchronization of the two networks; parameter $\varepsilon \in [0,1]$ for exploration

Initialize weights θ_k of network $\widetilde{Q}_{\theta_k}$ and weights θ'_k of target network $\widetilde{Q}'_{\theta'_k}$

Initialize replay memory B

for $e = 1, \dots, N_{\text{episodes}}$ **do**

  Sample initial $m_0 \in \mathcal{M}$ and get the associated $\bar{\boldsymbol{\mu}}_k^{m_0}$ from $\bar{\mathcal{M}}_k$

  Sample $x_0 \sim m_0$

  for $n = 0, \dots, N-1$ **do**

    With probability ε select random action a_n, otherwise select $a_n \in \arg\max_a \widetilde{Q}'_k(a|x_n, \bar{\boldsymbol{\mu}}_{k,n}^{m_0})$

    Execute action a_n, observe reward r_n and state x_{n+1}

    Add the transition $(x_n, a_n, \bar{\boldsymbol{\mu}}_{k,n}^{m_0}, r_n, \bar{\boldsymbol{\mu}}_{k,n+1}^{m_0})$ to B

  end

  Sample a random minibatch of N_T transitions $\{(x_n, a_n, \mu_n, r_n, \mu_{n+1}), n = 1, \dots, N_T\}$ from B

  Let $v_n = r_n + \gamma \max_{a'} \widetilde{Q}_{\theta_k}(x_{n+1}, \mu_{n+1}, a')$ for $n = 1, \dots, N_T$

  Update θ_k by performing a gradient step in the direction of minimizing w.r.t. θ:

$$\frac{1}{N_T} \sum_{n=1}^{N_T} \left| v_n - \widetilde{Q}_\theta(x_n, \mu_n, a_n) \right|^2$$

  Every C steps, copy weights θ_k of $\widetilde{Q}_{\theta_k}$ to the weights θ'_k of $\widetilde{Q}'_{\theta'_k}$

end

return $\widetilde{Q}_{\theta_k}$

Fix m_0. Let $\tilde{\boldsymbol{\mu}}^{m_0}$ and $\tilde{\boldsymbol{\pi}}^{m_0}$ be the MF flow and the population-agnostic policy induced by using $\widetilde{\pi}$ starting from m_0, *i.e.*, for $n \geq 0$,

$$\tilde{\boldsymbol{\pi}}_n^{m_0}(x) = \widetilde{\pi}(x, \tilde{\boldsymbol{\mu}}_n^{m_0}), \qquad \tilde{\boldsymbol{\mu}}_{n+1}^{m_0} = \phi(\tilde{\boldsymbol{\mu}}_n^{m_0}, \tilde{\boldsymbol{\pi}}_n^{m_0}). \tag{E.2}$$

We check that it is a Nash equilibrium starting with m_0. The second condition in Definition 6 is automatically satisfied by definition of $\tilde{\boldsymbol{\mu}}_{n+1}^{m_0}$, see (E.2). For the optimality condition, we proceed by induction to show that for every $n \geq 0$, $\tilde{\boldsymbol{\mu}}_n^{m_0} = \hat{\boldsymbol{\mu}}_n^{m_0}$, which is the unique equilibrium MF flow starting from m_0. Note first that, by (E.1) and dynamic programming,

$$\tilde{\boldsymbol{\pi}}_0^{m_0}(x) = \widetilde{\pi}(x, m_0) = \hat{\boldsymbol{\pi}}_0^{m_0}(x) \in \arg\max_{\pi \in \Pi} \mathbb{E}\Big[r(x, a, m_0) + \gamma \hat{V}(x_1; \hat{\boldsymbol{\mu}}_1^{m_0}) \;\Big|\; x_1 \sim p(.|x, a, m_0),\ a \sim \pi(.|x) \Big],$$

where $\hat{V}$ is the stationary and population-dependent value function for a representative agent facing a population playing according to a Nash equilibrium starting from a given distribution. Moreover,

$$\tilde{\boldsymbol{\mu}}_1^{m_0} = \phi(\tilde{\boldsymbol{\mu}}_0^{m_0}, \tilde{\boldsymbol{\pi}}_0^{m_0}) = \phi(m_0, \tilde{\pi}(\cdot, m_0)) = \phi(m_0, \hat{\boldsymbol{\pi}}_0^{m_0}) = \hat{\boldsymbol{\mu}}_1^{m_0},$$

where we used (E.2) for the first and second equalities, and (E.1) for the third equality. The last equality holds because $(\hat{\boldsymbol{\mu}}^{m_0}, \hat{\boldsymbol{\pi}}^{m_0})$ is an MFG Nash equilibrium consistent with m_0. So:

$$\tilde{\boldsymbol{\pi}}_0^{m_0}(x) \in \arg\max_{\pi \in \Pi} \mathbb{E}\Big[r(x, a, m_0) + \gamma \hat{V}(x_1; \tilde{\boldsymbol{\mu}}_1^{m_0}) \;\Big|\; x_1 \sim p(.|x, a, m_0),\ a \sim \pi(.|x) \Big],$$

At time $n \geq 1$, for the sake of induction, assume $\tilde{\boldsymbol{\mu}}_i^{m_0} = \hat{\boldsymbol{\mu}}_i^{m_0}$ for all $i \leq n$. Then

$$\tilde{\boldsymbol{\pi}}_n^{m_0}(x) = \tilde{\pi}(x, \tilde{\boldsymbol{\mu}}_n^{m_0}) = \hat{\boldsymbol{\pi}}_0^{\hat{\boldsymbol{\mu}}_n^{m_0}}(x) \in \arg\max_{\pi \in \Pi} \mathbb{E}\Big[r(x, a, \tilde{\boldsymbol{\mu}}_n^{m_0}) + \gamma \hat{V}(x_1; \hat{\boldsymbol{\mu}}_1^{\tilde{\boldsymbol{\mu}}_n^{m_0}}) \;\Big|\; x_1 \sim p(.|x, a, \tilde{\boldsymbol{\mu}}_n^{m_0}),\ a \sim \pi(.|x) \Big]. \tag{E.3}$$

Moreover,

$$\tilde{\boldsymbol{\mu}}_{n+1}^{m_0} = \phi(\tilde{\boldsymbol{\mu}}_n^{m_0}, \tilde{\boldsymbol{\pi}}_n^{m_0}) = \phi(\hat{\boldsymbol{\mu}}_n^{m_0}, \hat{\boldsymbol{\pi}}_0^{\hat{\boldsymbol{\mu}}_n^{m_0}}) \underbrace{=}_{(\star)} \phi(\hat{\boldsymbol{\mu}}_n^{m_0}, \hat{\boldsymbol{\pi}}_n^{m_0}) = \hat{\boldsymbol{\mu}}_{n+1}^{m_0},$$

where the first equality is by (E.2) and the second equality is by the induction hypothesis and (E.3). Equality $(\star)$ means that the population distributions generated at the next time step by $\hat{\boldsymbol{\pi}}_0^{\hat{\boldsymbol{\mu}}_n^{m_0}}$ and $\hat{\boldsymbol{\pi}}_n^{m_0}$ when starting from $\hat{\boldsymbol{\mu}}_n^{m_0}$ are the same (although these two policies could be different). This is because both of them are best responses to this population distribution and because we assumed uniqueness of the equilibrium MF flow. Indeed, by definition, $\hat{\boldsymbol{\pi}}_0^{\hat{\boldsymbol{\mu}}_n^{m_0}}$ is the initial step of a policy which is part of an MFG Nash equilibrium consistent with $\hat{\boldsymbol{\mu}}_n^{m_0}$. Furthermore, $(\hat{\boldsymbol{\pi}}_n^{m_0}, \hat{\boldsymbol{\mu}}_n^{m_0})_{n \geq 0}$ is an MFG Nash equilibrium consistent with m_0 and, as a consequence, for any $n_0 \geq 0$, $(\hat{\boldsymbol{\pi}}_n^{m_0}, \hat{\boldsymbol{\mu}}_n^{m_0})_{n \geq n_0}$ is an MFG Nash equilibrium consistent with $\hat{\boldsymbol{\mu}}_{n_0}^{m_0}$.

We conclude that $(\star)$ holds by using the fact that we assumed uniqueness of the equilibrium MF flow so these two policies must have the same result in terms of generated population distribution.

So we proved that $\tilde{\boldsymbol{\mu}}_n^{m_0} = \hat{\boldsymbol{\mu}}_n^{m_0}$ for all $n \geq 0$ and $(\tilde{\boldsymbol{\pi}}_n^{m_0})_{n \geq 0}$ is an associated equilibrium policy.

□

E.5 On the Convergence of Master Fictitious Play

In this section we study the evolution of the averaged MF flow generated by the Master Fictitious Play algorithm, see Algorithm 7.1. We then introduce a continuous time version of this algorithm and prove its convergence at a linear rate.

On the mixture of policies. Given $\overline{\widetilde{\pi}}_K = \text{UNIFORM}(\widetilde{\pi}_1, \dots, \widetilde{\pi}_K)$ and given an initial m_0, we first compute an average population distribution composed of K subpopulations where subpopulation k uses $\widetilde{\pi}_k$ to react to the current average population. Formally, recall that we define:

$$\begin{cases} \boldsymbol{\mu}_{k,0}^{m_0} = m_0, & k = 1, \dots, K \\ \bar{\boldsymbol{\mu}}_{K,0}^{m_0} = \frac{1}{K} \sum_{k=1}^K \boldsymbol{\mu}_{k,0}^{m_0} \end{cases}$$

and for $n \geq 0$,

$$\begin{cases} \boldsymbol{\mu}_{k,n+1}^{m_0} = \phi(\boldsymbol{\mu}_{k,n}^{m_0}, \widetilde{\pi}_k(\cdot|\cdot, \bar{\boldsymbol{\mu}}_{K,n}^{m_0})), & k = 1, \dots, K \\ \bar{\boldsymbol{\mu}}_{K,n+1}^{m_0} = \frac{1}{K} \sum_{k=1}^K \boldsymbol{\mu}_{k,n+1}^{m_0}. \end{cases}$$

We recall that the notation $\boldsymbol{\mu}_{k,n+1}^{m_0} = \phi(\boldsymbol{\mu}_{k,n}^{m_0}, \widetilde{\pi}_k(\cdot|\cdot, \bar{\boldsymbol{\mu}}_{K,n}^{m_0}))$ means:

$$\boldsymbol{\mu}_{k,n+1}^{m_0}(x) = \phi(\boldsymbol{\mu}_{k,n}^{m_0}, \widetilde{\pi}_k(\cdot|x, \bar{\boldsymbol{\mu}}_{K,n}^{m_0})) = \sum_{x'} \boldsymbol{\mu}_{k,n}^{m_0}(x') \sum_a \widetilde{\pi}_k(a|x', \bar{\boldsymbol{\mu}}_{K,n}^{m_0}) p(x|x', a, \bar{\boldsymbol{\mu}}_{K,n}^{m_0}), \qquad x \in \mathcal{X}. \tag{E.4}$$

Hence: for all $x \in \mathcal{X}$,

$$\bar{\boldsymbol{\mu}}_{K,n+1}^{m_0}(x) = \frac{1}{K} \sum_{k=1}^K \sum_{x'} \boldsymbol{\mu}_{k,n}^{m_0}(x') \sum_a \widetilde{\pi}_k(a|x', \bar{\boldsymbol{\mu}}_{K,n}^{m_0}) p(x|x', a, \bar{\boldsymbol{\mu}}_{K,n}^{m_0}) \tag{E.5}$$

$$= \sum_{x'} \bar{\boldsymbol{\mu}}_{K,n}^{m_0}(x') \sum_a \underbrace{\left(\frac{1}{K} \sum_{k=1}^K \frac{\boldsymbol{\mu}_{k,n}^{m_0}(x')}{\bar{\boldsymbol{\mu}}_{K,n}^{m_0}(x')} \widetilde{\pi}_k(a|x', \bar{\boldsymbol{\mu}}_{K,n}^{m_0}) \right)}_{=: \bar{\widetilde{\pi}}_{K,n}^{m_0}(a|x', \bar{\boldsymbol{\mu}}_{K,n}^{m_0})} p(x|x', a, \bar{\boldsymbol{\mu}}_{K,n}^{m_0}), \tag{E.6}$$

where, in the last expression, the first sum over $\{x' \in \mathcal{X} : \bar{\boldsymbol{\mu}}_{K,n}^{m_0}(x') > 0\}$. So the evolution of the average population can be interpreted as the fact that all the agents use the policy $\bar{\widetilde{\pi}}_{K,n}^{m_0}(a|x', \bar{\boldsymbol{\mu}}_{K,n}^{m_0})$ given by the terms between parentheses above. Note that this policy depends on m_0 and n.

We then consider the reward obtained by an infinitesimal player from the average population. This player belongs to subpopulation k with probability $1/K$. So the reward can be expressed as:

$$\frac{1}{K} \sum_{k=1}^K J(m_0, \widetilde{\pi}_k; \bar{\boldsymbol{\mu}}_K^{m_0}).$$

We expect that when $K \to +\infty$ (*i.e.*, we run more iterations of the Master Fictitious Play algorithm, see Algorithm 7.1), then this quantity converges to the one obtained by a typical player in the Nash equilibrium starting from m_0, *i.e.*:

$$J(m_0, \hat{\boldsymbol{\pi}}; \hat{\boldsymbol{\mu}}^{m_0})$$

where $\hat{\boldsymbol{\mu}}^{m_0} = \Phi(m_0, \hat{\boldsymbol{\pi}})$ with $\hat{\boldsymbol{\pi}} \in \arg\max_{\boldsymbol{\pi}} J(m_0, \boldsymbol{\pi}; \hat{\boldsymbol{\mu}}^{m_0})$.

Note that $\bar{\tilde{\pi}}^{m_0}_{K,n}(a|x', \bar{\boldsymbol{\mu}}^{m_0}_{K,n})$ takes $\bar{\boldsymbol{\mu}}^{m_0}_{K,n}$ as an input. However, this dependence is superfluous because $\bar{\boldsymbol{\mu}}^{m_0}_{K,n}$ can be derived from m_0 and $(\bar{\tilde{\pi}}^{m_0}_{K,m}(a|x', \bar{\boldsymbol{\mu}}^{m_0}_{K,m}))_{m\leq n}$. Proceeding by induction, we can show that there exists $\bar{\boldsymbol{\pi}}^{m_0}_K \in \boldsymbol{\Pi}$ s.t.

$$\bar{\tilde{\pi}}^{m_0}_{K,n}(a|x', \bar{\boldsymbol{\mu}}^{m_0}_{K,n}) = \bar{\boldsymbol{\pi}}^{m_0}_{K,n}(a|x')$$

Continuous Time Master Fictitious Play. We now describe the Continuous Time Master Fictitious Play (CTMFP) scheme in our setting. Here the iteration index $k \in \{1, 2, 3, \dots\}$ is replaced by a time t, which takes continuous values in $[1, +\infty)$. Intuitively, it corresponds to the limiting regime where the updates happen continuously.

Based on (E.5), we introduce the CTMFP mean-field flow defined for all $t \geq 1$ by: $\bar{\boldsymbol{\mu}}^{m_0}_{t,n} = \boldsymbol{\mu}^{m_0,\mathtt{BR}}_{t,n} = m_0$, and for $n = 1, 2, \dots,$

$$\bar{\boldsymbol{\mu}}^{m_0}_{t,n}(x) = \frac{1}{t}\int_{s=0}^{t} \boldsymbol{\mu}^{m_0,\mathtt{BR}}_{s,n}(x)ds, \quad \text{or in differential form:} \quad \frac{d}{dt}\bar{\boldsymbol{\mu}}^{m_0}_{t,n}(x) = \frac{1}{t}\left(\boldsymbol{\mu}^{m_0,\mathtt{BR}}_{t,n}(x) - \bar{\boldsymbol{\mu}}^{m_0}_{t,n}(x)\right), \tag{E.7}$$

where $\boldsymbol{\mu}^{m_0,\mathtt{BR}}_{t,n}$ denotes the distribution induced by a best response policy $(\boldsymbol{\pi}^{m_0,\mathtt{BR}}_{t,n})_{n\geq 0}$ against $\bar{\boldsymbol{\mu}}^{m_0}_{t,n}(x)$.

As in (E.6) for the discrete update case, the distribution $\bar{\boldsymbol{\mu}}^{m_0}_{t,n}$ corresponds to the population distribution induced by the averaged policy $(\bar{\boldsymbol{\pi}}^{m_0}_{t,n})_n$ defined as follows: for all $n = 1, 2, \dots,$ and all $t \geq 1$:

$$\bar{\boldsymbol{\pi}}^{m_0}_{t,n}(a|x)\int_{s=0}^{t} \boldsymbol{\mu}^{m_0,\mathtt{BR}}_{s,n}(x)ds = \int_{s=0}^{t} \boldsymbol{\mu}^{m_0,\mathtt{BR}}_{s,n}(x)\boldsymbol{\pi}^{m_0,\mathtt{BR}}_{s,n}(a|x)ds \tag{E.8}$$

$$\text{or in differential form: } \bar{\boldsymbol{\mu}}^{m_0}_{t,n}(x)\frac{d}{dt}\bar{\boldsymbol{\pi}}^{m_0}_{t,n}(a|x) = \frac{1}{t}\boldsymbol{\mu}^{m_0,\mathtt{BR}}_{t,n}(x)[\boldsymbol{\pi}^{m_0,\mathtt{BR}}_{t,n}(a|x) - \bar{\boldsymbol{\pi}}^{m_0}_{t,n}(a|x)]. \tag{E.9}$$

The CTMFP process really starts from time $t = 1$, but it is necessary to define what happens just before this starting time. For $t \in [0, 1)$, we define $\bar{\boldsymbol{\pi}}^{m_0}_{t<1} = (\bar{\boldsymbol{\pi}}^{m_0}_{t<1,n})_n = (\boldsymbol{\pi}^{m_0,\mathtt{BR}}_{t<1,n})_n$, where $\boldsymbol{\pi}^{m_0,\mathtt{BR}}_{t<1}$ is constant and equal to an arbitrary policy. The induced distribution between time 0 and 1 is $\bar{\boldsymbol{\mu}}^{m_0}_{t<1} = \boldsymbol{\mu}^{m_0}_{t<1} = \boldsymbol{\mu}^{m_0,\boldsymbol{\pi}^{m_0}_{t<1}} = (\boldsymbol{\mu}^{m_0,\boldsymbol{\pi}^{m_0}_{t<1}}_n)_{n\geq 0}$.

Proof of convergence. We assume the transition p is independent of the distribution: $x_{n+1} \sim p(.|x_n, a_n)$, and we assume the reward can be split as:

$$r(x, a, \mu) = r_A(x, a) + r_M(x, \mu). \tag{E.10}$$

A useful property is the so-called monotonicity condition, introduced by Lasry and Lions (2007).

Definition 9. *The MFG is said to be* monotone *if: for all* $\mu \neq \mu' \in \mathcal{M}$,

$$\sum_x (\mu(x) - \mu'(x))(r_M(x,\mu) - r_M(x,\mu')) < 0. \tag{E.11}$$

This condition intuitively means that the agent gets a lower reward if the population density is larger at its current state. Monotonicity implies that for every m_0, there exists at most one MF Nash equilibrium consistent with m_0; see (Lasry and Lions, 2007). This can be checked by considering the exploitability.

Here, we are going to use the average exploitability as introduced in (7.3):

$$\bar{\mathcal{E}}_{\mathcal{M}}(\bar{\boldsymbol{\pi}}_t) = \mathbb{E}_{m_0 \sim \text{UNIFORM}(\mathcal{M})}\left[\bar{\mathcal{E}}(m_0, \bar{\boldsymbol{\pi}}_t^{m_0})\right],$$

where $\bar{\boldsymbol{\pi}}_t = (\bar{\boldsymbol{\pi}}_t^{m_0})_{m_0 \in \mathcal{M}}$ is the uniform distribution over past best responses $(\boldsymbol{\pi}_s^{m_0,\text{BR}})_{s\in[0,t], m_0 \in \mathcal{M}}$, and we define in the continuous-time setting:

$$\bar{\mathcal{E}}(m_0, \bar{\boldsymbol{\pi}}_t^{m_0}) = \max_{\boldsymbol{\pi}'} J(m_0, \boldsymbol{\pi}'; \bar{\boldsymbol{\mu}}_t^{m_0}) - \frac{1}{t}\int_{s=0}^{t} J(m_0, \boldsymbol{\pi}_t^{m_0,\text{BR}}; \bar{\boldsymbol{\mu}}_t^{m_0}).$$

Theorem E.1 (Theorem 7.2 restated)**.** *Assume the reward is separable, the MFG is monotone, and the transition is independent of the population. Then, for every* $m_0 \in \mathcal{M}$, $\bar{\mathcal{E}}(\bar{\boldsymbol{\pi}}_t) \in O(1/t)$.

Proof. We follow the proof strategy of Perrin, Perolat, et al. (2020), adapted to our setting. To alleviate the notation, we denote $\langle f, g\rangle_{\mathcal{A}} = \sum_{a\in\mathcal{A}} f(a)g(a)$ for two functions f, g defined on $\mathcal{A}$, and similarly for $\langle \cdot, \cdot\rangle_{\mathcal{X}}$. We also denote: $r^{\pi}(x,\mu) = \langle \pi(\cdot|x), r(x,\cdot,\mu)\rangle_{\mathcal{A}}$.

We first note that, by the structure of the reward function given in (E.10),

$$\nabla_\mu r^{\boldsymbol{\pi}_{t,n}^{m_0,\text{BR}}}(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0}) = \nabla_\mu r_M(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0}) \text{ and } \nabla_\mu r^{\bar{\boldsymbol{\pi}}_{t,n}^{m_0}}(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0}) = \nabla_\mu r_M(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0}).$$

Moreover, using (E.9) and (E.7) respectively, we have, for every $x \in \mathcal{X}$,

$$-\langle \frac{d}{dt}\bar{\boldsymbol{\pi}}_{t,n}^{m_0}(.|x), r(x,.,\bar{\boldsymbol{\mu}}_{t,n}^{m_0})\rangle_{\mathcal{A}}\, \bar{\boldsymbol{\mu}}_{t,n}^{m_0}(x) = -\frac{1}{t} r^{\boldsymbol{\pi}_{t,n}^{m_0,\text{BR}}}(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0})\boldsymbol{\mu}_{t,n}^{m_0,\text{BR}}(x) + \frac{1}{t} r^{\bar{\boldsymbol{\pi}}_{t,n}^{m_0}}(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0})\boldsymbol{\mu}_{t,n}^{m_0,\text{BR}}(x),$$

$$-r^{\bar{\boldsymbol{\pi}}_{t,n}^{m_0}}(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0})\frac{d}{dt}\bar{\boldsymbol{\mu}}_{t,n}^{m_0}(x) = \frac{1}{t} r^{\bar{\boldsymbol{\pi}}_{t,n}^{m_0}}(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0})\bar{\boldsymbol{\mu}}_{t,n}^{m_0}(x) - \frac{1}{t} r^{\bar{\boldsymbol{\pi}}_{t,n}^{m_0}}(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0})\boldsymbol{\mu}_{t,n}^{m_0,\text{BR}}(x).$$

Using the definition of exploitability together with the above remarks, we deduce:

$$\frac{d}{dt}\bar{\mathcal{E}}(m_0, \bar{\boldsymbol{\pi}}_t^{m_0}) = \frac{d}{dt}\left[\max_{\boldsymbol{\pi}'} J(m_0, \boldsymbol{\pi}'; \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_t^{m_0}, m_0}) - J(m_0, \bar{\boldsymbol{\pi}}_t^{m_0}; \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_t^{m_0}, m_0})\right]$$

$$
\begin{aligned}
&= \sum_{n=0}^{+\infty} \gamma^n \sum_{x\in\mathcal{X}} \Big[\langle \nabla_\mu r^{\boldsymbol{\pi}_{t,n}^{m_0,\mathtt{BR}}}(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0}), \frac{d}{dt}\bar{\boldsymbol{\mu}}_{t,n}^{m_0} \rangle_{\mathcal{X}} \, \boldsymbol{\mu}_{t,n}^{m_0,\mathtt{BR}}(x) \\
&\qquad - \langle \nabla_\mu r^{\bar{\boldsymbol{\pi}}_{t,n}^{m_0}}(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0}), \frac{d}{dt}\bar{\boldsymbol{\mu}}_{t,n}^{m_0} \rangle_{\mathcal{X}} \, \bar{\boldsymbol{\mu}}_{t,n}^{m_0}(x) \\
&\qquad - \langle \frac{d}{dt}\bar{\boldsymbol{\pi}}_{t,n}^{m_0}(\cdot|x), r(x,\cdot,\bar{\boldsymbol{\mu}}_{t,n}^{m_0}) \rangle_{\mathcal{A}} \, \bar{\boldsymbol{\mu}}_{t,n}^{m_0}(x) - r^{\bar{\boldsymbol{\pi}}_{t,n}^{m_0}}(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0}) \frac{d}{dt}\bar{\boldsymbol{\mu}}_{t,n}^{m_0}(x) \Big] \\
&= \sum_{n=0}^{+\infty} \gamma^n \sum_{x\in\mathcal{X}} \Big[t\langle \nabla_\mu r_M(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0}), \frac{d}{dt}\bar{\boldsymbol{\mu}}_{t,n}^{m_0} \rangle_{\mathcal{X}} \, \frac{1}{t}\Big(\boldsymbol{\mu}_{t,n}^{m_0,\mathtt{BR}}(x) - \bar{\boldsymbol{\mu}}_{t,n}^{m_0}(x) \Big) \Big] \\
&\quad + \sum_{n=0}^{+\infty} \gamma^n \sum_{x\in\mathcal{X}} \Big[\frac{1}{t} r^{\bar{\boldsymbol{\pi}}_{t,n}^{m_0}}(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0}) \bar{\boldsymbol{\mu}}_{t,n}^{m_0}(x) - \frac{1}{t} r^{\boldsymbol{\pi}_{t,n}^{m_0,\mathtt{BR}}}(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0}) \boldsymbol{\mu}_{t,n}^{m_0,\mathtt{BR}}(x) \Big] \\
&= -\frac{1}{t}\bar{\mathcal{E}}(m_0, \bar{\boldsymbol{\pi}}_t^{m_0}) + \sum_{n=0}^{+\infty} \gamma^n \sum_{x\in\mathcal{X}} \Big[t\langle \nabla_\mu r_M(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0}), \frac{d}{dt}\bar{\boldsymbol{\mu}}_{t,n}^{m_0} \rangle_{\mathcal{X}} \, \frac{d}{dt}\bar{\boldsymbol{\mu}}_{t,n}^{m_0}(x) \Big],
\end{aligned}
$$

where the last term is non-positive. Indeed, the monotonicity condition (E.11) implies that, for all $\tau \geq 0$, we have:

$$
\sum_{x\in\mathcal{X}} (\bar{\boldsymbol{\mu}}_{t,n}^{m_0}(x) - \bar{\boldsymbol{\mu}}_{t+\tau,n}^{m_0}(x))(r_M(x, \bar{\boldsymbol{\mu}}_{t,n}^{m_0}) - r_M(x, \bar{\boldsymbol{\mu}}_{t+\tau,n}^{m_0})) \leq 0.
$$

The result follows after dividing by τ^2 and letting τ tend to 0.

□

Appendix F

Complements on Chapter 8

F.1 Algorithms in the exact case

In this section we present algorithms to compute MFG equilibria when the model is fully known.

F.1.1 Subroutines

The distribution computation for a given policy π is described in Algorithm F.1 using forward (in time) iterations. The evaluation of the state-action value function for a *given policy* π and mean field flow μ is described in Algorithm F.2. The computation of the optimal value function $Q^{*,\mu}$ for a given μ is described in Algorithm F.3. A best response against μ can be obtained by running this algorithm and then taking an optimizer of $Q_n^{*,\mu}(x,\cdot)$ for each n, x.

F.1.2 Main algorithms with fully known model

Banach-Picard Fixed point iterations are presented in Algorithm F.4. Fictitious Play is described in Algorithm F.5. Policy iteration (for MFG) is presented in Algorithm F.6. Online Mirror Descent is described in Algorithm F.7.

Algorithm F.1: Forward update for the distribution

input : **Parameter**: policy $\pi = (\pi_n)_{n=0,\dots,N_T}$
$\mu = (\mu_n)_{n=0,\dots,N_T} = \mu^\pi = (\mu_n^\pi)_{n=0,\dots,N_T}$ Let $\mu_0 = m_0$;
for $n = 1 \dots, N_T$: **do**
  $\mu_n^\pi = P_{n-1}^{\mu_{n-1}^\pi, \pi_{n-1}} \mu_{n-1}^\pi$
output: Mean field flow $\mu = (\mu_n)_{n=0,\dots,N_T}$

Algorithm F.2: Backward induction for the value function evaluation

input :Parameters: policy $\pi = (\pi_n)_{n=0,\dots,N_T}$, mean field flow $\mu = (\mu_n)_{n=0,\dots,N_T}$

Let $Q^\pi_{N_T}(x,a) = r_{N_T}(x,a,\mu_{N_T})$;

for $n = N_T - 1, \dots, 0$ **do**

$$Q^\pi_n(x,a) = r_n(x,a,\mu_n) + \mathbb{E}_{x' \sim p_n(\cdot|x,a,\mu_n), a' \sim \pi_n(\cdot|x')}[Q^\pi_{n+1}(x',a')]$$

where the expectation is computed in an exact way using the knowledge of the transition:

$$\mathbb{E}[Q^\pi_{n+1}(x',a')] = \sum_{x'} p_n(x'|x,a,\mu_n) \sum_{a'} \pi_{n+1}(a'|x') Q^\pi_{n+1}(x',a')$$

output: Return state-action value function $Q^\pi = (Q^\pi_n)_{n=0,\dots,N_T}$

Algorithm F.3: Backward induction for the optimal value function

input :Parameters: mean field flow $\mu = (\mu_n)_{n=0,\dots,N_T}$

Let $Q^*_{N_T}(x,a) = r_{N_T}(x,a,\mu_{N_T})$;

for $n = N_T - 1, \dots, 0$ **do**

$$Q^*_n(x,a) = r_n(x,a,\mu_n) + \underbrace{\mathbb{E}_{x' \sim p_n(\cdot|x,a,\mu_n)}[\max_{a' \in \mathcal{A}} Q^*_{n+1}(x',a')]}_{= \sum_{x'} p_n(x'|x,a,\mu_n) \max_{a'} Q^*_{n+1}(x',a')}$$

where the expectation is computed in an exact way using the knowledge of the transition p_n

output: Optimal state-action value function $Q^* = (Q^*_n)_{n=0,\dots,N_T}$

Algorithm F.4: Banach-Picard (BP) fixed point

input : Number of iterations K; optional softmax temperature η

Initialize π^0;

for $k = 0, \dots, K$ **do**

Forward update: Compute $\mu^k = \mu^{\pi^{k-1}}$, *e.g.* using Algorithm F.1 with $\pi = \pi^{k-1}$;

Best response computation: Compute a BR π^k against μ^k, *e.g.* by computing Q^{*,μ^k} using Algorithm F.3 with $\mu = \mu^k$ and then taking $\pi^k_n(\cdot|x)$ as a(ny) distribution over $\arg\max Q^{*,\bar{\mu}^k}_n(x,\cdot)$ for every n, x ;

alternatively, compute a soft version: $\pi^k_n(\cdot|x) = \mathrm{softmax}(Q^{*,\mu^k}_n(x,\cdot)/\eta)$;

output: $\mu^K = (\mu^K_n)_{n=0,\dots,N_T}$ and policy $\pi^K = (\pi^K_n)_{n=0,\dots,N_T}$

F.2 Deep RL Algorithms

We now present details on the deep RL algorithms used or developed in this paper. In this work, we focus on the use of deep RL for policy computation from the point of view of a representative

Algorithm F.5: Fictitious Play (FP)

input : Number of iterations K

Initialize π^0;

for $k = 0, \dots, K$ **do**

Forward update: Compute $\mu^k = \mu^{\pi^{k-1}}$, *e.g.* using Algorithm F.1 with $\pi = \pi^{k-1}$;

Average distribution update: Compute $\bar{\mu}^k$ as the average of $(\mu^0, \dots, \mu^{\pi^k})$:

$$\bar{\mu}_n^k(x) = \frac{1}{k}\sum_{i=1}^{k} \mu_n^i(x) = \frac{k-1}{k}\bar{\mu}_n^{k-1}(x) + \frac{1}{k}\mu_n^k(x)$$

Best response computation: Compute a BR π^k against $\bar{\mu}^k$, *e.g.* by computing $Q^{*,\bar{\mu}^k}$ using Algorithm F.3 and then taking $\pi_n^k(\cdot|x)$ as a(ny) distribution over $\arg\max Q_n^{*,\bar{\mu}^k}(x, \cdot)$ for every n, x;

output : $\bar{\mu}^K = (\bar{\mu}_n^K)_{n=0,\dots,N_T}$ and policy $\bar{\pi}^K = (\bar{\pi}_n^K)_{n=0,\dots,N_T}$ generating this mean field flow

Algorithm F.6: Policy Iteration (PI)

input : **Parameters:** softmax temperature η; number of iterations K

Initialize the sequence of tables $(\bar{q}_n^0)_{n=0,\dots,N_T}$, *e.g.* with $\bar{q}_n^0(x, a) = 0$ for all n, x, a;

Let the projected policy be: $\pi_n^0(a|x) = \mathrm{softmax}(\bar{q}_n^0(x, \cdot)/\eta)(a)$ for all n, x, a;

for $k = 1, \dots, K$ **do**

Forward Update: Compute $\mu^k = \mu^{\pi^{k-1}}$, *e.g.* using Algorithm F.1 with $\pi = \pi^{k-1}$;

Backward Update: Compute $Q^k = Q^{\pi^{k-1},\mu^k}$, *e.g.* using backward induction (c.f. Algorithm F.2) with $\pi = \pi^{k-1}$ and $\mu = \mu^k$ and then let $\pi_n^k(\cdot|x)$ be a(ny) distribution over $\arg\max Q_n^{*,\bar{\mu}^k}(x, \cdot)$ for every n, x; alternatively, compute a soft version: $\pi_n^k(\cdot|x) = \mathrm{softmax}(Q_n^k(x, \cdot)/\eta)$;

output : Q^K, π^K

agent. We assume that this agent has access to an oracle that can return $r_n(x_n, a_n, \mu_n)$ and a sample of $p_n(\cdot|x_n, a_n, \mu_n)$ when the agent follows is in state x_n and uses action a_n. In fact, the collection of samples is split into episodes. At each episode, the agent start from some x_0 sampled from m_0. Then it evolves by following the current policy, and the transitions are added to a replay buffer.

In order to focus on the errors due to the deep RL algorithm, we assume that the distribution is updated in an exact way following Algorithm F.1.

For the Deep RL part, the approaches can be summarized as follows:

Algorithm F.7: Online Mirror Descent (OMD)

input : **Parameters:** inverse learning rate parameter τ; number of iterations K

Initialize the sequence of tables $(\bar{q}_n^0)_{n=0,\dots,N_T}$, *e.g.* with $\bar{q}_n^0(x,a) = 0$ for all n, x, a;

Let the projected policy be: $\pi_n^0(a|x) = \mathrm{softmax}(\bar{q}_n^0(x,\cdot))(a)$ for all n, x, a;

for $k = 1, \dots, K$ **do**

  Forward Update: Compute $\mu^k = \mu^{\pi^{k-1}}$, *e.g.* using Algorithm F.1 with $\pi = \pi^{k-1}$;

  Backward Update: Compute $Q^k = Q^{\pi^{k-1},\mu^k}$, *e.g.* using backward induction (c.f. Algorithm F.2) with $\pi = \pi^{k-1}$ and $\mu = \mu^k$;

  Update the regularized Q-function and the projected policy: for all n, x, a,

$$\bar{q}_n^k(x,a) = \bar{q}_n^{k-1}(x,a) + \frac{1}{\tau} Q_n^k(x,a)$$
$$\pi_n^k(a|x) = \mathrm{softmax}(\bar{q}_n^k(x,\cdot))(a)$$

output: $\bar{q}^K, \pi^K$

- Algorithm F.2: Intuitively, he updates are replaced by stochastic gradient steps so as to minimize the following loss with respect to θ:

$$\left| Q_\theta((n,x_n),a_n) - r_n(x_n,a_n,\mu_n) - \hat{\mathbb{E}}_{x'_{n+1}\sim p_n(\cdot|x_n,a_n,\mu_n), a'_{n+1}\sim\pi_{n+1}}[q((n+1,x'_{n+1}),a'_{n+1})] \right|^2, \tag{F.1}$$

  where $\hat{\mathbb{E}}$ denotes an empirical expectation over a finite number of samples picked from the replay buffer and q is replaced by a target network $Q_{\theta'}$ whose parameters are frozen while training θ and that are updated less frequently than θ. Combined with Policy Iteration (Algorithm F.6), this leads to the algorithm referred to as **Deep Policy Iteration (D-PI)**.

- Algorithm F.3: We can proceed similarly, except that the target becomes:

$$r_n(x_n,a_n,\mu_n) + \hat{\mathbb{E}}_{x'_{n+1}\sim p_n(\cdot|x_n,a_n,\mu_n)}[\max_{a'} q((n+1,x'_{n+1}),a')].$$

  In fact, in our implementation we use DQN (Mnih et al., 2013) as a subroutine for the BR computation. Combined with Banach-Picard iterations (Algorithm F.4), this leads directly to the algorithm referred to as **Deep Banach-Picard (D-BP)**.

- To obtain **Deep Average-network Fictitious Play (D-AFP)** (Algorithm 8.1), at each iteration, the best response against the current distribution is learnt using DQN (Mnih et al., 2013). This policy is used to generate trajectories, whose state-action samples are added to a reservoir buffer $\mathcal{M}_{SL}$. This buffer stores state-actions generated using past policies from previous iterations. Then, an auxiliary neural network for the logits representing the average policy is trained using supervised learning using $\mathcal{M}_{SL}$: stochastic gradient is

used to find $\bar{\theta}$ minimizing approximately the loss:

$$\mathcal{L}(\bar{\theta}) = \mathbb{E}_{(s,a)\sim\mathcal{M}_{SL}}\left[-\log\left(\bar{\pi}_{\bar{\theta}}(a|s)\right)\right]$$

In our implementation, for the representation of the average policy, we use a neural network $\bar{\ell}_\omega$ with parameters ω for the logits, and then we compute the policy as: $\bar{\pi} = \mathrm{softmax}(\bar{\ell}_\omega)$. This step is reminiscent of Neural Fictitious Self Play (NFSP) introduced in Heinrich and Silver (2016), which was used to solve Leduc poker and Limit Texas Hold'em poker. However, the overall algorithm is different. Indeed, in NFSP as described in Algorithm 1 of Heinrich and Silver (2016), the neural network for the average policy and the neural network for the Q-function are both updated at each iteration. We reuse the idea of having a buffer of past actions but in our case, between each update of the average policy network, we do two operations: first, we update the mean field (sequence of population distributions) and second, we learn a best response against this mean field.

- To obtain **Deep Munchausen Online Mirror Descent (D-MOMD)** (Algorithm 8.2), we can simply modify the target in (F.1) as follows:

$$\Big|Q_\theta((n,x_n),a_n) - r_n(x_n,a_n,\mu_n) - \tau\ln\pi_{n-1}(a_n|x_n) \\ - \hat{\mathbb{E}}_{x'_{n+1}\sim p_n(\cdot|x_n,a_n,\mu_n),}\sum_{a'}\pi_n(a'|x_n)[q((n+1,x'_{n+1}),a'_{n+1}) - \tau\ln\pi_n(a'|x'_{n+1})]\Big|^2,$$

where $q = Q_{\theta'}$ is a target network whose parameters θ' are frozen while training θ. This is similar to equation (7) of Vieillard, Pietquin, and Geist (2020) which introduced the Munchausen RL method. However, in our case the distribution μ_n appears in the reward r_n and in the transition leading to the new state x'_{n+1}. So while Vieillard, Pietquin, and Geist (2020) simply repeatedly update the Q-network, we intertwine the udpates of the cumulative Q-network with the updates of the population distribution.

F.3 Details on the link between MOMD and regularized MDPs

Consider regularizing the MFG with only entropy, that is

$$J(\pi,\mu) = \mathbb{E}_\pi\left[\sum_{n=0}^{N_T}(r_n(s_n,a_n,\mu_n) - (1-\alpha)\tau\ln\pi_n(a_n|s_n))\right]. \tag{F.2}$$

Notice that we choose $(1-\alpha)\tau$ here because it will simplify later, but it would work with any temperature (or learning rate from the OMD perspective).

Now, let's solve this MFG with OMD with learning rate $(\alpha\tau)^{-1}$, adopting the KL perspective. The corresponding algorithm is:

$$\pi_n^{k+1} \in \arg\max \langle \pi_n, q_n^k \rangle - \alpha\tau \mathrm{KL}(\pi_n || \pi_n^k) + (1-\alpha)\tau \mathcal{H}(\pi_n) \tag{F.3}$$

$$\begin{cases} q_{N_T}^{k+1} = r_{N_T}^{k+1} \\ q_n^{k+1} = r_n^{k+1} + \gamma P \langle \pi_{n+1}^{k+1}, q_{n+1}^{k+1} - (1-\alpha)\tau \ln \pi_{n+1}^{k+1} \rangle \end{cases} \tag{F.4}$$

Next, using $Q_n^k = q_n^k + \alpha\tau \ln \pi_n^k$, we can rewrite the evaluation part as:

$$\pi_n^{k+1} = \mathrm{softmax}\left(\frac{Q_n^k}{\tau}\right) \tag{F.5}$$

$$\begin{cases} Q_{N_T}^{k+1} = r_{N_T}^{k+1} + \alpha\tau \ln \pi_{N_T}^{k+1} \\ Q_n^{k+1} = r_n^{k+1} + \alpha\tau \ln \pi_n^{k+1} + \gamma P \langle \pi_{n+1}^{k+1}, Q_{n+1}^{k+1} - \tau \ln \pi_{n+1}^{k+1} \rangle \end{cases} \tag{F.6}$$

We remark that it corresponds to the "scaled" version of Munchausen OMD, meaning that it amounts to solving the MFG regularized with $(1-\alpha)\tau\mathcal{H}(\pi)$ with OMD. We retrieve with $\alpha = 1$ the unscaled version of Munchausen OMD, addressing the unregularized MFG. It also makes a connection with the Boltzmannn iteration method of Cui and Koeppl (2021), in which a similar penalization except that the penalty involves a fixed policy instead of using the current policy. Their prior descent method, in which the reference policy is updated from time to time, can thus be viewed as a first step towards Munchausen OMD.

F.4 Hyperparameters sweeps

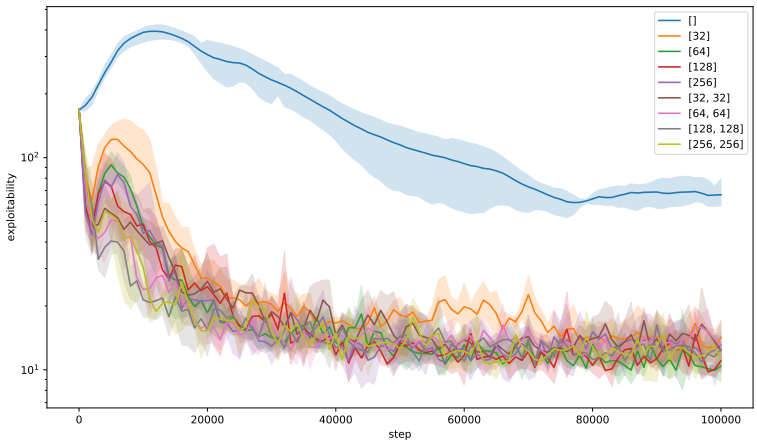


Figure F.1 – D-MOMD, Exploration game with four rooms: Sweep over the network size. The neural network architecture is feedforward fully connected with one or two hidden layers, except for the curve with label `[]`, which refers to a linear function. This illustrates in particular that the policy can not be well approximated using only linear functions, hence the need for non-linear approximations, which raises the difficulty of averaging or summing such approximations (here neural networks).

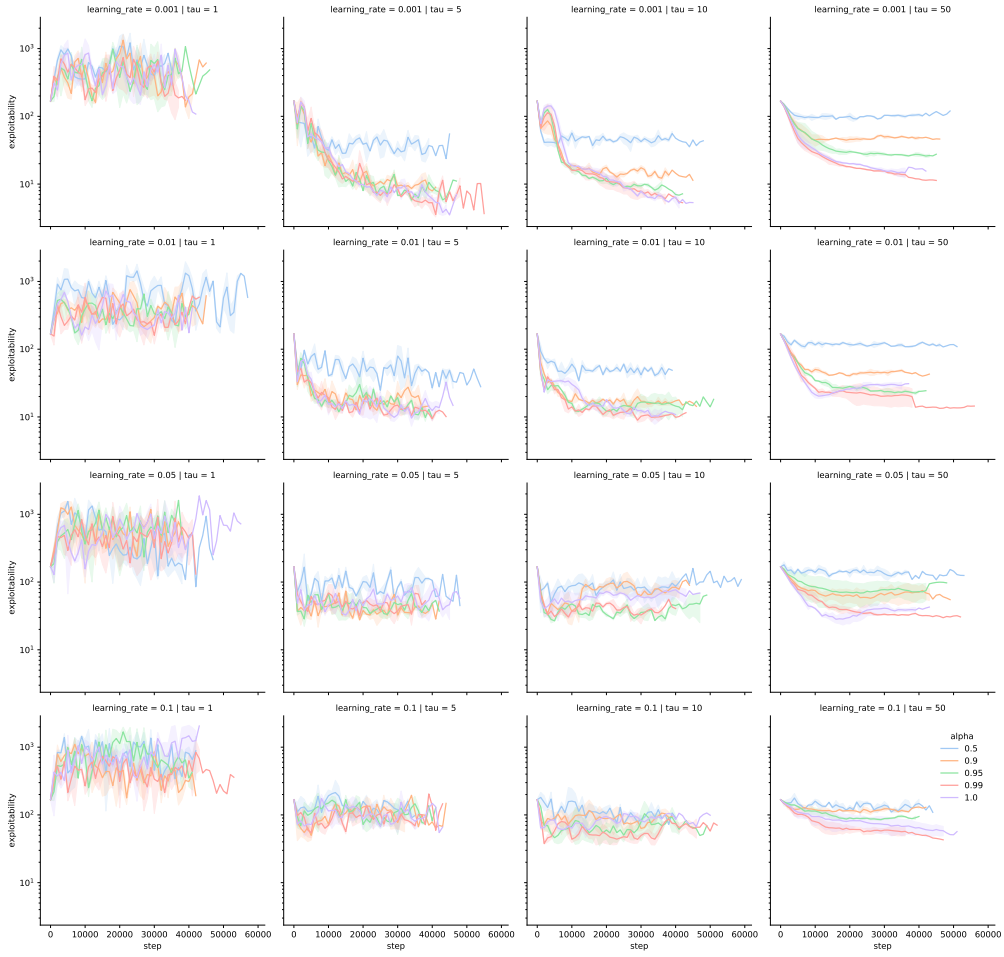


Figure F.2 – D-MOMD, Exploration game with four rooms: Sweep over τ, α and learning rate.

List of Figures

1.1 The mean field approximation: atomic players are replaced by a distribution of players over the state space, here a two dimensional domain. 4

1.2 Reading order: **Left** All identical and anonymous players allow to elect a representative player; **Right** The representative player adapts its policy to the distribution of players μ. In return, the distribution is influenced by all the players using the policy of the representative player. 4

1.3 Figure from Cui, Tahir, et al. (2022). Scheme of the links between multi-agent systems, mean field games, mean field control and there solutions. 5

1.4 Mean field Reinforcement Learning. 9

2.1 Reinforcement learning environment: classical single-agent setup. Here, at iteration n, the current state of the MDP is x_n, the action taken by the agent is a_n, the new state is $x_{n+1} \sim p(\cdot|x_n, a_n)$ and the reward is $r_n = r(x_n, a_n)$. The new state x_{n+1} is observed by the agent and is also used for the next step of the environment's evolution. 25

3.1 Environment for MFGs: Here, the current state of the MDP is the representative agent's state x_n and the population distribution μ_n, the action taken by the agent is a_n, the new state is $x_{n+1} \sim p(\cdot|x_n, a_n, \mu_n)$ and the reward is $r_n = r(x_n, a_n, \mu_n)$. The new state x_{n+1} is observed by the agent and is also used for the next step of the environment's evolution along with μ_n. 58

3.2 Environment for MFC and MFMDP. 62

3.3 Reading order: **(a)** the considered environment initial state in yellow, walls in white); **(b)** the log-density of a uniform policy (to illustrate entropy maximization). 67

3.4 **Entropy maximization.** From left to right: Terminal distribution induced by (**a**) Fixed Point; (**b**) Fictitious Play; (**c**) OMD; (**d**) Damped Fixed Point; (**e**) Softmax Fixed Point; (**f**) Softmax Fictitious Play; (**g**) Boltzmann Policy Iteration; (**h**) Exploitability curves for these methods. 69

4.1 Evolution of the distribution in the linear quadratic MFG with finite horizon. . 80

4.2 The beach bar process. 81

4.3 Beach bar process in finite horizon: (a, b) evolution of the distribution, (c) exploitability. 82

4.4 Linear Quadratic with Common Noise. 84

4.5 First Common Noise setting, the bar has a probability 0.5 of closing at time step 15. 84

4.6 2D crowd modeling example. 85

5.1 5 Garnet sampled with param $n_x = 20000$, $n_a = 10$, $t = 2000$, $s_f = 10$ 96

5.2 Population distribution at consecutive dates (three first figures on the left). Each plot of a subfigure is a different floor, the bottom floor is the bottom-right plot, the top floor is the top-left plot. The figure on the right displays the exploitability of: Fictitious Play (red, $\alpha = 10^{-5}$), Fictitious Play damped (green, $\alpha = 10^{-3}$) and OMD (blue, $\alpha = 10^{-4}$). 98

5.3 Crowd position at different consecutive dates when the point of interest is randomly shifted to the right by a common noise. The fourth graph is displaying the exploitability of MD. 98

5.4 4-population chasing. Right figure : Fictitious Play (red, $\alpha = 10^{-3}$), Fictitious Play damped (green, $\alpha = 10^{-5}$) and OMD (blue, $\alpha = 10^{-5}$). From left to right, 3 picture chowing the distribution evolving through time and a fourth one displaying the exploitability. 100

5.5 Building environment. 100

5.6 $\bar{r}^{i,j}$ for three-population. 102

6.1 Multi-group flocking with noise and $\beta = 100$. 118

6.2 Flocking with noise and many obstacles. 119

7.1 Neural network architecture of the Q-function for the 2D beach bar experience. 130

7.2 Exploration 1D: Performance matrices when the training set is made of Gaussian distributions. From left to right: (**a**) Log of Wasserstein distances to the exact solution (time average); (**b**) Log of exploitabilities. The x-axis is the initial distribution index: on the left (resp. right) of the vertical red line are the training (resp. testing) distributions. 134

7.3 Beach bar 2D: Environment. From left to right: (**a**) an initial distribution $m_0 \in \mathcal{M}$; (**b**) MF state at equilibrium (specialized policy); (**c**) MF state at equilibrium (learned Master policy); (**d**) MF state at equilibrium (specialized policy of another initial distribution). Note that the scale is very different for the last figure. 134

7.4 Beach bar 2D: Performance matrices with Gaussian distributions. From left to right: (**a**) Log of Wasserstein distances to the exact solution (average over time steps); (**b**) Log of exploitabilities. Each row is a policy. Top part: a row j gives the performance of the equilibrium policy for the j-th initial distribution. Bottom part: policies given in the text). 135

8.1 Left: exploitability. Right: evolution of the distribution obtained by the policy learnt with D-MOMD. 149

8.2 Top: Evolution of the distribution generated by the policy learnt by D-MOMD. Bottom left: Exploitability of different algorithms on the Linear Quadratic environment. Bottom right: Wasserstein distance between the solution learnt by D-MOMD and the benchmark one, over its iterations. 150

8.3 Top: exploitability. Bottom: evolution of the distribution obtained by the policy learnt with D-MOMD. 151

8.4 Maze example. Top: exploitability. Bottom: evolution of the distribution obtained by the policy learnt with D-MOMD. 152

8.5 Multi-population chasing example. Top: Exploitability. Bottom: evolution of the distributions for the three populations. 153

B.1 2$^{\text{nd}}$ common noise setting, the bar has a probability $p = 0.5$ to close at every time step before $\frac{N}{2}$. 172

B.2 Final distributions and exploitability in the γ-discounted case 175

C.1 5 garnet sampled with param $n_x = 2000$, $n_a = 20$, $N_T = 2000$, $s_f = 10$ 190

C.2 5 garnet sampled with param $n_x = 20000$, $n_a = 10$, $N_T = 2000$, $s_f = 10$ 190

C.3 5 garnet sampled with param $n_x = 2000$, $n_a = 10$, $N_T = 2000$, $s_f = 10$ 190

C.4 Garnet Experiments performances . . . 190
C.5 Building Experiment performances . . . 191
C.6 After a few timesteps . . . 192
C.7 Intermediate time . . . 192
C.8 Almost at arrival timestep . . . 192
C.9 Building Experiment solution (ground floor on the upper left corner) . . . 192
C.10 Torus topology and corner initialization . . . 195
C.11 Square topology and corner initialization . . . 195
C.12 Donut topology and corner initialization . . . 195
C.13 Torus topology and random initialization . . . 195
C.14 Square topology and random initialization . . . 195
C.15 Donut topology and random initialization . . . 195
C.16 Multi-population experiments, performances with different topologies . . . 195

D.1 Flocking with an intuitive example . . . 198
D.2 Flocking for a simple example in six dimensions. . . . 199
D.3 Flocking in 4D with one obstacle. . . . 200
D.4 Flocking in 6D with one obstacle. . . . 200
D.5 Flocking in 4D with many obstacles. . . . 201
D.6 Visual rendering with Unity . . . 202

E.1 Pure exploration 1D: Training set . . . 204
E.2 Pure exploration 1D: Testing set . . . 204
E.3 Beach bar 2D: Training set . . . 205
E.4 Beach bar 2D: Testing set . . . 205

F.1 D-MOMD, Exploration game with four rooms: Sweep over the network size. The neural network architecture is feedforward fully connected with one or two hidden layers, except for the curve with label `[]`, which refers to a linear function. This illustrates in particular that the policy can not be well approximated using only linear functions, hence the need for non-linear approximations, which raises the difficulty of averaging or summing such approximations (here neural networks). 219

F.2 D-MOMD, Exploration game with four rooms: Sweep over τ, α and learning rate. 220

List of Algorithms

4.1 Fictitious Play in Mean Field Games . . . 78

5.1 Online Mirror Descent (OMD) . . . 94

6.1 Flock'n RL . . . 114

7.1 Master Fictitious Play . . . 128

8.1 D-AFP . . . 143

8.2 D-MOMD . . . 147

B.1 Q-Learning in Mean Field Games . . . 176

B.2 Empirical Density Estimation . . . 176

B.3 Backward Induction in Mean Field Games . . . 176

B.4 Density Estimation . . . 176

E.1 DQN for a population-dependent Best Response . . . 206

F.1 Forward update for the distribution . . . 213

F.2 Backward induction for the value function evaluation . . . 214

F.3 Backward induction for the optimal value function . . . 214

F.4 Banach-Picard (BP) fixed point . . . 214

F.5 Fictitious Play (FP) . . . 215

F.6 Policy Iteration (PI) . . . 215

F.7 Online Mirror Descent (OMD) . . . 216

List of Tables

1.1 Notations for various number of agents 7

5.1 Number of states, action-states pairs & RAM memory required for the experiments. $|\mathcal{X}|$ = positions $\times$ timesteps $\times$ common noise $\times$ number of populations (KB stands for Kilo Byte, G stands for Giga and T stands for Tera). 95

List of References

Achdou, Y., M. Bardi, and M. Cirant (2017). Mean field games models of segregation. *Mathematical Models and Methods in Applied Sciences* 27.1, pp. 75–113.

Achdou, Y., F. Buera, J.-M. Lasry, P.-L. Lions, and B. Moll (2014). PDE Models in Macroeconomics. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*.

Achdou, Y., F. Camilli, and I. Capuzzo-Dolcetta (2012). Mean field games: numerical methods for the planning problem. *SIAM Journal on Control and Optimization* 50.1.

Achdou, Y. and I. Capuzzo-Dolcetta (2010). Mean field games: numerical methods. *SIAM Journal on Numerical Analysis* 48.3.

Achdou, Y., P. Cardaliaguet, F. Delarue, A. Porretta, and F. Santambrogio (2020). *Mean Field Games: Cetraro, Italy 2019*. Vol. 2281. Springer Nature.

Achdou, Y., P.-N. Giraud, J.-M. Lasry, and P.-L. Lions (2016). A long-term mathematical model for mining industries. *Applied Mathematics & Optimization* 74.3.

Achdou, Y., J. Han, J.-M. Lasry, P.-L. Lions, and B. Moll (2017). *Income and Wealth Distribution in Macroeconomics: A Continuous-Time Approach*. Tech. rep. *National Bureau of Economic Research*.

Achdou, Y. and J.-M. Lasry (2019). Mean field games for modeling crowd motion. In *Contributions to partial differential equations and applications*. Springer, pp. 17–42.

Achdou, Y. and M. Laurière (2015). On the system of partial differential equations arising in mean field type control. *Discrete and Continuous Dynamical Systems. Series A* 35.9.

— (2016). Mean field type control with congestion. *Applied Mathematics & Optimization* 73.3.

— (2020). Mean field games and applications: Numerical aspects. *Mean field games*, pp. 249–307.

Achdou, Y., P. Mannucci, C. Marchi, and N. Tchou (2020). Deterministic mean field games with control on the acceleration. *NoDEA*.

Alasseur, C., I. B. Taher, and A. Matoussi (2020). An extended mean field game for storage in smart grids. *Journal of Optimization Theory and Applications* 184.2.

Almulla, N., R. Ferreira, and D. Gomes (2017). Two numerical approaches to stationary mean-field games. *Dynamic Games and Applications* 7.4.

Anahtarci, B., C. D. Kariksiz, and N. Saldi (2021). Learning in Discrete-time Average-cost Mean-field Games. In *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, pp. 3048–3053.

Anahtarcı, B., C. D. Karıksız, and N. Saldi (2019a). Fitted Q-learning in mean-field games. *arXiv preprint arXiv:1912.13309*.

— (2019b). Learning in Discounted-cost and Average-cost Mean-field Games. *arXiv preprint arXiv:1912.13309*.

— (2020a). Q-learning in regularized mean-field games. *arXiv preprint arXiv:2003.12151*.

— (2020b). Value iteration algorithm for mean-field games. *Systems & Control Letters* 143, p. 104744.

Angiuli, A., J.-P. Fouque, and M. Laurière (2021). Reinforcement learning for mean field games, with applications to economics. *To appear in Machine Learning And Data Sciences For Financial Markets* (*arXiv preprint arXiv:2106.13755*).

— (2022). Unified reinforcement Q-learning for mean field game and control problems. *Mathematics of Control, Signals, and Systems*, pp. 1–55.

Angiuli, A., C. V. Graves, H. Li, J.-F. Chassagneux, F. Delarue, and R. Carmona (2019). Cemracs 2017: numerical probabilistic approach to MFG. *ESAIM: Proceedings and Surveys* 65.

Anthony, T., Z. Tian, and D. Barber (2017). Thinking Fast and Slow with Deep Learning and Tree Search. In *Proceedings of NeurIPS*.

Al-Aradi, A., A. Correia, D. Naiff, G. Jardim, and Y. Saporito (2018). Solving nonlinear and high-dimensional partial differential equations via deep learning. *arXiv preprint arXiv:1811.08782*.

Archibald, T., K. McKinnon, and L. Thomas (1995). On the generation of markov decision processes. *Journal of the Operational Research Society* 46.3, pp. 354–361.

Arthur, W. B. (1994). Inductive reasoning and bounded rationality. *The American Economic Review* 84.2.

Arulkumaran, K., M. P. Deisenroth, M. Brundage, and A. A. Bharath (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* 34.6, pp. 26–38.

Aumann, R. J. (1964). Markets with a continuum of traders. *Econometrica: Journal of the Econometric Society*, pp. 39–50.

— (1987). Correlated Equilibrium as an Expression of Bayesian Rationality. In

Aumann, R. J. and L. S. Shapley (2015). *Values of non-atomic games*. Princeton University Press.

Aurell, A., R. Carmona, G. Dayanikli, and M. Laurière (2022). Optimal incentives to mitigate epidemics: a Stackelberg mean field game approach. *SIAM Journal on Control and Optimization* 60.2, S294–S322.

Aurell, A. and B. Djehiche (2018). Mean-field type modeling of nonlocal crowd aversion in pedestrian crowd dynamics. *SIAM Journal on Control and Optimization* 56.1, pp. 434–455.

— (2019). Modeling tagged pedestrian motion: A mean-field type game approach. *Transportation Research Part B: Methodological* 121.

Bagagiolo, F. and D. Bauso (2014). Mean-field games and dynamic demand management in power grids. *Dynamic Games and Applications* 4.2.

Bailo, R., M. Bongini, J. A. Carrillo, and D. Kalise (2018). Optimal consensus control of the Cucker-Smale model. *IFAC*.

Bakhtin, A., D. J. Wu, A. Lerer, J. Gray, A. P. Jacob, G. Farina, et al. (2022). *Mastering the Game of No-Press Diplomacy via Human-Regularized Reinforcement Learning and Planning*.

Balaguer, J., R. Koster, C. Summerfield, and A. Tacchetti (2022). *The Good Shepherd: An Oracle Agent for Mechanism Design*.

Bäuerle, N. (2021). Mean Field Markov Decision Processes. *arXiv preprint arXiv:2106.08755*.

Bauso, D., H. Tembine, and T. Basar (2016). Opinion dynamics in social networks through mean-field games. *SIAM Journal on Control and Optimization* 54.6.

Bauso, D., X. Zhang, and A. Papachristodoulou (2016). Density flow in dynamical networks via mean-field games. *IEEE Transactions on Automatic Control* 62.3, pp. 1342–1355.

Bellemare, M. G., Y. Naddaf, J. Veness, and M. Bowling (2013). The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research* 47, pp. 253–279.

Bellman, R. (1957). A Markovian decision process. *Journal of mathematics and mechanics*, pp. 679–684.

Bensoussan, A., J. Frehse, and P. Yam (2013). *Mean field games and mean field type control theory*. Vol. 101. Springer.

Bensoussan, A., J. Frehse, and S. C. P. Yam (2013). *Mean Field Games and Mean Field Type Control Theory*. Springer Briefs in Mathematics. Springer, New York.

— (2015). The Master equation in mean field theory. *Journal de Mathématiques Pures et Appliquées* 103.6, pp. 1441–1474.

Bensoussan, A., T. Huang, and M. Laurière (2018). Mean field control and mean field game models with several populations. *Minimax Theory and its Applications* 3.2, pp. 173–209.

Bensoussan, A., K. Sung, S. C. P. Yam, and S.-P. Yung (2016). Linear-quadratic mean field games. *Journal of Optimization Theory and Applications* 169.2.

Bertsekas, D. (2012). *Dynamic programming and optimal control*. Vol. 1. Athena scientific.

Bertsekas, D. and S. E. Shreve (1996). *Stochastic optimal control: the discrete-time case*. Vol. 5. Athena Scientific.

Blum, A. and Y. Mansour (2005). *From External to Internal Regret*.

Bonnans, J. F., P. Lavigne, and L. Pfeiffer (2021). Generalized conditional gradient and learning in potential mean field games. *arXiv preprint arXiv:2109.05785*.

Borkar, V. S. (2009). *Stochastic approximation: a dynamical systems viewpoint*. Vol. 48. Springer.

Bowling, M., N. Burch, M. Johanson, and O. Tammelin (2015). Heads-up limit hold'em poker is solved. *Science* 347.6218.

Bravo, M., D. S. Leslie, and P. Mertikopoulos (2018). Bandit learning in concave N-person games. *arXiv preprint arXiv:1810.01925*.

Briceño-Arias, L. M., D. Kalise, Z. Kobeissi, M. Laurière, Á. Mateos González, and F. J. Silva (2019). On the implementation of a primal-dual algorithm for second order time-dependent Mean Field Games with local couplings. *ESAIM: Proceedings* 65.

Briceño-Arias, L. M., D. Kalise, and F. J. Silva (2018). Proximal methods for stationary mean field games with local couplings. *SIAM Journal on Control and Optimization* 56.2.

Brockman, G., V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, et al. (2016). *OpenAI Gym*.

Brown, G. W. (1951). Iterative solution of games by fictitious play. *Activity analysis of production and allocation* 13.1, pp. 374–376.

Brown, N. and T. Sandholm (2017). Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* 360.6385.

— (2019). Superhuman AI for multiplayer poker. *Science* 365.6456.

Burch, N., M. Johanson, and M. Bowling (2014). Solving imperfect information games using decomposition. In *Proceedings of AAAI*.

Burger, M., M. Francesco, P. Markowich, and M.-T. Wolfram (Apr. 2013). Mean field games with nonlinear mobilities in pedestrian dynamics. *Discrete and Continuous Dynamical Systems - Series B* 19.

Cabannes, T., M. Lauriere, J. Perolat, R. Marinier, S. Girgin, S. Perrin, et al. (2021). Solving N-player dynamic routing games with congestion: a mean field approach. *Extended abstract at AAMAS 2022* (*long version: arXiv preprint arXiv:2110.11943*).

Cacace, Simone, Camilli, Fabio, and Goffi, Alessandro (2021). A policy iteration method for mean field games. *ESAIM: COCV* 27, p. 85.

Cai, Y., O. Candogan, C. Daskalakis, and C. Papadimitriou (2016). Zero-Sum Polymatrix Games: A Generalization of Minmax. *Mathematics of Operations Research* 41.2, pp. 648–655.

Camilli, F. and Q. Tang (2022). Rates of convergence for the policy iteration method for Mean Field Games systems. *Journal of Mathematical Analysis and Applications* 512.1, p. 126138.

Campbell, M., A. J. Hoane Jr, and F.-h. Hsu (2002). Deep blue. *Artificial intelligence* 134.1-2, pp. 57–83.

Campi, L. and M. Fischer (2020). Correlated equilibria and mean field games: a simple model. *arXiv preprint arXiv:2004.06185*.

Cao, H., X. Guo, and M. Laurière (2020). Connecting GANs, MFGs, and OT. *arXiv preprint arXiv:2002.04112*.

Caponigro, M., M. Fornasier, B. Piccoli, and E. Trélat (2013). Sparse stabilization and optimal control of the Cucker-Smale Model. *Mathematical Control and Related Fields*.

Cardaliaguet, P. (2012). Notes on mean field games. *P.-L. Lions' Lectures at Collège de France*.

Cardaliaguet, P., F. Delarue, J.-M. Lasry, and P. L. Lions (2019). *The master equation and the convergence problem in mean field games*. Ed. by P. U. Press. Vol. 381. AMS-201.

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

Cardaliaguet, P. and C.-A. Lehalle (2018). Mean field game of controls and an application to trade crowding. *Mathematics and Financial Economics* 12.3, pp. 335–363.

Cardaliaguet, P., A. Porretta, and D. Tonon (2016). A segregation problem in multi-population mean field games. In *International Symposium on Dynamic Games and Applications*. Springer.

Carlini, E. and F. J. Silva (2014). A fully discrete semi-Lagrangian scheme for a first order mean field game problem. *SIAM Journal on Numerical Analysis* 52.1.

— (2015). A semi-Lagrangian scheme for a degenerate second order mean field game system. *Discrete and Continuous Dynamical Systems* 35.9.

Carmona, R. (2020). Applications of mean field games in financial engineering and economic theory. *arXiv preprint arXiv:2012.05237*.

Carmona, R. and F. Delarue (2013). Mean field forward-backward stochastic differential equations. *Electronic Communications in Probability* 18, pp. 1–15.

— (2018a). *Probabilistic Theory of Mean Field Games with Applications I-II*. Springer.

— (2018b). *Probabilistic theory of mean field games with applications. I*. Vol. 83. Probability Theory and Stochastic Modelling. Mean field FBSDEs, control, and games. Springer, Cham.

— (2018c). *Probabilistic theory of mean field games with applications. II*. Vol. 84. Probability Theory and Stochastic Modelling. Mean field games with common noise and master equations. Springer, Cham.

Carmona, R., F. Delarue, and D. Lacker (2016a). Mean field games with common noise. *The Annals of Probability* 44.6, pp. 3740–3803.

— (2016b). Mean field games with common noise. *Annals of Probability* 44.6.

Carmona, R., J.-P. Fouque, and L.-H. Sun (2015a). Mean Field Games and systemic risk. *Communications in Mathematical Sciences* 13.4, pp. 911–933.

— (2015b). Mean field games and systemic risk. *Commun. Math. Sci.* 13.4.

Carmona, R., C. V. Graves, and Z. Tan (2019). Price of anarchy for mean field games. *ESAIM: Proceedings and Surveys* 65, pp. 349–383.

Carmona, R., K. Hamidouche, M. Laurière, and Z. Tan (2020). Policy optimization for linear-quadratic zero-sum mean-field type games. In *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, pp. 1038–1043.

— (2021). Linear-quadratic zero-sum mean-field type games: Optimality conditions and policy optimization. *Journal of Dynamics & Games* 8.4, p. 403.

Carmona, R. and M. Laurière (2019). Convergence Analysis of Machine Learning Algorithms for the Numerical Solution of Mean Field Control and Games: II - The Finite Horizon Case. *To appear in Annals of Applied Probability (preprint arXiv:1908.01613)*.

Carmona, R. and M. Laurière (2021). Convergence Analysis of Machine Learning Algorithms for the Numerical Solution of Mean Field Control and Games I: The Ergodic Case. *SIAM Journal on Numerical Analysis* 59.3, pp. 1455–1485.

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

— (2019b). Model-free mean-field reinforcement learning: mean-field MDP and mean-field Q-learning. *arXiv preprint arXiv:1910.12802*.

Cecchin, A. and G. Pelino (2019). Convergence, fluctuations and large deviations for finite state mean field games via the master equation. *Stochastic Processes and their Applications* 129.11, pp. 4510–4555.

Cesa-Bianchi, N. and G. Lugosi (2006). *Prediction, Learning, and Games*. Cambridge University Press.

Chan, P. and R. Sircar (2015). Bertrand and Cournot mean field games. *Applied Mathematics & Optimization* 71.3.

— (2017). Fracking, renewables, and mean field games. *SIAM Review* 59.3.

Chassagneux, J.-F., D. Crisan, and F. Delarue (2019). Numerical method for FBSDEs of McKean–Vlasov type. *The Annals of Applied Probability* 29.3, pp. 1640–1684.

Chen, M., Y. Li, E. Wang, Z. Yang, Z. Wang, and T. Zhao (2021). Pessimism Meets Invariance: Provably Efficient Offline Mean-Field Multi-Agent RL. *Advances in Neural Information Processing Systems* 34.

Chevalier, G., J. Le Ny, and R. Malhamé (2015). A micro-macro traffic model based on mean-field games. In *American Control Conference (ACC)*. IEEE.

Chotibut, T., F. Falniowski, M. Misiurewicz, and G. Piliouras (2019). The route to chaos in routing games: When is Price of Anarchy too optimistic? *arXiv preprint arXiv:1906.02486*.

Cirant, M. (2015). Multi-population mean field games systems with Neumann boundary conditions. *Journal de Mathématiques Pures et Appliquées* 103.5, pp. 1294–1315.

Couillet, R., S. M. Perlaza, H. Tembine, and M. Debbah (2012). Electrical vehicles in the smart grid: A mean field game analysis. *IEEE Journal on Selected Areas in Communications* 30.6.

Cucker, F. and E. Mordecki (2008). Flocking in noisy environments. *Journal de mathématiques pures et appliquées* 89.3, pp. 278–296.

Cucker, F. and S. Smale (2007). Emergent behavior in flocks. *IEEE Transactions on automatic control*.

Cui, K. and H. Koeppl (2021). Approximately Solving Mean Field Games via Entropy-Regularized Deep Reinforcement Learning. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Ed. by A. Banerjee and K. Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, pp. 1909–1917.

Cui, K., A. Tahir, G. Ekinci, A. Elshamanhory, Y. Eich, M. Li, et al. (2022). *A Survey on Large-Population Systems and Scalable Multi-Agent Reinforcement Learning*.

Daskalakis, C. and Q. Pan (2014). A counter-example to Karlin's strong conjecture for fictitious play. In *IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE.

Degl'Innocenti, L. (2018). Correlated Equilibria in Static Mean-Field Games. PhD thesis. Università degli Studi di Padova.

Delarue, F. and A. Vasileiadis (2021). Exploration noise for learning linear-quadratic mean field games. *arXiv preprint arXiv:2107.00839*.

Djehiche, B., A. Tcheukam Siwe, and H. Tembine (Nov. 2017). Mean-Field-Type Games in Engineering. *AIMS Electronics and Electrical Engineering* 1.

Djehiche, B., A. Tcheukam, and H. Tembine (2017). A mean-field game of evacuation in multi-level building. *IEEE Transactions on Automatic Control* 62.10.

Doncel, J., N. Gast, and B. Gaujal (2022). A mean field game analysis of SIR dynamics with vaccination. *Probability in the Engineering and Informational Sciences* 36.2, pp. 482–499.

Ducatelle, F., G. A. Di Caro, A. Förster, M. Bonani, M. Dorigo, S. Magnenat, et al. (2014). Cooperative navigation in robotic swarms. *Swarm Intelligence* 8.1.

Duffie, D. and Y. Sun (2012). The exact law of large numbers for independent random matching. *Journal of Economic Theory* 147.3, pp. 1105–1139.

Duncan, T. E. and H. Tembine (2018). Linear–quadratic mean-field-type games: A direct method. *Games* 9.1.

Durkan, C., A. Bekasov, I. Murray, and G. Papamakarios (2019). *Neural Spline Flows*.

Elie, R., E. Hubert, T. Mastrolia, and D. Possamai (2019). Mean-field moral hazard for optimal energy demand response management. *Mathematical Finance* (*to appear*).

Elie, R., E. Hubert, and G. Turinici (2020). Contact rate epidemic control of COVID-19: an equilibrium view. *Mathematical Modelling of Natural Phenomena*.

Elie, R., T. Ichiba, and M. Laurière (2020). Large banking systems with default and recovery: A mean field game model. *arXiv preprint arXiv:2001.10206*.

Elie, R., T. Mastrolia, and D. Possamai (2019). A tale of a principal and many, many agents. *Mathematics of Operations Research* 44.2.

Elie, R., J. Perolat, M. Laurière, M. Geist, and O. Pietquin (2020). On the convergence of model free learning in mean field games. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05, pp. 7143–7150.

Farago, J., A. Greenwald, and K. Hall (2002). Fair and Efficient Solutions to the Santa Fe Bar Problem. In *Proceedings of the Grace Hopper Conference on Women in Computing*.

Feleqi, E. (2013). The derivation of ergodic mean field game equations for several populations of players. *Dynamic Games and Applications* 3.4, pp. 523–536.

Firoozi, D. and S. Jaimungal (2022). Exploratory LQG mean field games with entropy regularization. *Automatica* 139, p. 110177.

Foerster, J. N., G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson (2018). Counterfactual multi-agent policy gradients. In *Proceedings of AAAI*.

Foerster, J., R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, and I. Mordatch (2018). Learning with opponent-learning awareness. In *Proceedings of AAMAS*.

Foerster, J., N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, et al. (2017). Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of ICML*. JMLR. org.

Fouque, J.-P. and Z. Zhang (2020). Deep Learning Methods for Mean Field Control Problems With Delay. *Frontiers in Applied Mathematics and Statistics* 6.

François-Lavet, V., P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau (2018). An Introduction to Deep Reinforcement Learning. *Foundations and Trends® in Machine Learning* 11.3-4, pp. 219–354.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences* 3.4, pp. 128–135.

Freund, Y. and R. E. Schapire (1999). Adaptive game playing using multiplicative weights. *Games and Economic Behavior* 29.1-2, pp. 79–103.

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

Fudenberg, D. and D. K. Levine (1998a). The Theory of Learning in Games. *MIT Press Books* 1.

— (1998b). *The Theory of Learning in Games*. Vol. 2. MIT press.

Fudenberg, D. and J. Tirole (1991). *Game theory*. MIT press.

Fujimoto, S., H. Hoof, and D. Meger (2018). Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*. PMLR, pp. 1587–1596.

Gast, N., B. Gaujal, and J.-Y. Le Boudec (2012). Mean field for Markov decision processes: from discrete to continuous optimization. *IEEE Transactions on Automatic Control* 57.9, pp. 2266–2280.

Geist, M., J. Pérolat, M. Laurière, R. Elie, S. Perrin, O. Bachem, et al. (2021). Concave utility reinforcement learning: the mean-field game viewpoint. *Autonomous Agents and Multiagent Systems (AAMAS 2022) (arXiv preprint arXiv:2106.03787)*.

Geist, M., B. Scherrer, and O. Pietquin (Sept. 2019). A Theory of Regularized Markov Decision Processes. In *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 2160–2169.

Germain, M., J. Mikael, and X. Warin (2022). Numerical resolution of McKean-Vlasov FBSDEs using neural networks. *Methodology and Computing in Applied Probability*, pp. 1–30.

Gomes, D. A., J. Mohr, and R. R. Souza (2010). Discrete time, finite state space mean field games. *Journal de mathématiques pures et appliquées* 93.3, pp. 308–328.

Gomes, D. A., S. Patrizi, and V. Voskanyan (2014). On the existence of classical solutions for stationary extended mean field games. *Nonlinear Analysis: Theory, Methods & Applications* 99, pp. 49–79.

Gomes, D. A. and J. Saúde (2020). A Mean-Field Game Approach to Price Formation. *Dynamic Games and Applications*.

Gomes, D. A. and V. K. Voskanyan (2016). Extended deterministic mean-field games. *SIAM Journal on Control and Optimization* 54.2, pp. 1030–1055.

Gomes, D., R. M. Velho, and M.-T. Wolfram (2014). Socio-economic applications of finite state mean field games. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 372.2028.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Goodfellow, I., M. Mirza, D. Xiao, A. Courville, and Y. Bengio (2014). An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *Proceedings of ICLR*.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, et al. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27 (NIPS)*.

Graber, P. J. (2016). Linear quadratic mean field type control and mean field games with common noise, with application to production of an exhaustible resource. *Applied Mathematics & Optimization* 74.3.

Graber, P. J. and A. Bensoussan (2018). Existence and uniqueness of solutions for Bertrand and Cournot mean field games. *Applied Mathematics & Optimization* 77.1.

Greenwald, A., K. Hall, and R. Serrano (2003). Correlated Q-learning. In *Proceedings of ICML*. Vol. 20.

Grover, P., K. Bakshi, and E. A. Theodorou (2018). A mean-field game model for homogeneous flocking. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 28.6, p. 061103.

Gu, H., X. Guo, X. Wei, and R. Xu (2019). Dynamic Programming Principles for Mean-Field Controls with Learning. *arXiv preprint arXiv:1911.07314*.

— (2020). Q-learning for mean-field controls. *arXiv preprint arXiv:2002.04131*.

— (2021a). Mean-field controls with Q-learning for cooperative MARL: convergence and complexity analysis. *SIAM Journal on Mathematics of Data Science* 3.4, pp. 1168–1196.

— (2021b). Mean-field multi-agent reinforcement learning: A decentralized network approach. *arXiv preprint arXiv:2108.02731*.

Guéant, O., J.-M. Lasry, and P.-L. Lions (2011). Mean field games and applications. In *Paris-Princeton lectures on mathematical finance 2010*. Springer, pp. 205–266.

Guo, X., A. Hu, R. Xu, and J. Zhang (2019). Learning mean-field games. *Advances in Neural Information Processing Systems* 32.

— (2020). A General Framework for Learning Mean-Field Games. *CoRR* abs/2003.06069.

Guo, X., R. Xu, and T. Zariphopoulou (2020). Entropy regularization for mean field games with learning. *arXiv preprint arXiv:2010.00145*.

Haarnoja, T., A. Zhou, P. Abbeel, and S. Levine (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, pp. 1861–1870.

Hadikhanloo, S. (2017). Learning in anonymous nonatomic games with applications to first-order mean field games. *arXiv preprint arXiv:1704.00378*.

— (2018). Learning in mean field games. PhD thesis. PhD thesis. Université Paris sciences et lettres.

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

Hamidouche, K., W. Saad, M. Debbah, and H. V. Poor (2016). Mean-field games for distributed caching in ultra-dense small cell networks. In *2016 American Control Conference (ACC)*. IEEE.

Harris, C. (1998). On the rate of convergence of continuous-time fictitious play. *Games and Economic Behavior* 22.2.

Heinrich, J., M. Lanctot, and D. Silver (2015). Fictitious self-play in extensive-form games. In *Proceedings of ICML*.

Heinrich, J. and D. Silver (2016). Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*.

Hill, A., A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, et al. (2018). *Stable Baselines*. https://github.com/hill-a/stable-baselines.

Hofbauer, J. and W. H. Sandholm (2002). On the global convergence of stochastic fictitious play. *Econometrica* 70.6.

Hu, J. and M. P. Wellman (2003). Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research* 4.Nov.

Hu, R. (2021). Deep fictitious play for stochastic differential games. *Communications in mathematical sciences* 19.2.

Hu, R. and M. Laurière (2022). Recent Developments in Machine Learning Methods for Stochastic Control and Games. *SSRN preprint:4096569*.

Huang, K., X. Di, Q. Du, and X. Chen (2017). A game-theoretic framework for autonomous vehicles velocity control: Bridging microscopic differential games and macroscopic mean field games. *Discrete & Continuous Dynamical Systems - B* 22.11.

— (2019). Stabilizing traffic via autonomous vehicles: A continuum mean field game approach. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, pp. 3269–3274.

Huang, M., R. P. Malhamé, and P. E. Caines (2006). Large population stochastic dynamic games: closed-loop McKean-Vlasov systems and the Nash certainty equivalence principle. *Communications in Information & Systems* 6.3, pp. 221–252.

Hubert, E. and G. Turinici (2018). Nash-MFG equilibrium in a SIR model with time dependent newborn vaccination. *Ricerche di Matematica* 67.1.

Karlin, S. (1959). *Mathematical Methods and Theory in Games, Programming and Economics. Addison-Wesley*. American Association for the Advancement of Science.

Khalil, H. K. (2002). *Nonlinear systems; 3rd ed.* Prentice-Hall.

Kingma, D. P. and M. Welling (2014). Auto-Encoding Variational Bayes. In *Proceedings of ICLR*.

Kizilkale, A. C., R. Salhab, and R. P. Malhamé (2019). An integral control formulation of mean field game based large scale coordination of loads in smart grids. *Automatica* 100.

Kobeissi, Z. (2022). On classical solutions to the mean field game system of controls. *Communications in Partial Differential Equations* 47.3, pp. 453–488.

Kobyzev, I., S. Prince, and M. Brubaker (2020). Normalizing flows: An introduction and review of current methods. *PAMI*.

Kolokoltsov, V. N. and A. Bensoussan (2016). Mean-field-game model for botnet defense in cyber-security. *Applied Mathematics & Optimization* 74.3.

Kolokoltsov, V. N. and O. A. Malafeyev (2018). Corruption and botnet defense: a mean field game approach. *International Journal of Game Theory* 47.3.

Koutsoupias, E. and C. Papadimitriou (1999). Worst-case equilibria. In *Annual symposium on theoretical aspects of computer science*. Springer, pp. 404–413.

Lachapelle, A., J.-M. Lasry, C.-A. Lehalle, and P.-L. Lions (2016). Efficiency of the price formation process in presence of high frequency participants: a mean field game analysis. *Mathematics and Financial Economics* 10.3.

Lachapelle, A. and M.-T. Wolfram (2011). On a mean field game approach modeling congestion and aversion in pedestrian crowds. *Transportation research part B: methodological* 45.10, pp. 1572–1589.

Lacker, D. (2017). Limit theory for controlled McKean–Vlasov dynamics. *SIAM Journal on Control and Optimization* 55.3, pp. 1641–1672.

— (2020). On the convergence of closed-loop Nash equilibria to the mean field game limit. *The Annals of Applied Probability* 30.4, pp. 1693–1761.

Lacker, D. and K. Ramanan (2019). Rare Nash equilibria and the price of anarchy in large static games. *Mathematics of Operations Research* 44.2, pp. 400–422.

Laguzet, L. and G. Turinici (2015). Individual vaccination as Nash equilibrium in a SIR model with application to the 2009–2010 influenza A (H1N1) epidemic in France. *Bulletin of Mathematical Biology* 77.10, pp. 1955–1984.

Lanctot, M., E. Lockhart, J.-B. Lespiau, V. Zambaldi, S. Upadhyay, J. Pérolat, et al. (2019). OpenSpiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*.

Lanctot, M., K. Waugh, M. Zinkevich, and M. Bowling (2009). Monte Carlo sampling for regret minimization in extensive games. In vol. 22, pp. 1078–1086.

Lanctot, M., V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Perolat, et al. (2017). A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning. In *Proceedings of NeurIPS*.

Lasry, J.-M. and P.-L. Lions (2007). Mean field games. *Japanese journal of mathematics* 2.1, pp. 229–260.

Laurière, M. (2021). Numerical methods for mean field games and mean field type control. *Mean Field Games* 78, p. 221.

Laurière, M., S. Perrin, M. Geist, and O. Pietquin (2022). Learning Mean Field Games: A Survey. *arXiv:2205.12944*.

Lauriere, M., S. Perrin, S. Girgin, P. Muller, A. Jain, T. Cabannes, et al. (2022). Scalable Deep Reinforcement Learning Algorithms for Mean Field Games. In *Proceedings of the 39th International Conference on Machine Learning (ICML 2022)*. Vol. 162, pp. 12078–12095.

Laurière, M., J. Song, and Q. Tang (2021). Policy iteration method for time-dependent Mean Field Games systems with non-separable Hamiltonians. *arXiv preprint arXiv:2110.02552*.

Lee, W., S. Liu, H. Tembine, W. Li, and S. Osher (2021). Controlling propagation of epidemics via mean-field control. *SIAM Journal on Applied Mathematics* 81.1, pp. 190–207.

Li, F., R. P. Malhamé, and J. Le Ny (2016). Mean field game based control of dispersed energy storage devices with constrained inputs. In *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE.

Li, Y., L. Wang, J. Yang, E. Wang, Z. Wang, T. Zhao, et al. (2021). Permutation invariant policy optimization for mean-field multi-agent reinforcement learning: A principled approach. *arXiv preprint arXiv:2105.08268*.

Liang, E., R. Liaw, R. Nishihara, P. Moritz, R. Fox, J. Gonzalez, et al. (2017). Ray RLLib: A Composable and Scalable Reinforcement Learning Library. *CoRR* abs/1712.09381.

Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, et al. (2016). Continuous control with deep reinforcement learning. In *Proceedings of ICLR*.

Lin, A. T., S. W. Fung, W. Li, L. Nurbekyan, and S. J. Osher (2020). APAC-Net: Alternating the Population and Agent Control via Two Neural Networks to Solve High-Dimensional Stochastic Mean Field Games. *arXiv preprint arXiv:2002.10113*.

Lions, P.-L. (2012). *Lecture at the Collège de France*.

McGuire, K., C. De Wagter, K. Tuyls, H. Kappen, and G. C. de Croon (2019). Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Science Robotics* 4.35.

Mériaux, F., V. Varma, and S. Lasaulce (2012). Mean field energy games in wireless networks. In *2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*. IEEE.

Mertikopoulos, P., C. Papadimitriou, and G. Piliouras (2018). Cycles in adversarial regularized learning. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, pp. 2703–2717.

Mertikopoulos, P. and W. H. Sandholm (2016). Learning in games via reinforcement and regularization. *Mathematics of Operations Research* 41.4, pp. 1297–1324.

Meyn, S. (2022). *Control Systems and Reinforcement Learning*. Cambridge University Press.

Mguni, D., J. Jennings, and E. Munoz de Cote (2018). Decentralised Learning in Systems With Many, Many Strategic Agents. In *Proceedings of AAAI*.

Miehling, E. and T. Başar (2022). Reinforcement Learning for Non-stationary Discrete-Time Linear–Quadratic Mean-Field Games in Multiple Populations. *Dynamic Games and Applications*, pp. 1–47.

Mishra, R. K., D. Vasal, and S. Vishwanath (2020). Model-free reinforcement learning for non-stationary mean field games. In *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, pp. 1032–1037.

Mitchell, T. M. (1997). Machine learning.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, et al. (2013). Playing atari with deep reinforcement learning. *In NIPS Workshop on Deep Learning and Unsupervised Feature Learning. arXiv preprint arXiv:1312.5602*.

Moravcik, M., M. Schmid, N. Burch, V. Lisy, D. Morrill, N. Bard, et al. (2017). Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* 356.6337, pp. 508–513.

Morrill, D., R. D'Orazio, M. Lanctot, J. R. Wright, M. Bowling, and A. R. Greenwald (2021). Efficient Deviation Types and Learning for Hindsight Rationality in Extensive-Form Games. In *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 7818–7828.

Morrill, D., R. D'Orazio, R. Sarfati, M. Lanctot, J. R. Wright, A. Greenwald, et al. (2020). Hindsight and Sequential Rationality of Correlated Play. *arXiv preprint arXiv:2012.05874*.

Motte, M. and H. Pham (2019). Mean-field Markov decision processes with common noise and open-loop controls. *arXiv preprint arXiv:1912.07883*.

Muller, P., R. Elie, M. Rowland, M. Laurière, J. Perolat, S. Perrin, et al. (2022). Learning Correlated Equilibria in Mean-Field Games. *arXiv:2208.10138*.

Muller, P., M. Rowland, R. Elie, G. Piliouras, J. Perolat, M. Laurière, et al. (2021). Learning Equilibria in Mean-Field Games: Introducing Mean-Field PSRO. *AAMAS 2022 (arXiv preprint arXiv:2111.08350)*.

Nash, J. F. (1950). Equilibrium points in n-person games. *Proceedings of the national academy of sciences* 36.1, pp. 48–49.

— (1951). Non-cooperative games. *Annals of mathematics*, pp. 286–295.

Nemirovsky, A. and D. Yudin (1979). Problem Complexity and Optimization Method Efficiency. *M.: Nauka*.

Neu, G., A. Jonsson, and V. Gómez (2017). *A unified view of entropy-regularized Markov decision processes*.

Neumann, J. von (1928). Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen* 100, pp. 295–300.

Neumann, J. von and O. Morgenstern (1944). *Theory of Games and Economic Behavior*. Princeton University Press.

Nisan, N., T. Roughgarden, E. Tardos, and V. V. Vazirani (2007). *Algorithmic Game Theory*. USA: Cambridge University Press.

Nourian, M., P. E. Caines, and R. P. Malhamé (2010). Synthesis of Cucker-Smale type flocking via mean field stochastic control theory: Nash equilibria. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, pp. 814–819.

— (2011). Mean field analysis of controlled Cucker-Smale type flocking: Linear analysis and perturbation equations. *IFAC Proceedings Volumes* 44.1, pp. 4471–4476.

Okubo, A. (1986). Dynamical aspects of animal grouping: swarms, schools, flocks, and herds. *Advances in biophysics*.

Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on automatic control*.

Omidshafiei, S., D. Hennes, D. Morrill, R. Munos, J. Perolat, M. Lanctot, et al. (2019). Neural Replicator Dynamics. *arXiv preprint arXiv:1906.00190*.

Ostrovski, G. and S. Strien (Aug. 2013). Payoff Performance of Fictitious Play. *Journal of Dynamics and Games* 1.

Palaiopanos, G., I. Panageas, and G. Piliouras (2017). Multiplicative weights update with constant step-size in congestion games: Convergence, limit cycles and chaos. In *Advances in Neural Information Processing Systems*, pp. 5872–5882.

Papamakarios, G., E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research* 22.57, pp. 1–64.

Parise, F., S. Grammatico, B. Gentile, and J. Lygeros (2015). Network aggregative games and distributed mean field control via consensus theory. *arXiv preprint arXiv:1506.07719*.

Partridge, B. L. (1982). The structure and function of fish schools. *Scientific american* 246.6, pp. 114–123.

Pérolat, J., R. Munos, J.-B. Lespiau, S. Omidshafiei, M. Rowland, P. Ortega, et al. (2021). From Poincaré recurrence to convergence in imperfect information games: Finding equilibrium via regularization. In *Proceedings of ICML*.

Perolat, J., S. Perrin, R. Elie, M. Laurière, G. Piliouras, M. Geist, et al. (2021). Scaling up Mean Field Games with Online Mirror Descent. *Autonomous Agents and Multiagent Systems (AAMAS 2022) (arXiv preprint arXiv:2103.00623)*.

Perolat, J., B. Piot, and O. Pietquin (2018). Actor-critic fictitious play in simultaneous move multistage games. In *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 919–928.

Perolat, J., B. de Vylder, D. Hennes, E. Tarassov, F. Strub, V. de Boer, et al. (2022). *Mastering the Game of Stratego with Model-Free Multiagent Reinforcement Learning*.

Perrin, S., M. Laurière, J. Pérolat, R. Élie, M. Geist, and O. Pietquin (2021). Generalization in Mean Field Games by Learning Master Policies. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2022) (arXiv preprint arXiv:2109.09717)*.

Perrin, S., M. Laurière, J. Pérolat, M. Geist, R. Élie, and O. Pietquin (2021). Mean Field Games Flock! The Reinforcement Learning Way. *International Joint Conference of Artificial Intelligence* (*IJCAI 2021*) (*arXiv preprint arXiv:2105.07933*).

Perrin, S., J. Perolat, M. Laurière, M. Geist, R. Elie, and O. Pietquin (2020). Fictitious Play for Mean Field Games: Continuous Time Analysis and Applications. In *Advances in Neural Information Processing Systems* (*NeurIPS 2020*). Vol. 33. Curran Associates, Inc., pp. 13199–13213.

Piliouras, G. and J. S. Shamma (2014). Optimization despite chaos: Convex relaxations to complex limit sets via Poincaré recurrence. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, pp. 861–873.

Postlethwaite, C. M. and A. M. Rucklidge (2017). Spirals and heteroclinic cycles in a spatially extended Rock-Paper-Scissors model of cyclic dominance. *EPL* (*Europhysics Letters*) 117.4, p. 48006.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Rawlik, K., M. Toussaint, and S. Vijayakumar (2012). On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII*.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of SIGGRAPH*.

Rezende, D. and S. Mohamed (2015). Variational inference with normalizing flows. In *International conference on machine learning*. PMLR, pp. 1530–1538.

Robinson, J. (1951). An iterative method of solving a game. *Annals of mathematics*, pp. 296–301.

Roughgarden, T. (2009). Intrinsic robustness of the price of anarchy. In *Proceedings of STOC*, pp. 513–522.

Roughgarden, T. and E. Tardos (2007). Introduction to the inefficiency of equilibria. *Algorithmic game theory* 17, pp. 443–459.

Ruthotto, L., S. J. Osher, W. Li, L. Nurbekyan, and S. W. Fung (2020). A machine learning framework for solving high-dimensional mean field game and mean field control problems. *Proceedings of the National Academy of Sciences* 117.17.

Saldi, N., T. Başar, and M. Raginsky (2018). Markov-Nash equilibria in mean-field games with discounted cost. *SIAM Journal on Control and Optimization* 56.6.

Salhab, R., J. Le Ny, and R. P. Malhamé (2018). A mean field route choice game model. In *2018 IEEE Conference on Decision and Control* (*CDC*). IEEE, pp. 1005–1010.

Samarakoon, S., M. Bennis, W. Saad, M. Debbah, and M. Latva-Aho (2015). Energy-efficient resource management in ultra dense small cell networks: A mean-field approach. In *IEEE Global Communications Conference* (*GLOBECOM*).

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development* 3.3.

Sato, Y., E. Akiyama, and J. D. Farmer (2002). Chaos in learning a simple two-person game. *Proceedings of the National Academy of Sciences* 99.7, pp. 4748–4751.

Schaeffer, J., N. Burch, Y. Björnsson, A. Kishimoto, M. Müller, R. Lake, et al. (2007). Checkers is solved. *Science* 317.5844.

Schmeidler, D. (1973). Equilibrium points of nonatomic games. *Journal of statistical Physics* 7.4, pp. 295–300.

Schulman, J., S. Levine, P. Abbeel, M. Jordan, and P. Moritz (2015). Trust region policy optimization. In *International conference on machine learning*. PMLR, pp. 1889–1897.

Shalev-Shwartz, S. (2011). Online learning and online convex optimization. *Foundations and trends in Machine Learning* 4.2, pp. 107–194.

Shannon, C. E. (1959). Programming a Computer Playing Chess. *Philosophical Magazine* Series 7, 41.312.

Shapiro, H. N. (1958). Note on a computation method in the theory of games. In *Communications on Pure and Applied Mathematics*.

Shaw, E. (1975). Naturalist at large-fish in schools. *Natural History*.

Shiri, H., J. Park, and M. Bennis (2019). Massive Autonomous UAV Path Planning: A Neural Network Based Mean-Field Game Theoretic Approach. In *IEEE Global Communications Conference (GLOBECOM)*.

Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence* 60.1, pp. 51–92.

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature* 529.7587.

Silver, D., T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 632.6419.

Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, et al. (2017). Mastering the game of Go without human knowledge. *Nature* 550.7676.

Srebro, N., K. Sridharan, and A. Tewari (2011). On the universality of online mirror descent. *arXiv preprint arXiv:1107.4080*.

Srinivasan, S., M. Lanctot, V. Zambaldi, J. Pérolat, K. Tuyls, R. Munos, et al. (2018). Actor-critic policy optimization in partially observable multiagent environments. In *Proceedings of NeurIPS*.

Stella, L., F. Bagagiolo, D. Bauso, and G. Como (2013). Opinion dynamics and stubbornness through mean-field games. In *52nd IEEE Conference on Decision and Control*. IEEE.

Subramanian, J. and A. Mahajan (2019). Reinforcement learning in stationary mean-field games. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 251–259.

Subramanian, J., R. Seraj, and A. Mahajan (2018). Reinforcement learning for mean-field teams. In *Workshop on Adaptive and Learning Agents at International Conference on Autonomous Agents and Multi-Agent Systems.*

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

Sun, Y. (2006). The exact law of large numbers via Fubini extension and characterization of insurable risks. *Journal of Economic Theory* 126.1, pp. 31–69.

Sutton, R. S. and A. G. Barto (2018). *Reinforcement Learning: An Introduction*. 2nd. The MIT Press.

Szepesvári, C. (2010). Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning* 4.1, pp. 1–103.

Sznitman, A.-S. (1991). Topics in propagation of chaos. In *Ecole d'été de probabilités de Saint-Flour XIX—1989*. Springer, pp. 165–251.

Szolnoki, A., B. de Oliveira, and D. Bazeia (2020). Pattern formations driven by cyclic interactions: A brief review of recent developments. *EPL (Europhysics Letters)* 131.6, p. 68001.

Szymanski, M., T. Breitling, J. Seyfried, and H. Wörn (2006). Distributed shortest-path finding by a micro-robot swarm. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*. Springer.

Tanaka, T., E. Nekouei, A. R. Pedram, and K. H. Johansson (2020). Linearly solvable mean-field traffic routing games. *IEEE Transactions on Automatic Control* 66.2, pp. 880–887.

Tembine, H., R. Tempone, and P. Vilanova (2012). Mean-field learning: a survey. *arXiv preprint arXiv:1210.4657*.

Todorov, E. (2008). General duality between optimal control and estimation. In *2008 47th IEEE Conference on Decision and Control*. IEEE, pp. 4286–4292.

Toner, J. and Y. Tu (1998). Flocks, herds, and schools: A quantitative theory of flocking. *Physical review E* 58.4, p. 4828.

Toussaint, M. (2009). Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, pp. 1049–1056.

Vieillard, N., O. Pietquin, and M. Geist (2020). Munchausen Reinforcement Learning. In *Proceedings of NeurIPS*.

Vinyals, O., I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, et al. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575.7782, pp. 350–354.

Wang, L., Z. Yang, and Z. Wang (2020). Breaking the curse of many agents: Provable mean embedding q-iteration for mean-field reinforcement learning. In *International Conference on Machine Learning*. PMLR, pp. 10092–10103.

Wang, W., J. Han, Z. Yang, and Z. Wang (2021). Global convergence of policy gradient for linear-quadratic mean-field control/game in continuous time. In *International Conference on Machine Learning*. PMLR, pp. 10772–10782.

Wang, X., J. Cerny, S. Li, C. Yang, Z. Yin, H. Chan, et al. (2022). A Unified Perspective on Deep Equilibrium Finding. *arXiv preprint arXiv:2204.04930*.

Watkins, C. J. and P. Dayan (1992). Q-learning. *Machine learning* 8.3-4, pp. 279–292.

Watkins, C. (1989). Learning form delayed rewards. *Ph. D. thesis, King's College, University of Cambridge*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8.3, pp. 229–256.

Wooldridge, M. and N. R. Jennings (1995). Agent theories, architectures, and languages: A survey. In *Intelligent Agents*. Ed. by M. J. Wooldridge and N. R. Jennings. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–39.

Xie, Q., Z. Yang, Z. Wang, and A. Minca (2021). Learning While Playing in Mean-Field Games: Convergence and Optimality. In *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 11436–11447.

Yang, C., J. Li, M. Sheng, A. Anpalagan, and J. Xiao (2017). Mean field game-theoretic framework for interference and energy-aware control in 5G ultra-dense networks. *IEEE Wireless Communications* 25.1.

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

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

Yin, H., P. G. Mehta, S. P. Meyn, and U. V. Shanbhag (2010). Learning in mean-field oscillator games. In *49th IEEE Conference on Decision and Control (CDC)*. IEEE.

Zaman, M. A. uz, K. Zhang, E. Miehling, and T. Başar (2020). Reinforcement learning in non-stationary discrete-time linear-quadratic mean-field games. In *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, pp. 2278–2284.

Zhang, K., Z. Yang, and T. Başar (2021). Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, pp. 321–384.

Zhou, Z., P. Mertikopoulos, A. L. Moustakas, N. Bambos, and P. Glynn (2017). Mirror descent learning in continuous games. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, pp. 5776–5783.

Zinkevich, M., M. Johanson, M. Bowling, and C. Piccione (2007). Regret minimization in games with incomplete information. In *Proceedings of NeurIPS*. Vol. 20, pp. 1729–1736.