Reinforcement Learning: The Multi-Player Case

Julien Pérolat

Université des Sciences et des Technologies de Lille
École Doctorale Sciences Pour l'Ingénieur

Thèse de Doctorat

préparée au sein du laboratoire CRIStAL, UMR 9189 Lille 1/CNRS

Spécialité : **Informatique**

présentée par
Julien PEROLAT

Reinforcement Learning: The Multi-Player Case

---

sous la direction de **Olivier PIETQUIN**

Soutenue publiquement à **Villeneuve d'Ascq**, le 18 **decembre 2017** devant le jury composé de :

D. **ERNST**	Université de Liège	Rapporteur
D. **PRECUP**	Université McGill	Rapporteur
O. **PIETQUIN**	DeepMind	Directeur
L. **DUCHIEN**	Université de Lille	Examinateur
R. **ORTNER**	Université de Leoben	Examinateur
B. **PIOT**	DeepMind	Examinateur
B. **SCHERRER**	Inria	Examinateur
K. **TUYLS**	Université de Liverpool	Examinateur

English title: Reinforcement Learning : The Multi-Player Case

Mots clés: apprentissage automatique, algorithmes, intelligence artificielle, apprentissage par renforcement, théorie des jeux, apprentissage multi-agent.

Key words: machine learning, algorithm, artificial intelligence, reinforcement learning, game theory, multi-agent learning.

Acknowledgement & Remerciements

Ce manuscrit de thèse a vu le jour grâce à l'aide de nombreuses personnes que je tiens à remercier. En premier lieu, je remercie mon directeur de thèse Olivier Pietquin qui m'a accompagné durant ces trois années. Je tiens aussi à remercier Bilal Piot et Bruno Scherrer pour leur soutien dès le premier jour. Je n'oublie pas Marie-Annick et Hugh Gash qui ont relu et corrigé une grande partie de ce manuscrit. Cette thèse doit aussi beaucoup à l'environnement qu'est l'équipe SequeL dirigé par Philippe Preux. Je souhaite bien sur remercier les permanents de cette équipe: Michal, Emilie, Odalric, Romaric, Jérémie, Alessandro, Danil ainsi que les doctorants avec qui j'ai partagé une partie de cette thèse Florian, Alexandre, Daniele, Marc, Merwan, Pratik, Frederic, Tomas, Hadrien, Marta, Lilian, Nicolas, Ronan, Jean-Bastien, Guillaume et Romain pour les tous les bons moments passés ensemble. Cette thèse m'a permis d'effectuer un stage à DeepMind où j'ai eu l'opportunité de travailler dans l'équipe de Thore Graepel que je tiens à remercier ainsi que les chercheurs et ingénieurs avec qui j'ai pu travailler Joel, Karl, Marc et Vinicius. Je terminerai en remerciant ma famille pour la constance de son soutien tout au long de ces longues années d'étude.

Abstract.

This thesis mainly focuses on learning from historical data in a sequential multi-agent environment. We studied the problem of batch learning in Markov games (MGs). Markov games are a generalization of Markov decision processes (MDPs) to the multi-agent setting. Our approach was to propose learning algorithms to find equilibria in games where the knowledge of the game is limited to interaction samples (also named batch data). To achieve this task, we explore two main approaches.

The first approach explored in this thesis is to study approximate dynamic programming techniques. We generalize several batch algorithms from MDPs to zero-sum two-player MGs. This part of our work generalizes several approximate dynamic programming bounds from a $\mathcal{L}_\infty$-norm to a $\mathcal{L}_p$-norm. Then we describe, test and compare algorithms based on those dynamic programming schemes. But these algorithms are highly sensitive to the discount factor (a parameter that controls the time horizon of the problem). To improve those algorithms, we studied many non-stationary variants of approximate dynamic methods to the zero sum two player case. In the end, we show that using non-stationary strategies can be used in general sum games. However, the resulting guarantees are very loose compared to the one on MDPs or zero-sum two-player MGs.

The second approach studied in this manuscript is the Bellman residual approach. This approach reduces the problem of learning from batch data to the minimization of a loss function. In a zero-sum two-player MG, we prove that using a Newton's method on some Bellman residuals is either equivalent to the Least Squares Policy Iteration (LSPI) algorithm or to the Bellman Residual Minimizing Policy Iteration (BRMPI) algorithm. We leverage this link to address the oscillation of LSPI in MDPs and in MGs. Then we show that a Bellman residual approach could be used to learn from batch data in general-sum MGs.

Finally in the last part of this dissertation, we study multi-agent independent learning in Multi-Stage Games (MSGs). We provide an actor-critic independent learning algorithm that provably converges in zero-sum two-player MSGs and in cooperative MSGs and empirically converges using function approximation on the game of Alesia.

Contents

Summary of the Notations

1 Summary of the Notations

This section we will summarize the definitions and notations introduced in this chapter. It provides a recapitulation of all general notations and a table comparing the cases of MDPs, zero-sum two-player MGs and general-sum MGs.

1.1 General Notations

- the set ΔA is the set of the probability distributions on A,
- $\mathbb{N}$ the set of integer $\{0, 1, 2, \ldots\}$,
- $\mathbb{R}$ the set of real numbers,
- $|S|$ the cardinal of set S,
- $\mathbf{1}_{\tilde{s}}$ is the function defined on S, such that $\mathbf{1}_{\tilde{s}}(s) = 1$ if $\tilde{s} = s$ and 0 otherwise,
- the $\mathcal{L}_{+\infty}$-norm is defined as $\|f\|_{+\infty} = \sup_{x \in X} f(x)$,
- I is the identity,
- the $\mathcal{L}_p$-norm of the function f (written $\|f\|_p$) is $\|f\|_p^p = \sum_{x \in X} f(x)^p$
- The $\mathcal{L}_{\rho,p}$ of function f is $\|f\|_{\rho,p}$ is defined here as $\|f\|_{\rho,p}^p = \sum_{x \in X} \rho(x) f(x)^p$
- $a \sim \pi$ means that a is a random variable drawn according to the law π

1.2 Statistic:

- Unbiased estimator: an estimator $\hat{\theta}_n$ of the quantity θ is said to be unbiased if $E\left[\hat{\theta}_n\right] - \theta = 0$.
- Consistent estimator: an estimator is said to be consistent if $\lim_{n \to +\infty} \hat{\theta}_n - \theta$ in probability.

1.3 Reinforcement Learning and Multi-Agent Reinforcement Learning Notations

Table 1 – Reinforcement Learning and Multi-Agent Reinforcement Learning Notations

	MDP	zero-sum two-player MGs	general-sum MGs
State space	S	S	S
Number of player	1	2	N
Action space	A	A^1, A^2	$\{A^i\}_{i\in\{1,\dots,N\}}$
reward	$r(s,a)$	$r(s,a^1,a^2)$	$r(s,\boldsymbol{a})$
policy	$\pi(a\|s)$	$\mu(a^1\|s), \nu(a^2\|s)$	$\{\pi^i(a\|s)\}_{i\in\{1,\dots,N\}}$
kernel	$p(s'\|s,a)$	$p(s'\|s,a^1,a^2)$	$p(s'\|s,\boldsymbol{a})$
reward averaged	$r_\pi(s)$	$r_{\mu,\nu}(s)$	$\{r^i_{\boldsymbol{\pi}}(s)\}_{i\in\{1,\dots,N\}}$
kernel averaged	$P_\pi(s'\|s)$	$P_{\mu,\nu}(s'\|s)$	$P_{\boldsymbol{\pi}}(s'\|s)$
kernel Q-function	$\mathcal{P}_\pi(s',a\|s)$	$\mathcal{P}_{\mu,\nu}(s',b^1,b^2\|s,a^1,a^2)$	$\mathcal{P}_{\boldsymbol{\pi}}(s',\boldsymbol{b}\|s,\boldsymbol{a})$
value function	$v_\pi = (I-\gamma P_\pi)^{-1} r_\pi$	$v_{\mu,\nu} = (I-\gamma P_{\mu,\nu})^{-1} r_{\mu,\nu}$	$v^i_{\boldsymbol{\pi}} = (I-\gamma P_{\boldsymbol{\pi}})^{-1} r^i_{\boldsymbol{\pi}}$
Q-function	$Q_\pi = (I-\gamma\mathcal{P}_\pi)^{-1} r$	$Q_{\mu,\nu} = (I-\gamma\mathcal{P}_{\mu,\nu})^{-1} r$	$Q^i_{\boldsymbol{\pi}} = (I-\gamma\mathcal{P}_{\boldsymbol{\pi}})^{-1} r^i$

Part I

Introduction, Background, and Related Work

Chapter 1
Introduction

In many areas of research and industry, interaction data are collected. For example, many databases record professional games of chess, checkers or go. In many areas of industry, records of interactions between humans or between human and machines are logged. These data demonstrate an empirical knowledge of the rules of an interaction that can't necessarily be explained. For example, it can be hard to program a machine to interact with a human. We intend to study whether or not it is possible to learn an interaction strategy from these data. We will take a machine learning perspective on this problem. We will design and evaluate algorithms to learn interaction strategies from data without prior knowledge.

This document is devoted to the study of reinforcement learning methods in multi-agent environments. In reinforcement learning, machines are not explicitly programmed to perform a task but have to learn through a numerical reward to perform well. From a single agent environment to multi-agent environments, a large amount of problems can be studied and many agendas can be followed. The range of problems arising in multi-agent systems goes from the study of communication to the emergence of coordination and includes game theoretic equilibrium computation. There is a trove of scientifically relevant problems that are worth studying in multi-agent systems. First, we shall specify the agenda this work contributes to. In this thesis, we take a game theoretic approach to multi-agent learning. In classical game theory, each game player is supposed to know the rules of the game and is aware of the strategies each player can play and of their pay-off. This thesis shows that a machine can learn equilibria in games if the knowledge of the game is limited to samples.

In the single agent case, it is fairly straightforward to define what should be learnt. A single agent should learn how to behave in order to collect maximum rewards. In the multi-agent case, the optimal behaviour of a player must take into account the other players. If one agent changes its strategy, the others must adapt to this new strategic interaction. Co-adaptation is the reason why multi-agent systems are fundamentally different from single-agent systems. Despite the fact that players can adapt to changes of their opponents, one can still define stable strategic interaction called equilibrium in game theory. An equilibrium is a solution concept where players do not have an incentive to vary their current strategy even if they could. Equilibria have played a central role in the development of game theory and this thesis will study two forms of equilibria *i.e.*, the minimax solution and the Nash equilibrium. In zero-sum games, the canonical solution concept is the minimax strategy which guarantees a minimal pay-off to a player whatever the opponent does. It also stands that if both players play their counterpart of the equilibrium, none of them will have an incentive to shift from

their current strategy. However, interactions are not necessarily purely adversarial in a complex multi-agent systems, players might pursue similar objectives in some situations and might have opposed ones in others. When no reward structure is assumed, the game is called a general-sum game as opposed to zero-sum two-player games. In general-sum games, when players choose their strategy secretly and independently the solution concept that is considered is the Nash equilibrium. Again, in a Nash equilibrium, no player has an incentive to switch from their current strategy but the goal is not to exploit the strategy of the others. Our aim is to find a strategy for each player such that none of them have an incentive to switch. However, in general sum games, playing a Nash equilibrium does not guarantee a minimal pay-off but it makes cooperation possible where the minimax solution explicitly prevents it by considering a worst case scenario.

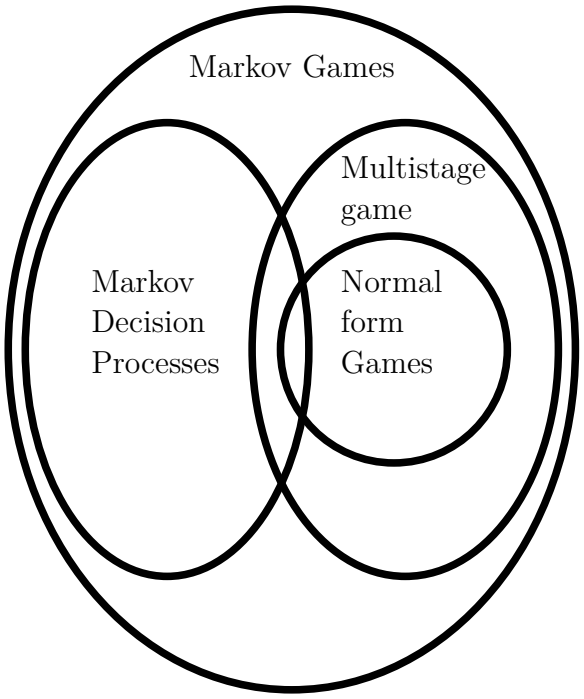


Figure 1.1 – Representation of the class of model we consider

However, one major drawback of classical game theory is that one needs to have a perfect knowledge of the interaction between all players. In a general-sum game, all players must know their own reward and the reward of the opponents to find a Nash equilibrium. But, in many situations this knowledge is not available and can only be observed through past interactions or while playing the game. The first case is named the batch scenario and is the focus of the main part of this dissertation. The second case is the online scenario and is explored later. In the batch scenario, the only information on the game available to learn a strategy for all players are historical interaction data between players. For example, the usual way to teach chess is to first explain the rules of the game and then to give a few basic strategies. However, we won't teach the machine that way since this method is game specific and since it uses game specific knowledge. As a comparison, instead of giving the machine the rules of the game, we give it details of several games and their outcome. The class of algorithms studied here leverage this historical information to learn an optimal strategy without being told the rules of the game. The second case studied in this thesis is independent reinforcement learning. In this setting agents no longer have access to historical data and must learn their strategy online while interacting with other learning agents. Agents are blind to each other's actions and only receive their own reward signal.

In the end, since the knowledge of the system in which players interact is limited to

samples of interaction, finding an exact Nash equilibrium or an exact minimax solution of the game will not be possible. We will learn these equilibria from interaction samples. To understand to what extent this can be done, we will need to analyse the impact this approximate knowledge of the game has on the solution.

1 Structure of the dissertation

The next chapter (Chapter 2) introduces the necessary background to game theory, reinforcement learning, and machine learning used in the dissertation. We provide an historical overview of the existing Dynamic Programming (DP) techniques to solve Markov Decision Processes (MDPs) and zero-sum two-player MGs. We also exhibit links between Bellman residual minimization and DP. Finally, we provide an overview of independent RL in games. The rest of the document is organized in three parts and six chapters.

The second part (Part II) of this manuscript is dedicated to the study of Approximate Dynamic Programming (ADP) techniques to learn from batch data. In Chapter 3 we introduce a novel family of batch algorithms based on a generic ADP scheme Approximate Generalized Policy Iteration (AGPI) that includes Approximate Value Iteration (AVI) and Approximate Policy Iteration (API). These algorithms are approximate since our knowledge of the game is limited to samples. These errors accumulate from one iteration to an other and have an impact on the accuracy of the final solution of the algorithm. We provide a unified sensitivity analysis of these algorithms to errors and provide an empirical evaluation of the algorithm on the game of Alesia. Chapter 4 explores the use of non-stationary (cyclic) strategies to reduce the sensitivity of the error on the final solution. We show that there is an explicit trade-off between the size of the cycle and the quality of the final solution. Finally, in Chapter 5, we study to which extend using non stationary strategies can be used to solve general-sum MGs. In this case the size of the cyclic strategy is the only factor that improves the solution. Whilst in MDPs and in zero-sum two player MGs the number of iterations was improving the solution, this is not the case for general-sum MGs.

The third part of this dissertation (Part III) studies a Bellman residual approach to learn from batch data. In zero-sum two-player MGs (Chapter 6) we draw connections between an existing and unstable algorithm LSPI and Newton's method on the Bellman residual. We leverage this connection to improve these algorithms and provide two stable and efficient algorithms. Then, in Chapter 7, we propose a Bellman residual approach to learn stationary Nash equilibria from batch data in general-sum MGs (whilst Chapter 5 only provides a method to learn a non-stationary strategy).

Although this thesis mainly treat the batch scenario, we also are interested in how agents can reach an equilibrium while acting independently and online. Finding independent learning algorithms for cooperative and competitive scenarios is usually treated as a separate agenda in multi-agent RL and we propose in Part IV a unified learning rule that applies in both cooperative multi-stage games and in competitive multi-stage games.

2 Contributions

The mathematical model we consider to study multi-agent interaction is a Markov game (or stochastic game). It generalizes MDPs to multi-agent systems. A Markov Game (MG) is a temporally and spatially extended model of interaction between N players. It is composed of a state space S which contains all the states s in which agents can be. In every state s, players simultaneously take an action a^i in a set $\mathcal{A}^i(s)$. As a result of this joint action $(a^1, \dots, a^N)$, each player receives a reward $r^i(s, a^1, \dots, a^N)$ and the system moves to the following state s' with a probability defined by the transition kernel $p(s'|s, a^1, \dots, a^N)$. Usually in these settings, each player's goal is to find a strategy $\pi^i(.|s)$ to maximize a long term objective. The long term objective studied in this thesis is the expected sum of γ discounted rewards ($\gamma \in [0, 1[$). Reinforcement learning is usually studied with a single agent using MDPs as a model while the canonical model used in standard game theory focuses on normal form games which is an MG with a single state. An-other common model studied in game theory are tree structured games named here multi-stage games. Multi-stage games are MGs with a probability transition function that forbids to visiting the same state twice. As an example, an MDP can model single agent problems such as Atari games, normal form games can model games such as rock-paper-scissors or the prisoner dilemma, and a multi-stage game can model the game of Alesia (also known as Oshi-zumo).

The rest of this section sums up the contributions of this thesis. Most of this work is published or under review.

Approximate Dynamic Programming for Zero-Sum Two-Player Markov Games: Chapter 3 provides an analysis of error propagation in Approximate Dynamic Programming applied to zero-sum two-player Stochastic Games. We provide a novel and unified error propagation analysis in L_p-norm of three well-known algorithms adapted to Stochastic Games (namely Approximate Value Iteration, Approximate Policy Iteration and Approximate Generalized Policy Iteration). We analyse it's sensitivity to value function approximation error and greedy error on the final solution. In addition, we provide a practical algorithm (AGPI-Q) to solve infinite horizon γ-discounted two-player zero-sum Stochastic Games in a batch setting. This is an extension of the Fitted-Q algorithm (which solves Markov Decisions Processes from data) and can be non-parametric. Finally, we demonstrate experimentally the performance of AGPI-Q on a simultaneous two-player game, namely Alesia.

Pérolat, J. Scherrer, B. Piot, B. Pietquin, O, Approximate Dynamic Programming for Two-Player Zero-Sum Markov Games. *In Proceedings of ICML (2015)*

Improved Error Propagation Bounds Using Non-Stationary Strategies: Chapter 4 extends several non-stationary Reinforcement Learning (RL) algorithms and their theoretical guarantees to the case of γ-discounted zero-sum Markov Games (MGs). As in the case of Markov Decision Processes (MDPs), non-stationary algo-

rithms are shown to exhibit better performance bounds compared to their stationary counterparts. This chapter empirically demonstrates, on generic MGs (called Garnets), that non-stationary algorithms outperform their stationary counterparts. In addition, we show that performance mostly depends on the nature of the propagation error. Indeed, algorithms where the error is due to the evaluation of a best-response are penalized (such as policy iteration like algorithms) compared to those suffering from a regression error (such as value iteration like algorithms) even if they exhibit better concentrability coefficients and dependencies on γ.

Pérolat, J. Piot, B. Scherrer, B. Pietquin, O, On the Use of Non-Stationary Strategies for Solving Two-Player Zero-Sum Markov Games. *In Proceedings of AISTATS (2016)*

Non-Stationary Strategies in general sum Markov Games: In Chapter 5 we analyse the use of non-stationary strategies in general sum MGs. The value iteration algorithm is known, in the context of Markov Decision Processes (MDPs) and zero-sum two-player MGs, to iteratively build a sequence of stationary strategies converging toward an optimal one. Usually, only the last strategy is used to behave by the agents. However, in general-sum MGs, running this stationary strategy does not guarantee convergence to a Nash equilibrium. We show that, if one runs the m last strategies in a cycle, there is convergence to an ϵ-Nash equilibrium where ϵ depends on the size of the cycle. This bound compares poorly with the MDPs case and the zero-sum two-player MG case since it does not improve with the number of iterations of the algorithm. This result is tight up to a constant and thus can't be improved.

Pérolat, J. Piot, B. Pietquin, O., A Study of Value Iteration with Non-Stationary Strategies in General Sum Markov Games. *Learning, Inference and Control of Multi-Agent Systems. NIPS workshop (2016)*

Bellman Residual in Zero-Sum Two-Player Markov Games: Chapter 6 reports theoretical and empirical investigations on the use of quasi-Newton methods on two kinds of Bellman residual. First, we demonstrate that existing approximate policy iteration algorithms for MGs and MDPs are Newton's method on different kinds of Bellman residual. Consequently, new algorithms are proposed, making use of quasi-Newton methods to minimize the Optimal Bellman Residual (OBR) and the Projected OBR (POBR) so as to benefit from enhanced empirical performances at low cost. Indeed, using a quasi-Newton method approach introduces slight modifications in terms of coding on those policy iteration algorithms and improves significantly both their stability and their performance. These phenomena are illustrated in an experiment conducted on artificially constructed games called Garnets.

Pérolat, J. Piot, B. Geist, M. Scherrer, B. Pietquin, O., Softened Approximate Policy Iteration for Markov Games. *In Proceedings of ICML (2016)*

Learning Nash Equilibrium in Markov Games: Chapter 7 reports a method to learn a Nash equilibrium in γ-discounted multiplayer general-sum Markov Games (MGs) in a batch setting. We introduce a new definition of the ϵ-Nash equilibrium in MGs which grasps the strategy's quality for multiplayer games. We prove that minimizing the norm of two Bellman-like residuals implies learning such an ϵ-Nash equilibrium. Then, we show that minimizing an empirical estimate of the $\mathcal{L}_p$ norm of these Bellman-like residuals allows learning for general-sum games within the batch setting. Finally, we introduce a neural network architecture that successfully learns a Nash equilibrium in generic multiplayer general-sum turn-based MGs. This work was done with Florian Strub who focused on the empirical evaluation of the method.

Pérolat, J. Strub, F. Piot, B. Pietquin, O.,Learning Nash Equilibrium for General-Sum Markov Games from Batch Data. *In Proceedings of AISTATS (2017)*

Actor-Critic Fictitious Play: Finally, Chapter 8 makes contributions to independent learning in games. We propose an actor-critic algorithm based on the Fictitious-play process. Our work defines a novel actor-critic method that provably converges in both zero-sum two-player and cooperative multi-stage games. This work is currently under review.

Chapter 2

Background and Related Work

Markov Games (MGs) introduced by Shapley (Shapley, 1953) are a generalization of Markov Decision Processes (MDPs) introduced by Bellman (Bellman, 1957). As many of the algorithms employed to solve MGs are either a generalization of algorithms for MDPs or rely on algorithms for MDPs, the first section (Section 1) of this chapter will attempt to sum up existing work in the reinforcement learning community. This part will mostly focus on batch and on online learning algorithms. Then, in Section 2, normal-form games are described. This section introduces basic techniques to solve zero-sum two-player normal-form games which will be used as a subroutine in some algorithms for zero-sum two-player MGs. In Section 3, we introduce general-sum MGs. This section will emphasize the special case of zero-sum two-player MGs and introduce specific notation as it is the focus of Chapters 3, 4 and 6.

1 Markov Decision Processes

A Markov Decision Process (MDP) is the canonical model of sequential decision making under uncertainty as it is a temporally and spatially extended single-agent process. The situation is the following, an agent lies in a state space S (here we only consider discrete state spaces). At time t, the agent is in state $s = s_t$ and has to choose an action $a = a_t$ in the finite set A of possible actions in state s. As a result, the agent receives a reward $r_t = r(s_t, a_t)$ and the state moves to state s_{t+1} with the probability $p(.|s_t, a_t)$. This model is a reductionist interaction model in multiple ways: first it assumes that the environment is stationary (*i.e.* the reward function $r(s, a)$ and the transition kernel $p(.|s, a)$ do not depend on the time t), second it assumes that no information is hidden from the agent (*i.e.* the agent completely knows the state of the environment). Finally, $\gamma \in [0, 1[$ is the discount factor of the MDP, it controls the horizon of the agent $\frac{1}{1-\gamma}$ (*i.e.* which is the horizon at which the agent will attempt to accumulate the maximum sum of rewards). In the end, an MDP M is fully represented by a tuple $< S, A, r, p, \gamma >$.

In such an environment, the goal of each agent is to find a policy. This is a decision rule an agent will apply in an MDP. This rule may depend on time, on the history of states, on the history of actions and on the history of rewards. It might be deterministic or stochastic. In an MDP, since the dynamics and the reward are Markovian (*i.e.* they only depend on the current state and action), the policy does not need to be history dependent (Puterman, 1994). In technical terms, a policy $\{\pi_t(.|s)\}_{t\in\mathbb{N}}$ (here a non-stationary policy) is (for a fixed t) a mapping from a state to a distribution on the action space $S \to \Delta A$ (where ΔA is the set of distributions over A and $\mathbb{N}$ is the set of positive integers $\{0, 1, 2, \ldots\}$). The situation is the following: the process starts at

state s_0 and the agent chooses an action a_0 with probability $\pi_0(.|s_0)$. The agent receives a reward $r_0 = r(s_0, a_0)$ and the environment moves to a next state s_1. This process is repeated sequentially and produces a sequence of states $s_0, s_1, s_2, \ldots, s_t, \ldots$, a sequence of actions $a_0, a_1, a_2, \ldots, a_t, \ldots$ and a sequence of rewards $r_0, r_1, r_2, \ldots, r_t, \ldots$. The goal of the agent is to cumulate a maximum amount of rewards over time. The criterion we will study in this dissertation is the expected γ-discounted sum of rewards written as $v_{\{\pi_t\}_{t\in\mathbb{N}}}(s)$ where $\{\pi_t\}_{t\in\mathbb{N}}$ is the policy of the agent (in the following expression $a \sim \pi$ means that a is a random variable that behaves according to the law π).

$$v_{\{\pi_t\}_{t\in\mathbb{N}}}(s) = E\left[\sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t) | s_0 = s,\ a_t \sim \pi_t(.|s_t),\ s_{t+1} \sim p(.|s_t, a_t)\right]. \tag{2.1}$$

The goal of an agent is to find a policy $\{\pi_t\}_{t\in\mathbb{N}}$ that maximizes the value function $v_{\{\pi_t\}_{t\in\mathbb{N}}}$. A policy that satisfies that condition is said to be optimal. The set of optimal policies is not empty and contains at least one stationary policy (*i.e.*, that does not depend on time) (Puterman, 1994). Usually, the purpose of an agent is to find a stationary policy π since it is easier to store a time independent policy rather than a non-stationary one. However, it might be easier to find a good policy in the set of non-stationary policies rather than in the set of stationary ones (Scherrer and Lesner, 2012). In the case of a stationary policy, the value function is defined as follows:

$$v_\pi(s) = E\left[\sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t) | s_0 = s,\ a_t \sim \pi(.|s_t),\ s_{t+1} \sim p(.|s_t, a_t)\right]. \tag{2.2}$$

The value function achieved by an optimal policy is called the optimal value function and is defined as follows:

$$v^*(s) = \max_{\{\pi_t\}_{t\in\mathbb{N}}} v_{\{\pi_t\}_{t\in\mathbb{N}}}(s) = \max_\pi v_\pi(s). \tag{2.3}$$

In addition to the value function, one usually defines the state-action value function or Q-function written $Q_\pi(s, a)$ for a stationary policy π. The Q-function $Q_\pi(s, a)$ corresponds to the expected γ-discounted return starting from state s if the agent takes a as the first action and then selects his next action according to the policy π.

$$Q_\pi(s, a) = E\left[\sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t) | s_0 = s,\ a_0 = a,\ a_t \sim \pi(.|s_t),\ s_{t+1} \sim p(.|s_t, a_t)\right], \tag{2.4}$$

$$= r(s, a) + \gamma \sum_{s'\in S} p(s'|s, a) v_\pi(s'). \tag{2.5}$$

One can also define an optimal state-action value function as follow:

$$Q^*(s, a) = \max_\pi Q_\pi(s, a), \tag{2.6}$$

$$= r(s, a) + \gamma \sum_{s'\in S} p(s'|s, a) v^*(s'). \tag{2.7}$$

Note that several policies might achieve such an optimal value function or Q-function. We will often write an optimal policy π^*.

Stochastic Matrices and Linear Algebra: To ease the notations and to simplify the proofs, we will see the value function as a vector of $\mathbb{R}^{|S|}$ in the canonical basis $\{\mathbf{1}_s\}_{s\in S}$ ($\mathbb{R}$ is the set of real numbers, $|X|$ is the cardinal of the set X and $\mathbf{1}_s$ is the function such that $\mathbf{1}_s(s) = 1$ and $\forall \tilde{s} \neq s,\ \mathbf{1}_s(\tilde{s}) = 0$). These notations will be introduced using a linear kernel for the sake of generality. Obviously, the vector $v_\pi = \sum_{s\in S} \mathbf{1}_s v_\pi(s)$. In this vector space, the vector $r_\pi = \sum_{s\in S} \mathbf{1}_s E_{a\sim\pi(.|s)}\left[r(s,a)\right]$. The probability transition kernel P_π is the stochastic matrix that defines the following linear transformation:

$$P_\pi v = \sum_{s\in S} \mathbf{1}_s \sum_{s'\in S} E_{a\sim\pi(.|s)}\left[p(s'|s,a)\right] v(s'). \tag{2.8}$$

Furthermore, we will define $r_a = \sum_{s\in S} \mathbf{1}_s r(s,a)$ and $P_a v = \sum_{s\in S} \mathbf{1}_s \sum_{s'\in S} p(s'|s,a)v(s')$ and $p_\pi(s'|s) = E_{a\sim\pi(.|s)}\left[p(s'|s,a)\right]$. With these notations we can define the value function as follows:

$$v_\pi = \left(I - \gamma P_\pi\right)^{-1} r_\pi, \text{ where } I \text{ is the identity application.} \tag{2.9}$$

This property is a direct consequence of eq. 2.13 defined below.

Bellman Operator: The key tools of dynamic programming are the so called Bellman operators. The first Bellman operator depends on a policy π and can be thought of as the solution to a one-step evaluation problem. Its application on the value v produces the value function $[\mathcal{T}_\pi v]$ which is the expected return of the following game: the agent chooses an action a with probability $\pi(.|s)$ and receives the random pay-off $r(s,a) + \gamma v(s')$ (where $s' \sim p(.|s,a)$). It is a linear operator and is defined as follows:

$$\left[\mathcal{T}_\pi v\right](s) = r_\pi(s) + \gamma \sum_{s'\in S} p_\pi(s'|s)v(s'). \tag{2.10}$$

Since this operator is linear in the value function, it can be written with the tools defined in the previous paragraph:

$$\left[\mathcal{T}_\pi v\right] = r_\pi + \gamma P_\pi v. \tag{2.11}$$

The second Bellman operator is non-linear. Its value $[\mathcal{T}^* v]$ for the value function v is the optimal return the agent can get in the game described above. It is often called the optimal Bellman operator and is defined as follows:

$$\left[\mathcal{T}^* v\right](s) = \max_a \left(r(s,a) + \gamma \sum_{s'\in S} p(s'|s,a)v(s') \right). \tag{2.12}$$

These two operators are γ-contractions with respect to the $\mathcal{L}_{+\infty}$-norm. Here the $\mathcal{L}_{+\infty}$-norm of function $f : X \to \mathbb{R}$ is defined as $\|f\|_{+\infty} = \sup_{x\in X} f(x)$. This property of the optimal Bellman operator is leveraged in the value iteration algorithm (see Section 1.1.1). Thus, each of these two operators admits a unique fixed point. The

fixed point of $\mathcal{T}_\pi$ is v_π and the fixed point of $\mathcal{T}^*$ is v^*. Meaning we have the two following identities:

$$\mathcal{T}_\pi v_\pi = v_\pi, \tag{2.13}$$
$$\mathcal{T}^* v^* = v^*. \tag{2.14}$$

In addition to these two Bellman operators, we will define $[\mathcal{T}_a v](s) = r(s,a) + \gamma \sum_{s \in S} p(s'|s,a)v(s')$. From this definition, we can redefine the Q-function as $Q_\pi(s,a) = \mathcal{T}_a v_\pi(s)$ and the optimal Q-function as $Q^*(s,a) = \mathcal{T}_a v^*(s)$.

Respectively, one can define operators on Q-functions as follow:

$$[\mathcal{B}_\pi Q](s,a) = r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) E_{a \sim \pi(.|s')} [Q(s',a)], \tag{2.15}$$
$$[\mathcal{B}^* Q](s,a) = r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) \max_a [Q(s',a)]. \tag{2.16}$$

These operators are also γ-contractions and their respective fixed points are Q_π (for operator 2.15) and Q^* (for 2.16). The linear Bellman operator on a Q-function can be expressed using linear transformations. Let's write the state-action transition kernel:

$$\mathcal{P}_\pi Q = \sum_{s,a \in S \times A} \mathbf{1}_{s,a} \sum_{s',b \in S \times A} \pi(b|s') p(s'|s,a) Q(s',b). \tag{2.17}$$

Then, the linear Bellman operator of 2.15 can be expressed as follow:

$$\mathcal{B}_\pi Q = r + \gamma \mathcal{P}_\pi Q. \tag{2.18}$$

Greedy Policy: When given a value function v_π, there is a simple way to retrieve a policy performing at least as well as π. The idea is to act greedily with respect to the value function v_π. A policy π is said to be greedy with respect to a value function v if $\mathcal{T}_\pi v = \mathcal{T}^* v$. In this case, we say that $\pi \in \mathcal{G}(v)$. A policy π is said to be greedy with respect to a Q-function Q if for all s in S the relation holds $E_{a \sim \pi(.|s)} [Q(s,a)] = \max_a Q(s,a)$. Again, we will say that $\pi \in \mathcal{G}(Q)$ if a policy π is greedy with respect to Q. One can prove the two following results:

Theorem 2.1. *A policy π is optimal if and only if it is greedy with respect to it's value v_π.*

$$v_\pi = v^* \text{ if and only if } \pi \in \mathcal{G}(v_\pi). \tag{2.19}$$

This first theorem proves that to find an optimal policy, it is sufficient to find an optimal value. This theorem will be extended for zero-sum two-player Markov-Games but has no extensions to general sum MGs.

Theorem 2.2. *If π' is greedy with respect to the value v_π of policy π, then $v_{\pi'} \geq v_\pi$*

This second theorem proves that a simple way to improve a policy π is to play the greedy policy π' with respect to the value v_π. This theorem will be the basis of the Policy iteration algorithm (Section 1.1.2). These two theorems are standard in ADP and are proven in (Puterman, 1994).

1.1 Exact Algorithms

This section intends to give an overview of the common algorithms used to solve MDPs. The first family of algorithms are dynamic programming algorithms (Section 1.1.1, 1.1.2, 1.1.3), the second directly minimize the Bellman residual (sec 1.1.4).

1.1.1 Value Iteration

The Value Iteration (VI) algorithm (Puterman, 1994) is based on the γ-contraction property of the optimal Bellman operator. It repetitively applies the optimal Bellman Operator on a value function starting from an arbitrary value $v_0 = 0$.

Algorithm 1 Value Iteration

Input: An MDP M, a value $v_0 = 0$ and a maximum number of iterations K.
for k=1,2,...,K-1 **do**
for all s compute $v_k(s) = \max_a r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v_{k-1}(s')$ (or $v_k = \mathcal{T}^* v_{k-1}$)
end for
for all s, a compute $Q_K(s) = r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v_{K-1}(s')$ and find $\pi_K \in \mathcal{G}(Q_K)$ (or find $\pi_K \in \mathcal{G}(v_K)$ where $v_K = \mathcal{T}^* v_{K-1}$)
Output: π_K

The performance of policy π_K can be expressed as a distance between v_π and the value of an optimal strategy v^*. The proof of the following result can be found in (Puterman, 1994):

$$\|v_{\pi_K} - v^*\|_{+\infty} \leq \frac{2\gamma^K}{1-\gamma} \|v^* - v_0\|_{+\infty}. \tag{2.20}$$

This algorithm performs iterations at a low computational cost but is slow to converge (in the number of iterations) compared to Policy Iteration.

1.1.2 Policy Iteration

The second iterative algorithm presented here is the Policy Iteration (PI) algorithm. It starts from an arbitrary value v_0 and proceeds in two steps. At iteration k it finds a greedy policy π_k with respect to the value v_{k-1} and then computes the value of that policy. Finally, it returns π_K after K iterations.

This algorithm produces an increasing sequence of values v_k. This property is implied by theorem 2.2. Furthermore, this algorithm is guaranteed to converge in a finite number of iterations.

Again, one can prove the following result on the performance of policy π_K:

$$\|v_{\pi_K} - v^*\|_{+\infty} \leq \frac{2\gamma^K}{1-\gamma} \|v^* - v_0\|_{+\infty}. \tag{2.21}$$

Algorithm 2 Policy Iteration

Input: An MDP M, a value $v_0 = 0$ and a maximum number of iterations K.
for k=1,2,...,K **do**
  find $\pi_k \in \mathcal{G}(v_{k-1})$
  compute $v_k = v_{\pi_k} = (I - \gamma P_{\pi_k})^{-1} r_{\pi_k}$
end for
Output: π_K

The policy iteration is guaranteed to converge in a finite number of steps and its complexity is polynomial in the size of the state space and of the action space for a fixed γ (Hansen et al., 2013a, Scherrer, 2016). In the end the Policy Iteration algorithm performs updates that are heavier VI's one. This is because the value update is required to solve a linear system. But PI converges faster in terms of the number of iterations compared to VI.

1.1.3 Modified Policy Iteration

Modified Policy Iteration (MPI) is an algorithm that bridges the gap between VI and PI by introducing a parameter m. The parameter m controls the cost of each iteration. This algorithm reduces to VI if $m = 1$ and to PI if $m = +\infty$. Indeed, in Algorithm 3 if $m = +\infty$, the value v_k will be the fixed point of $\mathcal{T}_{\pi_k}$ (*i.e.* the value v_{π_k}). And if $m = 1$, since $\pi_k \in \mathcal{G}(v_{k-1})$ we have $\mathcal{T}^* v_{k-1} = \mathcal{T}_{\pi_k} v_{k-1}$ and finally $v_k = \mathcal{T}_{\pi_k} v_{k-1} = \mathcal{T}^* v_{k-1}$. Again, this algorithm enjoys the same guarantees of convergence as PI and VI and surprisingly the performance policy π_k does not depends on m.

$$\|v_{\pi_K} - v^*\|_{+\infty} \leq \frac{2\gamma^K}{1-\gamma} \|v^* - v_0\|_{+\infty}. \tag{2.22}$$

This algorithm also enjoys a lower bound of complexity which is detailed in (Lesner and Scherrer, 2015) for a generalized algorithm.

Algorithm 3 Modified Policy Iteration

Input: An MDP M, a value $v_0 = 0$, a maximum number of iterations K and a parameter m.
for k=1,2,...,K **do**
  find $\pi_k \in \mathcal{G}(v_{k-1})$
  compute $v_k = (\mathcal{T}_{\pi_k})^m v_{k-1}$
end for
Output: π_K

1.1.4 Minimization of the Bellman Residual

The last method we will present here is the direct minimization of the Bellman residual. Indeed, solving an MDP means finding the fixed point of the operator $\mathcal{T}^*$. This means

that we have to find a value v such that $v = \mathcal{T}^* v$. The idea is thus to minimize the norm of the Bellman residual $v - \mathcal{T}^* v$ with some optimization method (Baird et al., 1995). For example, the minimization of the $\mathcal{L}_2$-norm of the Bellman residual with Newton's method is in fact the policy iteration algorithm Puterman (1994) (the $\mathcal{L}_p$-norm of the function f (written $\|f\|_p$) is $\|f\|_p^p = \sum_{x \in X} f(x)^p$).

1.2 Batch Algorithms

The previous section (Section 1.1) detailed several algorithms to exactly solve MDPs. These algorithms are based on the value function and proceed iteratively in two steps. The first step is a greedy step and requires knowing the model of the MDP (since for a given value v we need to find a policy π such that $\mathcal{T}^* v = \mathcal{T}_\pi v$). This section is devoted to batch algorithms and thus we will need to perform this step without knowledge of the model. That is why all the algorithms presented in this section perform iterations on the Q-function instead of the value function. The second step in these algorithms also requires the model and performs a linear transformation on the current value v. Again, since we don't have access to the model, we will replace that step with a supervised learning step. The underlying idea is that unbiased estimators of the Bellman operator can be built with samples. The algorithms introduced in Section 1.2.1 are an adaptation of the value iteration algorithm for the batch setting. In Section 1.2.2 we will present an adaptation of policy iteration and in Section 1.2.3 we will show how errors are propagated through iterations of modified policy iteration. Finally, we will show how to adapt the Bellman residual minimization approach to the Batch setting.

Remark 2.1. An estimator $\hat{\theta}_n$ of the quantity θ is said to be unbiased if $E\left[\hat{\theta}_n\right] - \theta = 0$ and it is said to be consistent if $\lim_{n \to +\infty} \hat{\theta}_n - \theta = 0$ in probability.

1.2.1 Fitted-Q iteration and Neural Fitted-Q

These algorithms are value iteration like algorithms. Instead of performing an exact update $Q_k = \mathcal{B}^* Q_{k-1}$ as the VI algorithm on Q-function would, it performs an approximate update $Q_k \simeq \mathcal{B}^* Q_{k-1}$. Any batch algorithm requires as inputs a set of data $D_n = \{(s_j, a_j, r_j, s'_j)\}_{j=1,\dots,n}$ where for all j, the reward $r_j = r(s_j, a_j)$ and where $s'_j \sim p(.|s_j, a_j)$. At iteration k, the VI algorithm on Q-functions would perform the following step:

$$\forall s, a \in S \times A,\ Q_k(s,a) = r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) \max_b Q_{k-1}(s', b). \tag{2.23}$$

But since batch algorithms do not have access to the model, they will approximate that step with a regression. First, they will build a dataset $\{(x_j, y_j)\}_{j=1,\dots,n}$ where $x_j = (s_j, a_j)$ and $y_j = r_j + \gamma \max_a Q_{k-1}(s'_j, a)$ and then find a best fit in an hypothesis

space $\mathcal{F}$. This means finding $Q_k \in \mathcal{F}$ such that:

$$Q_k \in \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{j=1}^{n} l(f(x_j), y_j). \tag{2.24}$$

This approach was first studied by Bellman (Bellman et al., 1963) with polynomial approximations. Fitted-Q (Ernst et al., 2005) iteration considers extra-trees for the regression method while Neural fitted-Q (Riedmiller, 2005) considers neural networks. The sample efficiency of this method is analysed in (Antos et al., 2008a).

Algorithm 4 Fitted - Q & Neural fitted-Q

Input: $D_n = \{(s_j, a_j, r_j, s'_j)\}_{j=1,\dots,n}$ some samples, $q_0 = 0$ a Q-function, $\mathcal{F}$ an hypothesis space and a number of iterations K.
for k=1,2,...,K **do**
  for all j **do**
    $q^j = r(s_j, a_j) + \gamma \max_a q_{k-1}(s'_j, a)$
  end for
  $q_k = \operatorname{argmin}_{q \in \mathcal{F}} \sum_{j=1}^{n} l(q(s_j, a_j), q_j)$ where l is a loss function.
end for
Output: q_K

As a remark, Ernst (Ernst et al., 2005) also uses his algorithm for continuous action spaces. The only problem is that it is cumbersome to find the maximum of the Q-function over the actions. Thus, in their experiment they discretize the action space to approximate the maximum step. This example emphasizes the necessity of analysing approximations in the greedy step in approximate dynamic programming algorithms (Section 1.2.3).

1.2.2 Least Squares Policy Iteration and Bellman Residual Minimizing Policy Iteration

The Least Squares Policy Iteration (LSPI) algorithm (Lagoudakis and Parr, 2003) and the Bellman Residual Minimizing Policy Iteration (BRMPI) algorithms are approximate PI algorithms. They will perform two steps as PI, and both approximate the policy evaluation steps of PI. Again, as we work with batch data, LSPI and BRMPI will perform their iterations on the Q-function instead of the value function. At iteration k, the PI algorithm's evaluation step computes the state-action value function $Q_k = Q_{\pi_k}$. In the case of LSPI and BRMPI, the Q-function are lying in a d dimensional linear function space $\mathcal{F}_\Phi = \{\Phi\omega, \omega \in \mathbb{R}^d\}$. The features $\Phi = [\phi_1, \dots, \phi_d]$ are linearly independent functions from $S \times A \to \mathbb{R}$ and can be thought of as vectors of size $|S| \times |A|$ in the vector space of canonical basis $\{\mathbf{1}_{s,a}\}_{s,a \in S \times A}$. The feature Φ can be seen as a matrix of size $(|S| \times |A|) \times d$ and ω a vector of size d. Furthermore, let ρ be a probability distribution over the state-action space $S \times A$ and we will write Δ_ρ the diagonal application that maps $\mathbf{1}_{s,a}$ to $\rho(s,a)\mathbf{1}_{s,a}$. Finally, the orthogonal projection

onto $\mathcal{F}_\Phi$ with respect to the $\mathcal{L}_{\rho,2}$-norm is the application $\Phi(\Phi^\top\Delta_\rho\Phi)^{-1}\Phi^\top\Delta_\rho = \Pi_{\rho,\Phi}$. The $\mathcal{L}_{\rho,p}$ of function f is defined here as $\|f\|_{\rho,p}^p = \sum_{x\in X}\rho(x)f(x)^p$.

Least Squares Policy Iteration: At each iterations for a policy π, the PI algorithm finds Q satisfying $Q = \mathcal{T}_\pi Q$. Instead, LSPI finds ω satisfying the projected fixed point equation:

$$\Phi\omega = \Pi_{\rho,\Phi}\mathcal{T}_\pi\Phi\omega. \tag{2.25}$$

The solution for ω of this equality is given by:

$$\omega = A_{\rho,\Phi,\pi}^{-1}b_{\rho,\Phi}, \tag{2.26}$$
$$\text{where } A_{\rho,\Phi,\pi} = \Phi^\top\Delta_\rho(\Phi - \gamma\mathcal{P}_\pi\Phi), \tag{2.27}$$
$$\text{and } b_{\rho,\Phi} = \Phi\Delta_\rho r. \tag{2.28}$$

Given a batch of data $D_n = \{(s_j, a_j, r_j, s'_j)\}_{j=1,\dots,n}$, the matrix $A_{\rho,\Phi,\pi}$ and the vector $b_{\rho,\Phi}$ can be estimated as follow:

$$\hat{A}_{\Phi,\pi,D_n} = \frac{1}{n}\sum_{j=1}^{n}\Phi(s_j,a_j)\left(\Phi(s_j,a_j) - \gamma\sum_{a\in A}\pi(a|s'_j)\Phi(s'_j,a)\right), \tag{2.29}$$

$$\hat{b}_{\Phi,D_n} = \frac{1}{n}\sum_{j=1}^{n}\Phi(s_j,a_j)r(s_j,a_j). \tag{2.30}$$

If the data in D_n are independent and distributed according the distribution ρ, $\hat{A}_{\Phi,\pi,D_n}$ and $\hat{b}_{\Phi,D_n}$ are consistent estimators of $A_{\rho,\Phi,\pi}$ and $b_{\rho,\Phi}$. Furthermore, $\hat{A}^{-1}_{\Phi,\pi,D_n}$ can be be efficiently computed with the Scherman-Morison formula which provides an optimized version of the algorithm. The algorithm in its most simple version is presented in Algorithm 5.

Algorithm 5 Least Squares Policy Iteration (LSPI)

Input: A batch $D_n = \{(s_j, a_j, r_j, s'_j)\}_{j=1,\dots,n}$, a feature space Φ, an initial parametrization $\omega_0 = 0$ and a number of iterations K.
for k=1,2,...,K **do**
  find $\pi_k \in \mathcal{G}(\Phi\omega_{k-1})$
  compute $\hat{A}_{\Phi,\pi_k,D_n}$ and $\hat{b}_{\Phi,D_n}$ as in formula 2.96 and 2.97.
  compute $\omega_k = \hat{A}^{-1}_{\Phi,\pi_k,D_n}\hat{b}_{\Phi,D_n}$
end for
Output: π_K

Bellman Residual Minimizing Policy Iteration: This second algorithm is less popular that LSPI. It is roughly sketched in (Lagoudakis and Parr, 2003). We provide details here as it will be improved in Chapter 6 and since it illustrates bias issues in the

estimation of the Bellman residual. These bias issues will be encountered in Chapter 6 and in Chapter 7.

Instead of finding the fixed point of the projected Bellman operator to approximate Q_π, BRMPI finds ω that minimizes the Bellman residual $\|\Phi\omega - \mathcal{T}_\pi\Phi\omega\|_{\rho,2}$. The solution to this minimization problem is:

$$\omega = C_{\rho,\Phi,\pi}^{-1} d_{\rho,\Phi}, \tag{2.31}$$

$$\text{where } C_{\rho,\Phi,\pi} = \Phi^\top (\mathcal{I} - \gamma\mathcal{P}_\pi)^\top \Delta_\rho (\mathcal{I} - \gamma\mathcal{P}_\pi)\Phi, \tag{2.32}$$

$$\text{and } b_{\rho,\Phi,\pi} = \Phi^\top (\mathcal{I} - \gamma\mathcal{P}_\pi)^\top \Delta_\rho r. \tag{2.33}$$

This approach can also be implemented with batch data but the resulting solution for this estimation proposed by Lagoudakis and Parr is a biased and inconsistent estimator of $C_{\rho,\Phi,\pi}$. The following estimators $\hat{C}_{\Phi,\pi,D_n}$ and $\hat{b}_{\Phi,\pi,D_n}$ of $C_{\rho,\Phi,\pi}$ and $b_{\rho,\Phi,\pi}$ are:

$$\hat{C}_{\Phi,\pi,D_n} = \frac{1}{n}\sum_{j=1}^{n}\left(\Phi(s_j,a_j) - \gamma\sum_{a\in A}\pi(a|s'_j)\Phi(s'_j,a)\right)\left(\Phi(s_j,a_j) - \gamma\sum_{a\in A}\pi(a|s'_j)\Phi(s'_j,a)\right)^\top, \tag{2.34}$$

$$\hat{d}_{\Phi,\pi,D_n} = \frac{1}{n}\sum_{j=1}^{n}\left(\Phi(s_j,a_j) - \gamma\sum_{a\in A}\pi(a|s'_j)\Phi(s'_j,a)\right)^\top r(s_j,a_j). \tag{2.35}$$

Algorithm 6 Bellman Residual Minimizing Policy Iteration (BRMPI)

Input: A batch $D_n = \{(s_j,a_j,r_j,s'_j)\}_{j=1,\dots,n}$, a feature space Φ, an initial parametrization $\omega_0 = 0$ and a number of iterations K.
for k=1,2,...,K **do**
  find $\pi_k \in \mathcal{G}(\Phi\omega_{k-1})$
  compute $\hat{A}_{\Phi,\pi_k,D_n}$ and $\hat{b}_{\Phi,D_n}$ as in formula 2.34 and 2.35.
  compute $\omega_k = \hat{A}^{-1}_{\Phi,\pi_k,D_n}\hat{b}_{\Phi,D_n}$
end for
Output: π_K

As described in (Lagoudakis and Parr, 2003) the BRMPI algorithm would be as in Algorithm 6. The bias of the estimator can be solved with a generative model of the MDP. If, for each sample of the bacth, the next state can be sampled two times independently (*i.e.* the batch would be $D_n = \{(s_j,a_j,r_j,s'_j,s''_j)\}_{j=1,\dots,n}$ where $s''_j \sim p(.|s^j,a^j)$), one can get the following unbiased estimator from the batch:

$$\hat{C}_{\Phi,\pi,D_n} = \frac{1}{n}\sum_{j=1}^{n}\left(\Phi(s_j,a_j) - \gamma\sum_{a\in A}\pi(a|s'_j)\Phi(s'_j,a)\right)\left(\Phi(s_j,a_j) - \gamma\sum_{a\in A}\pi(a|s'_j)\Phi(s''_j,a)\right)^\top. \tag{2.36}$$

This presentation of LSPI and BRMPI sums up useful background for this dissertation. These algorithms have been extensively studied in the literature. Although LSPI

is a batch algorithm, it has been adapted to the online setting (Buşoniu et al., 2010, Li et al., 2009). The sample complexity of LSPI has been analysed by Lazaric et al. (2012) and the one of BRMPI by Antos et al. (2008b). The sample-complexity of policy evaluation through Bellman residual minimization has been analysed in (Maillard et al., 2010).

1.2.3 Approximate Modified Policy Iteration

In the two previous sections (Section 1.2.1 and 1.2.2) we introduced approximations of VI and PI. These approximations are made in the evaluation step (*i.e.* $v_k = \mathcal{T}_{\pi_k} v_{k-1}$ for VI and $v_k = (\mathcal{T}_{\pi_k})^{+\infty} v_{k-1}$ for PI). In this section, we present results that shed light on the robustness of these algorithms to errors. Imagine that instead of performing the iteration $v_k = \mathcal{T}_{\pi_k} v_{k-1}$ (where $\pi_k \in \mathcal{G}(v_{k-1})$), the update of v_k is made up to an error ϵ_k such that $v_k = \mathcal{T}_{\pi_k} v_{k-1} + \epsilon_k$. But approximations can also occur in the greedy step (Gabillon et al., 2011, Scherrer et al., 2012) as discussed in Section 1.2.1 and, instead of taking a greedy action, some algorithms select suboptimal actions in some states. We shall write $\pi \in \hat{\mathcal{G}}_{\epsilon'}(v)$ if $\mathcal{T}^* v \leq \mathcal{T}_\pi v + \epsilon'$. In this section we provide the state of the art sensitivity analysis of Approximate Modified Policy Iteration (AMPI) like algorithms. The generic scheme studied is described in Algorithm 7 and applies to all algorithms described in Section 1.2.1 and in Section 1.2.2.

Algorithm 7 Approximate Modified Policy Iteration

Input: An MDP M, a value $v_0 = 0$, a maximum number of iterations K and a parameter m.
for k=1,2,...,K **do**
  find $\pi_k \in \hat{\mathcal{G}}_{\epsilon'_k}(v_{k-1})$
  compute $v_k = (\mathcal{T}_{\pi_k})^m v_{k-1} + \epsilon_k$
end for
Output: π_K

The following analysis is standard in approximate dynamic programming algorithms. As this kind of analysis will be done for several algorithms in Part II we detail here the case of AMPI of Scherrer et al. (2012). These bounds are decomposed in a sum of 3 terms: the value update error, the greedy error and a concentration term. Each of these terms is decomposed in a product of 3 quantities. All these terms can be controlled: the first one can be controlled by the accuracy of the value function update, the second one can be controlled making small approximation on the greedy step and the last one by the number of iterations of the algorithm. These analysis were performed in $\mathcal{L}_\infty$-norm and then were performed in $\mathcal{L}_p$-norm (Farahmand et al., 2010, Antos et al., 2008a, Munos, 2007).

The performance of AMPI after k-iterations measures in $\mathcal{L}_{\rho,p}$-norm the distance between v_{π_k} and v^*. The measure ρ specifies the distribution over the state space on which the performance needs to be accurate. Thus, we will bound the norm $\|v^* - v_{\pi_k}\|_{\rho,p}$. Since our method performs approximations at each iteration, our bound will

depend on $\epsilon_1, \dots, \epsilon_{k-1}$ and $\epsilon'_1, \dots, \epsilon'_k$. These errors are supposed to be controlled in $\mathcal{L}_{\sigma,pq'}$-norm where σ is the distribution on which the estimation is supposed to be accurate. For instance in Fitted-Q iteration, $\epsilon_j,\ j \leq k$ the error of the regression at iteration j. In that case, σ is the distribution of the data and $d = 2$ (since the $\mathcal{L}_2$ loss is often used for trees). Theorem 2.3 shows the impact of the approximation on the final solution.

Theorem 2.3. *Let ρ and σ be distributions over states. Let p, q and q' be such that $\frac{1}{q} + \frac{1}{q'} = 1$. Then, after k iterations, we have:*

$$\|v^* - v_{\pi_k}\|_{\rho,p} \leq \underbrace{\frac{2(\gamma - \gamma^k)(\mathcal{C}_q^{1,k,0})^{\frac{1}{p}}}{(1-\gamma)^2} \sup_{1 \leq j \leq k-1} \|\epsilon_j\|_{\sigma,pq'}}_{\text{value update error}} + \underbrace{\frac{(1-\gamma^k)(\mathcal{C}_q^{0,k,0})^{\frac{1}{p}}}{(1-\gamma)^2} \sup_{1 \leq j \leq k} \|\epsilon'_j\|_{\sigma,pq'}}_{\text{greedy error}}, \tag{2.37}$$

$$+ \underbrace{\frac{2\gamma^k}{1-\gamma}(\mathcal{C}_q^{k,k+1,0})^{\frac{1}{p}} \min\left(\|v^* - v_0\|_{\sigma,pq'}, \|v_0 - \mathcal{T}_{\pi_1} v_0\|_{\sigma,pq'}\right)}_{\text{contraction term}}. \tag{2.38}$$

where

$$\mathcal{C}_q^{l,k,d} = \frac{(1-\gamma)^2}{\gamma^l - \gamma^k} \sum_{i=l}^{k-1} \sum_{j=i}^{\infty} \gamma^j c_q(j+d), \tag{2.39}$$

with the following norm of a Radon-Nikodym derivative:

$$c_q(j) = \sup_{\pi_1,\dots,\pi_j} \left\| \frac{d(\rho P_{\pi_1} \dots P_{\pi_j})}{d\sigma} \right\|_{q,\sigma}. \tag{2.40}$$

The contribution of the value update error can be divided in three parts:

$$\underbrace{\frac{2(\gamma - \gamma^k)(\mathcal{C}_q^{1,k,0})^{\frac{1}{p}}}{(1-\gamma)^2} \sup_{1 \leq j \leq k-1} \|\epsilon_j\|_{\sigma,pq'}}_{\text{value update error}} = \underbrace{\frac{2(\gamma - \gamma^k)}{(1-\gamma)^2}}_{\gamma\text{-sensitivity}} \underbrace{(\mathcal{C}_q^{1,k,0})^{\frac{1}{p}}}_{\text{concentrability coeficient}} \underbrace{\sup_{1 \leq j \leq k-1} \|\epsilon_j\|_{\sigma,pq'}}_{\epsilon\text{-error}}. \tag{2.41}$$

The sensitivity to γ is of the order of the square of the horizon $\frac{1}{1-\gamma}$. The sensitivity to the error ϵ depends on the maximum of the norm of the errors. The concentrability coefficient $(\mathcal{C}_q^{1,k,0})^{\frac{1}{p}}$ measures the impact of controlling the error with respect to measure σ and guaranteeing a performance on measure ρ. It is a discounted sum of the norm of the Radon-Nikodym derivative $c_q(j)$ (in discrete actions spaces, this derivative is the ratio of $\rho P_{\pi_1} \dots P_{\pi_j}$ and σ). The larger j is the more $c_q(j)$ is discounted in the coefficient $\mathcal{C}_q^{l,k,d}$. Indeed, if we only have access to samples to control the regression error of a part of the state-action space that is never visited from the part of the state-action space on which we want to give guarantees, then the concentrability coefficient will be infinite. As an example, the concentrability coefficient will always be finite if the measure σ is

uniform. The reader can find in (Scherrer, 2014) a detailed and meticulous comparison of the different concentrability coefficients.

The greedy error term can be decomposed in the same way and is comparable to the value update error term:

$$\underbrace{\frac{(1-\gamma^k)(\mathcal{C}_q^{0,k,0})^{\frac{1}{p}}}{(1-\gamma)^2}\sup_{1\leq j\leq k}\|\epsilon'_j\|_{\sigma,pq'}}_{\text{greedy error}} = \underbrace{\frac{(1-\gamma^k)}{(1-\gamma)^2}}_{\gamma\text{-sensitivity}}\underbrace{(\mathcal{C}_q^{0,k,0})^{\frac{1}{p}}}_{\text{concentrability coeficient}}\underbrace{\sup_{1\leq j\leq k}\|\epsilon'_j\|_{\sigma,pq'}}_{\epsilon\text{-error}}. \tag{2.42}$$

Finally, the last term of that sum is the contraction term and can be decomposed in three terms:

$$\underbrace{\frac{2\gamma^k}{1-\gamma}(\mathcal{C}_q^{k,k+1,0})^{\frac{1}{p}}\min\left(\|v^*-v_0\|_{\sigma,pq'},\|v_0-\mathcal{T}_{\pi_1}v_0\|_{\sigma,pq'}\right)}_{\text{contraction term}} = \tag{2.43}$$

$$\underbrace{\frac{2\gamma^k}{1-\gamma}}_{\gamma\text{-contraction}}\underbrace{(\mathcal{C}_q^{k,k+1,0})^{\frac{1}{p}}}_{\text{concentrability coeficient}}\underbrace{\min\left(\|v^*-v_0\|_{\sigma,pq'},\|v_0-\mathcal{T}_{\pi_1}v_0\|_{\sigma,pq'}\right)}_{\text{error initial value}}. \tag{2.44}$$

This last term converges to zero at a geometrical rate (since the γ-contraction term is $O(\gamma^k)$). It also depends on a concentrability coefficient $(\mathcal{C}_q^{k,k+1,0})^{\frac{1}{p}}$ and on a term that reflects how close the initial solution v_0 is from the optimal solution v^*.

Non-Stationary Policies in MDPs: As described in this section, the sensitivity to γ is one of the strongest drawbacks of AMPI algorithms. One way to reduce this dependency is to use a non-stationary policy. In (Scherrer and Lesner, 2012), Scherrer shows that using cyclic policies reduces the dependency on γ. In this paper, they prove that playing the m last policies given by approximate value iterations (instead of playing π_K at each time step, they play the cyclical policy $\pi_K, \pi_{K-1}, \dots, \pi_{K-m+1}, \pi_K, \pi_{K-1}, \dots$). In the case of cyclical policies, the γ-sensitivity of approximate value iteration is $\frac{2(\gamma-\gamma^k)}{(1-\gamma)(1-\gamma^m)}$ instead of $\frac{2(\gamma-\gamma^k)}{(1-\gamma)^2}$ for the value update error and $\frac{1-\gamma^k}{(1-\gamma)(1-\gamma^m)}$ instead of $\frac{1-\gamma^k}{(1-\gamma)^2}$ for the greedy error term. The AMPI algorithm was adapted to use non-stationary strategies and analysed in (Lesner and Scherrer, 2015).

1.2.4 Bellman Residual Minimization

This section on Bellman residual minimization introduces an approach existing in MDPs that we will study in MGs (part III).

Finding an optimal policy can be achieved in MDPs if one finds the optimal Q-function Q^*. This can be achieved by the minimization of the Optimal Bellman Residual (OBR) $\|Q-\mathcal{B}^*Q\|_{\rho,2}$. If π is greedy with respect to any Q the following bounds on $\|Q^*-Q_\pi\|_{\rho,2}$ stands (Piot et al., 2014a)

$$\|Q^*-Q_\pi\|_{\rho,2} \leq \frac{2}{1-\gamma}\left(\frac{C_2(\rho,\pi)+C_2(\rho,\pi^*)}{2}\right)^{\frac{1}{2}}\|\mathcal{B}^*Q-Q\|_{\rho,2}. \tag{2.45}$$

with $C_2(\rho,\pi) = ||\frac{\partial \rho^\top (1-\gamma)(I-\gamma P_\pi)^{-1}}{\partial \rho^\top}||_{2,\rho}$ the Radon-Nikodym derivative of the Kernel $(1-\gamma)(I-\gamma P_\pi)^{-1}$.

The optimal Bellman residual minimization approach is to use a batch of data $D_n = \{(s_j, a_j, r_j, s'_j)\}_{j=1,\dots,n}$ to estimate the OBR. In (Piot et al., 2014a), the following estimator is proposed:

$$\hat{J}_{\text{OBR}}(Q) = \sum_{j=1}^{n} \left(Q(s_j, a_j) - r_j - \gamma \max_a Q(s'_j, a) \right)^2. \tag{2.46}$$

The goal is now to minimize a loss where $Q \in \mathcal{F}$ with any relevant optimization technique:

$$Q = \operatorname*{argmin}_{Q \in \mathcal{F}} \hat{J}_{\text{OBR}}(Q). \tag{2.47}$$

This estimator is biased and not consistent (Piot et al., 2014b,a) in the case of a stochastic dynamic. Many techniques have been developed to improve the estimation of the OBR including (Grunewalder et al., 2012, Taylor and Parr, 2012, Maillard et al., 2010).

1.3 Online Learning in MDPs

The last part of this dissertation will study independent learning in multi-stage games. As independent learning is a generalization to MGs of online learning in MDPs, we briefly introduce here two basic algorithms.

Whilst batch algorithms were designed to learn a near optimal behaviour without being able to access the environment, online algorithms were designed to learn a policy in interaction with the environment. At time t, the agent is in state s_t and takes an action a_t. With the reward r_t collected, the agent will perform an update.

1.3.1 Q-learning

The Q-learning algorithm (Watkins and Dayan, 1992) learns the optimal Q-function. This algorithm is an off-policy algorithm (there is a fixed policy used to behave which is different from the learnt policy). At each step the algorithm updates the Q-function in order to estimate the optimal Q-function Q^*. For the on-policy version of the Q-learning algorithm, the policy to select the action a_t at time t is usually an ϵ-greedy policy with respect to the current estimate of the Q-function.

Algorithm 8 Q-learning

Input: An MDP M, a Q-value $Q_0 = 0$, an initial state s_0 and a number of iterations T.
for t=0,1,2,...,T-1 **do**
take action a_t according to an exploration policy (usually ϵ greedy policy),
collect reward $r_t \sim r(s_t, a_t)$ and move to state $s_{t+1} \sim p(.|s_t, a_t)$,
update Q_{t+1}: $Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \right)$
end for
Output: Q_T

1.3.2 SARSA

The SARSA algorithm will instead update the value according to the so called "Temporal difference" error. For a given behavioural policy π, the Q-learning algorithm will learn the optimal Q-function whilst SARSA will learn the Q-function associated with the behaved policy Q_π.

Algorithm 9 SARSA

Input: An MDP M, a Q-value $Q_0 = 0$, an initial state s_0 and a number of iterations T.
for t=1,2,...,T **do**
take action a_t according to an exploration policy (usually the ϵ greedy policy),
collect reward $r_t \sim r(s_t, a_t)$ and move to state $s_{t+1} \sim p(.|s_t, a_t)$,
$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \left(r_t + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \right)$
end for
Output: Q_T

2 Normal-Form Games

Before introducing Markov Games (MGs), we will first shortly introduce Normal Form Games (NFGs) which is a stateless MG. In MDPs or in single agent problems, there is only a single way to define an optimal policy. In a multi-agent system, many solutions can be studied. In this section, we will introduce two of them: the minimax solution and the Nash equilibrium. These two notions will be generalized in the next section to Markov games.

In an N-player NFG, each player chooses actions a^i in a finite set A^i of available actions. Each player receives an individual reward $r^i(a^1, \dots, a^N) \in \mathbb{R}$. For the sake of simplicity, we will write the joint action $\boldsymbol{a} = (a^1, \dots, a^N) = (a^i, \boldsymbol{a^{-i}})$ where $\boldsymbol{a^{-i}}$ is the joint action of every player except player i. The goal of each player is to find a strategy $\pi^i \in \Delta(A^i)$. If every player plays his strategy, player i receives the average reward $E_{\boldsymbol{a} \sim \boldsymbol{\pi}}[r^i(\boldsymbol{a})]$. Again, the joint policy of all players is $\boldsymbol{\pi}$ and the joint policy

of every player except player i is $\boldsymbol{\pi}^{-i}$. Finally, an NFG can be represented as a tuple $< \{A^i\}_{i\in\{1,\dots,N\}}, \{r^i(\boldsymbol{a})\}_{i\in\{1,\dots,N\}} >$.

In an NFG, the equivalent of a value function for player i would be $r^i_{\boldsymbol{\pi}} = E_{\boldsymbol{a}\sim\boldsymbol{\pi}}\left[r^i(\boldsymbol{a})\right]$ and the equivalent of a Q-function would be $r^i_{\boldsymbol{\pi}^{-i}}(a^i) = E_{\boldsymbol{a}^{-i}\sim\boldsymbol{\pi}^{-i}}\left[r^i(a^i, \boldsymbol{a}^{-i})\right]$

2.1 Nash equilibrium:

This thesis will focus on Nash equilibrium as a solution concept. In a Nash equilibrium, no player has an incentive to switch from his current strategy if other players stick to their own. The common way to define a Nash equilibrium is the following:

Definition 2.1. *A strategy $\boldsymbol{\pi}$ is a Nash equilibrium if:*

$$\forall i \in \{1,\dots,N\},\ r^i_{\boldsymbol{\pi}} = \max_{a^i} r^i_{\boldsymbol{\pi}^{-i}}(a^i). \tag{2.48}$$

This definition can also be rewritten in reinforcement learning language as follows:

Definition 2.2. *A strategy $\boldsymbol{\pi}$ is a Nash equilibrium if:*

$$\forall i \in \{1,\dots,N\},\ \pi^i \in \mathcal{G}(r^i_{\boldsymbol{\pi}^{-i}}), \tag{2.49}$$

As in the previous section, $\mathcal{G}(r^i_{\boldsymbol{\pi}^{-i}}) = \{\pi |\ E_{a^i\sim\pi}[r^i_{\boldsymbol{\pi}^{-i}}(a^i)] = \max_{a^i} r^i_{\boldsymbol{\pi}^{-i}}(a^i)\}$

Finding a Nash equilibrium in Normal form games is considered as a hard algorithmic problem (Daskalakis et al., 2009). Several algorithms can be used to solve this problem such as the Lemke Howson algorithm or searching in the support space for the two-player case (Shoham and Leyton-Brown, 2008). The problem is considered to be "hopelessly impractical to solve exactly" for more than two players (Shoham and Leyton-Brown, 2008). These algorithms will not be presented in details here as these methods will not be used in this thesis(see (Nisan et al., 2007, Shoham and Leyton-Brown, 2008) for more involved explanations).

2.2 Zero-Sum Normal-Form Games

Zero-sum two-player NFG is one of the class of NFGs where a Nash equilibrium can be found in a reasonable time. A zero-sum two-player NFG is an NFG where $N = 2$ and where for all $a^1, a^2 \in A^1 \times A^2$, $r^1(a^1, a^2) = -r^2(a^1, a^2) = r(a^1, a^2)$. In that case, player 1 will attempt to maximize his expected outcome while the second player's goal is to minimize it. In that case, the concept of a Nash equilibrium and the minimax solution are equivalent. In the specific case of zero-sum two-player games we will define μ the policy of player 1 and ν the policy of player 2.

Definition 2.3. *A pair of strategies (μ, ν) is a Nash equilibrium if:*

$$\mu \in \operatorname*{argmax}_{\mu\in\Delta(A^1)} \min_{a^2} E_{a^1\sim\mu}[r(a^1, a^2)], \tag{2.50}$$

$$\nu \in \operatorname*{argmin}_{\nu\in\Delta(A^2)} \max_{a^1} E_{a^2\sim\nu}[r(a^1, a^2)]. \tag{2.51}$$

Furthermore, all Nash equilibria in zero-sum two-player NFGs achieve the same value:

$$v^* = \max_{\mu \in \Delta(A^1)} \min_{a^2} E_{a^1 \sim \mu}[r(a^1, a^2)] = \min_{\nu \in \Delta(A^2)} \max_{a^1} E_{a^2 \sim \nu}[r(a^1, a^2)]. \tag{2.52}$$

In a zero-sum two-player game, the minimax strategy can be found in the set of stochastic strategies. That's why in equation (2.50), the policy μ needs to be found in the set of distributions over A^1. But a best response of player 2 against μ can always be found in the set of deterministic strategies. That's why in equation (2.50), a^2 is located in the set of deterministic strategies even if ν (defined in (2.51)) would be a valid best response.

This optimal value v^* is the solution of two dual linear programs. The first linear program involves the strategy of player 1 (μ) and finds the value that maximizes over μ the minimum over a^2 of $E_{a^1 \sim \mu}[r(a^1, a^2)]$:

$$\begin{aligned} &\max\ v^* \\ \text{Subject to } &\sum_{a^1 \in A^1} r(a^1, a^2)\mu(a^1) \geq v^*,\ \forall a^2 \in A^2 \\ &\sum_{a^1 \in A^1} \mu(a^1) = 1 \\ &\mu(a^1) \geq 0,\ a^1 \in A^1 \end{aligned}$$

It's dual involves the strategy of player 2 ν and finds the value that minimizes over ν the maximum over a^2 of $E_{a^2 \sim \nu}[r(a^1, a^2)]$:

$$\begin{aligned} &\min\ v^* \\ \text{Subject to } &\sum_{a^2 \in A^2} r(a^1, a^2)\nu(a^2) \leq v^*,\ \forall a^1 \in A^1 \\ &\sum_{a^2 \in A^2} \nu(a^2) = 1 \\ &\nu(a^2) \geq 0,\ a^2 \in A^2 \end{aligned}$$

These two programs can be solved with any linear programming algorithms such as the simplex method or the interior point method. The complexity of a linear program depends on the number of constrains $c = |A^1| + |A^2| + 1$ or in the number of variables $|A^1|$ for the first linear program and $|A^2|$ the number of variables for the second one. For instance, the complexity of the simplex method is exponential in c in the worst case (for both linear programs) while the complexity of the interior point method is $O(|A^1|^{3.5})$ for the first linear program and $O(|A^2|^{3.5})$ for the second one (Karmarkar, 1984). Linear programming was adapted to solve large turn taking partial information games (Koller et al., 1994). It was the state of the art technique to solve poker before Counter Factual Regret minimization (CFR) (Zinkevich et al., 2008).

3 General-Sum Markov Games

General-sum Markov Games (MG) are a generalization of MDPs for the multi-player scenario and a generalization of NFGs to environments where the interactions also

depend on state. In an MG, N players evolve in a discrete state space S. As in MDPs, the interaction is sequential and at time t, players belong to state s_t. In that state, they all simultaneously choose an action a_t^i in the action space A^i (a_t^i is the action of player i at time t) and receive a reward $r^i(s_t, a_t^1, \dots, a_t^N)$. Again, we will write $\boldsymbol{a} = (a^1, \dots, a^N) = (a^i, \boldsymbol{a^{-i}})$ where $\boldsymbol{a^{-i}}$ is the joint action of every player except player i. Then, the state changes to state $s_{t+1} \sim p(.|s_t, a_t^1, \dots, a_t^N)$ where $p(.|s, a^1, \dots, a^N)$ is the transition kernel of the MG. The constant $\gamma \in [0, 1]$ is the discount factor of the MG. Finally, an MG is a tuple $< S, \{A^i\}_{i\in\{1,\dots,N\}}, \{r^i\}_{i\in\{1,\dots,N\}}, p, \gamma >$.

In an MG, each player's goal is to find a strategy $\{\pi_t^i\}_{t\in\mathbb{R}}$. The strategy π_t^i at time t is a mapping from the state space to the a distribution over the action space (*i.e.* $\pi_t^i \in \Delta A^S$). These strategies are independent and actions are supposed to be chosen simultaneously. Again, we will define $\{\boldsymbol{\pi}_t\}_{t\in\mathbb{N}}$ as the joint strategy of all players and $\{\boldsymbol{\pi}_t^{-i}\}_{t\in\mathbb{N}}$ as the strategy of every player except i. If the strategy is stationary, it will be written π^i, $\boldsymbol{\pi}$ and $\boldsymbol{\pi^{-i}}$. Furthermore, we will often use short notations such as $\boldsymbol{\pi}(\boldsymbol{b}|s) = \pi^1(b^1|s) \times \cdots \times \pi^N(b^N|s)$ or $\boldsymbol{\pi^{-i}}(\boldsymbol{b^{-i}}|s) = \pi^1(b^1|s) \times \cdots \times \pi^{i-1}(b^{i-1}|s) \times \pi^{i+1}(b^{i+1}|s) \times \cdots \times \pi^N(b^N|s)$

The value for player i in state s is his expected γ discounted sum of rewards starting from state s when all players play their part of the joint policy $\{\boldsymbol{\pi}_t\}_{t\in\mathbb{N}}$:

$$v^i_{\{\boldsymbol{\pi}_t\}_{t\in\mathbb{N}}}(s) = E\left[\sum_{t=0}^{+\infty} \gamma^t r^i(s_t, \boldsymbol{a}_t)|s_0 = s,\ \boldsymbol{a}_t \sim \boldsymbol{\pi}_t(.|s_t),\ s_{t+1} \sim p(.|s_t, \boldsymbol{a}_t)\right]. \tag{2.53}$$

When this strategy is stationary, the value function of player i is defined as:

$$v^i_{\boldsymbol{\pi}}(s) = E\left[\sum_{t=0}^{+\infty} \gamma^t r^i(s_t, \boldsymbol{a}_t)|s_0 = s,\ \boldsymbol{a}_t \sim \boldsymbol{\pi}(.|s_t),\ s_{t+1} \sim p(.|s_t, \boldsymbol{a}_t)\right]. \tag{2.54}$$

In the case of stationary strategies, if every player except player i sticks to his strategy $\boldsymbol{\pi^{-i}}$ the value the best response player i can achieve is:

$$v^{*i}_{\boldsymbol{\pi^{-i}}}(s) = \max_{\pi^i} E\left[\sum_{t=0}^{+\infty} \gamma^t r^i(s_t, \boldsymbol{a}_t)|s_0 = s,\ \boldsymbol{a}_t \sim \boldsymbol{\pi}(.|s_t),\ s_{t+1} \sim p(.|s_t, \boldsymbol{a}_t)\right]. \tag{2.55}$$

This value exists since when the strategy of the opponents $\boldsymbol{\pi^{-i}}$ is fixed, the model reduces to an MDP for player i. This value of the best response is the optimal value of that MDP. The following Q-function can be defined:

$$\begin{aligned} Q^i_{\boldsymbol{\pi}}(s, \boldsymbol{a}) &= E\left[\sum_{t=0}^{+\infty} \gamma^t r^i(s_t, \boldsymbol{a}_t)|s_0 = s,\ \boldsymbol{a}_0 = \boldsymbol{a},\ \boldsymbol{a}_t \sim \boldsymbol{\pi}(.|s_t),\ s_{t+1} \sim p(.|s_t, \boldsymbol{a}_t)\right], & (2.56)\\ &= r^i(s, \boldsymbol{a}) + \gamma \sum_{s'\in S} p(s'|s, \boldsymbol{a}) v^i_{\boldsymbol{\pi}}(s'). & (2.57)\end{aligned}$$

One can also define the Q-function for player i of the best response:

$$\begin{aligned} Q^{*i}_{\boldsymbol{\pi^{-i}}}(s, \boldsymbol{a}) &= \max_{\pi^i} Q^i_{\boldsymbol{\pi}}(s, \boldsymbol{a}), & (2.58)\\ &= r^i(s, \boldsymbol{a}) + \gamma \sum_{s'\in S} p(s'|s, \boldsymbol{a}) v^{i*}_{\boldsymbol{\pi^{-i}}}(s'). & (2.59)\end{aligned}$$

These Q-functions are respectively the value for player i in state s if all players start from action $\boldsymbol{a}$ when they play the joint strategy $\boldsymbol{\pi}$ and when player i plays a best response to policy $\boldsymbol{\pi}^{-i}$.

Stochastic Matrices and Linear Algebra: As for MDPs, we can define the linear operator as the transition kernel. The probability transition kernel $P_{\boldsymbol{\pi}}$ is the stochastic matrix that defines the following linear transformation:

$$P_{\boldsymbol{\pi}} v = \sum_{s \in S} \mathbf{1}_s \sum_{s' \in S} E_{\boldsymbol{a} \sim \boldsymbol{\pi}(.|s)} \left[p(s'|s, \boldsymbol{a}) \right] v(s'). \tag{2.60}$$

Furthermore, we will define $r^i_{\boldsymbol{\pi}} = \sum_{s \in S} \mathbf{1}_s E_{\boldsymbol{a} \sim \boldsymbol{\pi}}[r^i(s, \boldsymbol{a})]$ and $p_{\boldsymbol{\pi}}(s'|s) = E_{\boldsymbol{a} \sim \boldsymbol{\pi}(.|s)} \left[p(s'|s, \boldsymbol{a}) \right]$. With these notations we can rewrite the value function as follows:

$$v^i_{\boldsymbol{\pi}} = \left(I - \gamma P_{\boldsymbol{\pi}} \right)^{-1} r^i_{\boldsymbol{\pi}} \tag{2.61}$$

The linear Bellman operator on Q-function can be expressed with linear operators. Let's define the state-action transition kernel:

$$\mathcal{P}_{\boldsymbol{\pi}} Q = \sum_{s, \boldsymbol{a} \in S \times A} \mathbf{1}_{s, \boldsymbol{a}} \sum_{s', \boldsymbol{b} \in S \times A} \boldsymbol{\pi}(\boldsymbol{b}|s') p(s'|s, \boldsymbol{a}) Q(s', \boldsymbol{b}), \tag{2.62}$$

and the reward:

$$r^i = \sum_{s, \boldsymbol{a} \in S \times A} \mathbf{1}_{s, \boldsymbol{a}} r^i(s, \boldsymbol{a}). \tag{2.63}$$

with these definitions, we have:

$$Q^i_{\boldsymbol{\pi}} = \left(I - \gamma \mathcal{P}_{\boldsymbol{\pi}} \right)^{-1} r^i. \tag{2.64}$$

3.1 Nash Equilibrium ϵ-Nash Equilibrium

A Nash equilibrium is a game theoretic solution concept. In a Nash equilibrium, no player can improve his own value by changing his strategy if the other players stick to their strategies (Filar and Vrieze, 2012). In Markov games, this definition must stand in all states:

Definition 2.4. *In an MG, a strategy $\boldsymbol{\pi}$ is a Nash equilibrium if:* $\forall i \in \{1, ..., N\}$, $v^i_{\boldsymbol{\pi}} = v^{*i}_{\boldsymbol{\pi}^{-i}}$.

In an MDP, a Nash equilibrium is simply the optimal strategy. An ϵ-Nash equilibrium is a relaxed solution concept in game theory. When all players play an ϵ-Nash equilibrium the value they receive is at most ϵ sub-optimal compared to a best response. Formally (Filar and Vrieze, 2012):

Definition 2.5. *In an MG, a strategy $\boldsymbol{\pi}$ is an ϵ-Nash equilibrium if:*
$\forall i \in \{1, ..., N\}$, $v^i_{\boldsymbol{\pi}} + \epsilon \geq v^{*i}_{\boldsymbol{\pi}^{-i}}$

3.2 Bellman Operator in General-Sum Games

As explained above, when the strategy of all opponents is fixed, the problem reduces to an MDP. Thus, two Bellman operator can be defined per player.

$$\left[\mathcal{T}^i_{\boldsymbol{\pi}} v\right](s) = r^i_{\boldsymbol{\pi}}(s) + \gamma \sum_{s' \in S} p_{\boldsymbol{\pi}}(s'|s) v(s') \tag{2.65}$$

$$\left[\mathcal{T}^i_{\boldsymbol{\pi}} v\right] = r^i_{\boldsymbol{\pi}} + \gamma P_{\boldsymbol{\pi}} v \tag{2.66}$$

The Bellman operator of the best response of player i to strategy $\boldsymbol{\pi}^{\text{-}i}$:

$$\left[\mathcal{T}^{*i}_{\boldsymbol{\pi}^{\text{-}i}} v\right](s) = \max_{\pi^i} \left(r^i_{\pi^i, \boldsymbol{\pi}^{-i}} + \gamma \sum_{s' \in S} p_{\pi^i, \boldsymbol{\pi}^{-i}}(s'|s) v(s') \right) \tag{2.67}$$

Respectively, one can define operators on Q-functions as follow:

$$\left[\mathcal{B}^i_{\boldsymbol{\pi}} Q\right](s, \boldsymbol{a}) = r^i(s, \boldsymbol{a}) + \gamma \sum_{s' \in S} p(s'|s, \boldsymbol{a}) E_{\boldsymbol{b} \sim \boldsymbol{\pi}(.|s')} \left[Q(s', \boldsymbol{b})\right] \tag{2.68}$$

$$= r + \gamma \mathcal{P}_{\boldsymbol{\pi}} Q \tag{2.69}$$

$$\left[\mathcal{B}^{*i}_{\boldsymbol{\pi}} Q\right](s, \boldsymbol{a}) = r^i(s, \boldsymbol{a}) + \gamma \sum_{s' \in S} p(s'|s, \boldsymbol{a}) \max_{b^i} E_{\boldsymbol{b}^{\text{-}i} \sim \boldsymbol{\pi}^{\text{-}i}(.|s')} \left[Q(s', \boldsymbol{b})\right] \tag{2.70}$$

Again, the fixed point of $\mathcal{T}^i_{\boldsymbol{\pi}}$ is $v^i_{\boldsymbol{\pi}}$, the fixed point of $\mathcal{T}^{*i}_{\boldsymbol{\pi}^{\text{-}i}}$ is $v^{*i}_{\boldsymbol{\pi}^{\text{-}i}}$, the fixed point of $\mathcal{B}^i_{\boldsymbol{\pi}}$ is $Q^i_{\boldsymbol{\pi}}$ and the fixed point of $\mathcal{B}^{*i}_{\boldsymbol{\pi}}$ is $Q^{*i}_{\boldsymbol{\pi}}$.

3.3 Zero-Sum Two-Player Markov Games

As for NFGs we will introduce specific notations and highlight the properties specific to zero-sum two-player MGs. In a zero-sum two-player MG the reward of Player 1 is the loss of Player 2. Thus, for all $s, a^1, a^2 \in S \times A^1 \times A^2$ we have $r^1(s, a^1, a^2) = -r^2(s, a^1, a^2) = r(s, a^1, a^2)$. In this setting, player 1 will attempt to maximize his rewards while player 2 will attempt to minimize his losses. In this specific setting, we will define the strategy of player 1 (μ) instead of π^1 and the strategy of player 2 (ν) instead of π^2. The value of the joint policy (μ, ν) is:

$$v_{\mu,\nu} = v^1_{\mu,\nu} = v^1_{\pi^1, \pi^2}. \tag{2.71}$$

The value of the best response of player 2 to the strategy of player 1 μ is:

$$v_\mu = \min_\nu v_{\mu,\nu} = -v^{*2}_\mu. \tag{2.72}$$

In an MG, there exists a unique value for all Nash equilibrium:

$$v^* = \min_\nu \max_\mu v_{\mu,\nu} = \max_\mu \min_\nu v_{\mu,\nu}. \tag{2.73}$$

Bellman Operator: In zero-sum two-player MGs, five Bellman Operator can be described. Three of them correspond to the Bellman operators for general sum MGs (*i.e.* the Bellman operators introduced in eq. 2.74, eq. 2.75 and eq. 2.77).

$$\mathcal{T}_{\mu,\nu} v = r_{\mu,\nu} + \gamma P_{\mu,\nu} v \tag{2.74}$$

$$\mathcal{T}_{\mu} v = \min_{\nu} \mathcal{T}_{\mu,\nu} v \tag{2.75}$$

$$\mathcal{T} v = \max_{\mu} \mathcal{T}_{\mu} v \tag{2.76}$$

$$\hat{\mathcal{T}}_{\nu} v = \max_{\mu} \mathcal{T}_{\mu,\nu} v \tag{2.77}$$

$$\hat{\mathcal{T}} v = \min_{\nu} \hat{\mathcal{T}}_{\nu} v \tag{2.78}$$

The operator $\mathcal{T}_{\mu,\nu}$ is either $\mathcal{T}^1_{\mu,\nu}$ or $v \to -\mathcal{T}^2_{\mu,\nu}(-v)$. The operator $\mathcal{T}_{\mu}$ is $v \to -\mathcal{T}^{*2}_{\mu}(-v)$ and $\hat{\mathcal{T}}_{\nu}$ is $\mathcal{T}^{*1}_{\nu}$. However, operator $\mathcal{T}$ and $\hat{\mathcal{T}}$ have no equivalent in general-sum games. These two operators are equivalent (Patek, 1997) and are γ-contractions. The fixed point of $\mathcal{T}$ and $\hat{\mathcal{T}}$ is v^*.

Q-Functions and Operators on the Q-Functions: These five operators have their counterparts for Q-functions. For the sake of completeness, we briefly define them in this paragraph. The linear operator $\mathcal{B}_{\mu,\nu}$ is equal to $\mathcal{B}^1_{\mu,\nu}$ and equal to $Q \to -\mathcal{B}^2_{\mu,\nu}(-Q)$. The operator $\mathcal{B}_{\mu}$ is $v \to -\mathcal{B}^{*2}_{\mu}(-v)$ and the operator $\hat{\mathcal{B}}_{\nu}$ is $\mathcal{T}^{*1}_{\nu}$. Finally, the equivalents of $\mathcal{T}$ and $\hat{\mathcal{T}}$ are:

$$\left[\mathcal{B}Q\right](s, a^1, a^2) = r(s, a^1, a^2) + \gamma \sum_{s' \in S} p(s'|s, a^1, a^2) \max_{\mu \in \Delta A^1} \min_{\nu \in \Delta A^2} E_{b^1 \sim \mu, b^2 \sim \nu}[Q(s', b^1, b^2)], \tag{2.79}$$

$$\left[\hat{\mathcal{B}}Q\right](s, a^1, a^2) = r(s, a^1, a^2) + \gamma \sum_{s' \in S} p(s'|s, a^1, a^2) \min_{\nu \in \Delta A^2} \max_{\mu \in \Delta A^1} E_{b^1 \sim \mu, b^2 \sim \nu}[Q(s', b^1, b^2)]. \tag{2.80}$$

The fixed point of $\mathcal{B}_{\mu,\nu}$ is the state-action value function of the joint strategy μ, ν and will be written $Q_{\mu,\nu} = Q^1_{\mu,\nu} = -Q^2_{\mu,\nu}$. The fixed point of $\mathcal{B}_{\mu}$ is the Q-function of the best response of player 2 to the strategy μ and will be written $Q_{\mu} = -Q^{*2}_{\mu}$. The minimax state-action value function is Q^* and is the fixed point of $\mathcal{B}$ or $\hat{\mathcal{B}}$.

Greedy Strategy in Zero-Sum Two-Player Markov Games: The notion of greedy strategy in zero-sum two-player MGs is slightly different from the case of MDPs. In zero-sum two-player MGs we need to take the point of view of one player (player 1 or player 2). The canonical way to do it is to take the point of view of player 1 but the two point of view are dual. A strategy μ is said to be greedy with respect to a value function v if $\mathcal{T}_{\mu} v = \mathcal{T} v$ and we will define $\mathcal{G}(v) = \{\mu |\ \mathcal{T}_{\mu} v = \mathcal{T} v\}$. A strategy μ is greedy with respect to a Q-function if for all s we have $\min_{\nu \in \Delta A^2} \max_{\mu \in \Delta A^1} E_{a^1 \sim \mu, a^2 \sim \nu}[Q(s, a^1, a^2)] = \min_{\nu \in \Delta A^2} E_{a^1 \sim \mu(.|s), a^2 \sim \nu}[Q(s, a^1, a^2)]$ and we will define it $\mu \in \mathcal{G}(Q)$. We choose this definition of a greedy strategy with the intention to build policy iteration algorithms that converges to the minimax value v^* but others operators could be considered if the problem was to find a best response to some strategy. Again, two results similar to the case of MDPs and can be found:

Theorem 2.4. *A strategy μ is optimal if and only if it is greedy with respect to it's value v_μ.*

$$v_\mu = v^* \text{ if and only if } \mu \in \mathcal{G}(v_\mu) \tag{2.81}$$

This first theorem implies that the optimal policy can be retrieved from the optimal value.

Theorem 2.5. *If μ' is greedy with respect to the value v_μ of strategy μ, then $v_{\mu'} \geq v_\mu$*

This second theorem proves that a simple way to improve a strategy π is to play the greedy strategy π' with respect to the value v_π. This theorem will be the basis of the Policy Iteration algorithm for zero-sum two-player MGs (Section 3.4.2). These two theorems can be deduced from (Patek, 1997).

3.4 Exact Algorithms for Zero-Sum Two-Player Markov-Games

The Value Iteration (VI) algorithm for zero-sum two-player MGs (Shapley, 1953) will be detailed in Section 3.4.1 and is a generalization of the VI algorithm for MDPs (Section 1.1.1). As in MDPs, VI is relatively slow at the beginning even if iterations are computationally cheap. This motivates the use of algorithms that have more computationally intensive iterations such as the Policy Iteration algorithm. Howard's PI algorithm for MDPs (detailed in sec 1.1.2) has two extensions to zero-sum two-player MGs. The first one is Hoffman and Karp's Policy Iteration Hoffman and Karp (1966). This algorithm is cumbersome since it requires solving an MDP as a subroutine (see Section 3.4.2). To address this issue, Pollatschek and Avi-Itzhak (Pollatschek and Avi-Itzhak, 1969) proposed their own generalization of the Howard PI algorithm that only requires a joint strategy evaluation subroutine (see Section 3.4.3). Despite being efficient in practice, this algorithm was not proven to converge in all zero-sum two-player MGs. In fact, less than 10 years later Van Der Wal presented a simple counterexample (Van Der Wal, 1978) on which the Pollatschek and Avi-Itzhak's algorithm was oscillating. In the same paper, he presented his own improvement of the Hoffman and Karp's PI algorithm named the Generalized Policy Iteration (GPI) (detailed in Section 3.4.4). This algorithm is a generalization to zero-sum two-player MGs of MPI (Section 1.1.3). More than 10 year later, Filar and Tolwinski revisited the Pollatschek and Avi-Itzhak's algorithm. They proved that this algorithm was in fact a Newton's method and the $\mathcal{L}_2$-norm of the Optimal Bellman Residual (OBR). Furthermore, after showing that every local minimum of the OBR is in fact a global minimum, they suggest the use of a quasi-Newton method on the OBR instead of a Newton's method. This last method is detailed in 3.4.5.

3.4.1 Value Iteration

The value iteration algorithm (see Algorithm 10) was proposed by Shapley in the same paper introducing MGs (Shapley, 1953). It iteratively applies the Bellman operator

$\mathcal{T}$ at each iteration. This algorithm produces a sequence of values v_k and an implicit sequence of greedy strategies μ_k with respect to v_k. Each iteration's cost is $|S|$ times the cost of solving a minimax problem. This cost thus depends on the choice of the algorithm for the minimax problem(see Section 2.2).

Algorithm 10 Value Iteration

Input: An zero-sum two-player MG MG, a value $v_0 = 0$ and a maximum number of iterations K.
for k=1,2,...,K-1 **do**
  $v_k = \mathcal{T} v_{k-1}$
end for
find $\mu_K \in \mathcal{G}(v_{K-1})$
Output: μ_K

The performance of μ_K is usually determined as the norm of the difference between v_{μ_K} and the minimax value v^*. The prof of the following result can be found in (Patek, 1997):

$$\|v_{\mu_K} - v^*\|_{+\infty} \leq \frac{2\gamma^K}{1-\gamma} \|v^* - v_0\|_{+\infty}. \tag{2.82}$$

As for MDPs, the cost of each iteration is relatively low compared to other methods but the convergence of this method is slow in the number of iterations.

3.4.2 Policy Iteration by Hoffman and Karp

Hoffman and Karp's PI has two steps per iteration. Suppose we are at iteration k. The first step is to find a greedy strategy μ_k with respect to the previous value v_{k-1}. This greedy step requires solving a minimax problem per state. The second step of this process is to find the value of a best response to μ_k. This step requires building the MDP of the kernel $E_{a^1 \sim \mu(.|s)}[p(.|s, a^1, a^2)]$ and of the reward $E_{a^1 \sim \mu(.|s)}[r(s, a^1, a^2)]$ and to solve it by minimizing the sum of rewards instead of maximizing them. This second step computes v_{μ_k}. Any algorithm solving an MDP can be selected for this second step and the complexity of the overall method will depend on the algorithm selected. This algorithm reduces to policy iteration in the case the MG is an MDP (*i.e.* in the case $|A^2| = 1$).

This algorithm produces an increasing sequence of values v_k. This property is implied by theorem 2.5.

Again, one can prove the following result on the performance of policy π_K:

$$\|v_{\mu_K} - v^*\|_{+\infty} \leq \frac{2\gamma^K}{1-\gamma} \|v^* - v_0\|_{+\infty}. \tag{2.83}$$

This bound shows the same asymptotic performance as the one of Value Iteration. But in practice, PI for zero-sum two-player MGs converges faster than VI (Van

Algorithm 11 Policy Iteration

Input: An zero-sum two-player MG MG, a value $v_0 = 0$ and a maximum number of iterations K.
for k=1,2,...,K **do**
  find $\mu_k \in \mathcal{G}(v_{k-1})$ (min max step)
  compute $v_k = v_{\mu_k}$ (solving an MDP)
end for
Output: μ_K

Der Wal, 1978). Further results can be proven on the number of iterations of PI for turn taking games (Hansen et al., 2013b). However, to the best of our knowledge, no better convergence guarantees exist in the case of simultaneous move games.

3.4.3 The Algorithm of Pollatschek and Avi-Itzhak

As described in the previous section, Hoffman and Karp's PI algorithm is cumbersome since it requires solving an MDP at each iteration. Pollatschek and Avi-Itzhak's algorithm (Algorithm 12) is an attempt to address this issue by changing the second step of the algorithm. Instead of solving an MDP at the second step of the iteration, this algorithm evaluates the joint strategy μ_k, ν_k. Pollatschek and Avi-Itzhak's algorithm is said to be efficient in practice (Van Der Wal, 1978) but lacks theoretical guarantees of convergence to a minimax solution. In fact, Van Der Wal proved that it was oscillating in certain cases.

Algorithm 12 Algorithm of Pollatschek and Avi-Itzhak

Input: An zero-sum two-player MG MG, a value $v_0 = 0$ and a maximum number of iterations K.
for k=1,2,...,K **do**
  find μ_k, ν_k such that $\mathcal{T}_{\mu_k,\nu_k} v_{k-1} = \mathcal{T}_{\mu_k} v_{k-1} = \hat{\mathcal{T}}_{\nu_k} v_{k-1} = \mathcal{T} v_{k-1}$
  compute $v_k = v_{\mu_k,\nu_k} = (I - \gamma P_{\mu_k,\nu_k})^{-1} r_{\mu_k,\nu_k}$
end for
Output: μ_K

3.4.4 Generalized Policy Iteration

In the same paper proving the flaws of Pollatschek and Avi-Itzhak's algorithm (Van Der Wal, 1978), Van Der Wal gave his own solution to the Hoffman and Karp's PI problem. He introduces the Generalize Policy Iteration algorithm (GPI) that generalizes the MPI (Section 1.1.3) to MGs. GPI also introduces a parameter m that controls the cost of each iteration. This algorithm also reduces to VI if $m = 1$ and also reduces to Hoffman and Karp' PI if $m = +\infty$. Indeed, in Algorithm 13 if $m = +\infty$, the value v_k is the value v_{μ_k}. If $m = 1$, since $\mu_k \in \mathcal{G}(v_{k-1})$ we have $\mathcal{T} v_{k-1} = \mathcal{T}_{\mu_k} v_{k-1}$ and

finally $v_k = \mathcal{T}_{\mu_k} v_{k-1} = \mathcal{T} v_{k-1}$. Again, this algorithm enjoys the same guarantees of convergence as PI and VI and again the performance policy μ_k does not depend on m.

$$\|v_{\mu_K} - v^*\|_{+\infty} \leq \frac{2\gamma^K}{1-\gamma} \|v^* - v_0\|_{+\infty}. \tag{2.84}$$

But as shown by in (Van Der Wal, 1978), this algorithm converges faster than VI. And its cost per iteration is lower than the PI by Hoffman and Karp.

Algorithm 13 Generalized Policy Iteration

Input: An MDP M, a value $v_0 = 0$, a maximum number of iterations K and a parameter m.
for k=1,2,...,K **do**
  find $\mu_k \in \mathcal{G}(v_{k-1})$
  compute $v_k = (\mathcal{T}_{\mu_k})^m v_{k-1}$
end for
Output: μ_K

3.4.5 Minimizing the Bellman Residual: Filar and Tolwinski's Algorithm

We will take extra care to detail this algorithm as it inspired the work detailed in Chapter 6.

In an enlightening paper, Filar and Tolwinski prove that the Pollatschek and Avi-Itzhak algorithm is a Newton's method on the $\mathcal{L}_2$-norm of the Optimal Bellman Residual (OBR) $\mathcal{J}_{\text{OBR}}(v) = \|v - \mathcal{T}v\|_2 = (v - \mathcal{T}v)^\top (v - \mathcal{T}v)$. Indeed, the gradient of $\mathcal{J}_{\text{OBR}}(v)$ is $\nabla \mathcal{J}_{\text{OBR}}(v) = (I - \gamma P_{\mu,\nu})^\top (v - \mathcal{T}v)$ and the Hessian matrix is $H\mathcal{J}_{\text{OBR}}(v) = (I - \gamma P_{\mu,\nu})^\top (I - \gamma P_{\mu,\nu})$ where μ, ν are such that $\mathcal{T}_{\mu,\nu} v = \mathcal{T}_\mu v = \hat{\mathcal{T}}_\nu v = \mathcal{T} v$. From value v_k, the update v_{k+1} of the Newton's method is

$$\begin{aligned}
v_{k+1} &= v_k - (H\mathcal{J}_{\text{OBR}}(v_k))^{-1} \nabla \mathcal{J}_{\text{OBR}}(v_k), && (2.85)\\
&= v_k - \left((I - \gamma P_{\mu_k,\nu_k})^\top (I - \gamma P_{\mu_k,\nu_k})\right)^{-1} (I - \gamma P_{\mu_k,\nu_k})^\top (v_k - \mathcal{T}v_k), && (2.86)\\
&= v_k - (I - \gamma P_{\mu_k,\nu_k})^{-1} (v_k - \mathcal{T}_{\mu_k,\nu_k} v_k), && (2.87)\\
&= v_k - (I - \gamma P_{\mu_k,\nu_k})^{-1} ((I - \gamma P_{\mu_k,\nu_k}) v_k - r_{\mu_k,\nu_k}), && (2.88)\\
&= v_k - v_k + (I - \gamma P_{\mu_k,\nu_k})^{-1} r_{\mu_k,\nu_k}, && (2.89)\\
&= (I - \gamma P_{\mu_k,\nu_k})^{-1} r_{\mu_k,\nu_k} = v_{\mu_k,\nu_k}. && (2.90)
\end{aligned}$$

Thus, the idea of Filar and Tolwinski is to make the use of a quasi-Newton method instead of Newton's method. At each iteration, they introduce a step size α_k chosen by line search (Amijo's rule (Filar and Tolwinski, 1991)) instead of using a step size of 1. The update becomes:

$$v_{k+1} = (1 - \alpha_k) v_k + \alpha_k v_{\mu_k,\nu_k}. \tag{2.91}$$

Furthermore, they notice that the gradient of $\mathcal{J}_{\text{OBR}}(v)$ is null if and only if $v - \mathcal{T}v = 0$. Thus every local minimum of the OBR is a global minimum. Thus, they claim that their algorithm converges to a global minimum. The algorithm is described in Algorithm 14.

Algorithm 14 Filar and Tolwinski's Algorithm

Input: An zero-sum two-player MG MG, a value $v_0 = 0$ and a maximum number of iterations K.
for k=1,2,...,K **do**
find μ_k, ν_k such that $\mathcal{T}_{\mu_k,\nu_k} v_{k-1} = \mathcal{T}_{\mu_k} v_{k-1} = \hat{\mathcal{T}}_{\nu_k} v_{k-1} = \mathcal{T} v_{k-1}$
compute $v_{\mu_k,\nu_k} = (I - \gamma P_{\mu_k,\nu_k})^{-1} r_{\mu_k,\nu_k}$
find α_k according to Amijo's rule for the function $\mathcal{J}_{\text{OBR}}$
update $v_{k+1} = (1 - \alpha_k) v_k + \alpha_k v_{\mu_k,\nu_k}$
end for
Output: μ_K

3.5 Batch Algorithms for Zero-Sum Two-Player Markov-Games

Even if many batch algorithms were developed for MDPs, this topic remains superficially explored in Markov Games. The only algorithm handling batch data is LSPI for zero-sum two-player MGs (Lagoudakis and Parr, 2002) is a generalization of the LSPI algorithm from MDPs to zero-sum two-player MGs. Again, as any batch algorithm LSPI will perform its iterations on the Q-function and will take as input a batch of data $D_n = \{(s_j, a_j, r_j, s'_j)\}_{j=1,\dots,n}$ where for all j, the reward $r_j = r(s_j, a_j)$ and where $s'_j \sim p(.|s_j, a_j)$

3.5.1 Least-Squares Policy Iteration

The LSPI algorithm for MGs (Lagoudakis and Parr, 2002) is a batch version of the algorithm of Pollatschek and Avi-Itzhak algorithm. The first step of the iteration of the algorithm is the greedy part. This part is similar to the LSPI algorithm for MDPs but instead of finding an argmax, it will find a minimax solution. The second step of an iteration of LSPI finds an approximation of the Q-function of the joint policy found at the previous iteration. This approximation of the value function lies in a feature space $\mathcal{F}_\Phi = \{\Phi\omega, \omega \in \mathbb{R}^d\}$ where $\Phi = [\phi_1, \dots, \phi_d]$ are linearly independent functions from $S \times A^1 \times A^2 \to \mathbb{R}$. These features can be thought of as vectors of an $|S| \times |A^1| \times |A^2|$-dimensional vector space. We will define ρ as a distribution of the state-action space and Δ_ρ the diagonal application that maps $\mathbf{1}_{s,a^1,a^2}$ to $\rho(s, a^1, a^2)\mathbf{1}_{s,a^1,a^2}$ and finally, the projection on the feature space $\mathcal{F}_\Phi$ with respect to the $\mathcal{L}_{\rho,2}$ is $\Pi_{\rho,\Phi}$.

Instead, LSPI finds ω satisfying the projected fixed point equation:

$$\Phi\omega = \Pi_{\rho,\Phi} \mathcal{B}_{\mu,\nu} \Phi\omega. \tag{2.92}$$

The solution for ω of this equality is given by:

$$\omega = A^{-1}_{\rho,\Phi,\mu,\nu} b_{\rho,\Phi}, \tag{2.93}$$
$$\text{where } A_{\rho,\Phi,\mu,\nu} = \Phi^\top \Delta_\rho (\Phi - \gamma \mathcal{P}_{\mu,\nu}\Phi), \tag{2.94}$$
$$\text{and } b_{\rho,\Phi} = \Phi \Delta_\rho r. \tag{2.95}$$

Given a batch of data $D_n = \{(s_j, a^1_j, a^2_j, r_j, s'_j)\}_{j=1,\dots,n}$, the matrix $A_{\rho,\Phi,\mu,\nu}$ and the vector $b_{\rho,\Phi}$ can be estimated as follow:

$$\hat{A}_{\Phi,\mu,\nu,D_n} = \frac{1}{n}\sum_{j=1}^{n} \Phi(s_j, a^1_j, a^2_j) \left(\Phi(s_j, a^1_j, a^2_j) - \gamma \sum_{a^1,a^2 \in A^1 \times A^2} \mu(a^1|s'_j)\nu(a^2|s'_j)\Phi(s'_j, a^1, a^2) \right), \tag{2.96}$$

$$\hat{b}_{\Phi,D_n} = \frac{1}{n}\sum_{j=1}^{n} \Phi(s_j, a^1_j, a^2_j) r(s_j, a^1_j, a^2_j). \tag{2.97}$$

If the data in D_n are independent and distributed according the distribution ρ, $\hat{A}_{\Phi,\mu,\nu,D_n}$ and $\hat{b}_{\Phi,D_n}$ are consistent estimators of $A_{\rho,\Phi,\mu,\nu}$ and $b_{\rho,\Phi}$. Furthermore, $\hat{A}^{-1}_{\Phi,\mu,\nu,D_n}$ can be be efficiently computed with the Scherman-Morison formula which provides an optimized version of the algorithm. The algorithm in it's most simple version is presented in Algorithm 15.

Algorithm 15 Least Squares Policy Iteration for MGs(LSPI)

Input: A batch $D_n = \{(s_j, a^1_j, a^2_j, r_j, s'_j)\}_{j=1,\dots,n}$, a feature space Φ, an initial parametrization $\omega_0 = 0$ and a number of iterations K.
for k=1,2,...,K **do**
  find μ_k, ν_k such that $\mu_k \in \mathcal{G}(\Phi\omega_{k-1})$ and ν_k is a policy such that $E_{a^1\sim\mu(.|s),\ a^2\sim\nu(.|s)}[\Phi\omega_{k-1}(a^1, a^2, s)] = \min_{a^2} E_{a^1\sim\mu(.|s)}[\Phi\omega_{k-1}(a^1, a^2, s)]$
  compute $\hat{A}_{\Phi,\mu_k,\nu_k,D_n}$ and $\hat{b}_{\Phi,D_n}$ as in formula 2.96 and 2.97.
  compute $\omega_k = \hat{A}^{-1}_{\Phi,\mu_k,\nu_k,D_n}\hat{b}_{\Phi,D_n}$
end for
Output: μ_K

3.6 Exact Algorithms for General-Sum Markov-Games

There is a trove of approaches to solve general sum MGs. This section will provide an overview of those methods with an emphasis on independent reinforcement learning methods.

Multi Objective Linear Programming and Homotopy: One way to solve MGs is to generalize the homotopy method. This method was studied widely in the two last decades and is model based (Herings and Peeters, 2004, 2010, Borkovsky et al., 2010). Like many algorithms to solve MGs, it scales poorly with the number of players

(the homotopy method scales exponentially). Multi Objective Linear Programming was studied in (Dermed and Isbell, 2009). These two methods are only tractable for small size problems (as reported in (Prasad et al., 2015))

Rational Learning: Rational learning is guaranteed to converge to a Nash equilibrium (Kalai and Lehrer, 1993) but is not an independent learning procedure. In a Bayesian manner, players maintain a prior on the strategy of the opponents. This algorithm is only guaranteed to converge in repeated games.

Evolutionary Game Theory: In (Akchurina, 2009), an evolutionary game theoretic approach is generalized to Markov games. This work uses numerical methods to find rest points of the replicator dynamic equation. This procedure is also model-based and hardly scale to large state spaces and large action spaces. Furthermore, no guarantee on the accuracy of the approximation is provided.

Two-timescale algorithms in MGs: Two-timescale algorithms have been the subject of a wide literature in MDPs that starts from the seminal work of Borkar (1997a). These works mainly analyze the use of linear function approximations. However, when applied independently in a multiagent setting, those algorithms no longer have convergence guarantees (even without function approximation). In (Prasad et al., 2015), the authors attempt to provide an on-line model-free algorithm with guarantees. This approach requires each player to observe the rewards of the others. The authors claim that their algorithm is guaranteed to converge to a Nash equilibrium but the proof of this algorithm is broken (details in App. 14).

Sampling Based Algorithms in MSGs and Conterfactual Regret Minimization Algorithms (CFR): A trove variety of algorithms exists to compute Nash equilibria in MSGs. A review of those algorithms can be found in (Bošanský et al., 2016). Those algorithms require that one can query the model at each stage of the game but they do not belong to the family of independent RL methods. Another family of algorithms related to our work are the CFR algorithms and their wide number of variations. Even if these algorithms have sample-based variations (Lanctot et al., 2009), none of those variations are based on independent learning rules.

3.7 Independent Reinforcement Learning for Markov Games

Several major issues prevent direct use of standard RL algorithms with multi-agent systems. First, blindly applying single agent RL in a decentralized fashion implies that, from each agent's point of view, the other agents are part of the environment. Such an hypothesis breaks a crucial RL assumption that the environment is (at least almost) stationary (Laurent et al., 2011). Second, it introduces partial observability as each agent's knowledge is restricted to its own actions and rewards while its behavior should depend on others' strategies.

Independent reinforcement learning in games has been studied widely in the case of normal form games and include regret minimization approaches (Bubeck and Cesa-Bianchi, 2012, Cesa-Bianchi and Lugosi, 2006) or stochastic approximation algorithms (Leslie and Collins, 2006). However, to our knowledge, none of the previous methods have been extended to independent reinforcement learning in Markov Games or any intermediate models such as MSGs with guarantees of convergence both for cooperative and zero-sum cases. Addressing both cases are still treated as separate agendas since the seminal paper by Shoham et al. (2007).

3.7.1 *Q*-Learning Like Algorithms

On-line algorithms like Q-learning (Watkins and Dayan, 1992) are often used in cooperative multi-agent learning environments but fail to learn a stationary strategy in simultaneous zero-sum two-player games. They fail in this setting because, in simultaneous zero-sum two-player games, it is not sufficient to use a greedy strategy to learn a Nash equilibrium. In (Littman, 1994), the Q-learning method is adapted to guarantee convergence to zero-sum two-player MGs. Other adaptations to N-player games were developed (Hu and Wellman, 2003, Greenwald et al., 2003). However, all these methods require the observation of the opponents' action and the two last ones are guaranteed to converge only under very conservative hypotheses. Moreover, these are not independent methods as each player needs to observe the rewards of the others.

3.7.2 Independent Policy Gradient

In MGs and in NFGs, previous work on decentralized learning includes the study of policy hill climbing (Bowling and Veloso, 2001). In this approach, each agent follows a gradient ascent on his expected outcome. This method enjoys limited guarantees and fails to converge in zero-sum games. The behavior of this algorithm can be improved using heuristics on the learning rate as reported in (Bowling and Veloso, 2001). It can also scale up using function approximation as in (Banerjee and Peng, 2003) which results in an actor-critic like method and fits in our setting.

3.7.3 Fictitious Play

Fictitious play is a model-based process that learns Nash equilibria in some subclass normal form games. It has been widely studied and required assumptions were weakened over time (Leslie and Collins, 2006, Hofbauer and Sandholm, 2002) since the original article by Robinson Robinson (1951). It has been extended to extensive form games (game trees) and, to a lesser extent, to function approximation (Heinrich et al., 2015). However, theoretical work on this process is neither on-line nor decentralized except from Leslie and Collins (2006) work which focus on NFGs. Fictitious play enjoys several convergence guarantees (Hofbauer and Sandholm, 2002) which makes it a good candidate for learning in simultaneous multi-stage games. The original fictitious play creates a sequence of policies $\boldsymbol{\pi}_n$. At each iteration n, players play their best

response against the average policy played by the opponent until iteration n. This average policy per players evolve as follows:

$$\pi^i_{n+1} = \left(1 - \frac{1}{n}\right) \pi^i_{n+1} + \frac{1}{n} B\left(r^i_{\boldsymbol{\pi}^{-i}_n}\right)$$

Where $B\left(r^i_{\boldsymbol{\pi}^{-i}_n}\right)$ is a greedy policy of with respect to the reward $r^i_{\boldsymbol{\pi}^{-i}_n}$ (formally we have $B\left(r^i_{\boldsymbol{\pi}^{-i}_n}\right) \in \mathcal{G}(r^i_{\boldsymbol{\pi}^{-i}_n})$).

This process is often studied as a continuous time process formalized as a differential inclusion. This continuous time process follows the following differential inclusion equation:

$$\dot{\pi}^i_t \in \mathcal{G}\left(r^i_{\boldsymbol{\pi}^{-i}_t}\right) - \pi^i_t \tag{2.98}$$

The process defined in Equation (2.99) has been weakened to allow the use of approximate best responses. In Hofbauer and Sandholm (2002), the approximate best response is defined in a generalized manner but can be thought as a softmax over the reward function. The differential inclusion defined above becomes a differential equation:

$$\dot{\pi}^i_t \in B_\sigma\left(r^i_{\boldsymbol{\pi}^{-i}_t}\right) - \pi^i_t \tag{2.99}$$

This process will be generalized to multi stage games in Chapter 8.

Part II

Approximate Dynamic Programming in Zero-Sum Two-Player Markov Games

Chapter 3

Approximate Dynamic Programming in Games : A Unified Analysis

---

This chapter details the contributions of (Perolat et al., 2015). The first part of this chapter reports a unified analysis of several approximate dynamic programming algorithms for MGs comparable to the one of section 1.2.3 on MDP. The second part of this chapter proposes a batch algorithm based on generalized policy iteration (Section 3.4.4). First we briefly recall the VI algorithm, the PI algorithm and the GPI algorithm. Then we present the scheme of approximation and analyse it. Finally, we propose our novel Batch algorithm for MGs and report its empirical evaluation.

1 Approximate Dynamic Programming: A Unified Scheme

The three algorithms we intend to analyse are VI, PI and GPI. Each iteration of these algorithms can be decomposed in two steps. The greedy step is common to all of them and the evaluation step implements a different strategy for each of them. Each iteration of the VI algorithm can be written as follows:

$$v_{k+1} = \mathcal{T} v_k.$$

This can equivalently be written in two steps as follows:

$$\begin{aligned} \mu_{k+1} &= \mathcal{G}(v_k), \\ v_{k+1} &= \mathcal{T}_{\mu_{k+1}} v_k. \end{aligned}$$

This two-step decomposition of these algorithms is straightforward for PI and GPI. Indeed, each iterations of PI can be written as:

$$\begin{aligned} \mu_{k+1} &= \mathcal{G}(v_k), \\ v_{k+1} &= v_{\mu_{k+1}} = (\mathcal{T}_{\mu_{k+1}})^{+\infty} v_k. \end{aligned}$$

Finally, GPI bridges the gap between PI and VI:

$$\mu_{k+1} = \mathcal{G}(v_k),$$
$$v_{k+1} = (\mathcal{T}_{\mu_{k+1}})^m v_k.$$

It is clear that GPI generalizes VI and PI. In particular, an error propagation bound for VI (when $m = 1$) and PI (when $m = \infty$) will follow from the analysis we shall provide for GPI. Thus, only an approximate version of GPI will be analysed. This analysis will give insights on the VI and PI algorithms as they are limit cases.

Remark 3.1. In turn-based games, each state is controlled by a single player. In the zero-sum two-player MGs framework, they can be seen as a game where $\forall s \in S$, $card(A^1(s)) = 1$ or $card(A^2(s)) = 1$. In this special case, optimal strategies are deterministic (Hansen et al., 2013b). Furthermore, and as we already mentioned, the greedy step (see definition in Section 2) reduces to finding a maximum rather than a minimax equilibrium and is thus significantly simpler. Furthermore, one can see an MDP as a zero-sum two-player MG where one player has no influence on both the reward and the dynamics. Therefore, our analysis should be consistent with previous MDP analyses.

1.1 Approximate Generalized Policy Iteration

As its exact counterpart, Approximate Generalized Policy Iteration (AGPI) proceeds in two steps (Algorithm 16). Unlike GPI, each steps of AGPI accounts for errors in each step of the iteration. The greedy step can select a strategy that is not a minimax strategy. At iteration k, this strategy can be ϵ'_k suboptimal and we shall write $\mu_k \leftarrow \hat{\mathcal{G}}_{\epsilon'_k}(v_{k-1})$ for:

$$\mathcal{T} v_{k-1} \leq \mathcal{T}_{\mu_k} v_{k-1} + \epsilon'_k \tag{3.1}$$
$$\text{or, } \forall \mu' \ \mathcal{T}_{\mu'} v_{k-1} \leq \mathcal{T}_{\mu_k} v_{k-1} + \epsilon'_k. \tag{3.2}$$

In other words, the strategy μ is not necessarily the best strategy, but it must be guaranteed to be ϵ' away from the best strategy in the worst case.

At iteration k the evaluation step may have an additive error ϵ_k:

$$v_k = (\mathcal{T}_{\mu_k})^m v_{k-1} + \epsilon_k. \tag{3.3}$$

In the evaluation step the strategy μ_k is fixed and the operator $\mathcal{T}_{\mu_k} \cdot = \min_\nu \mathcal{T}_{\mu_k,\nu} \cdot$ is applied m times. From the point of view of the minimizer, this step finds an optimal value for a γ-discounted m-horizon problem.

1.2 Error Propagation

The goal of our analysis is to report the impact of previous errors $\{\epsilon_i\}_{i\in\{1,\dots,k-1\}}$ and $\{\epsilon'_i\}_{i\in\{1,\dots,k\}}$ on the current strategy μ_k. Thus, we are interested in bounding the difference

$$l_k = v_* - v_{\mu_k} \geq 0,$$

Algorithm 16 Approximate Generalized Policy Iteration

Input: An MDP M, a value $v_0 = 0$, a maximum number of iterations K and a parameter m.
for k=1,2,...,K **do**
  find $\mu_k \in \hat{\mathcal{G}}_{\epsilon'_k}(v_{k-1})$
  compute $v_k = (\mathcal{T}_{\mu_k})^m v_{k-1} + \epsilon_k$
end for
Output: μ_K

where v_* is the minimax value of the game (obtained when both players play the Nash equilibrium μ_* and ν_*) and where v_{μ_k} is the value when the maximizer plays μ_k and the minimizer plays the optimal counter-strategy against μ_k. This is a natural measure of quality for the strategy μ_k that would be output by the approximate algorithm.

The following elements will be key properties in our analysis. By definition, we have:

$$\forall \nu, \forall v,\ \mathcal{T}_\mu v \leq \mathcal{T}_{\mu,\nu} v. \tag{3.4}$$

In addition, we shall consider a few notations. The minimizer policies ν_k^i, $\tilde{\nu}_k$, $\hat{\nu}_k$ and $\overline{\nu}_k$ are policies that respectively satisfy:

$$(\mathcal{T}_{\mu_k})^{i+1} v_{k-1} = \mathcal{T}_{\mu_k,\nu_k^i} ... \mathcal{T}_{\mu_k,\nu_k^1} \mathcal{T}_{\mu_k} v_{k-1}. \tag{3.5}$$

The strategy ν_k^i is an argmin strategy of Player 2 when Player 1 plays the strategy μ_k with respect to the value $\mathcal{T}_{\mu_k})^i v_{k-1}$. The strategy $\tilde{\nu}_k$ is is an argmin strategy for Player 2 when Player 1 chooses the strategy μ^* with respect to the value v_k:

$$\mathcal{T}_{\mu_*} v_k = \mathcal{T}_{\mu_*,\tilde{\nu}_k} v_k. \tag{3.6}$$

The strategy $\hat{\nu}_k$ (respectively $\overline{\nu}_k$) is also an argmin strategy for Player 2 when Player 1 chooses the strategy μ_k with respect to the value v_k (respectively v_{μ_k}).

$$\mathcal{T}_{\mu_k} v_k = \mathcal{T}_{\mu_k,\hat{\nu}_k} v_k, \tag{3.7}$$

$$\mathcal{T}_{\mu_k} v_{\mu_k} = \mathcal{T}_{\mu_k,\overline{\nu}_k} v_{\mu_k}. \tag{3.8}$$

In order to bound l_k, we will study the following quantities similar to those introduced by Scherrer et al. (2012) (recall that $\mathcal{T}_\mu$ and $\mathcal{T}_{\mu,\nu}$ are defined in Section 3.3 and the stochastic kernel is defined in Section 2.62):

$$d_k = v_* - (\mathcal{T}_{\mu_k})^m v_{k-1} = v_* - (v_k - \epsilon_k), \tag{3.9}$$
$$s_k = (\mathcal{T}_{\mu_k})^m v_{k-1} - v_{\mu_k} = (v_k - \epsilon_k) - v_{\mu_k}, \tag{3.10}$$
$$b_k = v_k - \mathcal{T}_{\mu_{k+1}} v_k, \tag{3.11}$$
$$x_k = (\mathcal{I} - \gamma P_{\mu_k,\hat{\nu}_k})\epsilon_k + \epsilon'_{k+1}, \tag{3.12}$$
$$y_k = -\gamma P_{\mu_*,\tilde{\nu}_k} \epsilon_k + \epsilon'_{k+1}. \tag{3.13}$$

Notice that $l_k = d_k + s_k$. We shall prove the following relations, similar to the one proved in Scherrer et al. (2012).

Lemma 3.1. *The following linear relations hold:*

$$b_k \leq \gamma P_{\mu_k,\hat{\nu}_k}\gamma P_{\mu_k,\nu_k^{m-1}}...\gamma P_{\mu_k,\nu_k^1}b_{k-1} + x_k, \tag{3.14}$$

$$d_{k+1} \leq \gamma P_{\mu_*,\tilde{\nu}_k}d_k + y_k + \sum_{j=1}^{m-1}\gamma P_{\mu_k,\nu_k^j}...\gamma P_{\mu_k,\nu_k^1}b_k, \tag{3.15}$$

$$s_k \leq (\gamma P_{\mu_k,\overline{\nu}_k})^m(\sum_{i=1}^{\infty}\gamma P_{\mu_k,\nu_k^i}...\gamma P_{\mu_k,\nu_k^1}b_{k-1}). \tag{3.16}$$

Contrary to the analysis of Scherrer et al. (2012) for MDPs in which the operator $\mathcal{T}_\mu$ is affine, it is in our case non-linear. The proof that we now develop is thus more technical.

Proof. Let us start with b_k:

$$\begin{aligned} b_k &= v_k - \mathcal{T}_{\mu_{k+1}}v_k, \\ &= v_k - \mathcal{T}_{\mu_k}v_k + \mathcal{T}_{\mu_k}v_k - \mathcal{T}_{\mu_{k+1}}v_k. \end{aligned}$$

From equation (3.2) we have $\mathcal{T}_{\mu_k}v_k \leq \mathcal{T}_{\mu_{k+1}}v_k + \epsilon'_{k+1}$ (since $\mu_{k+1} \in \hat{\mathcal{G}}_{\epsilon'_{k+1}}(v_k)$) then:

$$\begin{aligned} b_k &\leq v_k - \mathcal{T}_{\mu_k}v_k + \epsilon'_{k+1}, \\ &= v_k - \epsilon_k - \underbrace{\mathcal{T}_{\mu_k}v_k}_{=\mathcal{T}_{\mu_k,\hat{\nu}_k}v_k} + \gamma P_{\mu_k,\hat{\nu}_k}\epsilon_k + \epsilon_k - \gamma P_{\mu_k,\hat{\nu}_k}\epsilon_k + \epsilon'_{k+1}, \\ &= v_k - \epsilon_k - \underbrace{\mathcal{T}_{\mu_k,\hat{\nu}_k}(v_k - \epsilon_k)}_{\mathcal{T}_{\mu_k,\hat{\nu}_k}\text{ is affine}} + (\mathcal{I} - \gamma P_{\mu_k,\hat{\nu}_k})\epsilon_k + \epsilon'_{k+1}. \end{aligned}$$

From Equation (3.3), we have $v_k - \epsilon_k = (\mathcal{T}_{\mu_k})^m v_{k-1}$. Thus,

$$\begin{aligned} b_k &\leq (\mathcal{T}_{\mu_k})^m v_{k-1} - \mathcal{T}_{\mu_k,\hat{\nu}_k}(\mathcal{T}_{\mu_k})^m v_{k-1} + x_k \quad \text{(Eq. (3.5))}\ , \\ &= (\mathcal{T}_{\mu_k})^m v_{k-1} - \mathcal{T}_{\mu_k,\hat{\nu}_k}\mathcal{T}_{\mu_k,\nu_k^{m-1}}...\mathcal{T}_{\mu_k,\nu_k^1}(\mathcal{T}_{\mu_k}v_{k-1}) + x_k \quad \text{(Eq. (3.5))}, \\ &\leq \mathcal{T}_{\mu_k,\hat{\nu}_k}\mathcal{T}_{\mu_k,\nu_k^{m-1}}...\mathcal{T}_{\mu_k,\nu_k^1}v_{k-1} - \mathcal{T}_{\mu_k,\hat{\nu}_k}\mathcal{T}_{\mu_k,\nu_k^{m-1}}...\mathcal{T}_{\mu_k,\nu_k^1}(\mathcal{T}_{\mu_k}v_{k-1}) + x_k \quad \text{(Eq. (3.4))}, \\ &= \gamma P_{\mu_k,\hat{\nu}_k}\gamma P_{\mu_k,\nu_k^{m-1}}...\gamma P_{\mu_k,\nu_k^1}(v_{k-1} - \mathcal{T}_{\mu_k}v_{k-1}) + x_k, \\ &\leq \gamma P_{\mu_k,\hat{\nu}_k}\gamma P_{\mu_k,\nu_k^{m-1}}...\gamma P_{\mu_k,\nu_k^1}b_{k-1} + x_k. \end{aligned}$$

To bound d_{k+1}, we decompose it in the three following terms:

$$\begin{aligned} d_{k+1} &= v_* - (\mathcal{T}_{\mu_{k+1}})^m v_k, \\ &= \underbrace{\mathcal{T}_{\mu_*}v_* - \mathcal{T}_{\mu_*}v_k}_{①} + \underbrace{\mathcal{T}_{\mu_*}v_k - \mathcal{T}_{\mu_{k+1}}v_k}_{②} \\ &\quad + \underbrace{\mathcal{T}_{\mu_{k+1}}v_k - (\mathcal{T}_{\mu_{k+1}})^m v_k}_{③}. \end{aligned}$$

In this equation, term ① can be upper-bounded as follows:

$$\begin{aligned}
&\mathcal{T}_{\mu_*} v_* - \mathcal{T}_{\mu_*} v_k \\
&= \mathcal{T}_{\mu_*} v_* - \mathcal{T}_{\mu_*,\tilde{\nu}_k} v_k \text{ with } \tilde{\nu}_k \text{ defined in Eq. (3.6)}, \\
&\leq \mathcal{T}_{\mu_*,\tilde{\nu}_k} v_* - \mathcal{T}_{\mu_*,\tilde{\nu}_k} v_k \text{ since } \forall \nu,\ \mathcal{T}_{\mu_*}. \leq \mathcal{T}_{\mu_*,\nu}. \\
&= \gamma P_{\mu_*,\tilde{\nu}_k}(v_* - v_k).
\end{aligned}$$

By definition ② is bounded by the greedy error:

$$\mathcal{T}_{\mu_*} v_k - \mathcal{T}_{\mu_{k+1}} v_k \leq \epsilon'_{k+1} \text{(3.2) with } \mu \leftarrow \mu_*,\ k \leftarrow k+1$$

Finally, bounding term ③ involves the b_k quantity:

$$\begin{aligned}
&\mathcal{T}_{\mu_{k+1}} v_k - (\mathcal{T}_{\mu_{k+1}})^m v_k, \\
&= \sum_{j=1}^{m-1} (\mathcal{T}_{\mu_{k+1}})^j v_k - (\mathcal{T}_{\mu_{k+1}})^{j+1} v_k,
\end{aligned}$$

$$\begin{aligned}
&= \sum_{j=1}^{m-1} (\mathcal{T}_{\mu_{k+1}})^j v_k - \mathcal{T}_{\mu_{k+1},\nu^j_{k+1}} ... \mathcal{T}_{\mu_{k+1},\nu^1_{k+1}} \mathcal{T}_{\mu_k+1} v_k, \\
&\leq \sum_{j=1}^{m-1} [\mathcal{T}_{\mu_{k+1},\nu^j_{k+1}} ... \mathcal{T}_{\mu_{k+1},\nu^1_{k+1}} v_k - \mathcal{T}_{\mu_{k+1},\nu^j_{k+1}} ... \mathcal{T}_{\mu_{k+1},\nu^1_{k+1}} \mathcal{T}_{\mu_k+1} v_k] \text{ see (3.4)}, \\
&= \sum_{j=1}^{m-1} \gamma P_{\mu_{k+1},\nu^j_{k+1}} ... \gamma P_{\mu_{k+1},\nu^1_{k+1}} (v_k - \mathcal{T}_{\mu_k+1} v_k), \\
&= \sum_{j=1}^{m-1} \gamma P_{\mu_{k+1},\nu^j_{k+1}} ... \gamma P_{\mu_{k+1},\nu^1_{k+1}} b_k.
\end{aligned}$$

Then d_{k+1} becomes:

$$\begin{aligned}
d_{k+1} &\leq \gamma P_{\mu_*,\tilde{\nu}_k}(v_* - v_k) + \epsilon'_{k+1} + \sum_{j=1}^{m-1} \gamma P_{\mu_{k+1},\nu^j_{k+1}} ... \gamma P_{\mu_{k+1},\nu^1_{k+1}} b_k, \\
&\leq \gamma P_{\mu_*,\tilde{\nu}_k}(v_* - v_k) + \gamma P_{\mu_*,\tilde{\nu}_k} \epsilon_k - P_{\mu_*,\tilde{\nu}_k} \epsilon_k + \epsilon'_{k+1} + \sum_{j=1}^{m-1} \gamma P_{\mu_{k+1},\nu^j_{k+1}} ... \gamma P_{\mu_{k+1},\nu^1_{k+1}} b_k, \\
&\leq \gamma P_{\mu_*,\tilde{\nu}_k}(v_* - (\underbrace{v_k - \epsilon_k}_{(\mathcal{T}_{\mu_k})^m v_{k-1}})) \underbrace{-P_{\mu_*,\tilde{\nu}_k} \epsilon_k + \epsilon'_{k+1}}_{y_k} + \sum_{j=1}^{m-1} \gamma P_{\mu_{k+1},\nu^j_{k+1}} ... \gamma P_{\mu_{k+1},\nu^1_{k+1}} b_k, \\
&\leq \gamma P_{\mu_*,\tilde{\nu}_k} d_k + y_k + \sum_{j=1}^{m-1} \gamma P_{\mu_{k+1},\nu^j_{k+1}} ... \gamma P_{\mu_{k+1},\nu^1_{k+1}} b_k.
\end{aligned}$$

Let us finally bound s_k:

$$\begin{aligned}
s_k &= (\mathcal{T}_{\mu_k})^m v_{k-1} - v_{\mu_k}, \\
&= (\mathcal{T}_{\mu_k})^m v_{k-1} - (\mathcal{T}_{\mu_k})^\infty v_{k-1}, \\
&= (\mathcal{T}_{\mu_k})^m v_{k-1} - (\mathcal{T}_{\mu_k})^m (\mathcal{T}_{\mu_k})^\infty v_{k-1}, \\
&= (\mathcal{T}_{\mu_k})^m v_{k-1} - (\mathcal{T}_{\mu_k,\overline{\nu}_k})^m (\mathcal{T}_{\mu_k})^\infty v_{k-1}, \text{ with } \overline{\nu}_k \text{ defined in (3.8)} \\
&\leq (\mathcal{T}_{\mu_k,\overline{\nu}_k})^m v_{k-1} - (\mathcal{T}_{\mu_k,\overline{\nu}_k})^m (\mathcal{T}_{\mu_k})^\infty v_{k-1} \text{ since (3.4)}, \\
&= (\gamma P_{\mu_k,\overline{\nu}_k})^m (v_{k-1} - (\mathcal{T}_{\mu_k})^\infty v_{k-1}), \\
&= (\gamma P_{\mu_k,\overline{\nu}_k})^m (\sum_{i=0}^{\infty} (\mathcal{T}_{\mu_k})^i v_{k-1} - (\mathcal{T}_{\mu_k})^i (\mathcal{T}_{\mu_k} v_{k-1})), \\
&\leq (\gamma P_{\mu_k,\overline{\nu}_k})^m \sum_{i=0}^{\infty} \gamma P_{\mu_k,\nu_k^i} ... \gamma P_{\mu_k,\nu_k^1} (v_{k-1} - \mathcal{T}_{\mu_k} v_{k-1}), \\
&\leq (\gamma P_{\mu_k,\overline{\nu}_k})^m \sum_{i=0}^{\infty} \gamma P_{\mu_k,\nu_k^i} ... \gamma P_{\mu_k,\nu_k^1} b_{k-1}.
\end{aligned}$$

□

From linear recursive relations of the kind of Lemma 3.1, Scherrer et al. (2012) show how to deduce a bound on the L_p-norm of l_k. This part of the proof being identical to that of Scherrer et al. (2012), we do not develop it here. For completeness however, we include it in Appendix 4.1 of this chapter.

Theorem 3.1. *Let ρ and σ be distributions over states. Let p, q and q' be such that $\frac{1}{q} + \frac{1}{q'} = 1$. Then, after k iterations, we have:*

$$\|l_k\|_{p,\rho} \leq \underbrace{\frac{2(\gamma - \gamma^k)(\mathcal{C}_q^{1,k,0})^{\frac{1}{p}}}{(1-\gamma)^2} \sup_{1 \leq j \leq k-1} \|\epsilon_j\|_{pq',\sigma}}_{\text{value update error}} + \underbrace{\frac{(1 - \gamma^k)(\mathcal{C}_q^{0,k,0})^{\frac{1}{p}}}{(1-\gamma)^2} \sup_{1 \leq j \leq k} \left\|\epsilon'_j\right\|_{pq',\sigma}}_{\text{greedy error}},$$
$$+ \underbrace{\frac{2\gamma^k}{1-\gamma} (\mathcal{C}_q^{k,k+1,0})^{\frac{1}{p}} \min(\|d_0\|_{pq',\sigma}, \|b_0\|_{pq',\sigma})}_{\text{contraction term}}.$$

where

$$\mathcal{C}_q^{l,k,d} = \frac{(1-\gamma)^2}{\gamma^l - \gamma^k} \sum_{i=l}^{k-1} \sum_{j=i}^{\infty} \gamma^j c_q(j+d), \tag{3.17}$$

with the following norm of a Radon-Nikodym derivative:

$$c_q(j) = \sup_{\mu_1,\nu_1,...,\mu_j,\nu_j} \left\| \frac{d(\rho P_{\mu_1,\nu_1} ... P_{\mu_j,\nu_j})}{d\sigma} \right\|_{q,\sigma}. \tag{3.18}$$

This bound is again similar to the one of Section 1.2.3. It involves three terms similar to the one of theorem 2.3. The value update error, the greedy error and the concentration are comparable to the previous terms of theorem 2.3. The only major difference is the definition of the concentrability coeficient which generalizes the one

for MDPs to zero-sum two-player MGs. The bound is consistent with previous analysis on MDPs (Section 1.2.3). Furthermore, if player 2 has no influence on the MG, the bound of theorem 3.1 is the one theorem 2.3. In addition, when p tends to infinity, the bound becomes:

$$\liminf_{k\to+\infty} \|l_k\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2}\epsilon + \frac{1}{(1-\gamma)^2}\epsilon',$$

where ϵ and ϵ' are respectively the sup of errors at the evaluation step and the sup of errors at the greedy step in ∞-norm. We thus recover the bounds computed by Patek (1997).

Remark 3.2. In the case of an MG with no discount factor but with an absorbing state the expression given in lemma 3.1 is still valid. Instead of having a γ-discounted transition kernel we would have a simple transition kernel. And if this transition kernel has the following property

$$\exists l, \quad \sup_{\mu_0,\nu_0,\ldots,\mu_l,\nu_l} | \left\| \prod_{i=0}^{l} P_{\mu_i,\nu_i} \right\|_\infty \leq \gamma < 1,$$

we could still have an upper bounds on the propagation of errors.

2 Empirical Evaluation

The analysis of error propagation presented in Section 1.2 is general enough to develop several implementations. From the moment one can control the error made at each iteration step, the bound presented in Theorem 3.1 applies.

2.1 Algorithm

In this section, we present the Approximate Generalized Policy Iteration-Q (AGPI-Q) algorithm which is an extension to MG of Fitted-Q. This algorithm is offline and uses the so-called state-action value function Q. The state-action value function extends the value function by adding two degrees of freedom for the first action of each player. More formally, the state-action value function $Q^{\mu,\nu}(s,a,b)$ is defined as

$$Q_{\mu,\nu}(s,a^1,a^2) = E[r(s,a^1,a^2)] + \sum_{s'\in S} p(s'|s,a^1,a^2)v_{\mu,\nu}(s').$$

We assume we are given some samples $((s_j,a_j^1,a_j^2),r_j,s'_j)_{j=1,\ldots,N}$ and an initial Q-function (here we chose the null function). As it is an instance of AGPI, each iteration of this algorithm is made of a greedy step and an estimation step. The algorithm is precisely described in Algorithm 17.

For the *greedy step*, the minimax policy for the maximizer on each matrix game defined by $(q_k(s'_j,a^1,a^2))_{a^1,a^2}$. In general, this step involves solving N linear programs; recall that in the case of a turn-based game this step reduces to finding a maximum.

Algorithm 17 AGPI - Q for Batch sample

Input: $((s_j, a_j^1, a_j^2), r_j, x'_j)_{j=1,...,N}$ some samples,
$q_0 = 0$ a Q-function,
$\mathcal{F}$ an hypothesis space
for k=1,2,...,K **do**
  Greedy step:
  for all j **do**
    $\bar{a}_j^1 = \text{argmax}_{\bar{a}} \min_{\bar{b}} q_{k-1}(s'_j, \bar{a}, \bar{b})$ (solving a matrix game)
  end for
  Evaluation step:
  $q_{k,0} = q_{k-1}$
  for i=1,...,m **do**
    for all j **do**
      $q_j = r(s_j, a_j^1, a_j^2) + \gamma \min_b q_{k,i-1}(s'_j, \bar{a}_j^1, b)$
    end for
    $q_{k,i} = \text{argmin}_{q \in \mathcal{F}} \sum_{j=1}^{N} l(q(s_j, a_j^1, a_j^2), q_j)$
    Where l is a loss function.
    $q_k = q_{k,m}$
  end for
end for

The *evaluation step* involves solving the MDP with an horizon m for the minimizer. This part is similar to fitted-Q iteration. At each step, we try to find the best fit over our hypothesis space for the next Q-function according to some loss function $l(x, y)$ (often, $l(x, y) = |x - y|^2$).

2.2 Analysis

For this algorithm, we have $\epsilon'_k = 0$ and ϵ_k the error made on q_k at each iteration. Let us note $\epsilon_{k,i}$ the error for fitting the Q-function on the feature space.

We have $q_{k,i+1} = \mathcal{B}_{\mu_k} q_{k,i} + \epsilon_i$. Let us define $\nu_{k,i}$ such as $\mathcal{B}_{\mu_k}^m q_{k,0} = \mathcal{B}_{\mu_k,\nu_{k,m-1}} ... \mathcal{B}_{\mu_k,\nu_{k,0}} q_{k,0}$. Furthermore we have $q_{k,i+1} \leq \mathcal{B}_{\mu_k,\nu_{k,i}} q_{k,i} + \epsilon_{k,i}$. On the one hand, we have:

$$\epsilon_k = q_{k,m} - \mathcal{B}_{\mu_k}^m q_{k,0}, \tag{3.19}$$

$$\leq \mathcal{B}_{\mu_k,\nu_{k,m-1}} ... \mathcal{B}_{\mu_k,\nu_{k,0}} q_{k,0} + \sum_{i=0}^{m-1} \mathcal{P}_{\mu_k,\nu_{k,m-1}} ... \mathcal{P}_{\mu_k,\nu_{k,i+1}} \epsilon_{k,i} - \mathcal{B}_{\mu_k,\nu_{k,m-1}} ... \mathcal{B}_{\mu_k,\nu_{k,0}} q_{k,0}, \tag{3.20}$$

$$\leq \sum_{i=0}^{m-1} \mathcal{P}_{\mu_k,\nu_{k,m-1}} ... \mathcal{P}_{\mu_k,\nu_{k,i+1}} \epsilon_{k,i}. \tag{3.21}$$

On the other hand (with $\tilde{\nu}_{k,i}$ as $q_{k,i+1} = \mathcal{B}_{\mu_k,\tilde{\nu}_{k,i}} q_{k,i} + \epsilon_{k,i}$), we have:

$$\epsilon_k = q_{k,m} - \mathcal{B}^m_{\mu_k} q_{k,0}, \tag{3.22}$$

$$\geq \mathcal{B}_{\mu_k,\tilde{\nu}_{k,m-1}} ... \mathcal{B}_{\mu_k,\tilde{\nu}_{k,0}} q_{k,0} + \sum_{i=0}^{m-1} \mathcal{P}_{\mu_k,\tilde{\nu}_{k,m-1}} ... \mathcal{P}_{\mu_k,\tilde{\nu}_{k,i+1}} \epsilon_{k,i} - \mathcal{B}_{\mu_k,\tilde{\nu}_{k,m-1}} ... \mathcal{B}_{\mu_k,\tilde{\nu}_{k,0}} q_{k,0}, \tag{3.23}$$

$$\geq \sum_{i=0}^{m-1} \mathcal{P}_{\mu_k,\tilde{\nu}_{k,m-1}} ... \mathcal{P}_{\mu_k,\tilde{\nu}_{k,i+1}} \epsilon_{k,i}. \tag{3.24}$$

From these inequalities, we can provide the following bound (the proof is given in Appendix 5):

$$\|l_k\|_{p,\rho} \leq \frac{2(\gamma - \gamma^k)(1-\gamma^m)}{(1-\gamma)^3} (\mathcal{C}_q^{1,k,0,m,0})^{\frac{1}{p}} \sup_{i,l} \|\epsilon_{i,l}\|_{pq',\sigma} \tag{3.25}$$

$$+ \frac{2\gamma^k}{1-\gamma} (\mathcal{C}_q^{k,k+1,0})^{\frac{1}{p}} \min(\|d_0\|_{pq',\sigma}, \|b_0\|_{pq',\sigma}), \tag{3.26}$$

with

$$\mathcal{C}_q^{l,k,l',k',d} = \tag{3.27}$$

$$\frac{(1-\gamma)^3}{(\gamma^l - \gamma^k)(\gamma^{l'} - \gamma^{k'})} \sum_{i=l}^{k-1} \sum_{i'=l'}^{k'-1} \sum_{j=i+i'}^{\infty} \gamma^j c_q(j+d). \tag{3.28}$$

2.3 Complexity analysis

At the *greedy step*, the algorithm solves N minimax equilibria for a zero-sum matrix game. This is usually done by linear programming (Section 2.2). For state s, the complexity of such an operation is the complexity of solving a linear program with $c_s = 1 + card(A^1(s)) + card(A^2(s))$ constrains and with $card(A^1(s))$ variables. Let us note $\mathcal{L}(c_s, card(A^1(s)))$ this complexity. Then, the complexity of this step is bounded by $N\mathcal{L}(c,a)$ (with $c = \sup_{s \in \{x'^1,...,x'^N\}} c_s$ and $a = \sup_{s \in \{x'^1,...,x'^N\}} card(A^1(s))$). Using the simplex method, $\mathcal{L}(c,a)$ may grow exponentially with c while with the interior point method, $\mathcal{L}(c,a)$ is $O(a^{3.5})$ (Karmarkar, 1984). The time to compute q_j in the *evaluation step* depends on finding a maximum over $A^2(x'^j)$. And the regression complexity to find $q_{k,i}$ depends on the regression technique. Let us note this complexity $\mathcal{R}(N)$. Finally, the complexity of this step is $m\mathcal{R}(N)$.

The overall complexity of an iteration is thus $O(N\mathcal{L}(c,a) + m\mathcal{R}(N))$; in general, the complexity of solving the linear program will be the limiting factor for games with a large amount of actions.

2.4 The Game of Alesia

Alesia (Perolat et al., 2015, Meyer et al., 1997), which resembles the Oshi-Zumo game in (Buro, 2004, Bošanský et al., 2016) (meaning "the pushing sumo"), is a two-player

board game where each player starts with N coins. They are positioned in the middle of a board with $2K + 1$ different positions (see Figure 3.1). At each round, each player chooses secretly a certain amount of coins to bid (let's say (a^0, a^1)). If the player's budget is not null, he has to bet at least one coin. If player 1's bid is larger (respectively smaller) than the one of player 2, player 1 (respectively player 1) moves toward his side of the board. If the bids are equal, then players will stay on their current position. In all cases, all bets is discounted from the budget. This process continues until players have no coins left or until one player reaches one side of the board. The final position determines the winner. The game ends with a draw if no one succeeded to reach his side of the board. If player 2 (respectively player 1) reaches his side of the board, the reward is $(-1, 1)$ (respectively $(1, -1)$).

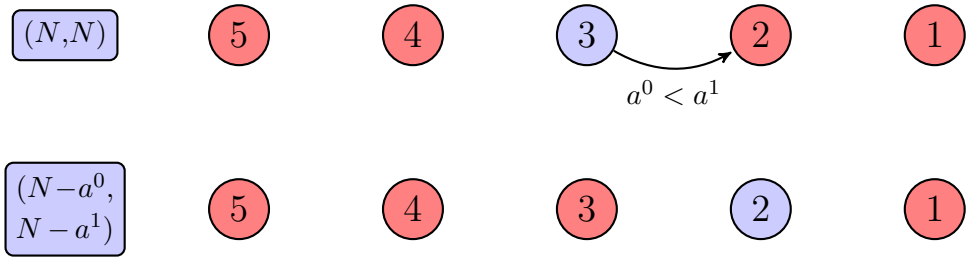


Figure 3.1 – Alesia rules for $K = 2$.

We perform the evaluation of AGPI-Q on the Alesia game described above. We assume that both players start with a budget $N = 20$ et $K = 2$. As a baseline, we use the exact solution of the problem provided by VI. We have run the algorithm for $k = 10$ iterations and for $m \in \{1, 2, 3, 4, 5\}$ evaluation steps. We have considered different sample set sizes, $n = 2500, 5000, 10000$. Each experiment is repeated 20 times. First, we generate n uniform samples (s_j) over the state space. Then, for each state, we draw uniformly the actions of each player in the set of their own action space in that state a_j^1, a_j^2, $r_j = r(s_j, a_j^1, a_j^2)$ and compute the next state s'_j. As hypothesis space, we use CART trees (Breiman et al., 1984) which exemplifies the non-parametric property of the algorithm.

The performance of the algorithm is measured as the mean-squared error between the value function $v_k(s) = \min_{\overline{b}} \max_{\overline{a}} q_k(s, \overline{a}, \overline{b})$ where q_k is the output of the algorithm AGPI-Q and the actual value function computed via VI. Figure 3.2 shows the evolution of performance along iterations for $n = 10000$ for the different values of the parameter m. Figure 3.3 shows the exact value function (Fig. 3.3(a)) and the approximated one v_k (Fig. 3.3(b)). The complete list of experiments results can be found in the appendix, especially for different size of sample set $n = 2500$ and $n = 5000$.

For each size of sample set, the asymptotic convergence is better for small values of m. This conforms to Eq. (3.26), in which the term $\frac{2(\gamma-\gamma^k)(1-\gamma^m)}{(1-\gamma)^3}$ increases with m. However, for small values of k, the mean-squared error is reducing when m is increasing. This is consistent with experimental results when using MPI for MDP: the bigger m, the higher the convergence rate. The price to pay for this acceleration of convergence towards the optimal value is an heavier evaluation step. This is similar to results in

the exact case (Puterman, 1994). Overall, this suggests using large values of m at the beginning of the algorithm and reducing the values of m as k grows to get a smaller asymptotic error.

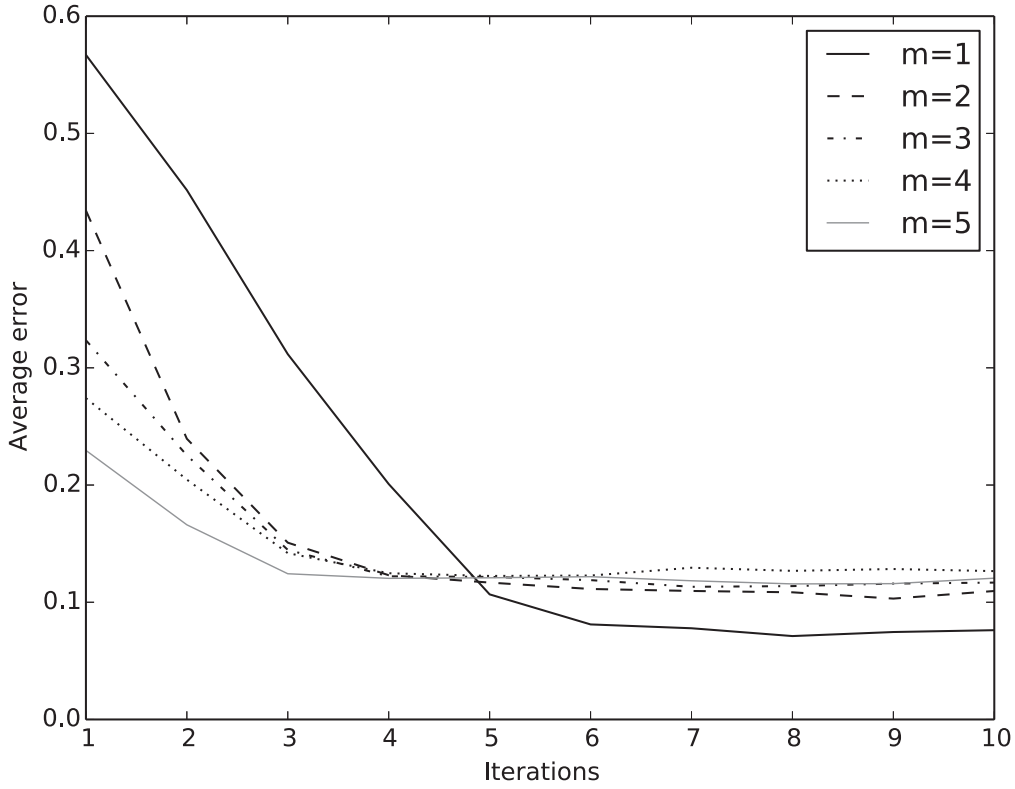
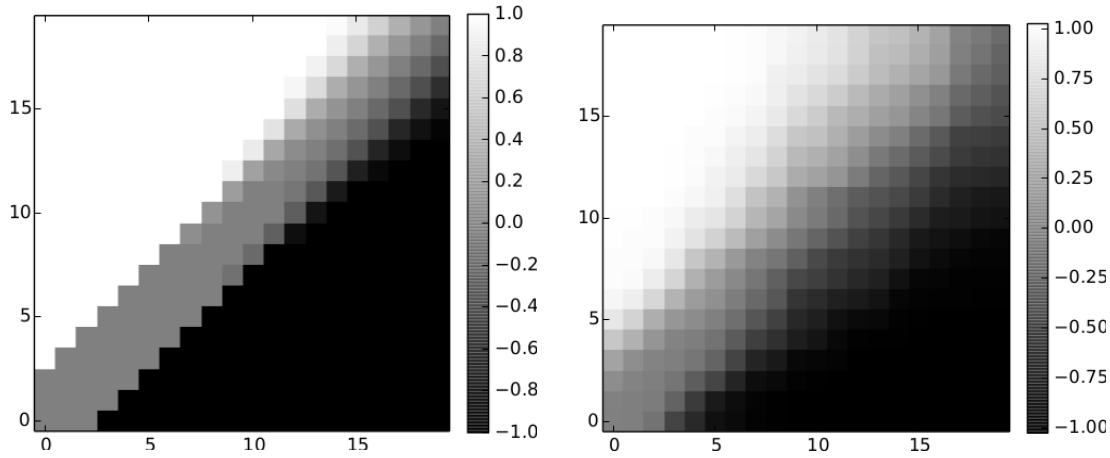


Figure 3.2 – Mean-squared error (y-axis) between the estimated value function and the true value function at step k (x-axis). For $N = 20$ and $n = 10000$

3 Conclusion and Perspectives

This work provides a novel and unified error propagation analysis in L_p-norm of well-known algorithms (API, AVI and AGPI) for zero-sum two-player MGs. It extends the error propagation analyses of Scherrer et al. (2012) for MDPs to zero-sum two-player MGs and of Patek (1997) which is an L_∞-norm analysis for only API. In addition, we provide a practical algorithm (AGPI-Q) which learns a good approximation of the Nash Equilibrium from batch data provided in the form of transitions sampled from actual games (the dynamics is not known). This algorithm is an extension of Fitted-Q for zero-sum two-player MGs and can thus be non-parametric. No features need to be provided or hand-crafted for each different application which is a significant advantage.



(a) Exact value for the Alesia game for player's budget 0...20 and for token position 3

(b) Value computed by AGPI-Q for the game of Alesia for player's budget 0...20 and for token position 3

Figure 3.3 – Value functions at token position 3

Finally, we empirically demonstrate that AGPI-Q performs well on a simultaneous two-player game, namely Alesia.

It appears that the provided bound is highly sensitive to γ (which is a common problem of ADP). This is critical and further work should concentrate on reducing the impact of γ in the final error bound. One solution explored in the next chapter is the use of non stationary strategies as they reduce the γ-sensitivity in the case of MDPs.

4 Appendix: Demonstration of Lemma 3.2

The following proof is similar to the one for MDP Scherrer et al. (2012) we write it for the sake of completeness. Furthermore, equation (3.34) is the starting point of the proof of (3.26).

We will use the same abusive but simplifying notation as in (Scherrer et al., 2012). We will note Γ^n any product of n discounted transition Kernel. Then Γ^n represents the set $\{\gamma P_{\mu_1,\nu_1}, \ \dots \ , \gamma P_{\mu_n,\nu_n}, \text{ with } \mu_i, \nu_i \text{ random strategies}\}$. Then Γ^n is the represent a class of discounted stochastic matrix. One should read Γ^n as there exists $\gamma\mathcal{P}_1, ..., \gamma\mathcal{P}_n$ such as Γ^n represents the product $\gamma\mathcal{P}_1 ... \gamma\mathcal{P}_n$. For example we have the following property:

$$\alpha_1\Gamma^i\alpha_2\Gamma^j + \alpha_3\Gamma^k = \alpha_1\alpha_2\Gamma^{i+j} + \alpha_3\Gamma^k.$$

We can rewrite Lemma 3.1 in this Simple way:

$$b_k \leq \Gamma^m b_{k-1} + x_k, \tag{3.29}$$

$$d_{k+1} \leq \Gamma d_k + y_k + \sum_{j=1}^{m-1} \Gamma^j b_k, \tag{3.30}$$

$$s_k \leq \Gamma^m (\sum_{i=1}^{\infty} \Gamma^i b_{k-1}). \tag{3.31}$$

We prove these three inequalities:

$$b_k \leq \sum_{i=1}^{k} \Gamma^{m(k-i)} x_k + \Gamma^{mk} b_0$$

$$d_k \leq \sum_{i=1}^{k} \Gamma^{i-1} y_{k-i} + \sum_{i=1}^{k-1} \sum_{j=i}^{mi-1} \Gamma^j x_{k-i} + \underbrace{\sum_{i=k}^{mk-1} \Gamma^i b_0 + \Gamma^k d_0}_{z_k}$$

$$s_k \leq \sum_{i=1}^{k-1} \sum_{j=mi}^{\infty} \Gamma^{j+m(k-i)} x_i + \underbrace{(\sum_{j=mk}^{\infty} \Gamma^j) b_0}_{z'_k}$$

First inequality. After expanding (3.29) we get:

$$b_k \leq \sum_{i=1}^{k} \Gamma^{m(k-i)} x_k + \Gamma^{mk} b_0 \tag{3.32}$$

Second inequality :

$$\begin{aligned}
d_k &\leq \Gamma^k d_0 + \sum_{j=0}^{k-1} \Gamma^{k-1-j}(y_j + (\sum_{l=1}^{m-1} \Gamma^l) b_j) \\
&\leq \Gamma^k d_0 + \sum_{j=0}^{k-1} \Gamma^{k-1-j}[y_j + (\sum_{l=1}^{m-1} \Gamma^l)(\sum_{i=1}^{j} \Gamma^{m(j-i)} x_i + \Gamma^{mj} b_0)] \\
&= \sum_{i=1}^{k} \Gamma^{i-1} y_{k-i} + \sum_{j=0}^{k-1} \sum_{l=1}^{m-1} \sum_{i=1}^{j} \Gamma^{k-1-j+l+m(j-i)} x_i + z_k \\
&= \sum_{i=1}^{k} \Gamma^{i-1} y_{k-i} + \sum_{i=1}^{k-1} \sum_{j=i}^{k-1} \sum_{l=1}^{m-1} \Gamma^{k-1+l+j(m-1)-mi} x_i + z_k \\
&= \sum_{i=1}^{k} \Gamma^{i-1} y_{k-i} + \sum_{i=1}^{k-1} \sum_{j=k-i}^{m(k-i)-1} \Gamma^j x_i + z_k \\
&= \sum_{i=1}^{k} \Gamma^{i-1} y_{k-i} + \sum_{i=1}^{k-1} \sum_{j=i}^{mi-1} \Gamma^j x_{k-i} + z_k
\end{aligned}$$

With z_k:

$$\begin{aligned}
z_k &= \sum_{j=0}^{k-1} \sum_{l=1}^{m-1} \Gamma^{k-1-j+l+mj} b_0 + \Gamma^k d_0 \\
&= \sum_{i=k}^{mk-1} \Gamma^i b_0 + \Gamma^k d_0
\end{aligned}$$

last inequality (we replace b_{k-1} with it's expression (3.32)):

$$s_k \leq \sum_{i=1}^{k-1} \sum_{j=mi}^{\infty} \Gamma^{j+m(k-i)} x_i + \underbrace{(\sum_{j=mk}^{\infty} \Gamma^j) b_0}_{z'_k}$$

And finally bounding l_k

$$l_k = d_k + s_k \tag{3.33}$$

$$\leq \underbrace{\sum_{i=1}^{k} \Gamma^{i-1} y_{k-i} + \sum_{i=1}^{k-1} \sum_{j=i}^{\infty} \Gamma^j x_{k-i}}_{\text{Depend on errors}} + \underbrace{z_k + z'_k}_{\text{Depend on initial condition}} \tag{3.34}$$

$$\eta_k = z_k + z'_k = \sum_{i=k}^{\infty} \Gamma^i b_0 + \Gamma^k d_0$$

Then we prove the following lemma which is similar to that for MDP given by Scherrer et al. (2012):

Lemma 3.2. $\forall k \geq 1$

$$|l_k| \leq 2\sum_{i=1}^{k-1}\sum_{j=i}^{\infty}\Gamma^j\,|\epsilon_{k-i}| + \sum_{i=0}^{k-1}\sum_{j=i}^{\infty}\Gamma^j\left|\epsilon'_{k-i}\right| + h(k), \tag{3.35}$$

$$with\ h(k) = 2\sum_{i=k}^{\infty}\Gamma^i\,|b_0| \ \ or\ 2\sum_{i=k}^{\infty}\Gamma^i\,|d_0|\,. \tag{3.36}$$

Relation between b_0 and d_0:

$$\begin{aligned}
b_0 &= v_0 - \mathcal{T}_{\mu_1}v_0 \\
&= v_0 - v_* + \mathcal{T}_{\mu_*}v_* - \mathcal{T}_{\mu_*}v_0 + \mathcal{T}_{\mu_*}v_0 - \mathcal{T}_{\mu_1}v_0 \\
&\leq -d_0 + \gamma P_{\mu_*,\tilde{\nu}_0}d_0 + \epsilon'_1 \\
&\leq (\mathcal{I} - \gamma P_{\mu_*,\tilde{\nu}_0})(-d_0) + \epsilon'_1 \\
d_0 &\leq (\mathcal{I} - \gamma P_{\mu_*,\tilde{\nu}_0})^{-1}(\epsilon'_1 - b_0)
\end{aligned}$$

One should notice that in a discrete state space $P_{\mu,\nu}$ is a stochastic matrix. Then $\mathcal{I} - \gamma P_{\mu_*,\tilde{\nu}_0}$ is invertible since $\gamma P_{\mu_*,\tilde{\nu}_0}$ has a spectral radius < 1.

then:

$$\begin{aligned}
|\eta_k| &\leq \sum_{i=k}^{\infty}\Gamma^i((\mathcal{I} - \gamma P_{\mu_*,\tilde{\nu}_0})\,|d_0| + |\epsilon'_1|) + \Gamma^k\,|d_0| \\
&\leq \sum_{i=k}^{\infty}\Gamma^i((\mathcal{I} + \Gamma)\,|d_0| + |\epsilon'_1|) + \Gamma^k\,|d_0| \\
&\leq 2\sum_{i=k}^{\infty}\Gamma^i\,|d_0| + \sum_{i=k}^{\infty}\Gamma^i\,|\epsilon'_1|
\end{aligned}$$

and also:

$$\begin{aligned}
|\eta_k| &\leq \sum_{i=k}^{\infty}\Gamma^i\,|b_0| + \Gamma^k(\mathcal{I} - \gamma P_{\mu_*,\tilde{\nu}_0})^{-1}(|\epsilon'_1| + |b_0|) \\
&\leq \sum_{i=k}^{\infty}\Gamma^i\,|b_0| + \Gamma^k\sum_{i=0}^{\infty}\Gamma(|\epsilon'_1| + |b_0|) \\
&\leq 2\sum_{i=k}^{\infty}\Gamma^i\,|b_0| + \sum_{i=k}^{\infty}\Gamma^i\,|\epsilon'_1|
\end{aligned}$$

because l_k is positive we have then:

$$
\begin{aligned}
|l_k| &\leq \sum_{i=1}^{k} \Gamma^{i-1} |y_{k-i}| + \sum_{i=1}^{k-1} \sum_{j=i}^{\infty} \Gamma^j |x_{k-i}| + |\eta_k| \\
&\leq \sum_{i=1}^{k} \Gamma^{i-1} (\Gamma |\epsilon_{k-i}| + \left|\epsilon'_{k+1-i}\right|) + \sum_{i=1}^{k-1} \sum_{j=i}^{\infty} \Gamma^j ((\mathcal{I} + \Gamma) |\epsilon_{k-i}| + \left|\epsilon'_{k+1-i}\right|) + |\eta_k| \text{ note that } \epsilon_0 = 0 \\
&\leq \sum_{i=1}^{k-1} (\Gamma^i + \sum_{j=i}^{\infty} (\Gamma^j + \Gamma^{j+1})) |\epsilon_{k-i}| + \sum_{i=1}^{k-1} (\Gamma^{i-1} + \sum_{j=i}^{\infty} \Gamma^j) \left|\epsilon'_{k+1-i}\right| + \Gamma^{k-1} |\epsilon'_1| + \sum_{i=k}^{\infty} \Gamma^i |\epsilon'_1| + h(k) \\
&\leq 2 \sum_{i=1}^{k-1} \sum_{j=i}^{\infty} \Gamma^j |\epsilon_{k-i}| + \sum_{i=0}^{k-1} \sum_{j=i}^{\infty} \Gamma^j \left|\epsilon'_{k-i}\right| + h(k) \text{ with } h(k) = 2 \sum_{i=k}^{\infty} \Gamma^i |b_0| \text{ or } 2 \sum_{i=k}^{\infty} \Gamma^i |d_0|
\end{aligned}
\tag{3.37}
$$

4.1 Demonstration of Theorem 3.1

Let us recall Scherrer's lemma (demonstration can be found in Scherrer et al. (2012)).

Lemma 3.3. *Let $\mathcal{I}$ and $(\mathcal{J}_i)_{i \in \mathcal{I}}$ be a sets of positive integers, $\{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ a partition of $\mathcal{I}$. Let f and $(g_i)_{i \in \mathcal{I}}$ be function such as:*

$$
|f| \leq \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}_i} \Gamma^j |g_i| = \sum_{l=1}^{n} \sum_{i \in \mathcal{I}_l} \sum_{j \in \mathcal{J}_i} \Gamma^j |g_i| .
$$

Then for all p, q and q' such as $\frac{1}{q} + \frac{1}{q'} = 1$ and for all distribution ρ and σ we have

$$
\|f\|_{p,\rho} \leq \sum_{l=1}^{n} (\mathcal{C}_q(l))^{\frac{1}{p}} \sup_{i \in \mathcal{I}_l} \|g_i\|_{pq',\sigma} \sum_{i \in \mathcal{I}_l} \sum_{j \in \mathcal{J}_i} \gamma^j .
$$

with the concentrability coefficient written:

$$
\mathcal{C}_q(l) = \frac{\sum_{i \in \mathcal{I}_l} \sum_{j \in \mathcal{J}_i} \gamma^j c_q(j)}{\sum_{i \in \mathcal{I}_l} \sum_{j \in \mathcal{J}_i} \gamma^j} .
$$

Theorem 3.1 can be proven by applying lemma 4.2 with:

$$
\begin{aligned}
&\mathcal{I} = \{1, ..., 2k\} \\
&\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3\},\ \mathcal{I}_1 = \{1, 2, ..., k-1\},\ \mathcal{I}_2 = \{k, ..., 2k-1\},\ \mathcal{I}_3 = \{2k\}
\end{aligned}
$$

$$\forall i \in \mathcal{I}_1$$
$$g_i = 2\epsilon_{k-i}$$
$$\mathcal{J}_i = \{i, i+1, ...\}$$

$$\forall i \in \mathcal{I}_2$$
$$g_i = \epsilon'_{k-(i-k)}$$
$$\mathcal{J}_i = \{i-k, i-k+1, ...\}$$

$$\forall i \in \mathcal{I}_3$$
$$g_i = 2d_0 \text{ or } 2b_0$$
$$\mathcal{J}_i = \{k, k+1, ...\}$$

5 Appendix: Bound for AGPI-Q

From (3.34) and with $x_k = (\mathcal{I} - \Gamma)\epsilon_k$ and $y_k = -\Gamma\epsilon_k$ we can compute:

$$l_k = d_k + s_k \tag{3.38}$$

$$\leq \sum_{i=1}^{k} \Gamma^{i-1}\Gamma(-\epsilon_{k-i}) + \sum_{i=1}^{k-1}\sum_{j=i}^{\infty} \Gamma^j(\mathcal{I} - \Gamma)\epsilon_{k-i} + z_k + z'_k \tag{3.39}$$

$$\leq \sum_{i=1}^{k} \Gamma^{i-1}\Gamma\left[\sum_{l=0}^{m-1} \Gamma^{m-l-1}(-\epsilon_{k-i,l})\right] + \sum_{i=1}^{k-1}\sum_{j=i}^{\infty} \Gamma^j(\mathcal{I} - \Gamma)\left[\sum_{l=0}^{m-1} \Gamma^{m-l-1}\epsilon_{k-i,l}\right] + z_k + z'_k \tag{3.40}$$

In (3.39) we use (3.24) to bound $-\epsilon_{k-i}$ and (3.21) to bound ϵ_{k-i} Since $l_k \geq 0$:

$$|l_k| \leq \sum_{i=1}^{k} \Gamma^{i}\left[\sum_{l=0}^{m-1} \Gamma^{m-l-1}\left|\epsilon_{k-i,l}\right|\right] + \sum_{i=1}^{k-1}\sum_{j=i}^{\infty} \Gamma^j(\mathcal{I} + \Gamma)\left[\sum_{l=0}^{m-1} \Gamma^{m-l-1}\left|\epsilon_{k-i,l}\right|\right] + h(k) \text{ with } \epsilon_{0,l} = 0 \tag{3.41}$$

$$\leq 2\sum_{i=1}^{k-1}\sum_{j=i}^{\infty} \Gamma^{j}\left[\sum_{l=0}^{m-1} \Gamma^{l}\left|\epsilon_{k-i,m-l-1}\right|\right] + h(k) \tag{3.42}$$

$$\leq 2\sum_{i=1}^{k-1}\sum_{l=0}^{m-1}\sum_{j=i+l}^{\infty} \Gamma^{j}\left|\epsilon_{k-i,m-l-1}\right| + h(k) \tag{3.43}$$

with lemma 4.2 applied to:

$$\mathcal{I} = \{1, ..., (k-1)m + 1\}$$
$$\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2\},\ \mathcal{I}_1 = \{1, 2, ..., (k-1)m\},\ \mathcal{I}_2 = \{(k-1)m + 1\}$$

$$\forall \xi \in \mathcal{I}_1,\ \xi = i + (k-1)l$$
$$g_\xi = 2\epsilon_{k-i,m-l-1}$$
$$\mathcal{J}_\xi = \{i + l, i + l + 1, ...\}$$

$$\forall i \in \mathcal{I}_2$$
$$g_i = 2d_0 \text{ or } 2b_0$$
$$\mathcal{J}_i = \{k, k + 1, ...\}$$

The result is:

$$\|l_k\|_{p,\rho} \leq \frac{2(\gamma - \gamma^k)(1 - \gamma^m)}{(1 - \gamma)^3}(\mathcal{C}_q^{1,k,0,m,0})^{\frac{1}{p}} \sup_{i,l} \|\epsilon_{i,l}\|_{pq',\sigma} + \frac{2\gamma^k}{1 - \gamma}(\mathcal{C}_q^{k,k+1,0})^{\frac{1}{p}} \min(\|d_0\|_{pq',\sigma}, \|b_0\|_{pq',\sigma}).$$

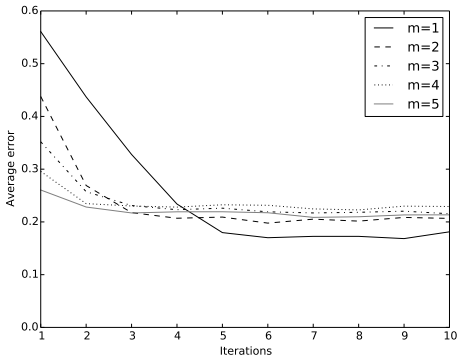
With:

$$\mathcal{C}_q^{l,k,l',k',d} = \frac{(1 - \gamma)^3}{(\gamma^l - \gamma^k)(\gamma^{l'} - \gamma^{k'})} \sum_{i=l}^{k-1}\sum_{i'=l'}^{k'-1}\sum_{j=i+i'}^{\infty} \gamma^j c_q(j + d). \tag{3.44}$$

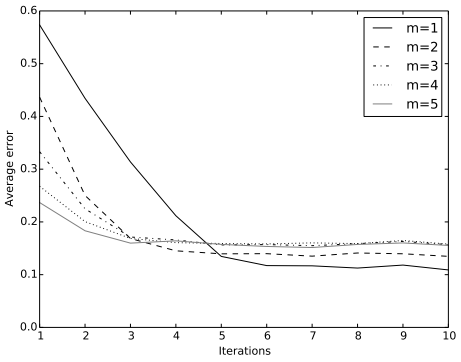
6 Appendix: Experiments

Results for experiments for $N = 2500$ (Figure 3.4(a)) and $N = 5000$ (Figure 3.4(b))

6.1 Mean square error between approximate value and exact value



(a) Mean square error (y-axis) between the estimated value function and the true value function at step k (x-axis). For $n = 20$ and $N = 2500$



(b) Mean square error (y-axis) between the estimated value function and the true value function at step k (x-axis). For $n = 20$ and $N = 5000$

6.2 Exact value function, approximate value function and error for $N = 10000$

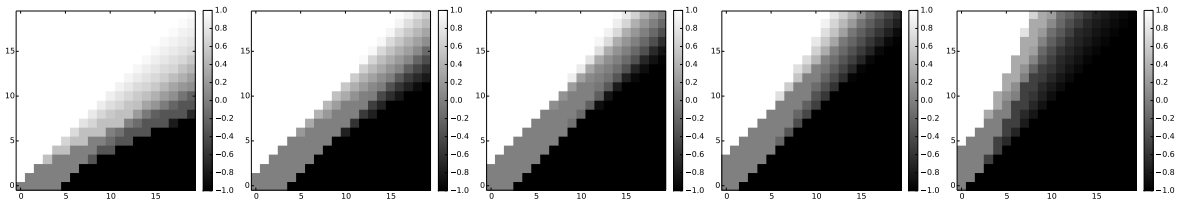


Figure 3.4 – Exact value for the game of ALESIA. From left to right the value of the game (for player's budget = 0...20) for token's position 5 to 1

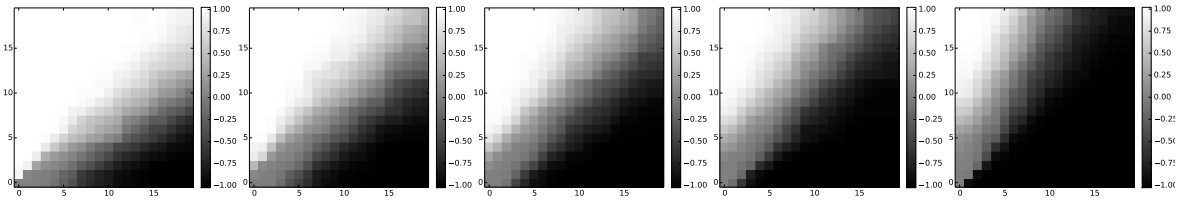


Figure 3.5 – Value computed by AGPI-Q for the game of ALESIA. From left to right the approximate value of the game (for player's budget = 0...20) for token's position 5 to 1

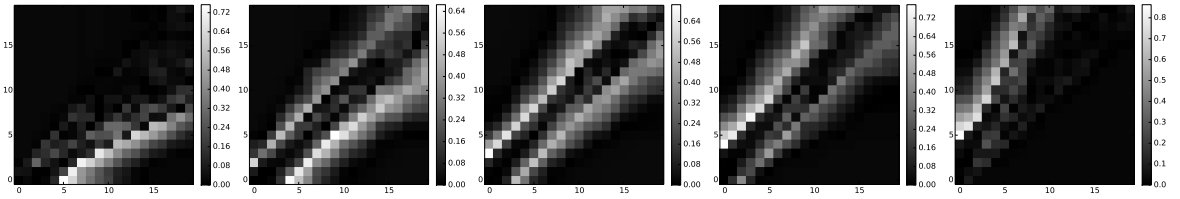


Figure 3.6 – Absolute difference between value computed by AGPI-Q and the exact value. From left to right the error of the game (for player's budget = 0...20) for token's position 5 to 1

Chapter 4

Improved bounds using non-stationary Strategies

In this chapter, we explore a solution to decrease the γ-sensitivity of classical ADP algorithms (Perolat et al., 2016). Section 1.2.3 describes how the use of non-stationary policies can be leveraged in MDPs to reduce the γ-sensitivity from $\frac{2(\gamma-\gamma^k)}{(1-\gamma)^2}$ to $\frac{2(\gamma-\gamma^k)}{(1-\gamma)(1-\gamma^m)}$ or to improve the concentrability coefficient. In that section, we explore how similar techniques can be used in zero-sum two-player MGs. From the point of view of player 1, the principle is the same. Instead of playing a stationary strategy μ_K given by some algorithm, the player will play a cyclic strategy $(\mu_K, \mu_{K-1}, \dots, \mu_{K-m+1}, \mu_K, \mu_{K-1}, \dots)$. To ease the notations, we will write such a strategy $\mu_K, \mu_{K-1}, \dots, \mu_{K-m+1} = \mu_{k,m}$. Using cyclic strategies in zero-sum two-player MGs provides a similar reduction in the γ-sensitivity of the algorithm as the one existing for MDPs. This chapter is organized in three parts. First, it provides definitions and properties required for this chapter. Then, we will present four ADP schemes using non-stationary strategies and provide their sensitivity analysis. Finally, an empirical evaluation of most algorithms is conducted in the last part of this chapter.

1 Non-Stationary Strategy in Zero-Sum Two-Player Markov Games

The value of a joint non-stationary strategy was defined in 2.53. If player 1 plays a non-stationary strategy of length M (a tuple $(\mu_0, \mu_1, \dots, \mu_{M-1})$), the value $(v_{\mu_0,\mu_1,\dots,\mu_{M-1}})$ of this non-stationary strategy is the expected cumulative γ-discounted reward the player gets when his adversary plays the optimal counter strategy. Formally:

$$v_{\mu_0,\mu_1,\dots,\mu_{M-1}}(s) = \min_{(\nu_t)_{t\in\mathbb{N}}} E\left[\sum_{t=0}^{+\infty} \gamma^t r_{\mu_i,\nu_t}(s_t)|s_0 = s,\ s_{t+1} \sim P_{\mu_i,\nu_t}(.|s_t),\ i = t\ mod(M)\right]. \tag{4.1}$$

In other words, the strategy used in state s and at time t will depend on t. Instead of always following a single strategy the player will follow a cyclic strategy. At time $t = 0$ the player will play μ_0, at time $t = 1$ he will play μ_1 and at time $t = Mj + i$ $(\forall i \in \{0, \dots, M-1\}, \forall j \in N)$ he will play strategy μ_i. This value is the fixed point of

the operator $\mathcal{T}_{\mu_0} \dots \mathcal{T}_{\mu_{M-1}}$. Thus, we have:

$$v_{\mu_0,\mu_1,\dots,\mu_{M-1}} = \mathcal{T}_{\mu_0} \dots \mathcal{T}_{\mu_{M-1}} v_{\mu_0,\mu_1,\dots,\mu_{M-1}}. \tag{4.2}$$

Proof. This property might appear intuitive but its proof remains technical. We want to show:

$$v_{\mu_0,\mu_1,\dots,\mu_{M-1}}(s) = \min_{(\nu_t)_{t\in\mathbb{N}}} E\left[\sum_{t=0}^{+\infty} \gamma^t r_{\mu_i,\nu_t}(s_t) | s_0 = s,\ s_{t+1} \sim P_{\mu_i,\nu_t}(.|s_t),\ i = t\ [M]\right].$$

Is the fixed point of operator $\mathcal{T}_{\mu_0} \dots \mathcal{T}_{\mu_{M-1}}$.

This property seems intuitive. However, it's demonstration is non standard. First we build an MDP with a state space of size $M \times |S|$. Then we prove the value we are interested in is a sub-vector of the optimal value of the large MDP. Finally we prove the sub-vector is a fixed point of $\mathcal{T}_{\mu_0} \dots \mathcal{T}_{\mu_{M-1}}$.

Let us define the following MDP with the same γ:
$\tilde{S} = S \times \{0, \dots, M-1\}$.
For $(s,i) \in \tilde{S}$ we have $\tilde{A}((s,i)) = A(s) \times \{i\}$.

$$\tilde{p}((s',i)|(s,j),(a^2,j)) = \delta_{\{i=j+1[M]\}} \sum_{a^1 \in A^1(s)} \mu_j(a^1|s) p(s'|s,a^1,a^2) \tag{4.3}$$

$$\tilde{r}((s,j),(a^2,j)) = \sum_{a^1 \in A^1(s)} \mu_j(a^1|s) r(s,a^1,a^2) \tag{4.4}$$

The Kernel and reward are defined as follow:

$$\tilde{\mathcal{P}}_{\tilde{\nu}}((s',i)|(s,j)) = E_{a^2 \sim \tilde{\nu}(.|(s,j))}[\tilde{p}((s',i)|(s,j),(a^2,j))]$$

One should notice that $(s',i) \sim \tilde{\mathcal{P}}_{\tilde{\nu}}(.|(s,j))$ then $i = j+1[M]$ (obvious consequence of (4.3))

$$\tilde{r}_{\tilde{\nu}}((s,j)) = E_{a^2 \sim \tilde{\nu}(.|(s,j))}[\tilde{r}((s,j),(a^2,j))]$$

Instead of trying to maximize we will try to minimize the cumulated γ-discounted reward. Let $\tilde{v}^*$ be the optimal value of that MDP:

$$\tilde{v}^*(\tilde{s}) = \min_{(\tilde{\nu}_t)_{t\in\mathbb{N}}} E[\sum_{t=0}^{+\infty} \gamma^t \tilde{r}_{\tilde{\nu}_t}(\tilde{s}_t) | s_0 = \tilde{s},\ \tilde{s}_{t+1} \sim \tilde{\mathcal{P}}_{\tilde{\nu}_t}(.|\tilde{s}_t)] \tag{4.5}$$

then:

$$\tilde{v}^*((s,0)) \tag{4.6}$$

$$= \min_{(\tilde{\nu}_t)_{t\in\mathbb{N}}} E[\sum_{t=0}^{+\infty} \gamma^t \tilde{r}_{\tilde{\nu}_t}(\tilde{s}_t)|s_0 = (s,0),\ \tilde{s}_{t+1} \sim \tilde{\mathcal{P}}_{\tilde{\nu}_t}(.|\tilde{s}_t)], \tag{4.7}$$

$$= \min_{(\tilde{\nu}_t)_{t\in\mathbb{N}}} E[\sum_{t=0}^{+\infty} \gamma^t \tilde{r}_{\tilde{\nu}_t}((s_t,i))|s_0 = (s,0),\ \tilde{s}_{t+1} \sim \tilde{\mathcal{P}}_{\tilde{\nu}_t}(.|(s_t,i)),\ i = t\ [M]], \tag{4.8}$$

$$= \min_{(\tilde{\nu}_t)_{t\in\mathbb{N}}} E[\sum_{t=0}^{+\infty} \gamma^t \tilde{r}_{\tilde{\nu}_t}((s_t,i))|s_0 = (s,0),\ s_{t+1} \sim P_{\mu_i,\nu_t}(.|s_t),\ i = t\ [M],\ \nu_t(.|s) = \tilde{\nu}_t(.|(s,j))], \tag{4.9}$$

$$= \min_{(\tilde{\nu}_t)_{t\in\mathbb{N}}} E[\sum_{t=0}^{+\infty} \gamma^t r_{\mu_i,\nu_t}(s_t)|s_0 = (s,0),\ s_{t+1} \sim P_{\mu_i,\nu_t}(.|s_t),\ i = t\ [M],\ \nu_t(.|s) = \tilde{\nu}_t(.|(s,j))], \tag{4.10}$$

$$= \min_{(\nu_t)_{t\in\mathbb{N}}} E[\sum_{t=0}^{+\infty} \gamma^t r_{\mu_i,\nu_t}(s_t)|s_0 = s,\ s_{t+1} \sim P_{\mu_i,\nu_t}(.|s_t),\ i = t\ [M]], \tag{4.11}$$

$$= v_{\mu_0,\mu_1,\ldots,\mu_{M-1}}(s). \tag{4.12}$$

Let $\tilde{v}$ be a value function and let $\tilde{v}_i, i \in \{0,\ldots,M-1\}$ be the restriction to $S \times \{i\}$ of $\tilde{v}$.

From (4.3) we also have: $\mathcal{B}_{\tilde{\nu}}\tilde{v} = \tilde{r}_{\tilde{\nu}} + \tilde{\mathcal{P}}_{\tilde{\nu}}\tilde{v} = \begin{pmatrix} \mathcal{T}_{\mu_0,\nu^0}\tilde{v}_1 \\ \mathcal{T}_{\mu_1,\nu^1}\tilde{v}_2 \\ \vdots \\ \mathcal{T}_{\mu_{M-2},\nu^{M-2}}\tilde{v}_{M-1} \\ \mathcal{T}_{\mu_{M-1},\nu^{M-1}}\tilde{v}_0 \end{pmatrix}$ where $\nu^j(.|s) \sim \tilde{\nu}(.|(s,j))$ thus, we have: $\mathcal{B}\tilde{v} = \min_{\tilde{\nu}}(\tilde{r}_{\tilde{\nu}} + \tilde{\mathcal{P}}_{\tilde{\nu}}\tilde{v}) = \begin{pmatrix} \mathcal{T}_{\mu_0}\tilde{v}_1 \\ \mathcal{T}_{\mu_1}\tilde{v}_2 \\ \vdots \\ \mathcal{T}_{\mu_{M-2}}\tilde{v}_{M-1} \\ \mathcal{T}_{\mu_{M-1}}\tilde{v}_0 \end{pmatrix}$

But from basic property of dynamic programming we have:

$$\mathcal{B}^M\tilde{v}^* = \tilde{v}^*$$

and finally, from the definition of $\mathcal{B}$ and from (4.12) we have:

$$(\mathcal{T}_{\mu_0}\ldots\mathcal{T}_{\mu_{M-1}}\tilde{v}_0)((s,0)) = \tilde{v}_0((s,0)) = v_{\mu_0,\mu_1,\ldots,\mu_{M-1}}(s)$$

□

This property is a key of our analysis. It becomes now possible to use similar proof techniques as the ones used in previous chapter.

2 Algorithms

This section presents extensions to two-player zero-sum MGs of three non-stationary algorithms, namely Policy Search by Dynamic Programming (PSDP), Non-Stationary Value Iteration (NSVI) and Non-Stationary Policy Iteration (NSPI), known for improving the error bounds for MDPs. For MDPs, PSDP is known to have the best concentrability coefficient (Scherrer, 2014), while NSVI and NSPI have a reduced dependency over γ compared to their stationary counterparts. Here, in addition to defining the extensions of those algorithms, we also prove theoretical guarantees of performance.

2.1 Value Iteration and Non-Stationary Value Iteration

The structure of the NSVI algorithm and of the VI algorithm are very similar. As a reminder, the VI algorithm proceeds in two steps. The first step of each iterations is the greedy step and the second step is an evaluation step. We consider the case where, at iteration k, the greedy step picks an ϵ'_k-greedy strategy (*i.e.* $\mu_k \in \mathcal{G}_{\epsilon'_k}(v_{k-1})$) and the evaluation step is made up to an evaluation error ϵ_k. Iteration k can be formally described as follow:

$$\mathcal{T} v_{k-1} \leq \mathcal{T}_{\mu_k} v_{k-1} + \epsilon'_k, \text{ (approximate greedy step)}$$

$$v_k = \mathcal{T}_{\mu_k} v_{k-1} + \epsilon_k. \text{ (approximate evaluation step)}$$

As in the previous chapter, we consider that those errors propagate from one iteration to the next, and we will analyse how far the final strategy may be from optimal. Again, to measure the performance of such an algorithm, we will bound (according to some norm) the distance between the optimal value and the value of the final strategy when the opponent is playing optimally.

The VI algorithm presented above produces a sequence of values $v_0, \ldots, v_k$ and, implicitly, strategies $\mu_0, \ldots, \mu_k$. The non-stationary variation of VI for MGs, NSVI (Algorithm 18), simply consists in playing the m last strategies generated by VI for MGs. As described in the introduction, this strategy is written $\mu_{k,m} = \mu_k, \mu_{k-1}, \ldots, \mu_{k-m+1}$. In the following, we provide a bound in L_p-norm for NSVI in the framework of zero-sum two-player MGs. The goal is to bound the difference between the optimal value v^* and the value $v_{\mu_{k,m}} = v_{\mu_k, \mu_{k-1}, \ldots, \mu_{k-m+1}}$ of the m last strategies generated by VI.

Algorithm 18 Non-Stationary Value Iteration

Input: An zero-sum two-player MG MG, a value $v_0 = 0$ and a maximum number of iterations K.
for k=1,2,...,K **do**
  find $\mu_k \in \mathcal{G}_{\epsilon'_k}(v_{k-1})$
  $v_k = \mathcal{T}_{\mu_k} v_{k-1} + \epsilon_k$
end for
Output: $\mu_{K,m} = (\mu_K, \mu_{K-1}, \ldots, \mu_{K-m+1})$

Usually, one is only able to control ϵ and ϵ' according to some norm $\|.\|_{pq',\sigma}$ and wants to control the difference of value functions according to some other norm $\|.\|_{p,\rho}$ (this is explained in 1.2.3 for MDPs and in the previous chapter for zero-sum two-player MGs). The following theorem provides a performance guarantee in L_p-norm:

Theorem 4.1. *Let ρ and σ be distributions over states. Let p,q and q′ be positive reals such that $\frac{1}{q}+\frac{1}{q'}=1$, then for a non-stationary policy of size m and after k iterations we have:*

$$\left\|v^*-v_{\mu_{k,m}}\right\|_{p,\rho} \leq \underbrace{\frac{2(\gamma-\gamma^k)(\mathcal{C}_q^{1,k,0,m})^{\frac{1}{p}}}{(1-\gamma)(1-\gamma^m)} \sup_{1\leq j\leq k-1} \|\epsilon_j\|_{pq',\sigma}}_{\text{value update error}} + \underbrace{\frac{(1-\gamma^k)(\mathcal{C}_q^{0,k,0,m})^{\frac{1}{p}}}{(1-\gamma)(1-\gamma^m)} \sup_{1\leq j\leq k} \left\|\epsilon'_j\right\|_{pq',\sigma}}_{\text{greedy error}} \tag{4.13}$$

$$+ \underbrace{\frac{2\gamma^k}{1-\gamma^m}(\mathcal{C}_q^{k,k+1,0,m})^{\frac{1}{p}} \|v^*-v_0\|_{pq',\sigma}}_{\text{contraction term}}, \tag{4.14}$$

With:

$$\mathcal{C}_q^{l,k,d,m} = \frac{(1-\gamma)(1-\gamma^m)}{\gamma^l-\gamma^k}\sum_{i=l}^{k-1}\sum_{j=0}^{\infty}\gamma^{i+jm}c_q(i+jm+d)$$

and where:

$$c_q(j) = \sup_{\mu_1,\nu_1,\ldots,\mu_j,\nu_j}\left\|\frac{d(\rho P_{\mu_1,\nu_1}\ldots P_{\mu_j,\nu_j})}{d\sigma}\right\|_{q,\sigma}.$$

Proof. The full proof is left in appendix 6. Techniques used for the previous bound are similar to the ones used to analyse AGPI. □

Each term of this bound is decomposed in three elements. The contraction term is essentially equivalent to the one of VI. The γ-contraction term $\frac{2\gamma^k}{1-\gamma^m}$ (instead of $\frac{2\gamma^k}{1-\gamma}$) is slightly improved and converges to zero at a geometrical rate. The concentrability coefficient is improved in some sense (Scherrer, 2014) (if it is infinite, it implies that the one of VI is infinite). Finally, the error on the initial value is slightly worse but we believe that extra care in the analysis would provide the same dependency on the initial value.

As for the error propagation bounds, the difference between VI and NSVI is way more significant (here we analyse both the value update error and the greedy error). Whilst the γ-sensitivity of VI is of the order of $\frac{1}{(1-\gamma)^2}$, the γ-sensitivity of VI is of the order of $\frac{1}{(1-\gamma)(1-\gamma^m)}$. For instance, if $\gamma=0.99$ (which is a typical choice in applications) we have $\frac{1}{(1-\gamma)^2}=10000$ for VI and $\frac{1}{(1-\gamma)(1-\gamma^m)}$ if of the order of 5000 for $m=2$, 2000 for $m=5$ and 1000 for $m=10$. Again, the concentrability coefficient is improved and the ϵ-error remains the same in both cases.

From an implementation point of view, this technique introduces an explicit trade-off between memory and error. Indeed, m strategies have to be stored instead of 1 to decrease the value function approximation error from $\frac{2\gamma\epsilon}{(1-\gamma)^2}$ to $\frac{2\gamma\epsilon}{(1-\gamma)(1-\gamma^m)}$. Moreover, a benefit of the use of a non-stationary strategy in VI is that it comes from a known algorithm and thus needs very little implementation effort.

Remark 4.1. We can notice the full bound matches the bound on stationary strategies in L_p-norm of the previous chapter for the first and second terms. It also matches the one in L_∞-norm for non-stationary policies of Scherrer and Lesner (2012) in the case of MDPs.

2.2 Policy Search by Dynamic Programming (PSDP)

PSDP was first introduced by Bagnell et al. (2003) for solving undiscounted MDPs and undiscounted Partially Observable MDPs (POMDPs), but a natural variant using non-stationary strategies can be used for the discounted case (Scherrer, 2014). When applied to MDPs, this algorithm enjoys the best concentrability coefficient among several algorithms based on policy iteration, namely Non-stationary Policy Iteration (NSPI), Conservative Policy Iteration (CPI), Approximate Policy Iteration (API) and NSPI(m) (see Scherrer (2014) for more details). In this section, we describe two extensions of PSDP (PSDP1 and PSDP2) to two-player zero-sum MGs. Both algorithms reduce to PSDP in the case of MDPs but only one of them can be practically implemented in the case of a MG (PSDP2). Still, we present PSDP1 since its concentrability coefficient is close to the one given by Scherrer for PSDP in MDPs. The main difference between those two schemes is that PSDP1 involves the non-linear operator $\mathcal{T}_\mu$ whilst PSDP2 involves operator $\mathcal{T}_{\mu,\nu}$

Algorithm 19 PSDP1

Input: An zero-sum two-player MG MG, a value $v_0 = 0$ and a maximum number of iterations K.
for k=1,2,...,K **do**
  find $\mu_k \in \mathcal{G}(v_{\sigma_{k-1}})$,
  $v_{\sigma_{k-1}} = \mathcal{T}_{\mu_{k-1}} \dots \mathcal{T}_{\mu_0} 0 + \epsilon_{k-1}$
end for
Output: $\mu_{K,K+1} = (\mu_K, \mu_{K-1}, \dots, \mu_0)$

PSDP1: A first natural extension of PSDP in the case of γ-discounted Markov games is the following. At each step the algorithm returns a strategy μ_k for the maximizer, such that $\mathcal{T} v_{\sigma_{k-1}} = \mathcal{T}_{\mu_k} v_{\sigma_{k-1}}$ where $v_{\sigma_{k-1}} = \mathcal{T}_{\mu_{k-1}} \dots \mathcal{T}_{\mu_0} 0 + \epsilon_{k-1}$. Following any non-stationary strategy that uses σ_k $(= \mu_k, \dots, \mu_0)$ for the $k+1$ first steps, we have the following performance guarantee:

Theorem 4.2. *Let ρ and σ be distributions over states. Let p,q and q' be positive reals such that $\frac{1}{q} + \frac{1}{q'} = 1$, then we have:*

$$\left\| v^* - v_{\mu_{k,k+1}} \right\|_{p,\rho} \leq \frac{2(\mathcal{C}^{1,k,0}_{\mu^*,q})^{\frac{1}{p}}}{1-\gamma} \sup_{0 \leq j \leq k} \|\epsilon_j\|_{pq',\sigma} + o(\gamma^k), \tag{4.15}$$

with:

$$\mathcal{C}_{\mu^*,q}^{l,k,d} = \frac{1-\gamma}{\gamma^l - \gamma^k} \sum_{i=l}^{k-1} \gamma^i c_{\mu^*,q}(i+d)$$

and where:

$$c_{\mu*,q}(j) = \sup_{\nu_1,\dots,\nu_j,\mu_j} \left\| \frac{d(\rho P_{\mu^*,\nu_1} \dots P_{\mu^*,\nu_{j-1}} P_{\mu_j,\nu_j})}{d\sigma} \right\|_{q,\sigma}.$$

Proof. The full proof is left in appendix 7.2. □

The concentrability coefficient of this algorithm is similar to the one of its MDP counterpart: with respect to the first player, it has the advantage of depending mostly on the optimal strategy μ_*. However, estimating at each iteration $\mathcal{T}_{\mu_{k-1}} \dots \mathcal{T}_{\mu_0} 0$ requires solving a control problem and thus might be computationally prohibitive. Therefore, we introduce a second version of PSDP for games, which does not require to solve a control problem at each iteration.

Algorithm 20 PSDP2

Input: An zero-sum two-player MG MG, a value $v_0 = 0$ and a maximum number of iterations K.
for k=1,2,...,K **do**
find (μ_k, ν_k) such that $\mathcal{T} v_{\sigma_{k-1}} = \mathcal{T}_{\mu_k} v_{\sigma_{k-1}}$ and $\mathcal{T}_{\mu_k} v_{\sigma_{k-1}} = \mathcal{T}_{\mu_k,\nu_k} v_{\sigma_{k-1}}$,
$v_{\sigma_{k-1}} = \mathcal{T}_{\mu_{k-1},\nu_{k-1}} \dots \mathcal{T}_{\mu_0,\nu_0} 0 + \epsilon_k$
end for
Output: $\mu_{K,K+1} = (\mu_K, \mu_{K-1}, \dots, \mu_0)$

PSDP2: This algorithm creates a sequence of strategies $((\mu_k, \nu_k), \dots, (\mu_0, \nu_0))$. At each step k the algorithm returns a pair of strategies (μ_k, ν_k) such that $\mathcal{T} v_{\sigma_{k-1}} = \mathcal{T}_{\mu_k} v_{\sigma_{k-1}}$ and $\mathcal{T}_{\mu_k} v_{\sigma_{k-1}} = \mathcal{T}_{\mu_k,\nu_k} v_{\sigma_{k-1}}$ (where $v_{\sigma_{k-1}} = \mathcal{T}_{\mu_{k-1},\nu_{k-1}} \dots \mathcal{T}_{\mu_0,\nu_0} 0 + \epsilon_k$). To analyze how the error propagates through iterations, we will compare the value of the non-stationary strategy $\mu_{k,k+1}$ $(=\mu_k, \dots, \mu_0)$ against the value of best response to the optimal strategy.

Theorem 4.3. *Let ρ and σ be distributions over states. Let p,q and q' be positive reals such that $\frac{1}{q} + \frac{1}{q'} = 1$, then we have:*

$$\left\| v^* - v_{\mu_{k,k+1}} \right\|_{p,\rho} \leq \frac{4(\mathcal{C}_q^{1,k,0})^{\frac{1}{p}}}{1-\gamma} \sup_{0 \leq j \leq k} \|\epsilon_j\|_{pq',\sigma} + o(\gamma^k), \tag{4.16}$$

with:

$$\mathcal{C}_q^{l,k,d} = \frac{1-\gamma}{\gamma^l - \gamma^k} \sum_{i=l}^{k-1} \gamma^i c_q(i+d).$$

Proof. The full proof is left in appendix 7.1. □

PSDP1 and PSDP2 are different algorithms since one requires to solve a finite horizon γ-discounted MDP (PSDP1) as a subroutine whilst the second one (PSDP2) only requires to evaluate a joint strategy. Thus, the error term ϵ_k in PSDP1 is an error due to solving a control problem, while the error term ϵ_k in PSDP2 comes from a pure estimation error.

However, both algorithms share the same issue. Both algorithms require the storage of all strategies from the very first iteration. The algorithm PSDP1 needs to store k strategies at iteration k while PSDP2 needs $2k$ at the same stage. However PSDP2 alleviates a major constraint of PSDP1: it doesn't need an optimization subroutine at each iteration. The price to pay for that simplicity is to store $2k$ strategies and a worse concentrability coefficient.

Indeed, one can notice that $\forall j \in N \ c_{\mu*,q}(j) \leq c_q(j)$ and thus $\mathcal{C}_{\mu^*,q}^{l,k,d} \leq \mathcal{C}_q^{l,k,d}$. And thus, the concentrability coefficient of PSDP2 is worse than PSDP1's. Moreover, $\mathcal{C}_q^{l,k,d,m} = (1-\gamma^m)\mathcal{C}_q^{l,k,d} + \gamma^m \mathcal{C}_q^{l,k,d+m,m}$ meaning intuitively that $\mathcal{C}_q^{l,k,d}$ is $\mathcal{C}_q^{l,k,d,m}$ when m goes to infinity. This also means that if $\mathcal{C}_q^{l,k,d} = \infty$, then we have $\mathcal{C}_q^{l,k,d,m} = \infty$. In that sense, one can argue that PSDP2 offers a better concentrability coefficient than NSVI. Finally, PSDP1 and PSDP2 enjoy a better γ-sensitivity than NSVI for the value update error ($\frac{4}{1-\gamma}$ instead of $\frac{2(\gamma-\gamma^k)}{(1-\gamma)(1-\gamma^m)}$)

2.3 Non Stationary Policy Iteration (NSPI(m))

The approximate version of PI has the same guarantees in terms of greedy error and approximation error as VI. Like VI, there exists a non-stationary version of policy iteration that was originally designed for MDPs by Scherrer and Lesner (2012). Instead of estimating the value of the current policy at each iteration, it estimates the value of the non-stationary policy formed by the last m policies. Generalized to MGs, NSPI(m) estimates the value of the best response to the last m strategies.

Algorithm 21 NSPI

Input: An zero-sum two-player MG MG, a value $v_0 = 0$ and a maximum number of iterations K.
for k=1,2,...,K **do**
  find $\mu_k \in \mathcal{G}(v_{k-1})$,
  $v_k = v_{\mu_{k,m}} + \epsilon_k$
end for
Output: $\mu_{K,m} = (\mu_K, \mu_{K-1}, \ldots, \mu_{K-m+1})$

Doing so, the algorithm NSPI(m) tackles the memory issue of PSDP. It allows controling the size of the stored non-stationary strategy. NSPI(m) proceeds in two steps. First, it computes an approximation v_k of $v_{\mu_{k,m}}$ ($v_k = v_{\mu_{k,m}} + \epsilon_k$). Here, $v_{\mu_{k,m}}$ is the value of the best response of the minimizer to strategy $\mu_{k,m} = \mu_k, \ldots, \mu_{k-m+1}$ (defined in Section 1). Then it moves to a new strategy μ_{k+1} satisfying $\mathcal{T} v_k = \mathcal{T}_{\mu_{k+1}} v_k$.

Theorem 4.4. *Let ρ and σ be distributions over states. Let p,q and q′ be positive reals such that $\frac{1}{q}+\frac{1}{q'}=1$, then for a non-stationary policy of size m and after k iterations we have:*

$$\left\|v^* - v_{\mu_{k,m}}\right\|_{p,\rho} \leq \frac{2\gamma(\mathcal{C}_q^{1,k-m+2,0,m})^{\frac{1}{p}}}{(1-\gamma)(1-\gamma^m)} \sup_{m\leq j\leq k-1} \|\epsilon_j\|_{pq',\sigma} + o(\gamma^k). \tag{4.17}$$

Proof. The full proof is left in appendix 8. □

The NSPI dependency over γ and the concentrability coefficient involved in the NSPI bound are similar to those found for NSVI. However, in the MDP case the policy evaluation error is responsible for the error ϵ_k and in the MG case the error comes from solving a full control problem.

2.4 Summary

To sum up, both NSVI and NSPI enjoy a reduced dependency over γ (i.e. $\frac{1}{(1-\gamma)(1-\gamma^m)}$) when considering non-stationary strategies. They exhibit similar concentrability coefficients but the origin of the error is different (a regression for NSVI and a policy evaluation or a control problem for NSPI). PSDP1 and PSDP2 enjoys an even better dependency on γ (i.e. $\frac{1}{1-\gamma}$). The concentrability coefficient of PSDP1 is better than that of PSDP2 which is better than those of NSVI and NSPI. However, the error involved in the analysis of PSDP1 is caused by solving a full control problem (which makes this algorithm impractical) while the error in PSDP2 comes from a simple regression.

3 Empirical Evaluation

The previous section provided, for each new algorithm, a performance bound that assumes a worst case error propagation. Examples suggesting the bound is tight were provided by Scherrer and Lesner (2012) for MDPs in the case of L_∞ analysis ($p=\infty$). Since those examples apply on MDPs, they also apply to MGs. Those specific examples are pathological problems and, in practice, the bounds will generally be conservative. Furthermore, our analysis of the term ϵ_k somehow hides the sources of errors that may vary a lot among the different algorithms. To ensure these techniques are relevant in practice and to go beyond the theoretical analysis, we tested them on synthetic MDPs and turn-based MGs, named Garnets (Archibald et al., 1995).

Garnets for MDPs: A Garnet is originally an abstract class of MDPs. It is generated according to three parameters (N_S,N_A,N_B). Parameters N_S and N_A are respectively the number of states and the number of actions. Parameter N_B is the branching factor defining the number of possible next states for any state-action pair. The procedure to generate the transition kernel $p(s'|s,a)$ is the following. First, one should

draw a partition of $[0,1]$ by drawing $N_B - 1$ cutting points uniformly over $[0,1]$ noted $(p_i)_{1 \leq i \leq N_B - 1}$ and sorted in increasing order (let us note $p_0 = 0$ and $p_{N_B} = 1$). Then, one draws a subset $\{s_1, \ldots, s_{N_B}\}$ of size N_B of the state space S. This can be done by drawing without replacement N_B states from the state space S. Finally, one assigns $p(s'_i|s,a)$ according to the following rule: $\forall i \in \{1, \ldots, N_B\},\ p(s_i|s,a) = p_i - p_{i-1}$. The reward function $r(s)$ depends on the experiment.

Garnet for two-player turn-based MGs We are interested in a special kind of MGs, namely turn-based games. Here, turn-based games are two-player zero-sum MGs where, at each state, only one player has the control on the game. The generating process for this kind of Garnet is the same as the one for MDPs. Then we will independently decide for each state which player has the control over the state. The probability of state s to be controlled by player 1 is $\frac{1}{2}$.

Experiments In the two categories of Garnets described previously we ran tests on Garnets of size $N_S = 100$, with $N_A \in \{2,5,8\}$ and $N_B \in \{1,2,5\}$. The experiment aims at analyzing the impact of the use of non-stationary strategies considering different amounts of samples at each iteration for each algorithm. Here, the reward for each state-action couple is null except for a given proportion (named the sparsity $\in \{0.05, 0.1, 0.5\}$) drawn according to a normal distribution of mean 0 and of variance 1. Algorithms are based on the state-actions value function (defined in 3.3): $Q_{\mu,\nu}(s,a^1,a^2) = r(s,a^1,a^2) + \sum_{s' \in S} p(s'|s,a^1,a^2) v_{\mu,\nu}(s')$.

The analysis of previous section still holds since one can consider an equivalent (but larger) MG whose state space is composed of state-action pairs. Moreover, each evaluation step consists in approximating the state-action(s) value function. We approximate the value function by minimizing a $\mathcal{L}_2$ norm on a tabular basis with a regularization also in $\mathcal{L}_2$ norm. All the following considers simultaneous MGs. To retrieve algorithms for MDPs, consider that the minimizer always has a single action. To retrieve algorithms for turn-based MGs, consider that at each state only a single player has more than one action.

Experiments are limited to finite states MGs. Moreover, Garnets have an erratic dynamic since next states are drawn without replacement within the set of states, thus the dynamic is not regular in any sens. Garnets are thus tough to optimize. Experiments on simultaneous games are not provided due to difficulties encountered to optimize such games. We believe Garnets are too hard to optimize when it comes to simultaneous games.

In all presented graphs, the performance of a strategy μ (which might be stationary or not) is measured as $\frac{\|v^* - v_\mu\|_{u,2}}{\|v^*\|_{u,2}}$ where u is the uniform measure over the state-action(s) space. The value v_μ is computed exactly with the policy iteration algorithm. In every curve, the confidence interval is proportional to the standard deviation. To compare algorithms on a fair basis, their implementation relies on a sample-based approximation involving an equivalent number of samples. In all tests, we could not notice a significant

influence of N_A. Moreover the sparsity only influences the amount of samples needed to solve the MG.

NSVI The NSVI algorithm starts with a null Q-functions $Q_0(s, a^1, a^2)$. At each iteration we draw uniformly over the state-actions space (s_i, a_i^1, a_i^2) then we compute $r^i = r(s_i, a_i^1, a_i^2)$ and draw $s'_i \sim p(.|s_i, a_i^1, a_i^2)$ for $i \in \{1, \ldots, N_k\}$. Then we compute $q_i = r_i + \gamma \min_{a^2} E_{a^1 \sim \mu_k(.|s'_i)}[Q_k(s'_i, a^1, a^2)]$. The next state-actions value function Q_{k+1} is the best fit over the training dataset $\{(s_i, a_i^1, a_i^2), q_i\}_{i \in \{1,\ldots,N_k\}}$. In all experiments on NSVI, all samples are refreshed after each iteration. The first advantage of using

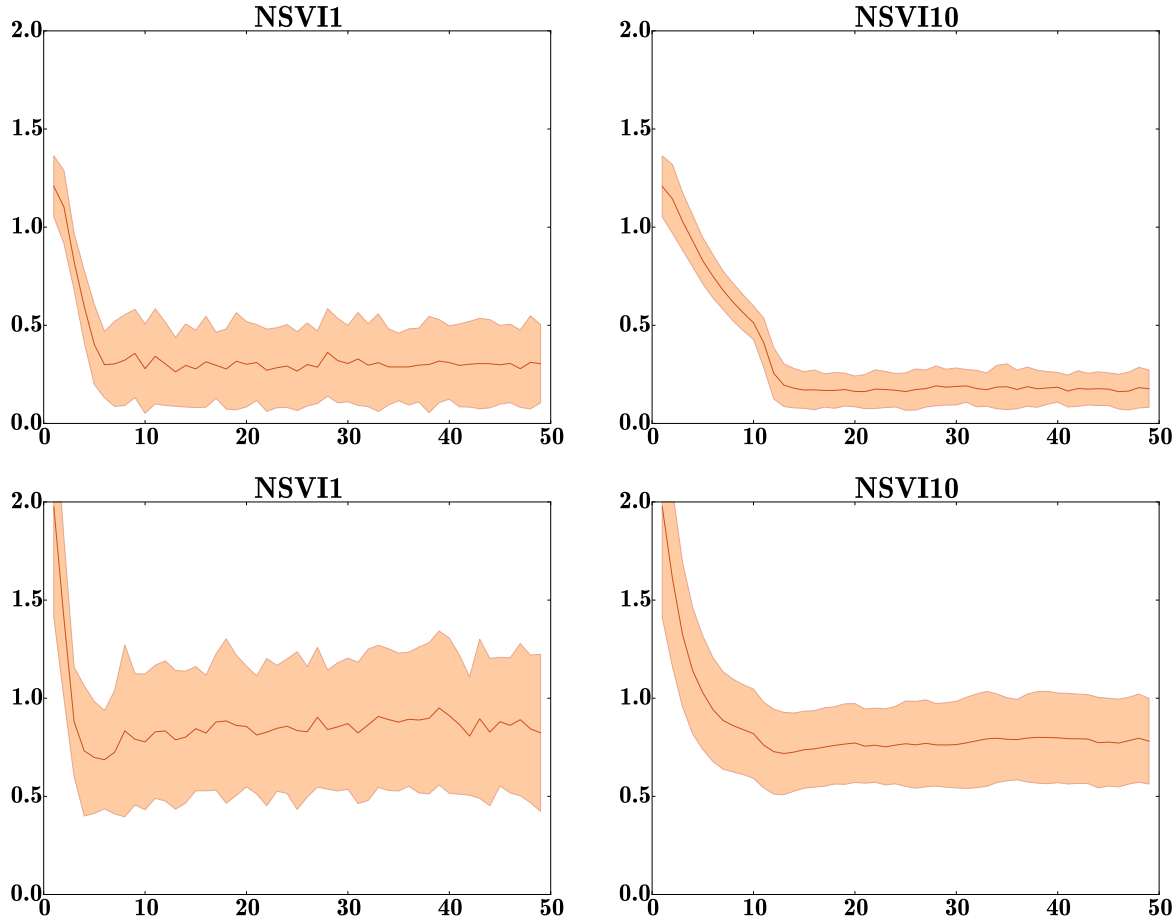


Figure 4.1 – Performance (y-axis) of the strategy at step k (x-axis) for NSVI for a strategy of length 10 (right) and length 1 (left). Results are averaged over 70 Garnets $N_S = 100$, $N_A = 5$, $N_B = 1$ (top) and $N_B = 2$ (bottom). All Garnets have a sparsity of 0.5 and $\gamma = 0.9$. Each step of the algorithm uses $2.25 \times N_A \times N_S$ samples.

non-stationary strategies in VI is the reduction of the standard deviation of the value $v_{\mu_k,\ldots,\mu_{k-m+1}}$. Figure 4.1 shows the reduction of the variance when running a non-stationary strategy. Intuitively one can think of it as a way of averaging over the last m strategies the resulting value. Moreover, the larger m is the more the performance concentrates (the parameter m is varied in figure 4.1, 4.4, 4.5, 4.6 and 4.7). A second

advantage is the improvement of the average performance when N_B (the branching factor of the problem) is low. One the negative side, since we are mixing last m strategies, the asymptotic performance is reached after more iterations (see Figure 4.1).

PSDP2 In practice PSDP2 builds N_k rollout (indexed by j) $\{(s_{i,j}, a^1_{i,j}, a^2_{i,j}, r_{i,j})\}_{i\in\{0,\dots,k+1\}}$ at iteration k. Where $(s_{0,j}, a^1_{0,j}, a^2_{0,j})$ are drawn uniformly over the state-actions space and where $s_{i+1,j} \sim p(.|s_{i,j}, a^1_{i,j}, a^2_{i,j})$, $a^1_{i+1,j} \sim \mu_{k-i}(.|s_{i+1,j})$, $a^2_{i+1,j} \sim \nu_{k-i}(.|s_{i+1,j})$ and $r_{i+1,j} = r(s_{i+1,j}, a^1_{i+1,j}, a^2_{i+1,j})$. Then we build the dataset $\{(s_{0,j}, a^1_{0,j}, a^2_{0,j}), \sum_{i=0}^{k+1} \gamma^i r_{i,j}\}_{j\in\{0,\dots,N_k\}}$. The state-actions value function Q_{k+1} is the best fit over the training dataset. Strategies μ_{k+1} and ν_{k+1} are the exact minmax strategies with respect to Q_{k+1}.

From an implementation point of view, PSDP2 uses $(k+2)\times N_k$ samples at each iteration (parameter N_k is varied in the figures 4.2 and 4.7). Furthermore, the algorithm uses rollouts of increasing size. As a side effect, the variance of $\sum_{i=0}^{k+1} \gamma^i r_{i,j}$ increases with iterations for non-deterministic MDPs and MGs. This makes the regression of Q_{k+1} less practical. To tackle this issue, we use a procedure, named resampling, that consists in averaging the cumulative γ-discounted reward $\sum_{i=0}^{k+1} \gamma^i r_{i,j}$ over different rollouts launched from the same state-actions triplet $(s_{0,j}, a^1_{0,j}, a^2_{0,j})$. In figure 4.2 the two top curves display the performance of PSDP2 with (on the right) and without (on the left) resampling trajectories on deterministic MGs. The two figures on the bottom are however obtained on non-deterministic MGs. One can notice a significant improvement of the algorithm when using resampling on non-deterministic MGs illustrating the variance issue raised in the foregoing paragraph.

PSDP1 We do not provide experiments within the PSDP1 scheme. Each iteration of PSDP1 consists in solving a finite horizon control problem (*i.e.* approximating $v_{\sigma_k} = \mathcal{T}_{\mu_k} \dots \mathcal{T}_{\mu_0} 0$). The problem of estimating v_{σ_k} reduces to solving a finite horizon MDP with non stationary dynamics. To do so, one should either use Fitted-Q iterations or PSDP for MDPs. In the first case, one would not see the benefit of such a scheme compared to the use NSVI. Indeed, each iterations of PSDP1 would be as heavy in term of computation as NSVI. In the second case, one would not see the benefit compared PSDP2 since each iterations would be as heavy as PSDP2.

NSPI(m) At iteration k, NSPI(m) approximates the best response of the non-stationary strategy $\mu_k, \dots, \mu_{k-m+1}$. In the case of an MDP, this results in evaluating a policy. The evaluation of a stationary policy is done by an approximate iteration of $\mathcal{T}_\mu$. This procedure can be done by a procedure close to Fitted-Q iteration in which the strategy μ is used at each iteration instead of the greedy policy. The evaluation of the non-stationary policy $\mu_k, \dots, \mu_{k-m+1}$ is done by approximately applying in a cycle $\mathcal{T}_{\mu_k}, \dots, \mathcal{T}_{\mu_{k-m+1}}$. For MG, the subroutine will contain $l \times m$ iterations. At iteration

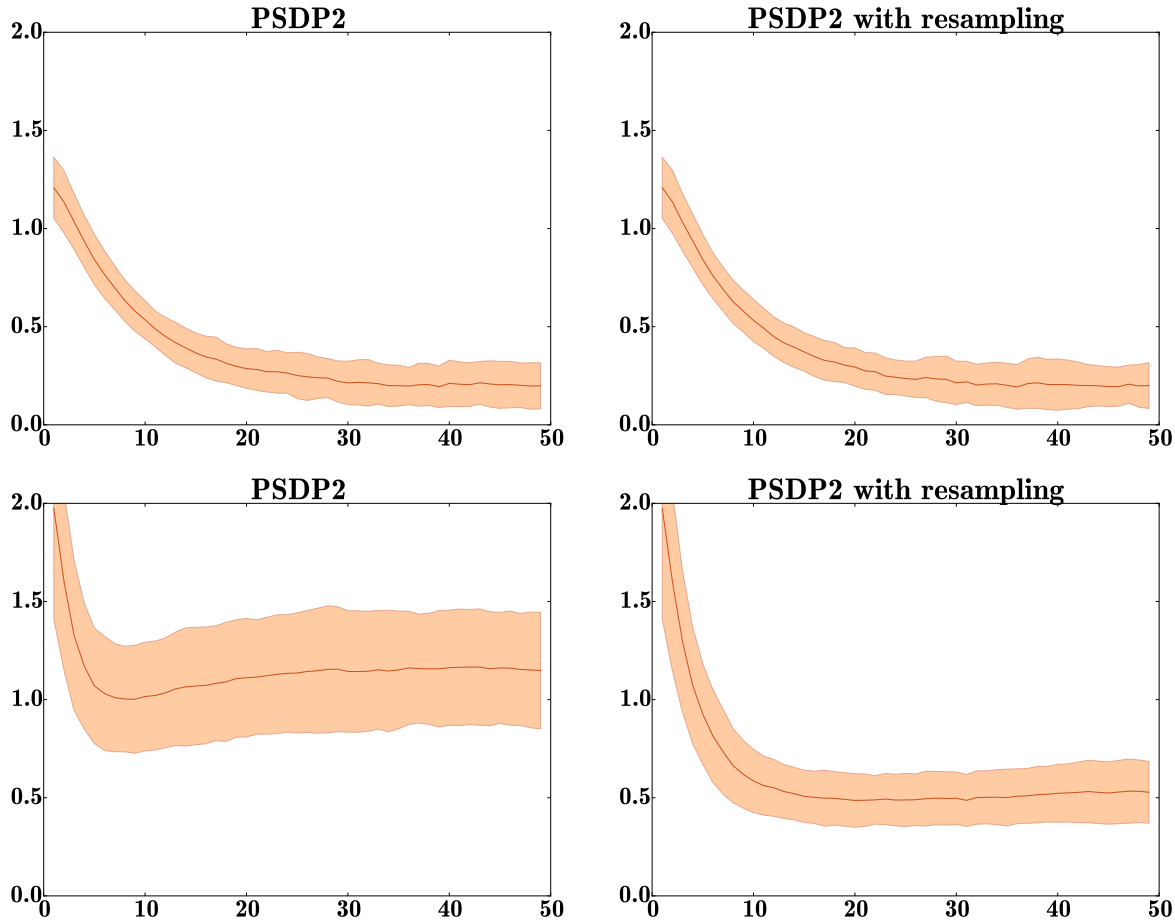


Figure 4.2 – Performance (y-axis) of the strategy of length $k+1$ at step k (x-axis) for PSDP2 without resampling (left) with a 10 times resampling (right). Results are averaged over 70 Garnets $N_S = 100$, $N_A = 5$, $N_B = 1$ (top) and $N_B = 2$ (bottom). All Garnets have a sparsity of 0.5 and $\gamma = 0.9$. Each step of the algorithm uses $2.25 \times N_A \times N_S$ rollouts.

$p \in \{1, \ldots, l \times m\}$ the subroutine computes one step of Fitted-Q iteration considering the maximizer's strategy is fixed and of value $\mu_{k-m+(p-1 \ mod(m))+1}$ and taking the greedy action for the minimizer. In this subroutine the dataset is fixed (it is only refreshed at each iteration of the overall NSPI(m) procedure). The parameter l is chosen large enough to achieve a given level of accuracy, that is having $\gamma^{m \times l}$ below a given threshold. Note that for small values of k (*i.e.* $k < m$) this implementation of the algorithm finds an approximate best response of the non-stationary strategy $\mu_k, \ldots, \mu_1$ (of size k and not m). As for VI, the use of non-stationary strategies reduces the standard deviation of the performance as m grows. Figure 4.3 shows also an improvement of the average performance as m grows.

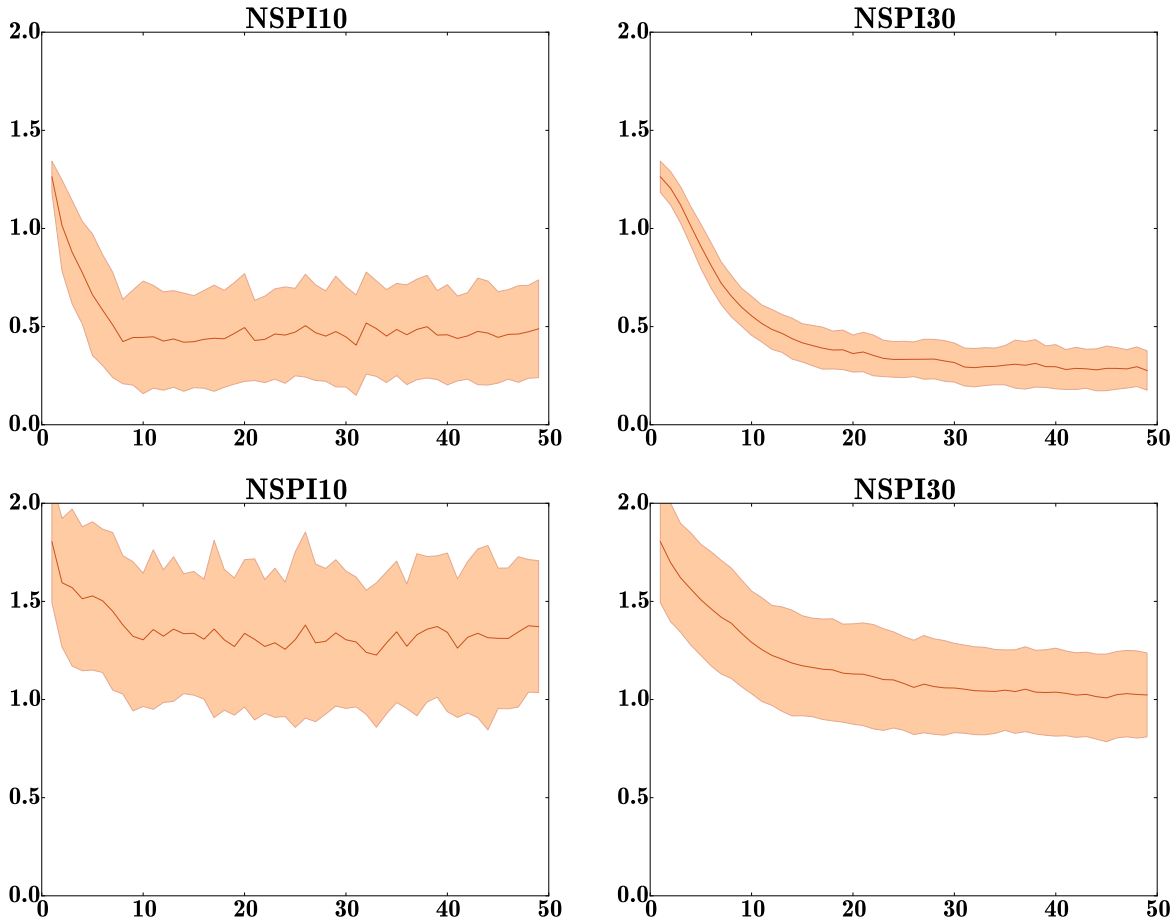


Figure 4.3 – Performance (y-axis) of the strategy at step k (x-axis) for NSPI for a strategy of length 10 (left) and 30 (right). results are averaged over 70 Garnets $N_S = 100$, $N_A = 8$, $N_B = 1$ (top) and $N_B = 2$ (bottom). All Garnets have a sparsity of 0.5 and $\gamma = 0.9$. Each step of the algorithm uses $2.25 \times N_A \times N_S$ samples.

4 A Comparison

From the theoretical analysis, one may conclude that PSDP1 is the best scheme to solve MGs. It's dependency over γ is the lowest among the analyzed algorithms and it exhibits the best concentrability coefficient. However, from the implementation point of view this scheme is a very cumbersome since it implies solving a finite horizon control problem of increasing size, meaning using an algorithm like Fitted-Q or PSDP as a subroutine at each iteration. PSDP2 tackles the main issue of PSDP1. This algorithm doesn't need to solve a control problem as a subroutine but a simple supervised learning step is enough. The price to pay is a worst bound and the storage of $2 \times k$ instead of k strategies.

As in PSDP1, the NSPI algorithm needs to solve a control problem as a subroutine but it only considers a constant number of strategies. Thus NSPI solves the memory issue of PSDP1 and PSDP2. The γ-sensitivity of the error bound is reduced from

roughly $\frac{1}{(1-\gamma)^2}$ to $\frac{1}{(1-\gamma)(1-\gamma^m)}$ where m is the length of the non-stationary strategy considered. Nevertheless, the error term of PSDP2 derives from a supervised learning problem while it comes from a control problem in NSPI. Thus, controlling the error of PSDP2 might be easier than controlling the error of NSPI.

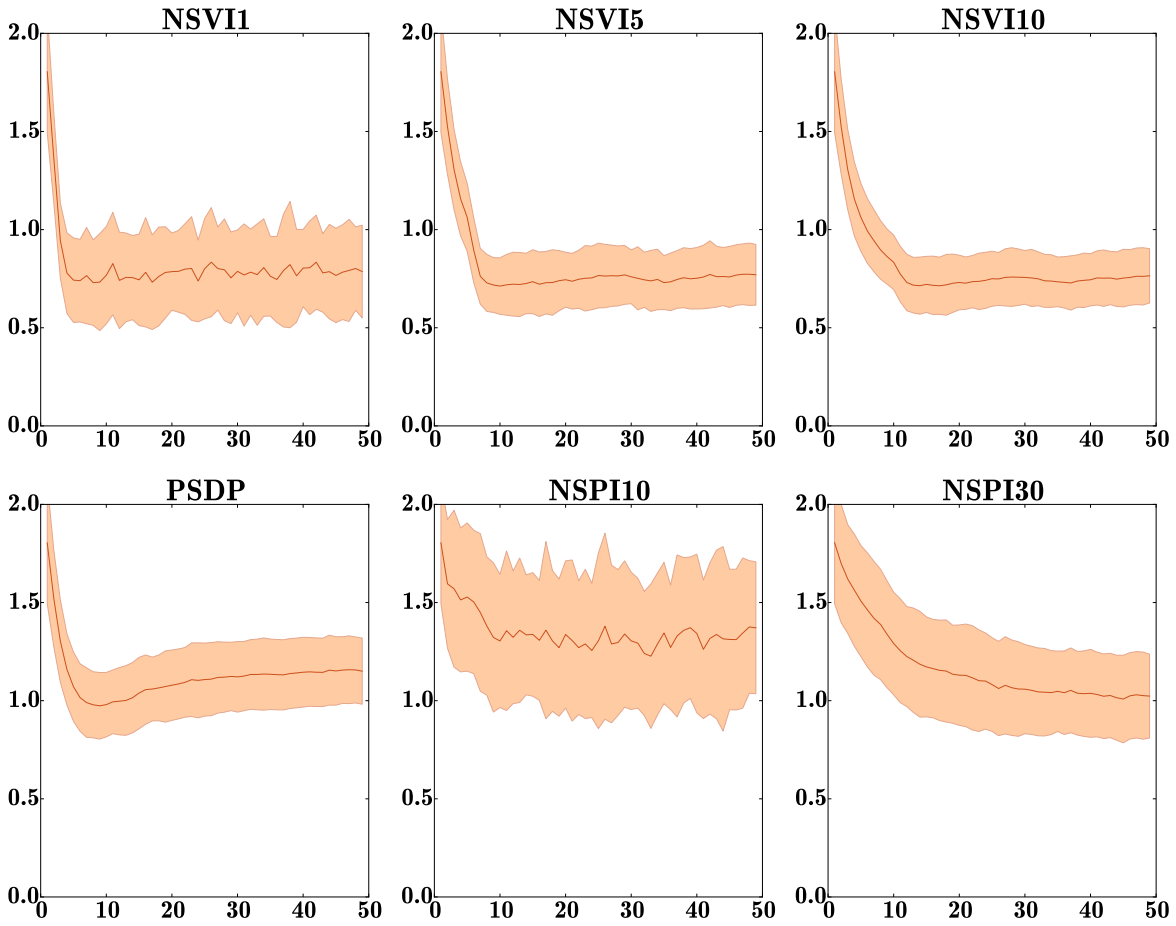


Figure 4.4 – Performance (y-axis) of the strategy at step k (x-axis) for NSVI, PSDP and NSPI. Results are averaged over 70 Garnets $N_S = 100$, $N_A = 8$, $N_B = 1$. All Garnets have a sparsity of 0.5 and $\gamma = 0.9$. Each step of NSPI and NSVI uses $2.25 \times N_A \times N_S$ samples. Each step of PSDP2 uses $2.25 \times N_A \times N_S$ rollouts.

The NSVI algorithm enjoys an error propagation bound similar to the NSPI one. However the error of NSVI derives from a simple supervised learning problem instead of a full control problem as for NSPI. Figure 4.4 compares NSPI and NSVI with the same number of samples at each iteration. It clearly shows NSVI performs better in average performance and regarding the standard deviation of the performance. For turn-based MGs the NSVI algorithm performs better than NSPI on Garnets. Furthermore one iteration of NSPI costs significantly more than an iteration of NSVI. Figure 4.4 also compares PSDP2 and NSVI. Even if PSDP2 uses k times more samples than NSVI at iteration k, it barely achieves the same performance as NSVI (in the case of a non-deterministic game this is not even the case, see Figure 4.6 in the appendix).

5 Conclusion

In this chapter, we explored the generalization of different techniques known for MDPs to zero-sum two-player MGs. These techniques make the use of non-stationary strategies to improve the error propagation bounds of several known dynamic programming algorithms. The theoretical analysis shows a reduced dependency over γ of non-stationary algorithms. For instance NSVI and NSPI have a dependency of $\frac{2\gamma}{(1-\gamma)(1-\gamma^m)}$ instead of $\frac{2\gamma}{(1-\gamma)^2}$ for the corresponding stationary algorithm. PSDP2 has a dependency of $\frac{2}{(1-\gamma)}$ over γ and it enjoys a better concentrability constant than NSPI and NSVI. The empirical study shows the dependency on the error is the main factor when comparing algorithms with the same budget of samples. The nature of the error seems to be crucial. NSVI outperforms NSPI since a simple regression produces less error than a policy evaluation or even a full control problem. NSVI outperforms PSDP2 since it is more thrifty in terms of samples per iteration.

One of the downside of this analysis is that it is limited to zero-sum two-player MGs whilst the use of cyclic strategies was suggested in general-sum games (Zinkevich et al., 2006). In the next chapter we will explore to which extent it is easier to find cyclic equilibria in general-sum MGs instead of a stationary one for the VI algorithm.

6 Appendix: NSVI

First, let us define a (somehow abusive) simplifying notation. Γ^n will represents any products of n discounted transition kernels. Then, Γ^n represents the class $\{\gamma P_{\mu_1,\nu_1}.....\gamma P_{\mu_n,\nu_n},$ with μ_i, ν_i random strategies$\}$. For example, the following property holds $a\Gamma^i b\Gamma^j + c\Gamma^k = ab\Gamma^{i+j} + c\Gamma^k$.

NSVI with a greedy and an evaluation error:

$$\mathcal{T} v_{k-1} \leq \mathcal{T}_{\mu_k} v_{k-1} + \epsilon'_k, \text{ (approximate greedy step)}$$

$$v_k = \mathcal{T}_{\mu_k} v_{k-1} + \epsilon_k. \text{ (approximate evaluation step)}$$

The following lemma shows how errors propagate through iterations.

Lemma 4.1. *$\forall M < k$:*

$$\left|v^* - v_{\mu_{k,M}}\right| \leq \sum_{j=0}^{\infty} \Gamma^{Mj} \left[2\Gamma^k \left|v^* - v_0\right| + 2\sum_{i=1}^{k-1} \Gamma^i \left|\epsilon_{k-i}\right| + \sum_{i=0}^{k-1} \Gamma^i \left|\epsilon'_{k-i}\right|\right]. \tag{4.18}$$

Proof. We will bound the error made while running the non-stationary strategy $(\mu_k, \dots, \mu_{k-M+1})$ rather than the optimal strategy. This means bounding the following positive quantity:
$v^* - v_{\mu_{k,M}}$

To do so let us first bound the following quantity:

$$\mathcal{T}_{\mu_k} v_{k-1} - v_{\mu_{k,M}} \tag{4.19}$$
$$= \mathcal{T}_{\mu_k} v_{k-1} - \mathcal{T}_{\mu_k} \dots \mathcal{T}_{\mu_{k-M+1}} v_{\mu_{k,M}}, \text{(with (4.2))} \tag{4.20}$$
$$= \mathcal{T}_{\mu_k} v_{k-1} - \mathcal{T}_{\mu_k,\hat{\nu}_k} \dots \mathcal{T}_{\mu_{k-M+1},\hat{\nu}_{k-M+1}} v_{\mu_{k,M}}, \tag{4.21}$$

Where $\hat{\nu}_{k-i}$ such as $\mathcal{T}_{\mu_{k-i},\hat{\nu}_{k-i}} \dots \mathcal{T}_{\mu_{k-M+1},\hat{\nu}_{k-M+1}} v_{\mu_{k,M}} = \mathcal{T}_{\mu_{k-i}} \dots \mathcal{T}_{\mu_{k-M+1}} v_{\mu_{k,M}}$

$$\leq \mathcal{T}_{\mu_k,\hat{\nu}_k} \dots \mathcal{T}_{\mu_{k-M+1},\hat{\nu}_{k-M+1}} v_{k-M} + \sum_{i=1}^{i=M-1} \gamma P_{\mu_k,\hat{\nu}_k} \dots \gamma P_{\mu_{k-i+1},\hat{\nu}_{k-i+1}} \epsilon_{k-i}$$
$$- \mathcal{T}_{\mu_k,\hat{\nu}_k} \dots \mathcal{T}_{\mu_{k-M+1},\hat{\nu}_{k-M+1}} v_{\mu_{k,M}}, \tag{4.22}$$

since $v_i = \mathcal{T}_{\mu_i} v_{i-1} + \epsilon_i$, $\forall v \mathcal{T}_{\mu_k} v \leq \mathcal{T}_{\mu_k,\hat{\nu}_k} v$ and since $\mathcal{T}_{\mu_k,\hat{\nu}_k}$ is affine

$$\leq \gamma P_{\mu_k,\hat{\nu}_k} \dots \gamma P_{\mu_{k-M+1},\hat{\nu}_{k-M+1}} (v_{k-M} - v_{\mu_{k,M}})$$
$$+ \sum_{i=1}^{i=M-1} \gamma P_{\mu_k,\hat{\nu}_k} \dots \gamma P_{\mu_{k-i+1},\hat{\nu}_{k-i+1}} \epsilon_{k-i}. \tag{4.23}$$

We also have:

$$v^* - v_k = \mathcal{T} v^* - \mathcal{T} v_{k-1} + \mathcal{T} v_{k-1} - v_k, \tag{4.24}$$
$$\leq \mathcal{T}_{\mu^*} v^* - \mathcal{T}_{\mu^*} v_{k-1} - \epsilon_k + \epsilon'_k, \tag{4.25}$$
$$\text{(since } \mathcal{T}_{\mu^*} v_{k-1} \leq \mathcal{T} v_{k-1} \text{ and } \mathcal{T}_{\mu^*} v^* = \mathcal{T} v^*)$$
$$\leq \mathcal{T}_{\mu^*,\tilde{\nu}_{k-1}} v^* - \mathcal{T}_{\mu^*,\tilde{\nu}_{k-1}} v_{k-1} - \epsilon_k + \epsilon'_k, \tag{4.26}$$
$$\text{(with } \mathcal{T}_{\mu^*,\tilde{\nu}_{k-1}} v_{k-1} = \mathcal{T}_{\mu^*} v_{k-1} \text{ and } \mathcal{T}_{\mu^*} v^* \leq \mathcal{T}_{\mu^*,\tilde{\nu}_{k-1}} v^*)$$
$$\leq \gamma P_{\mu^*,\tilde{\nu}_{k-1}} (v^* - v_{k-1}) - \epsilon_k + \epsilon'_k. \tag{4.27}$$

And we have:

$$
\begin{aligned}
v^* - v_k &= \mathcal{T} v^* - \mathcal{T}_{\mu_k} v_{k-1} + \mathcal{T}_{\mu_k} v_{k-1} - v_k, && (4.28)\\
&\geq \mathcal{T}_{\mu_k} v^* - \mathcal{T}_{\mu_k} v_{k-1} - \epsilon_k, && (4.29)\\
&\quad (\text{since } \mathcal{T}_{\mu_k} v^* \leq \mathcal{T} v^*)\\
&\geq \mathcal{T}_{\mu_k,\nu_k^*} v^* - \mathcal{T}_{\mu_k,\nu_k^*} v_{k-1} - \epsilon_k, && (4.30)\\
&\quad (\text{where } \mathcal{T}_{\mu_k,\nu_k^*} v^* = \mathcal{T}_{\mu_k} v^* \text{ and since } \mathcal{T}_{\mu_k} v_{k-1} \leq \mathcal{T}_{\mu_k,\nu_k^*} v_{k-1})\\
&\geq \gamma P_{\mu_k,\nu_k^*}(v^* - v_{k-1}) - \epsilon_k. && (4.31)\\
& && (4.32)
\end{aligned}
$$

which can also be written:

$$v_k - v^* \leq \gamma P_{\mu_k,\nu_k^*}(v_{k-1} - v^*) + \epsilon_k.$$

Using the Γ^n notation .

$$v^* - v_k \leq \Gamma^k (v^* - v_0) - \sum_{i=0}^{k-1} \Gamma^i \epsilon_{k-i} + \sum_{i=0}^{k-1} \Gamma^i \epsilon'_{k-i}, \tag{4.33}$$

$$v_k - v^* \leq \Gamma^k (v_0 - v^*) + \sum_{i=0}^{k-1} \Gamma^i \epsilon_{k-i}. \tag{4.34}$$

$$
\begin{aligned}
v^* - v_{\mu_{k,M}} &= \mathcal{T} v^* - \mathcal{T} v_{k-1} + \mathcal{T} v_{k-1} - v_{\mu_{k,M}}, && (4.35)\\
&\leq \mathcal{T}_{\mu_*} v^* - \mathcal{T}_{\mu^*} v_{k-1} + \mathcal{T} v_{k-1} - v_{\mu_{k,M}}, && (4.36)\\
&\quad (\text{since } \mathcal{T}_{\mu^*} v_{k-1} \leq \mathcal{T} v_{k-1})\\
&\leq \mathcal{T}_{\mu_*,\tilde{\nu}_{k-1}} v^* - \mathcal{T}_{\mu^*,\tilde{\nu}_{k-1}} v_{k-1} + \mathcal{T} v_{k-1} - v_{\mu_{k,M}}, && (4.37)\\
&\quad \text{with } \tilde{\nu}_{k-1} \text{ defined in (4.26)}\\
&\leq \gamma P_{\mu^*,\tilde{\nu}_{k-1}}(v^* - v_{k-1}) + \mathcal{T}_{\mu_k} v_{k-1} - v_{\mu_{k,M}} + \epsilon'_k, && (4.38)\\
&\leq \Gamma(v^* - v_{k-1}) + \Gamma^M (v_{k-M} - v_{\mu_{k,M}}) + \sum_{i=1}^{M-1} \Gamma^i \epsilon_{k-i} + \epsilon'_k, \text{With (4.23)} && (4.39)\\
&\leq \Gamma(v^* - v_{k-1}) + \Gamma^M (v_{k-M} - v^*) + \Gamma^M (v^* - v_{\mu_{k,M}})\\
&\quad + \sum_{i=1}^{M-1} \Gamma^i \epsilon_{k-i} + \epsilon'_k. && (4.40)
\end{aligned}
$$

Then combining (4.33), (4.34) and (4.40):

$$
\begin{aligned}
v^* - v_{\mu_{k,M}} \leq\ & \Gamma^k (v^* - v_0) - \sum_{i=1}^{k-1} \Gamma^i \epsilon_{k-i} + \sum_{i=1}^{k-1} \Gamma^i \epsilon'_{k-i} + \Gamma^k (v_0 - v^*)\\
& + \Gamma^M \sum_{i=0}^{k-M-1} \Gamma^i \epsilon_{k-M-i} + \sum_{i=1}^{M-1} \Gamma^i \epsilon_{k-i} + \epsilon'_k + \Gamma^M (v^* - v_{\mu_{k,M}}), && (4.41)
\end{aligned}
$$

$$\left|v^* - v_{\mu_{k,M}}\right| \leq 2\Gamma^k \left|v^* - v_0\right| + 2 \sum_{i=1}^{k-1} \Gamma^i \left|\epsilon_{k-i}\right| + \sum_{i=0}^{k-1} \Gamma^i \left|\epsilon'_{k-i}\right| + \Gamma^M \left|v^* - v_{\mu_{k,M}}\right|. \tag{4.42}$$

(4.43)

And finally:

$$\left|v^* - v_{\mu_{k,M}}\right| \leq \sum_{j=0}^{\infty} \Gamma^{Mj} [2\Gamma^k \left|v^* - v_0\right| + 2\sum_{i=1}^{k-1} \Gamma^i \left|\epsilon_{k-i}\right| + \sum_{i=0}^{k-1} \Gamma^i \left|\epsilon'_{k-i}\right|]. \tag{4.44}$$

□

Full analysis of NSVI Usually, one is only able to control the ϵ and ϵ' according to some norm $\|.\|_{q,\mu}$ and wants to control the difference of value according to some other norm $\|.\|_{p,\rho}$ where $\|f\|_{p,\sigma} = \left(\sum_{s\in S} |f(s)|^p \sigma(s)\right)^{\frac{1}{p}}$. Then the following theorem controls the convergence in L_p-norm:

Theorem 4.5. *Let ρ and σ be distributions over states. Let p,q and q′ be positive reals such that $\frac{1}{q} + \frac{1}{q'} = 1$, then for a non-stationary policy of size M and after k iterations we have:*

$$\begin{aligned}\left\|v^* - v_{\mu_{k,M}}\right\|_{p,\rho} \leq & \frac{2(\gamma - \gamma^k)(\mathcal{C}_q^{1,k,0,M})^{\frac{1}{p}}}{(1-\gamma)(1-\gamma^M)} \sup_{1\leq j\leq k-1} \|\epsilon_j\|_{pq',\sigma} \\ & + \frac{(1-\gamma^k)(\mathcal{C}_q^{0,k,0,M})^{\frac{1}{p}}}{(1-\gamma)(1-\gamma^M)} \sup_{1\leq j\leq k} \left\|\epsilon'_j\right\|_{pq',\sigma} \\ & + \frac{2\gamma^k}{1-\gamma^M} (\mathcal{C}_q^{k,k+1,0,M})^{\frac{1}{p}} \|v^* - v_0\|_{pq',\sigma}\,, \end{aligned} \tag{4.45}$$

With:

$$\mathcal{C}_q^{l,k,d,M} = \frac{(1-\gamma)(1-\gamma^M)}{\gamma^l - \gamma^k} \sum_{i=l}^{k-1} \sum_{j=0}^{\infty} \gamma^{i+jM} c_q(i + jM + d)$$

and where:

$$c_q(j) = \sup_{\mu_1,\nu_1,\ldots,\mu_j,\nu_j} \left\| \frac{d(\rho P_{\mu_1,\nu_1} \ldots P_{\mu_j,\nu_j})}{d\sigma} \right\|_{q,\sigma}.$$

Proof. The full proof uses standard techniques of ADP analysis. It involves a standard lemma of ADP analysis. Let us recall it (demonstration can be found in Scherrer et al. (2012)).

Lemma 4.2. *Let $\mathcal{I}$ and $(\mathcal{J}_i)_{i\in\mathcal{I}}$ be a sets of positive integers, $\{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ a partition of $\mathcal{I}$. Let f and $(g_i)_{i\in\mathcal{I}}$ be function such as:*

$$|f| \leq \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}_i} \Gamma^j |g_i| = \sum_{l=1}^{n} \sum_{i\in\mathcal{I}_l} \sum_{j\in\mathcal{J}_i} \Gamma^j |g_i|\,.$$

Then for all p, q and q′ such as $\frac{1}{q} + \frac{1}{q'} = 1$ and for all distribution ρ and σ we have

$$\|f\|_{p,\rho} \leq \sum_{l=1}^{n} (\mathcal{C}_q(l))^{\frac{1}{p}} \sup_{i\in\mathcal{I}_l} \|g_i\|_{pq',\sigma} \sum_{i\in\mathcal{I}_l} \sum_{j\in\mathcal{J}_i} \gamma^j.$$

with the concentrability coefficient written:

$$\mathcal{C}_q(l) = \frac{\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\gamma^j c_q(j)}{\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\gamma^j}.$$

Theorem 4.5 can be proven by applying lemma 4.2 with:

$$\mathcal{I} = \{1,\dots,2k\}$$
$$\mathcal{I} = \{\mathcal{I}_1,\mathcal{I}_2,\mathcal{I}_3\},\ \mathcal{I}_1 = \{1,2,\dots,k-1\},\ \mathcal{I}_2 = \{k,\dots,2k-1\},\ \mathcal{I}_3 = \{2k\}$$

$$\forall i \in \mathcal{I}_1$$
$$g_i = 2\epsilon_{k-i}$$
$$\mathcal{J}_i = \{i, i+M, i+2M, \dots\}$$

$$\forall i \in \mathcal{I}_2$$
$$g_i = \epsilon'_{k-(i-k)}$$
$$\mathcal{J}_i = \{i-k, i-k+M, i-k+2M, \dots\}$$

$$\forall i \in \mathcal{I}_3$$
$$g_i = |v^* - v_0|$$
$$\mathcal{J}_i = \{k, k+M, k+2M, \dots\}$$

With lemma 3 of Scherrer et al. (2012). we have:

$$\begin{aligned}\left\|v^* - v_{\mu_{k,M}}\right\|_{p,\rho} \leq{}& \frac{2(\gamma-\gamma^k)(\mathcal{C}_q^{1,k,0,M})^{\frac{1}{p}}}{(1-\gamma)(1-\gamma^M)} \sup_{1\leq j\leq k-1}\|\epsilon_j\|_{pq',\mu} \\ &+ \frac{(1-\gamma^k)(\mathcal{C}_q^{0,k,0,M})^{\frac{1}{p}}}{(1-\gamma)(1-\gamma^M)} \sup_{1\leq j\leq k}\left\|\epsilon'_j\right\|_{pq',\mu} \\ &+ \frac{2\gamma^k}{1-\gamma^M}(\mathcal{C}_q^{k,k+1,0,M})^{\frac{1}{p}}\|v^* - v_0\|_{pq',\mu}. \end{aligned} \tag{4.46}$$

□

7 Appendix: PSDP

In this section we prove the two theorems for PSDP schemes (theorem 4.2) and 4.3).

7.1 Appendix: PSDP2

First let us remind the PSDP1 algorithm.

$$v_{\sigma_k} = \mathcal{T}_{\mu_k,\nu_k} \dots \mathcal{T}_{\mu_0,\nu_0} 0 + \epsilon_k \tag{4.47}$$
$$\mathcal{T} v_{\sigma_k} = \mathcal{T}_{\mu_{k+1}} v_{\sigma_k} \text{ and } \mathcal{T}_{\mu_{k+1}} v_{\sigma_k} = \mathcal{T}_{\mu_{k+1},\nu_{k+1}} v_{\sigma_k} \tag{4.48}$$

Let's note $v_{\mu_{k,k+1}} = \mathcal{T}_{\mu_k} \dots \mathcal{T}_{\mu_0} v_{\mu_{k,k+1}}$ and only in this section $v_{\mu'_{k,k-i}} = \mathcal{T}_{\mu_k - i} \dots \mathcal{T}_{\mu_0} v_{\mu_{k,k+1}}$.

To prove theorem 4.2 we will first prove the following lemma:

Lemma 4.3. *$\forall k > 0$:*

$$0 \leq v_{\mu^*} - v_{\mu_{k,k+1}} \leq \Gamma_{\mu^*}^{k+1} v^* + \Gamma_{\mu^*} \Gamma^k v_{\mu_{k,k+1}} + \sum_{i=1}^{k} \Gamma_{\mu^*}^i \epsilon'_{k-i} + \sum_{i=1}^{k} \Gamma_{\mu^*} \Gamma^{i-1} \epsilon'_{k-i} \tag{4.49}$$

With $\epsilon'_k = \Gamma \epsilon_{k-1} - \epsilon_k$

Proof.

$$v^* - v_{\mu_{k,k+1}} = \mathcal{T}_{\mu^*} v^* - \mathcal{T}_{\mu^*} v_{\sigma_{k-1}} + \mathcal{T}_{\mu^*} v_{\sigma_{k-1}} - \mathcal{T}_{\mu_k} v_{\mu'_{k,k-1}} \tag{4.50}$$
$$\leq \gamma P_{\mu^*,\tilde{\nu}_k} (v^* - v_{\sigma_{k-1}}) + \mathcal{T}_{\mu^*} v_{\sigma_{k-1}} - \mathcal{T}_{\mu^*} v_{\mu'_{k,k-1}} \tag{4.51}$$
$$\leq \gamma P_{\mu^*,\tilde{\nu}_k} (v^* - v_{\sigma_{k-1}}) + \gamma P_{\mu^*,\bar{\nu}_k} (v_{\sigma_{k-1}} - v_{\mu'_{k,k-1}}) \tag{4.52}$$
$$\leq \Gamma_{\mu^*} \underbrace{(v^* - v_{\sigma_{k-1}})}_{(1)} + \Gamma_{\mu^*} \underbrace{(v_{\sigma_{k-1}} - v_{\mu'_{k,k-1}})}_{(2)} \tag{4.53}$$

To prove (1):

$$v^* - v_{\sigma_k} = \mathcal{T}_{\mu^*} v^* - \mathcal{T}_{\mu_k,\nu_k} (v_{\sigma_{k-1}} - \epsilon_{k-1}) - \epsilon_k, \tag{4.54}$$
$$= \mathcal{T}_{\mu^*} v^* - \mathcal{T}_{\mu_k,\nu_k} v_{\sigma_{k-1}} + \gamma P_{\mu_k,\nu_k} \epsilon_{k-1} - \epsilon_k, \tag{4.55}$$
$$= \mathcal{T}_{\mu^*} v^* - \mathcal{T}_{\mu_k} v_{\sigma_{k-1}} + \gamma P_{\mu_k,\nu_k} \epsilon_{k-1} - \epsilon_k, \tag{4.56}$$
$$= \mathcal{T}_{\mu^*} v^* - \mathcal{T}_{\mu^*} v_{\sigma_{k-1}} + \mathcal{T}_{\mu^*} v_{\sigma_{k-1}} - \mathcal{T}_{\mu_k} v_{\sigma_{k-1}} + \gamma P_{\mu_k,\nu_k} \epsilon_{k-1} - \epsilon_k, \tag{4.57}$$
$$\leq \mathcal{T}_{\mu^*} v^* - \mathcal{T}_{\mu^*,\tilde{\nu}_k} v_{\sigma_{k-1}} + \underbrace{\mathcal{T} v_{\sigma_{k-1}} - \mathcal{T}_{\mu_k} v_{\sigma_{k-1}}}_{=0} + \gamma P_{\mu_k,\nu_k} \epsilon_{k-1} - \epsilon_k, \tag{4.58}$$
$$\leq \mathcal{T}_{\mu^*,\tilde{\nu}_k} v^* - \mathcal{T}_{\mu^*,\tilde{\nu}_k} v_{\sigma_{k-1}} + \underbrace{\gamma P_{\mu_k,\nu_k} \epsilon_{k-1} - \epsilon_k}_{=\epsilon'_k}, \tag{4.59}$$
$$\leq \gamma P_{\mu^*,\tilde{\nu}_k} (v^* - v_{\sigma_{k-1}}) + \epsilon'_k, \tag{4.60}$$
$$\leq \gamma P_{\mu*,\tilde{\nu}_k} \dots \gamma P_{\mu*,\tilde{\nu}_0} (v^*) + \sum_{i=0}^{k-1} \gamma P_{\mu*,\tilde{\nu}_k} \dots \gamma P_{\mu*,\tilde{\nu}_{k-i+1}} \epsilon'_{k-i}, \tag{4.61}$$
$$\leq \Gamma_{\mu^*}^{k+1} (v_{\mu^*}) - \sum_{i=0}^{k-1} \Gamma_{\mu^*}^i \epsilon_{k-i} + \sum_{i=0}^{k-1} \Gamma_{\mu^*}^i \Gamma \epsilon_{k-i-1}. \tag{4.62}$$

To prove (2):

$$
\begin{aligned}
v_{\sigma_{k-1}} - v_{\mu'_{k,k-1}} &= v_{\sigma_{k-1}} - \mathcal{T}_{\mu_{k-1}} \dots \mathcal{T}_{\mu_0} v_{\mu_{k,k+1}}, && (4.63)\\
&= \mathcal{T}_{\mu_{k-1},\nu_{k-1}}(v_{\sigma_{k-2}} - \epsilon_{k-2}) + \epsilon_{k-1} - \mathcal{T}_{\mu_{k-1}} \dots \mathcal{T}_{\mu_0} v_{\mu_{k,k+1}}, && (4.64)\\
&= \mathcal{T}_{\mu_{k-1},\nu_{k-1}} v_{\sigma_{k-2}} - P_{\mu_{k-1},\nu_{k-1}} \epsilon_{k-2} + \epsilon_{k-1} - \mathcal{T}_{\mu_{k-1}} \dots \mathcal{T}_{\mu_0} v_{\mu_{k,k+1}}, && (4.65)\\
&= \mathcal{T}_{\mu_{k-1}} v_{\sigma_{k-2}} - \mathcal{T}_{\mu_{k-1}} \dots \mathcal{T}_{\mu_0} v_{\mu_{k,k+1}} - \epsilon'_{k-1}, && (4.66)\\
&= \mathcal{T}_{\mu_{k-1}} v_{\sigma_{k-2}} - \mathcal{T}_{\mu_{k-1},\hat{\nu}_{k-1}} \dots \mathcal{T}_{\mu_0} v_{\mu_{k,k+1}} - \epsilon'_{k-1}, && (4.67)\\
&= \mathcal{T}_{\mu_{k-1}} v_{\sigma_{k-2}} - \mathcal{T}_{\mu_{k-1},\hat{\nu}_{k-1}} \dots \mathcal{T}_{\mu_0} v_{\mu_{k,k+1}} - \epsilon'_{k-1}, && (4.68)\\
&\leq \mathcal{T}_{\mu_{k-1},\hat{\nu}_{k-1}} v_{\sigma_{k-2}} - \mathcal{T}_{\mu_{k-1},\hat{\nu}_{k-1}} \dots \mathcal{T}_{\mu_0} v_{\mu_{k,k+1}} - \epsilon'_{k-1}, && (4.69)\\
&\leq \gamma P_{\mu_{k-1},\hat{\nu}_{k-1}}(v_{\sigma_{k-2}} - v_{\mu'_{k,k-2}}) - \epsilon'_{k-1}, && (4.70)\\
&\leq \gamma P_{\mu_{k-1},\hat{\nu}_{k-1}} \dots \gamma P_{\mu_0,\hat{\nu}_0}(0 - v_{\mu_{k,k+1}}) \\
&\qquad - \sum_{i=1}^{k} \gamma P_{\mu_{k-1},\hat{\nu}_{k-1}} \dots \gamma P_{\mu_{k-i+1},\hat{\nu}_{k-i+1}} \epsilon'_{k-i}, && (4.71)\\
&\leq \Gamma^k v_{\mu_{k,k+1}} - \sum_{i=1}^{k} \Gamma^{i-1} \epsilon'_{k-i}. && (4.72)
\end{aligned}
$$

Finally:

$$
\begin{aligned}
v^* - v_{\mu_{k,k+1}} &\leq \Gamma_{\mu^*}(v^* - v_{\sigma_{k-1}}) + \Gamma_{\mu^*}(v_{\sigma_{k-1}} - v_{\mu'_{k,k-1}}) && (4.73)\\
&\leq \Gamma_{\mu^*}^{k+1} v^* + \sum_{i=0}^{k-1} \Gamma_{\mu^*}^{i+1} \epsilon'_{k-1-i} + \Gamma_{\mu^*} \Gamma^k v_{\mu_{k,k+1}} - \sum_{i=1}^{k} \Gamma_{\mu^*} \Gamma^{i-1} \epsilon'_{k-i} && (4.74)\\
&\leq \Gamma_{\mu^*}^{k+1} v^* + \Gamma_{\mu^*} \Gamma^k v_{\mu_{k,k+1}} + \sum_{i=1}^{k} \Gamma_{\mu^*}^{i} \epsilon'_{k-i} - \sum_{i=1}^{k} \Gamma_{\mu^*} \Gamma^{i-1} \epsilon'_{k-i} && (4.75)
\end{aligned}
$$

□

Finally, noticing v^* and $v_{\mu_{k,k+1}} \leq V_{\max}$ we have:

$$
\begin{aligned}
0 \leq v_{\mu^*} - v_{\mu_{k,k+1}} &\leq \Gamma_{\mu^*}^{k+1} v^* + \Gamma_{\mu^*} \Gamma^k v_{\mu_{k,k+1}} + \sum_{i=1}^{k} \Gamma_{\mu^*}^{i} \epsilon'_{k-i} + \sum_{i=1}^{k} \Gamma_{\mu^*} \Gamma^{i-1} \epsilon'_{k-i} && (4.76)\\
&\leq 2\gamma^{k+1} V_{\max} + \sum_{i=1}^{k} \Gamma_{\mu^*}^{i} (\Gamma \epsilon_{k-i-1} - \epsilon_{k-i}) + \sum_{i=1}^{k} \Gamma_{\mu^*} \Gamma^{i-1} (\Gamma \epsilon_{k-i-1} - \epsilon_{k-i}) && (4.77)
\end{aligned}
$$

$$
\left| v_{\mu^*} - v_{\mu_{k,k+1}} \right| \leq 2\gamma^{k+1} V_{\max} + 4 \sum_{i=0}^{k} \Gamma^i \left| \epsilon_{k-i} \right| \qquad (4.78)
$$

Lemma 4.2 concludes the proof of theorem 4.2.

7.2 Appendix: PSDP1

Below is reminded the scheme of PSDP1

$$
\begin{aligned}
v_{\sigma_k} &= \mathcal{T}_{\mu_k} \dots \mathcal{T}_{\mu_0} 0 + \epsilon_k && (4.79)\\
\mathcal{T} v_{\sigma_k} &= \mathcal{T}_{\mu_{k+1}} v_{\sigma_k} \text{ and } \mathcal{T}_{\mu_{k+1}} v_{\sigma_k} = \mathcal{T}_{\mu_{k+1},\nu_{k+1}} v_{\sigma_k} && (4.80)
\end{aligned}
$$

First we will prove the following lemma:

Lemma 4.4. *$\forall k > 0$:*

$$v_{\mu^*} - v_{\sigma_k} \leq \Gamma_{\mu^*}^{k+1} v^* + \sum_{i=1}^{k} \Gamma_{\mu^*}^{i} \epsilon'_{k-i} \tag{4.81}$$

With $\Gamma_{\mu^*}^n$ representing the class of kernel products $\{\gamma P_{\mu^*,\nu_1} \ldots . \gamma P_{\mu^*,\nu_n}$, with μ_i, ν_i random strategies$\}$. And with $\epsilon'_k = \Gamma\epsilon_{k-1} - \epsilon_k$

Proof. The proof comes from previous section. It is the bound of (1). □

Noticing $v_{\mu_{k,k+1}} \geq v_{\sigma_k} - \gamma^{k+1} V_{\max}$ and $v^* \leq V_{\max}$ we have:

$$0 \leq v_{\mu^*} - v_{\mu_{k,k+1}} \leq 2\gamma^{k+1} V_{\max} + 2 \sum_{i=0}^{k} \Gamma_{\mu^*}^{i-1} \Gamma \epsilon_{k-i} \tag{4.82}$$

Lemma 4.2 concludes the proof of theorem 4.3. However one has to do the proof with $c_{\mu*,q}(j)$ instead of $c_q(j)$.

8 Appendix: NSPI

We remind the non-stationary strategy of length m $\mu_k, \ldots, \mu_{k-m+1}$ is written $\mu_{k,m}$ and in this section $\mu'_{k,m} = \mu_{k-m+1}, \mu_k, \ldots, \mu_{k-m+1}, \mu_k, \ldots$. Let also note $\mathcal{T}_{\mu_{k,m}} = \mathcal{T}_{\mu_k} \ldots \mathcal{T}_{\mu_{k-m+1}}$. Then we will have $v_{\mu_{k,m}} = \mathcal{T}_{\mu_{k,m}} v_{\mu_{k,m}}$ and $v_{\mu'_{k,m}} = \mathcal{T}_{\mu_{k-m+1}} v_{\mu_{k,m}}$.

NSPI:

$$v_k = v_{\mu_{k,m}} + \epsilon_k \tag{4.83}$$

$$\mathcal{T} v_k = \mathcal{T}_{\mu_{k+1}} v_k \tag{4.84}$$

First let's prove the following lemma.

Lemma 4.5. *$\forall k \geq m$:*

$$0 \leq v^* - v_{\mu_{k+1,m}} \leq \Gamma_{\mu^*}^{k-m+1} (v^* - v_{\mu_{m,m}}) + 2 \sum_{j=0}^{k-m} \Gamma_{\mu^*}^{j} \Gamma (\sum_{i=0}^{+\infty} \Gamma^{im}) \epsilon_{k-j} \tag{4.85}$$

Proof. First we need an upper bound for:

$$
\begin{aligned}
v_{\mu'_{k,m}} - v_{\mu_{\mu_{k+1,m}}} &= \mathcal{T}_{\mu_{k-m+1}} v_{\mu_{k,m}} - v_{\mu_{k+1,m}} && (4.86)\\
&\leq \mathcal{T}_{\mu_{k-m+1},\hat{\nu}_{k-m+1}} v_{\mu k,m} - v_{\mu_{k+1,m}} && (4.87)\\
&\quad \text{with } \mathcal{T}_{\mu_{k-m+1},\hat{\nu}_{k-m+1}} v_k = \mathcal{T}_{\mu_{k-m+1}} v_k \\
&\leq \mathcal{T}_{\mu_{k-m+1},\hat{\nu}_{k-m+1}} (v_k - \epsilon_k) - v_{\mu_{k+1,m}} && (4.88)\\
&\leq \mathcal{T}_{\mu_{k-m+1},\hat{\nu}_{k-m+1}} v_k - \gamma P_{\mu_{k-m+1},\hat{\nu}_{k-m+1}} \epsilon_k - v_{\mu_{k+1,m}} && (4.89)\\
&\leq \mathcal{T}_{\mu_{k-m+1}} v_k - v_{\mu_{k+1,m}} - \gamma P_{\mu_{k-m+1},\hat{\nu}_{k-m+1}} \epsilon_k && (4.90)\\
&\leq \mathcal{T}_{\mu_{k+1}} v_k - v_{\mu_{k+1,m}} - \gamma P_{\mu_{k-m+1},\hat{\nu}_{k-m+1}} \epsilon_k && (4.91)\\
&\leq \mathcal{T}_{\mu_{k+1}} (v_{\mu_{k,m}} + \epsilon_k) - v_{\mu_{k+1,m}} - \gamma P_{\mu_{k-m+1},\hat{\nu}_{k-m+1}} \epsilon_k && (4.92)\\
&\leq \mathcal{T}_{\mu_{k+1},\tilde{\nu}_{k+1}} v_{\mu_{k,m}} - v_{\mu_{k+1,m}} + \gamma (P_{\mu_{k+1},\tilde{\nu}_{k+1}} - P_{\mu_{k-m+1},\hat{\nu}_{k-m+1}}) \epsilon_k && (4.93)\\
&\quad \text{with } \mathcal{T}_{\mu_{k+1},\hat{\nu}_{k+1}} v_{\mu_{k,m}} = \mathcal{T}_{\mu_{k+1}} v_{\mu_{k,m}} \\
&\leq \mathcal{T}_{\mu_{k+1}} v_{\mu_{k,m}} - v_{\mu_{k+1,m}} + \gamma (P_{\mu_{k+1},\tilde{\nu}_{k+1}} - P_{\mu_{k-m+1},\hat{\nu}_{k-m+1}}) \epsilon_k && (4.94)\\
&= \mathcal{T}_{\mu_{k+1}} \dots \mathcal{T}_{\mu_{k-m+1}} v_{\mu_{k,m}} - \mathcal{T}_{\mu_{k+1}} \dots \mathcal{T}_{\mu_{k-m+2}} v_{\mu_{k+1,m}} \\
&\quad + \gamma (P_{\mu_{k+1},\tilde{\nu}_{k+1}} - P_{\mu_{k-m+1},\hat{\nu}_{k-m+1}}) \epsilon_k && (4.95)\\
&= \mathcal{T}_{\mu_{k+1}} \dots \mathcal{T}_{\mu_{k-m+1}} v_{\mu_{k,m}} - \mathcal{T}_{\mu_{k+1},\bar{\nu}_{k+1}} \dots \mathcal{T}_{\mu_{k-m+2},\bar{\nu}_{k-m+2}} v_{\mu_{k+1,m}} \\
&\quad + \gamma (P_{\mu_{k+1},\tilde{\nu}_{k+1}} - P_{\mu_{k-m+1},\hat{\nu}_{k-m+1}}) \epsilon_k && (4.96)\\
&\quad \text{with } \mathcal{T}_{\mu_{k+1},\bar{\nu}_{k+1}} \dots \mathcal{T}_{\mu_{k-m+2},\bar{\nu}_{k-m+2}} v_{\mu_{k+1,m}} = \mathcal{T}_{\mu_{k+1}} \dots \mathcal{T}_{\mu_{k-m+2}} v_{\mu_{k+1,m}} \\
&\leq \underbrace{\gamma P_{\mu_{k+1},\bar{\nu}_{k+1}} \dots \gamma P_{\mu_{k-m+2},\bar{\nu}_{k-m+2}}}_{\bar{\Gamma}_{k+1,m}} (\underbrace{\mathcal{T}_{\mu_{k-m+1}} v_{\mu_{k,m}}}_{v_{\mu'_{k,m}}} - v_{\mu_{k+1,m}}) \\
&\quad + \gamma (P_{\mu_{k+1},\tilde{\nu}_{k+1}} - P_{\mu_{k-m+1},\hat{\nu}_{k-m+1}}) \epsilon_k && (4.97)\\
&\quad \text{Then:} && (4.98)\\
&\leq (\mathcal{I} - \bar{\Gamma}_{k+1,m})^{-1} \gamma (P_{\mu_{k+1},\tilde{\nu}_{k+1}} - P_{\mu_{k-m+1},\hat{\nu}_{k-m+1}}) \epsilon_k && (4.99)
\end{aligned}
$$

Proof of the lemma:

$$
\begin{aligned}
v^* - v_{\mu_{k+1,m}} &= \mathcal{T}_{\mu^*} v^* - \mathcal{T}_{\mu_{k+1,m}} v_{\mu_{k+1,m}} && (4.100)\\
&= \mathcal{T}_{\mu^*} v^* - \mathcal{T}_{\mu^*} v_{\mu_{k,m}} + \mathcal{T}_{\mu^*} v_{\mu_{k,m}} - \mathcal{T}_{\mu_{k+1,m+1}} v_{\mu_{k,m}} \\
&\quad + \mathcal{T}_{\mu_{k+1,m+1}} v_{\mu_{k,m}} - \mathcal{T}_{\mu_{k+1,m}} v_{\mu_{k+1,m}} && (4.101)\\
&\leq \gamma P_{\mu^*,\nu^*_{k,m}} (v^* - v_{\mu_{k,m}}) + \bar{\Gamma}_{k+1,m} (\mathcal{T}_{\mu_{k-m+1}} v_{\mu_{k,m}} - v_{\mu_{k+1,m}}) \\
&\quad + \underbrace{\mathcal{T}_{\mu^*} v_{\mu_{k,m}} - \mathcal{T}_{\mu_{k+1}} v_{\mu_{k,m}}}_{(1)} && (4.102)\\
&\quad \text{Where } \mathcal{T}_{\mu^*} v_{\mu_{k,m}} = \mathcal{T}_{\mu^*,\nu^*_{k,m}} v_{\mu_{k,m}} && (4.103)
\end{aligned}
$$

(1):

$$
\begin{aligned}
\mathcal{T}_{\mu^*} v_{\mu_{k,m}} - \mathcal{T}_{\mu_{k+1}} v_{\mu_{k,m}} &= \mathcal{T}_{\mu^*} v_{\mu_{k,m}} - \mathcal{T}_{\mu^*} v_k + \underbrace{\mathcal{T}_{\mu^*} v_k - \mathcal{T}_{\mu_{k+1}} v_k}_{\leq 0} \\
&\quad + \mathcal{T}_{\mu_{k+1}} v_k - \mathcal{T}_{\mu_{k+1}} v_{\mu_{k,m}} && (4.104)
\end{aligned}
$$

$$\leq \gamma P_{\mu^*,\nu^*_{k+1}}(v_{\mu_{k,m}} - v_k) + \gamma P_{\mu_{k+1},\tilde{\nu}_{k+1}}(v_k - v_{\mu_{k,m}}) \tag{4.105}$$

$$\text{with } \mathcal{T}_{\mu^*} v_k = \mathcal{T}_{\mu^*,\nu^*_{k+1}} v_k$$

$$\leq \gamma(P_{\mu_{k+1},\tilde{\nu}_{k+1}} - P_{\mu^*,\nu^*_{k+1}})\epsilon_k \tag{4.106}$$

And finally:

$$v^* - v_{\mu_{k+1,m}} \leq \Gamma_{\mu^*}(v^* - v_{\mu_{k,m}}) + \gamma(P_{\mu_{k+1},\tilde{\nu}_{k+1}} - P_{\mu^*,\nu^*_{k+1}})\epsilon_k \tag{4.107}$$

$$+ \bar{\Gamma}_{k+1,m}(\mathcal{I} - \bar{\Gamma}_{k+1,m})^{-1}\gamma(P_{\mu_{k+1},\tilde{\nu}_{k+1}} - P_{\mu_{k-m+1},\tilde{\nu}_{k-m+1}})\epsilon_k \tag{4.108}$$

$$\leq \Gamma_{\mu^*}(v^* - v_{\mu_{k,m}}) + 2\Gamma(\sum_{i=0}^{+\infty}\Gamma^{im})\epsilon_k \tag{4.109}$$

$$\leq \Gamma^{k-m+1}_{\mu^*}(v^* - v_{\mu_{m,m}}) + 2\sum_{j=0}^{k-m}\Gamma^j_{\mu^*}\Gamma(\sum_{i=0}^{+\infty}\Gamma^{im})\epsilon_{k-j} \tag{4.110}$$

□

Theorem 4.6. *Let ρ and σ be distributions over states. Let p,q and q' be positive reals such that $\frac{1}{q} + \frac{1}{q'} = 1$, then for a non-stationary policy of size M and after k iterations we have:*

$$\left\|v^* - v_{\mu_{k,m}}\right\|_{p,\rho} \leq \frac{2(\gamma - \gamma^{k-m+2})(\mathcal{C}_q^{1,k-m+2,0,m})^{\frac{1}{p}}}{(1-\gamma)(1-\gamma^m)} \sup_{m\leq j\leq k-1}\|\epsilon_j\|_{pq',\sigma}$$

$$+ \gamma^{k-m}(c_q(k-m))^{\frac{1}{p}}\left\|v^* - v_{\mu_{m,m}}\right\|_{pq',\sigma}, \tag{4.111}$$

Proof. The proof of the theorem 4.6 is done by applying lemma 4.2 □

Then theorem 4.4 falls using theorem 4.6.

9 Appendix: Figures

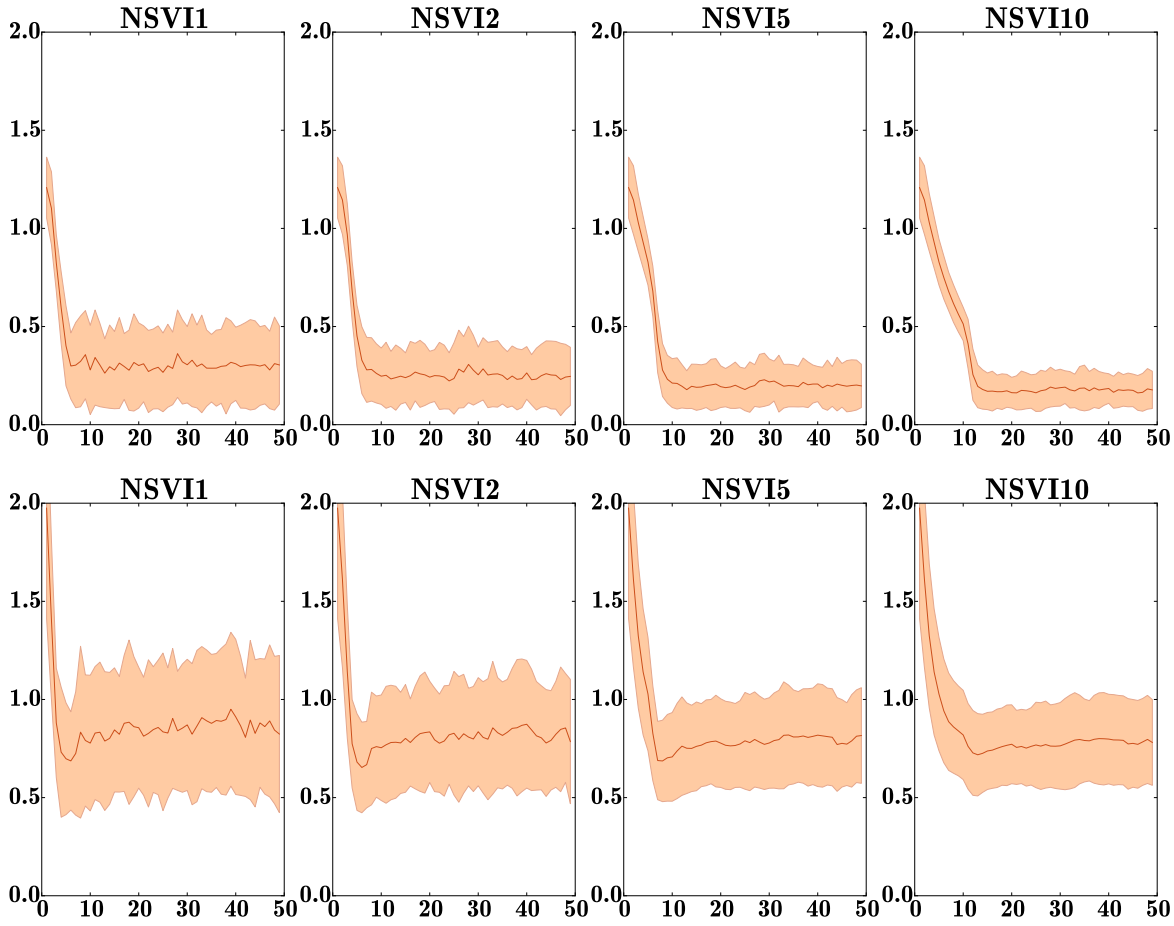


Figure 4.5 – Performance (y-axis) of the strategy at step k (x-axis) for NSVI for a strategy of length 10,5,2 and 1 from right to left. Those curves are averaged over 70 Garnet $N_S = 100$, $N_A = 5$, $N_B = 1$ (for the two curves on the top) and $N_B = 2$ (for the two curves on the bottom). All curves have a sparsity of 0.5. Each step of the algorithm uses $2.25 \times N_A \times N_S$ samples.

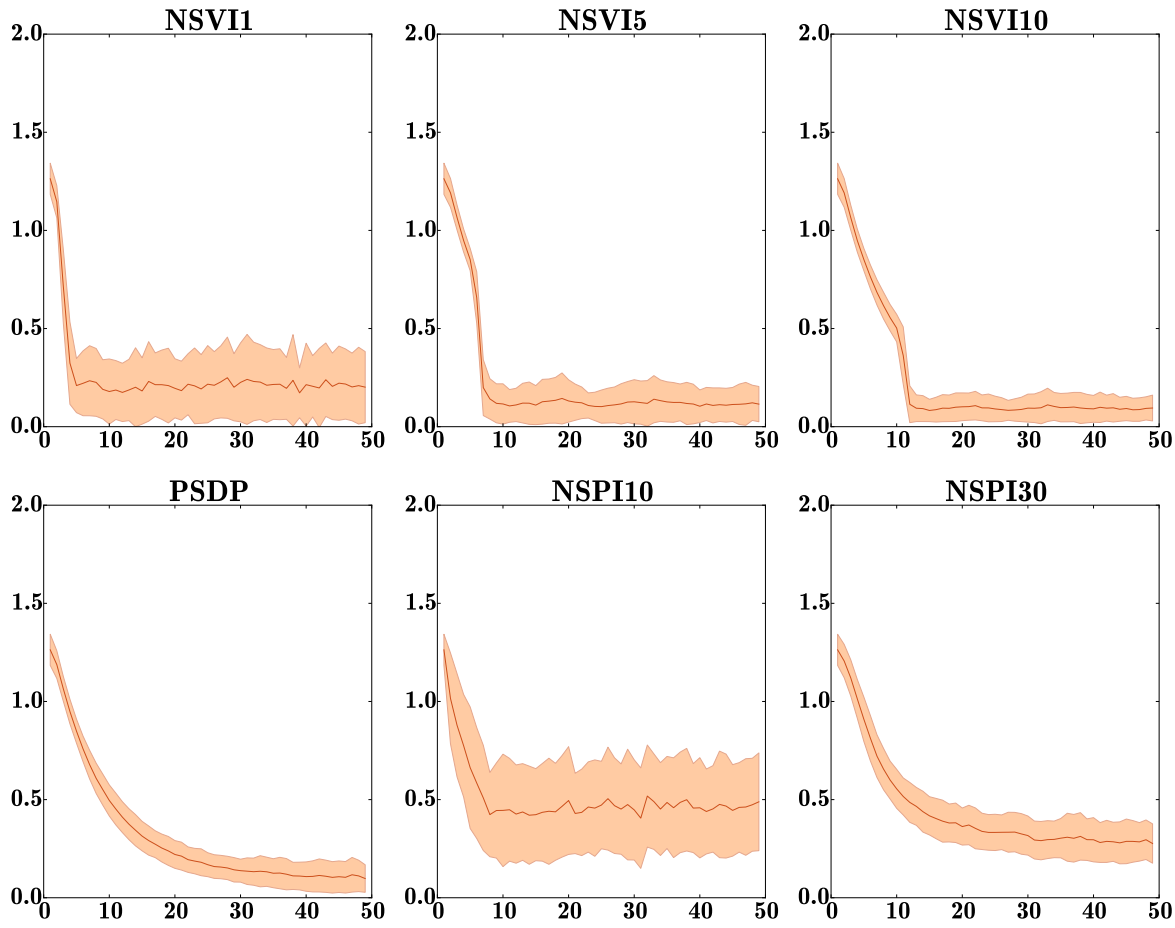


Figure 4.6 – Performance (y-axis) of the strategy at step k (x-axis) for NSVI, PSDP and NSPI. Those curves are averaged over 70 Garnet $N_S = 100$, $N_A = 8$, $N_B = 2$. All garnet have a sparsity of 0.5 and $\gamma = 0.9$. Each step of NSPI and NSVI uses $2.25 \times N_A \times N_S$ samples at each step. Each step of PSDP2 uses $2.25 \times N_A \times N_S$ rollout at each step.

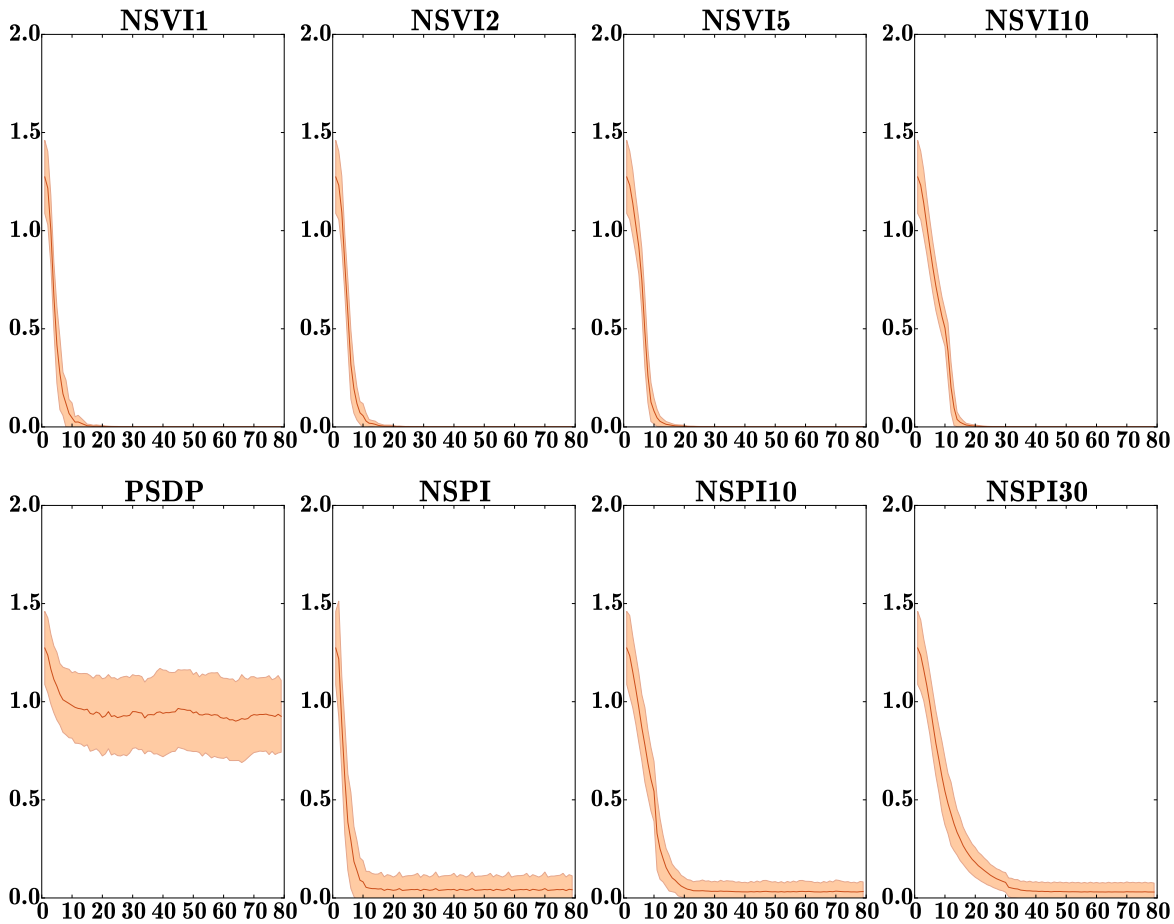


Figure 4.7 – Performance (y-axis) of the strategy at step k (x-axis) for NSVI, PSDP and NSPI. Those curves are averaged over 40 Garnet $N_S = 100$, $N_A = 5$, $N_B = 1$. All garnet have a sparsity of 0.5 and $\gamma = 0.9$. Each step of NSPI, NSVI and PSDP uses $0.75 \times k \times N_A \times N_S$ samples at step k.

Chapter 5

Non-Stationary Startegies in General-Sum Markov Games

This chapter focuses on the use of non-stationary strategies in general-sum MGs. In general sum games, two key properties that exist in zero-sum two-player MGs are missing. Those two properties are: i) the optimal Bellman operator is a γ-contraction (which guarantees the convergence toward the optimal value) and ii) acting greedily with respect to the optimal value generates the optimal strategy. In general-sum MGs, the first property does not hold since there is no known Bellman operator converging to the value of a Nash equilibrium (there is no known equivalent of $\mathcal{T}^*$ and $\mathcal{T}$). The existence of such an operator is unlikely since there is not a unique Nash equilibrium in general sum MGs. In addition, the existence of turn-based games with a unique stochastic Nash equilibrium contradicts the second property (Zinkevich et al., 2006). Indeed, acting greedily in that turn-based case would lead to a deterministic strategy which is not the unique stochastic Nash equilibrium we are looking for. Therefore and more generally, Zinkevich's example implies that no algorithm based on the value function would lead to a stationary Nash equilibrium in the case of general-sum MGs.

This is unfortunate since value function based algorithms, such as VI, benefit from a wide literature from MDPs and zero-sum two-player MGs. As described in the two previous chapters, these algorithms can handle value functions approximation as well as errors in the greedy step. It also holds in a wide range of practical scenarios such as when the dynamic is known, when only a simulator is available or when only a batch of data is provided (Perolat et al., 2015, Ernst et al., 2005, Munos, 2007, Bertsekas, 1995). But, even if VI does not converge, it seems, as mentioned by (Zinkevich et al., 2006), that VI cycles after a certain number of iterations. Thus, as suggested in (Zinkevich et al., 2006), it is worth studying to what extent the use of non-stationary (cyclic) strategies gives some guarantees of convergence to a Nash equilibrium. Loose guarantees of convergence under very strong hypotheses and under a constrained definition of an ϵ-correlated equilibrium are given in (Zinkevich et al., 2006). Even if their work focuses on correlated equilibrium (which is not the case here), two questions arise: i) how long would it take to converge to an ϵ-equilibrium (for a given ϵ), ii) how long should be the cycle for a given ϵ? In this chapter, we explore to which extent the techniques presented in the previous chapter can be used in general sum games. We first show that the non-stationary strategy found by the VI algorithm is an ϵ-Nash equilibrium where ϵ is only controlled by the length of the cycle and not by the number of iterations (which answers the two questions) contrary to MDPs. More precisely, we provide an upper bound of convergence $O(\frac{\gamma^m}{1-\gamma})$ where m is the length of the cycle whereas in

MDPs the upper bound is $O(\frac{\gamma^k}{(1-\gamma)(1-\gamma^m)})$ where k is the number of iterations. Second, we providing an example where the bound is tight up to a multiplicative constant. This example demonstrates that using value iteration with non-stationary strategies provides no better guarantees than approximating the infinite horizon γ-discounted problem with a finite horizon problem (of horizon m). In other words, playing the cyclic strategy (of VI) gives no better guarantee than playing first the m strategies of VI and then any arbitrary strategy.

1 Background on Cyclic Strategies and Nash Equilibrium

A cyclic strategy $\boldsymbol{\pi}_{\text{cycle},M} = (\boldsymbol{\pi}_0, \ldots, \boldsymbol{\pi}_{M-1})$ is a joint strategy where each player will play at time t the strategy $\boldsymbol{\pi}_j$ where $j \equiv t \pmod M$. The value of such a strategy is the following:

$$v^i_{\boldsymbol{\pi}_{\text{cycle},M}}(s) = E\Big[\sum_{t=0}^{+\infty} \gamma^t r^i_{\boldsymbol{\pi}_j}(s_t) | s_0 = s, s_{t+1} \sim \mathcal{P}_{\boldsymbol{\pi}_j}(.|s_t), j \equiv t \pmod M\Big]. \tag{5.1}$$

This value is the fixed point of the operator $\mathcal{T}^i_{\boldsymbol{\pi}_0} \ldots \mathcal{T}^i_{\boldsymbol{\pi}_{M-1}}$. The best response of player i against the cyclic strategy of all other players $\boldsymbol{\pi}^{-i}_{\text{cycle},M}$ is the following:

$$v^{*i}_{\boldsymbol{\pi}^{-i}_{\text{cycle},M}}(s) = \max_{\pi^i_t, t\in N} E\Big[\sum_{t=0}^{+\infty} \gamma^t r^i_{\pi^i_t, \boldsymbol{\pi}^{-i}_j}(s_t) | s_0 = s, s_{t+1} \sim \mathcal{P}_{\pi^i_t, \boldsymbol{\pi}^{-i}_j}(.|s_t), j \equiv t \pmod M\Big]. \tag{5.2}$$

This value is the fixed point of operator $\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_0} \ldots \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_{M-1}}$. The demonstration of this property was done in the context of zero-sum games in the previous chapter.

Nash and ϵ-Nash Equilibrium in N-Player MGs: A canonical definition of a Nash equilibrium for a cyclic strategy is the following: if everyone commits to play the cyclic strategy, no players has an incentive to deviate unilaterally from his strategy. And, in an ϵ-Nash equilibrium, no player has more than an ϵ incentive to deviate unilaterally from their own strategy. Formally:

Definition 5.1. *In a MG a cyclic strategy $\boldsymbol{\pi}_{cycle,M}$ is a Nash equilibrium if $\forall i$:*
$v^i_{\boldsymbol{\pi}_{cycle,M}} = v^{*i}_{\boldsymbol{\pi}^{-i}_{cycle,M}}$.

Definition 5.2. *In a MG, a cyclic strategy $\boldsymbol{\pi}_{cycle,M}$ is an ϵ-Nash equilibrium if:*
$\| \| v^i_{\boldsymbol{\pi}_{cycle,M}} - v^{*i}_{\boldsymbol{\pi}^{-i}_{cycle,M}} \|_{s,\infty} \|_{i,\infty} \leq \epsilon$.

Remark 5.1. This definition of a cyclic Nash equilibrium is consistent with the one of (Scherrer and Lesner, 2012) when reduced to an MDP. However, the definition of (Zinkevich et al., 2006) is stronger when reduced to Nash equilibrium. It requires the definition 5.2 to be valid for every cyclic permutation of the cyclic strategy.

2 The Value Iteration Algorithm for General-Sum MGs

In this section, we define the VI algorithm for general-sum MGs. Then, we prove a bound of performance of $\frac{4\gamma^m}{1-\gamma}$which is rather poor compared to the bound for MDPs and zero-sum two-player MGs ($\frac{2\gamma^k}{1-\gamma^m}$). However, this bound is tight up to a multiplicative constant as shown in App. 3. Finally, we empirically illustrate the use of non-stationary strategies in general-sum games.

The VI algorithm is a popular method used to find optimal or near optimal strategies for MDPs (Bertsekas, 1995) or zero-sum two-player MGs (Shapley, 1953, Perolat et al., 2015). However, it is known not to converge toward a Nash equilibrium for general-sum MGs. (Zinkevich et al., 2006) suggest to look for cyclic strategies instead of stationary strategies. Here, we present a non-stationary version of the VI algorithm for general-sum MGs (a stationary version is described in (Kearns et al., 2000)).

Algorithm 22 Non-Stationary Value Iteration

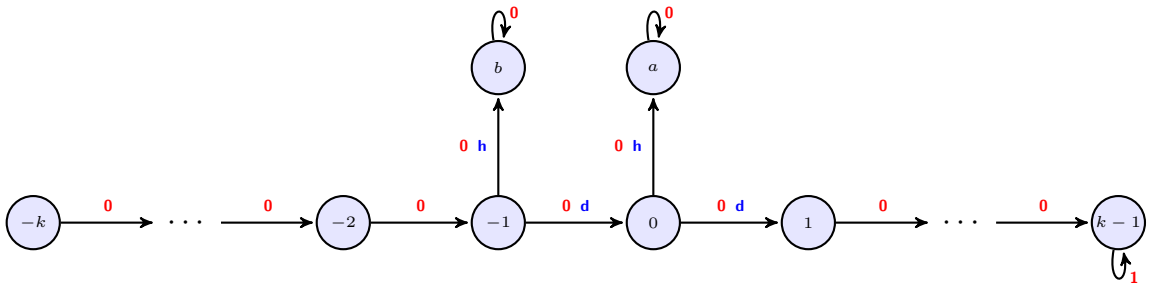
Input: constant m, k and $(v_0^1, \dots, v_0^N) = (0, \dots, 0)$
for $l = 1$ **to** k **do**
① Find $\boldsymbol{\pi}_l$ such as $\forall i \in \{1, \dots, N\}, \mathcal{T}^i_{\boldsymbol{\pi}_l} v^i_{l-1} = \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_l} v^i_{l-1}$ (Computing a Nash equilibrium in each state).
② $\forall i \in \{1, \dots, N\}$, compute $v^i_l = \mathcal{T}^i_{\boldsymbol{\pi}_l} v^i_{l-1}$ (Applying the Bellman operator)
end for
Return: $(\boldsymbol{\pi}_k, \dots, \boldsymbol{\pi}_{k-m+1})$

Theorem 5.1. *The NSVI algorithm produces a strategy* $\boldsymbol{\pi}_{k,m} = (\boldsymbol{\pi}_k, \dots, \boldsymbol{\pi}_{k-m+1})$ *which is a* $\frac{4\gamma^m R_{max}}{1-\gamma}$*-Nash equilibrium.*

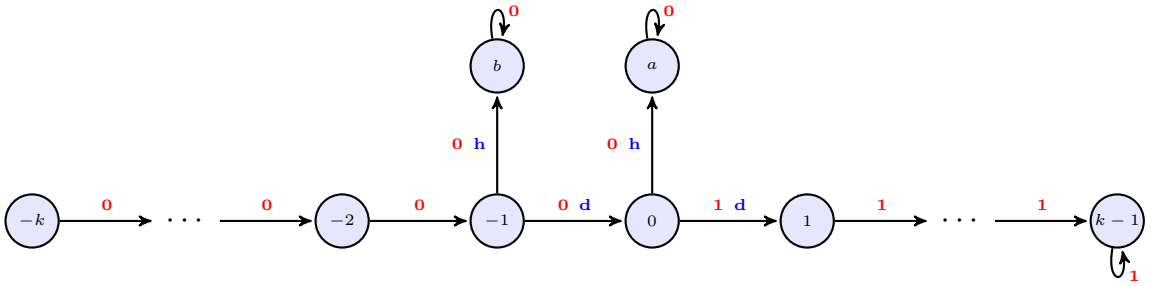
Proof. The proof is left in Appendix 7 □

The performance of the non-stationary VI algorithm for MDPs is $\frac{2R_{\max}\gamma^k}{(1-\gamma)(1-\gamma^m)}$ which is way better than the bound for general-sum MGs since the number of iterations k comes into account. In MDPs the number of iterations controls the quality of the strategy. In MGs, if we define the performance criterion as the norm of Def. 5.2, the corresponding bound is $\frac{4\gamma^m R_{\max}}{1-\gamma}$ (cf. Th. 5.1). Thus, in the general-sum case, the quality of the joint non-stationary strategy only depends on the length of the cycle. As a comparison, the guarantee would be the same if one would play first the m strategies $(\boldsymbol{\pi}_k, \dots, \boldsymbol{\pi}_{k-m+1})$ and then an arbitrary strategy. This bound seems limited since it means that we can do no better than planning m steps forward with the VI algorithm. We show in Section 3 that for all k in $\mathbb{N}$ and for all m in $\{1, \dots, k\}$ there exists an MG such that $\|\|v^i_{\boldsymbol{\pi}_{k,m}} - v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}}\|_{s,\infty}\|_{i,\infty} \geq \frac{\gamma^m}{1-\gamma}$ (with $R_{\max} = 1$). This example shows that our bound is tight up to a multiplicative constant. Still, Theorem 5.1 shows that one can compute a cyclic Nash equilibrium from value functions which is not the case of a stationary Nash equilibrium (Zinkevich et al., 2006).

3 Illustrating example of lower bound



(a) Reward of Player 1.



(b) Reward of Player 2.

Figure 5.1 – Example of game matching the lower bound. Actions are in blue and Rewards in red

In this section, we build an artificial example of a general-sum MG for which the performance criterion (cf Def. 5.2) matches, up to a multiplicative constant, the upper bound of Th. 5.1. The example is shown in Fig. 5.1. It is a turn-based two-player general-sum MG with state space $S = \{a, b, -k, -k+1, \dots, k-2, k-1\}$. In every state only one action is available to players but two states $\{0, 1\}$ where each player has two actions. The rewards depend on the state and on the action. In state 0, Player 1 (Fig. 5.1(a)) has two actions $A^1(0) = \{h, d\}$. In state -1, Player 2 (Fig. 5.1(b)) has also two actions $A^2(-1) = \{h, d\}$.

Since the game is turn-based, the greedy step which implies computing a Nash equilibrium in each state reduces to finding an argmax. When the argmax is not unique, we always choose action h over action d. As the greedy strategy is deterministic, instead of writing $\boldsymbol{\pi} = (\pi^2(.|-1), \pi^1(.|0))$ for the sake of clarity we write $\boldsymbol{\pi} = (a^2, a^1)$ with $\pi^2(a^2|-1) = 1$ and $\pi^1(a^1|0) = 1$. The value function will be written $(v(a), v(b), v(-k), \dots, v(k-1))$.

VI instantiation The VI algorithm produces the following sequence of values and strategies: Obviously $v_0^1 = v_0^2 = (0, \dots, 0)$ Then, for $0 < l < k$ we have:

$$\boldsymbol{\pi}_l = (h, h) \tag{5.3}$$

$$v_l^1 = (0, 0, 0, \ldots, v_l^1(k-l) = \frac{\gamma^{l-1} - \gamma^l}{1-\gamma}, \ldots, \frac{\gamma^0 - \gamma^l}{1-\gamma}), \tag{5.4}$$

$$v_l^2 = (0, 0, 0, \ldots, v_l^2(1) = \frac{\gamma^0 - \gamma^l}{1-\gamma}, \ldots, \frac{\gamma^0 - \gamma^l}{1-\gamma}). \tag{5.5}$$

And for $l = k$, Player 1 has an incentive to switch from action h to action d. Thus, we have:

$$\boldsymbol{\pi}_k = (h, d) \tag{5.6}$$

$$v_l^1 = (0, 0, 0, \ldots, v_l^1(0) = \frac{\gamma^{k-1} - \gamma^k}{1-\gamma}, \ldots, \frac{\gamma^0 - \gamma^k}{1-\gamma}), \tag{5.7}$$

$$v_l^2 = (0, 0, 0, \ldots, v_l^2(0) = \frac{\gamma^0 - \gamma^k}{1-\gamma}, \ldots, \frac{\gamma^0 - \gamma^k}{1-\gamma}). \tag{5.8}$$

Values, and Values of Bests Responses: Let's compute the performance criterion of Def. 5.2 for l in $\{1, \ldots, k\}$.

If $l < k$ we have:

$$v_{\boldsymbol{\pi}_{l,m}}^1 = (0, 0, 0, \ldots, 0, \frac{\gamma^{k-2}}{1-\gamma}, \ldots, \frac{1}{1-\gamma}), \tag{5.9}$$

$$v_{\boldsymbol{\pi}_{l,m}^{-1}}^{*1} = (0, 0, 0, \ldots, 0, \frac{\gamma^{k-1}}{1-\gamma}, \ldots, \frac{1}{1-\gamma}), \tag{5.10}$$

$$v_{\boldsymbol{\pi}_{l,m}}^2 = (0, 0, 0, \ldots, 0, v_{\boldsymbol{\pi}_{l,m}}^1(1) = \frac{1}{1-\gamma}, \ldots, \frac{1}{1-\gamma}), \tag{5.11}$$

$$v_{\boldsymbol{\pi}_{l,m}^{-2}}^{*2} = (0, 0, 0, \ldots, 0, v_{\boldsymbol{\pi}_{l,m}^{-2}}^{*2}(1) = \frac{1}{1-\gamma}, \ldots, \frac{1}{1-\gamma}). \tag{5.12}$$

When $l < k$, the cyclic joint strategy is in fact a stationary one (h, h). Then, the joint strategy when Player 1 plays his best response is (h, d) and is (h, h) when Player 2 plays his best response (since both actions are equivalent for him).

Therefore, the performance criterion is:

$$\forall l < k, \left\| \left\| v_{\boldsymbol{\pi}_{l,m}}^i - v_{\boldsymbol{\pi}_{l,m}^{-i}}^{*i} \right\|_{s,\infty} \right\|_{i,\infty} = \frac{\gamma^{k-1}}{1-\gamma}.$$

When $l = k$:

$$v^1_{\boldsymbol{\pi}_{l,m}} = (0,0,0,\dots,0,\frac{\gamma^{k-1}}{1-\gamma},\dots,\frac{1}{1-\gamma}), \tag{5.13}$$

$$v^{*1}_{\boldsymbol{\pi}^{-1}_{l,m}} = (0,0,0,\dots,0,\frac{\gamma^{k-1}}{1-\gamma},\dots,\frac{1}{1-\gamma}), \tag{5.14}$$

$$v^2_{\boldsymbol{\pi}_{l,m}} = (0,0,0,\dots,0,v^1_{\boldsymbol{\pi}_{l,m}}(0) = \frac{1}{1-\gamma},\dots,\frac{1}{1-\gamma}), \tag{5.15}$$

$$v^{*2}_{\boldsymbol{\pi}^{-2}_{l,m}} = (0,0,0,\dots 0, v^{*2}_{\boldsymbol{\pi}^{-2}_{l,m}}(-m) = \frac{\gamma^m}{1-\gamma}, 0,\dots,\dots,0, v^{*2}_{\boldsymbol{\pi}^{-2}_{l,m}}(0) = \frac{1}{1-\gamma},\dots,\frac{1}{1-\gamma}), \tag{5.16}$$

$$\forall i \in N,\ v^{*2}_{\boldsymbol{\pi}^{-2}_{l,m}}(-im) = \frac{\gamma^{im}}{1-\gamma}. \tag{5.17}$$

In this case, the cyclic joint strategy is $\boldsymbol{\pi}_{l,m} = ((h,d),(h,h),\dots,(h,h))$. The Player 1's best response is the joint strategy $((h,d),\dots,(h,d))$ and the Player 2' best is the joint strategy $((d,d),(d,h),\dots,(d,h))$.

When $l = k$ the performance criterion is:

$$\left\| \left\| v^i_{\boldsymbol{\pi}_{l,m}} - v^{*i}_{\boldsymbol{\pi}^{-i}_{l,m}} \right\|_{s,\infty} \right\|_{i,\infty} = \frac{\gamma^m}{1-\gamma}.$$

Then $\forall k \in \mathbb{N},\ \forall m \in \{1,\dots,k\}$, there exists an MG (our example for instance) such that:

$$\left\| \left\| v^i_{\boldsymbol{\pi}_{k,m}} - v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} \right\|_{s,\infty} \right\|_{i,\infty} \geq \frac{\gamma^m}{1-\gamma}.$$

Remark 5.2. The reader should notice that the size of the example matters. Indeed, if one runs the VI algorithm for $l > k$, one would find a Nash equilibrium of the MG (i.e. (d,d)).

The reason why this example gives the proper lower bound is that Player 1's actions control Player 2's rewards (in state 0). Thus, one can design an MG where the VI algorithm takes an arbitrary long time until Player 1 changes his action (that is the role played by states $1,\dots,k-1$). Until Player 1 switch from action h to action d in state 0 (as we said before this time can be arbitrarily long) Player 2 does not see the difference between action h and action d in state 1. That is why the number of iterations k does not come into account in the bound.

4 Approximate Value Iteration

In this section, we consider the same approach as in previous chapters to analyse the impact of such an approximation on the performance is to consider two sources of errors arising at each step of VI as illustrated in Algorithm 23.

Theorem 5.2. *The approximate NSVI algorithm produces a strategy $\boldsymbol{\pi}_{k,m}$ which is an ϵ-Nash equilibrium. Where:*

$$\epsilon = \frac{4\gamma^m R_{max}}{1-\gamma} + \frac{1-\gamma^k}{1-\gamma} \sup_{l\in\{0,\dots,k-1\}} \|\epsilon'^i_{k-l}\|_{i,\infty} + 2\frac{\gamma-\gamma^k}{1-\gamma} \sup_{l\in\{1,\dots,k-1\}} \|\epsilon^i_{k-l}\|_{i,\infty}. \tag{5.18}$$

Algorithm 23 Non Stationary Approximate Value Iteration

Input: constant m, K and $(v_0^1, \dots, v_0^N) = (0, \dots, 0)$
for $k = 1$ **to** K **do**
① Find $\boldsymbol{\pi}_k$ such as $\forall i \in \{1, \dots, N\}, \mathcal{T}^i_{\boldsymbol{\pi_k}} v^i_{k-1} + \epsilon'^i_k \geq \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_k} v^i_{k-1}$.
② $\forall i \in \{1, \dots, N\}$, compute $v^i_k = \mathcal{T}^i_{\boldsymbol{\pi_k}} v^i_{k-1} + \epsilon^i_k$
end for
Return: $(\boldsymbol{\pi}_K, \dots, \boldsymbol{\pi}_{K-m+1})$

Proof. The proof is left in Appendix 8. □

5 Experiments

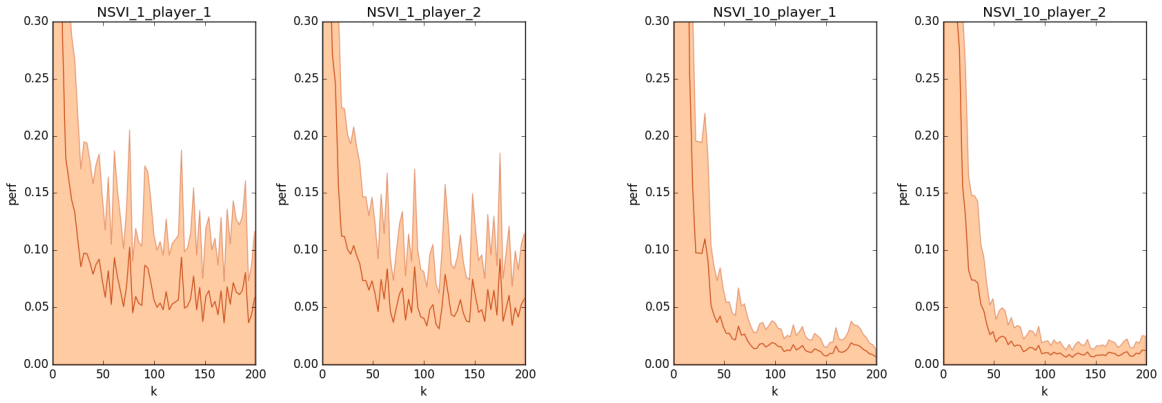
We empirically illustrate the previous theoretical results on Garnets (Chapter 4). They are randomly generated synthetic MGs with three parameters (N_S, N_A, N_B). The parameter N_S is the number of states, N_A is the number of actions and N_B is a parameter controlling the branching factor of the MDP. The process to generate a Garnet is the same as the one for two player zero-sum MGs except for the reward. Here, reward function only depends on the state and is drawn according to a centered normal law of variance 1. Only a given ratio (the sparsity) of rewards are non-null.

Figure 5.2(a) and 5.2(b) show $\|v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - v^i_{\boldsymbol{\pi_{k,m}}}\|_2 \backslash \|v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}}\|_2$ according to k. It is a normalized way to quantify what a player could win by switching his strategy unilaterally. From now on, we will refer to this quantity as the performance of the algorithm even though the lower the better. The envelope in figures 5.2(a) and 5.2(b) are proportional to the standard deviation of the performance measure.

The bound of theorem 5.1 does guarantee to improve the performance as m grows. Figure 5.2(a) shows the performance of the stationary strategy $\boldsymbol{\pi}_k$ of Players 1 and 2 according to the number of iterations. One can notice that, in the beginning, the performance seems to decrease exponentially toward a asymptotic regime. Then there is no more improvement. Figure 5.2(b) presents the performance for the cyclic strategy of length 10 $(\boldsymbol{\pi}_k, \dots, \boldsymbol{\pi}_{k-9})$ at iteration k. For the non-stationary strategy the performance still converges toward some stationary regime but the performance on that regime is improved.

Then we compare the performance with respect to m while varying γ. According to the bound of Theorem 5.1, $\|v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - v^i_{\boldsymbol{\pi_{k,m}}}\|_2$ should be lower than $\frac{4\gamma^m R_{\max}}{1-\gamma}$. Figure 5.3(a) reports an experiment where VI is ran with $k = 200$. It shows the logarithm of the expected performance of the cyclic strategy $\boldsymbol{\pi}_{k,m}$ for $m \in \{5, 10, 15, \dots, 100\}$ and $\gamma \in \{0.9, 0.95, 0.99\}$ for Player 1. In the worst case, the logarithm of the expected performance should decrease as $m \log(\gamma)$. This is confirmed even in the case of randomly generated MGs.

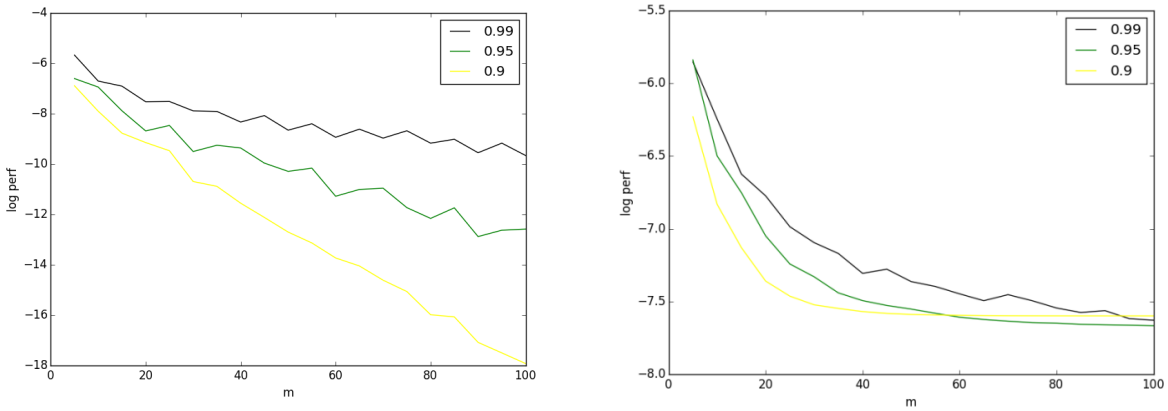
We applied approximate VI on Garnets to illustrate the empirical validity of the bound of Theorem 5.2. To do so, in Algorithm 23, we consider step ① as exact,



(a) Stationary VI

(b) Non-Stationary VI

Figure 5.2 – Performance (y-axis) as a function of k (x-axis) for VI with a stationary strategy 5.2(a) and a cyclic strategy of length 10 5.2(b). Results are averaged over 100 Garnets $N_S = 100$, $N_A = 10$, $N_B = 2$. All Garnets have a sparsity of 0.2 and $\gamma = 0.99$.



(a) Non-Stationary VI

(b) Approximate Non-Stationary VI

Figure 5.3 – Log of the expected performance (y-axis) as a function of m (x-axis) for NSVI with $k = 200$ for different value of γ ($\gamma \in \{0.9, 0.95, 0.99\}$). Results are averaged over 160 Garnets $N_S = 100$, $N_A = 10$, $N_B = 8$. All Garnets have a sparsity of 0.7.

but in step ② we introduce an error ϵ_k^i generated according to a centered Gaussian distribution of standard deviation $\sigma = 0.1$. For small lengths of the non-stationary strategy, the performance is lower when γ is small. For small cyclic strategies, the term $\frac{4\gamma^m R_{\max}}{1-\gamma}$ is dominant and it seems that the behavior of the algorithm is the same as the one on Fig. 5.3(a). Then, in Fig. 5.3(b) the performance seems to saturate for long cycles. Indeed, for large m the dominant term is $2\frac{\gamma-\gamma^k}{1-\gamma} \sup_{l\in\{1,\dots,k-1\}} \|\epsilon^i_{k-l}\|_{i,\infty} + \frac{1-\gamma^k}{1-\gamma} \sup_{l\in\{0,\dots,k-1\}} \|\epsilon'^i_{k-l}\|_{i,\infty}$while the performance is normalized by the norm of $v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}}$ (which is proportional to $\frac{1}{1-\gamma}$). Thus the sensitivity to γ is reduced for large m and all curves converge toward the same value.

6 Conclusion

The use of cyclic strategies has been the focus of recent works in approximate Dynamic Programming (DP) on MDPs (Scherrer and Lesner, 2012) and in zero-sum two-player MGs (see the previous chapter). However, these work concentrate on reducing the sensitivity to the error propagated through iterations. In this chapter, we highlight their use in exact DP for general-sum MGs.

Our main contribution is to show that Value Iteration (VI) can still be used to solve general-sum MGs when considering non-stationary strategies. Our approach relies on the fact that an infinite horizon γ-discounted MG can be well approximated by a finite horizon γ-discounted MG. We solve a m-finite horizon γ-discounted MG with somewhat similar techniques as in (Kearns et al., 2000) (which gives us a non-stationary strategy). Then, we show that playing this strategy in a cycle gives a satisfactory non-stationary solution to the infinite horizon γ-discounted problem. Finally, we exhibit an example showing that this guarantee can only be improved by a multiplicative constant. This example emphasizes a strong limitation of the use of cyclic strategies in VI for general-sum games. Indeed, the lower bound demonstrates that playing a cyclic strategy does not grant better guarantees than playing first the m strategies of VI and then playing any random strategy.

7 Proof of Theorem 5.1

From definition 5.2, we know that we should bound the following $\mathcal{L}_\infty$-norm $\left\| \left\| v^i_{\boldsymbol{\pi}_{k,m}} - v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} \right\|_{s,\infty} \right\|_{i,\infty}$.

In other words, we want to compare $\forall i \in \{1, \dots, N\}$ the value of $\boldsymbol{\pi_{k,m}}$ (i.e. $v^i_{\boldsymbol{\pi}_{k,m}} = \mathcal{T}^i_{\boldsymbol{\pi_k}} \dots \mathcal{T}^i_{\boldsymbol{\pi_{k-m+1}}} v^i_{\boldsymbol{\pi}_{k,m}}$) with the value of the best response of player i to others' joint strategy $\boldsymbol{\pi}^{-i}_{k,m} = (\boldsymbol{\pi}^{-i}_k, \dots, \boldsymbol{\pi}^{-i}_{k-m+1})$ (i.e. $v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} = \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_k} \dots \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_{k-m+1}} v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}}$). Let's define $\bar{\pi}^i_l$ the strategy such as $\mathcal{T}^i_{\bar{\pi}^i_k, \boldsymbol{\pi}^{-i}_k} \dots \mathcal{T}^i_{\bar{\pi}^i_{k-m+1}, \boldsymbol{\pi}^{-i}_{k-m+1}} v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} = \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_k} \dots \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_{k-m+1}} v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}}$. Then $\forall i \in \{1, \dots, N\}$ we have:

$$
\begin{aligned}
0 \leq & v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - v^i_{\boldsymbol{\pi_{k,m}}} && (5.19)\\
= & \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_k} \dots \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_{k-m+1}} v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - \mathcal{T}^i_{\boldsymbol{\pi_k}} \dots \mathcal{T}^i_{\boldsymbol{\pi_{k-m+1}}} v^i_{\boldsymbol{\pi_{k,m}}}, && (5.20)\\
= & \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_k} \dots \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_{k-m+1}} v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_k} \dots \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_{k-m+1}} v^i_{k-m} \\
& + \mathcal{T}^i_{\boldsymbol{\pi_k}} \dots \mathcal{T}^i_{\boldsymbol{\pi_{k-m+1}}} v^i_{k-m} - \mathcal{T}^i_{\boldsymbol{\pi_k}} \dots \mathcal{T}^i_{\boldsymbol{\pi_{k-m+1}}} v^i_{\boldsymbol{\pi_{k,m}}}, && (5.21)\\
= & \gamma \mathcal{P}_{\boldsymbol{\pi_k}} \dots \gamma \mathcal{P}_{\boldsymbol{\pi_{k-m+1}}} (v^i_{k-m} - v^i_{\boldsymbol{\pi_{k,m}}}) \\
& + \mathcal{T}^i_{\bar{\pi}^i_k, \boldsymbol{\pi}^{-i}_k} \dots \mathcal{T}^i_{\bar{\pi}^i_{k-m+1}, \boldsymbol{\pi}^{-i}_{k-m+1}} v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_k} \dots \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_{k-m+1}} v^i_{k-m}, && (5.22)\\
\leq & \gamma \mathcal{P}_{\boldsymbol{\pi_k}} \dots \gamma \mathcal{P}_{\boldsymbol{\pi_{k-m+1}}} (v^i_{k-m} - v^i_{\boldsymbol{\pi_{k,m}}}) + \mathcal{T}^i_{\bar{\pi}^i_k, \boldsymbol{\pi}^{-i}_k} \dots \mathcal{T}^i_{\bar{\pi}^i_{k-m+1}, \boldsymbol{\pi}^{-i}_{k-m+1}} v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} \\
& - \mathcal{T}^i_{\bar{\pi}^i_k, \boldsymbol{\pi}^{-i}_k} \dots \mathcal{T}^i_{\bar{\pi}^i_{k-m+1}, \boldsymbol{\pi}^{-i}_{k-m+1}} v^i_{k-m}, && (5.23)\\
\leq & \gamma \mathcal{P}_{\boldsymbol{\pi_k}} \dots \gamma \mathcal{P}_{\boldsymbol{\pi_{k-m+1}}} (v^i_{k-m} - v^i_{\boldsymbol{\pi_{k,m}}}) \\
& + \gamma \mathcal{P}_{\bar{\pi}^i_k, \boldsymbol{\pi}^{-i}_k} \dots \gamma \mathcal{P}_{\bar{\pi}^i_{k-m+1}, \boldsymbol{\pi}^{-i}_{k-m+1}} (v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - v^i_{k-m}). && (5.24)
\end{aligned}
$$

Finally, $\forall i \in \{1, \dots, N\}$ (noticing that the three quantities $v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}}$, v^i_{k-m} and $v^i_{\boldsymbol{\pi_{k,m}}}$ are smaller than $\frac{R_{\max}}{1-\gamma}$):

$$
v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - v^i_{\boldsymbol{\pi_{k,m}}} \leq \frac{4\gamma^m R_{\max}}{1-\gamma}. \tag{5.25}
$$

8 Proof of Theorem 5.2

Here we write $(\tilde{\pi}^i_k, \dots, \tilde{\pi}^i_1)$ such as $\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_k} \dots \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_1} v^i_0 = \mathcal{T}^i_{\tilde{\pi}^i_k, \boldsymbol{\pi}^{-i}_k} \dots \mathcal{T}^i_{\tilde{\pi}^i_1, \boldsymbol{\pi}^{-i}_1} v^i_0$.

$$
\begin{aligned}
0 \leq v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - v^i_{\boldsymbol{\pi_{k,m}}} & \leq v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - \mathcal{T}^i_{\boldsymbol{\pi_k}} v^i_{k-1} + \mathcal{T}^i_{\boldsymbol{\pi_k}} v^i_{k-1} - v^i_{\boldsymbol{\pi_{k,m}}}, && (5.26)\\
& \leq v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_k} v^i_{k-1} + \epsilon'^i_k + \mathcal{T}^i_{\boldsymbol{\pi_k}} v^i_{k-1} - v^i_{\boldsymbol{\pi_{k,m}}}, && (5.27)\\
& \leq v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - \mathcal{T}^i_{\tilde{\pi}^i_k, \boldsymbol{\pi}^{-i}_k} v^i_{k-1} + \epsilon'^i_k + \mathcal{T}^i_{\boldsymbol{\pi_k}} v^i_{k-1} - v^i_{\boldsymbol{\pi_{k,m}}}. && (5.28)
\end{aligned}
$$

With $v^i_{k-1} = \mathcal{T}^i_{\boldsymbol{\pi_{k-1}}} v^i_{k-2} + \epsilon^i_{k-1}$, with $\mathcal{T}^i_{\boldsymbol{\pi_k}} v^i_{k-1} \geq \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_k} v^i_{k-1} - \epsilon'^i_k$ and noticing that for all $v \leq v'$ and for all $\boldsymbol{\pi}$ we have $\mathcal{T}^i_{\boldsymbol{\pi}} v \leq \mathcal{T}^i_{\boldsymbol{\pi}} v'$, then:

$$
\begin{aligned}
v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - v^i_{\boldsymbol{\pi_{k,m}}} &\leq v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - \mathcal{T}^i_{\tilde{\pi}^i_k,\boldsymbol{\pi}^{-i}_k}\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_{k-1}} v^i_{k-2} + \mathcal{T}^i_{\boldsymbol{\pi_k}}\mathcal{T}^i_{\boldsymbol{\pi_{k-1}}} v^i_{k-2} - v^i_{\boldsymbol{\pi_{k,m}}} + \epsilon'^i_k \\
&\quad + \gamma\mathcal{P}_{\tilde{\pi}^i_k,\boldsymbol{\pi}^{-i}_k}\epsilon'^i_{k-1} - \gamma\mathcal{P}_{\tilde{\pi}^i_k,\boldsymbol{\pi}^{-i}_k}\epsilon^i_{k-1} + \gamma\mathcal{P}_{\boldsymbol{\pi}_k}\epsilon^i_{k-1}, && (5.29)\\
&\leq v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - \mathcal{T}^i_{\tilde{\pi}^i_k,\boldsymbol{\pi}^{-i}_k}\mathcal{T}^i_{\tilde{\pi}^i_{k-1},\boldsymbol{\pi}^{-i}_{k-1}} v^i_{k-2} + \mathcal{T}^i_{\boldsymbol{\pi_k}}\mathcal{T}^i_{\boldsymbol{\pi_{k-1}}} v^i_{k-2} - v^i_{\boldsymbol{\pi_{k,m}}} + \epsilon'^i_k \\
&\quad + \gamma\mathcal{P}_{\tilde{\pi}^i_k,\boldsymbol{\pi}^{-i}_k}\epsilon'^i_{k-1} - \gamma\mathcal{P}_{\tilde{\pi}^i_k,\boldsymbol{\pi}^{-i}_k}\epsilon^i_{k-1} + \gamma\mathcal{P}_{\boldsymbol{\pi}_k}\epsilon^i_{k-1}, && (5.30)\\
&\leq v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - \mathcal{T}^i_{\tilde{\pi}^i_k,\boldsymbol{\pi}^{-i}_k}\dots\mathcal{T}^i_{\tilde{\pi}^i_1,\boldsymbol{\pi}^{-i}_1} v^i_0 + \mathcal{T}^i_{\boldsymbol{\pi_k}}\dots\mathcal{T}^i_{\boldsymbol{\pi_1}} v^i_0 - v^i_{\boldsymbol{\pi_{k,m}}} && (5.31)\\
&\quad + \sum_{j=0}^{k-1} \gamma\mathcal{P}_{\tilde{\pi}^i_k,\boldsymbol{\pi}^{-i}_k}\dots\gamma\mathcal{P}_{\tilde{\pi}^i_{k-j+1},\boldsymbol{\pi}^{-i}_{k-j+1}}\epsilon'^i_{k-j} && (5.32)\\
&\quad - \sum_{j=1}^{k-1} \gamma\mathcal{P}_{\tilde{\pi}^i_k,\boldsymbol{\pi}^{-i}_k}\dots\gamma\mathcal{P}_{\tilde{\pi}^i_{k-j+1},\boldsymbol{\pi}^{-i}_{k-j+1}}\epsilon^i_{k-j} && (5.33)\\
&\quad + \sum_{j=1}^{k-1} \gamma\mathcal{P}_{\boldsymbol{\pi}_k}\dots\gamma\mathcal{P}_{\boldsymbol{\pi}_{k-j+1}}\epsilon^i_{k-j}. && (5.34)
\end{aligned}
$$

With $\tilde{v}^i_{k-m} = \mathcal{T}^i_{\boldsymbol{\pi_{k-m}}}\dots\mathcal{T}^i_{\boldsymbol{\pi_1}} v^i_0$, with $\bar{v}^i_{k-m} = \mathcal{T}^i_{\tilde{\pi}^i_{k-m},\boldsymbol{\pi}^{-i}_{k-m}}\dots\mathcal{T}^i_{\tilde{\pi}^i_1,\boldsymbol{\pi}^{-i}_1} v^i_0$, noticing $\bar{v}^i_{k-m}$ and $\tilde{v}^i_{k-m} \leq \frac{1}{1-\gamma}$ and from the proof of Th. 5.1 we have:

$$
\begin{aligned}
&v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - \mathcal{T}^i_{\tilde{\pi}^i_k,\boldsymbol{\pi}^{-i}_k}\dots\mathcal{T}^i_{\tilde{\pi}^i_{k-m+1},\boldsymbol{\pi}^{-i}_{k-m+1}}\dots\mathcal{T}^i_{\tilde{\pi}^i_1,\boldsymbol{\pi}^{-i}_1} v^i_0 + \mathcal{T}^i_{\boldsymbol{\pi_k}}\dots\mathcal{T}^i_{\boldsymbol{\pi_{k-m+1}}}\dots\mathcal{T}^i_{\boldsymbol{\pi_1}} v^i_0 - v^i_{\boldsymbol{\pi_{k,m}}} && (5.35)\\
&= \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_k}\dots\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_{k-m+1}} v^{*i}_{\boldsymbol{\pi}^{-i}_{k,m}} - \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_k}\dots\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}_{k-m+1}} \bar{v}^i_{k-m} \\
&\quad + \mathcal{T}^i_{\boldsymbol{\pi_k}}\dots\mathcal{T}^i_{\boldsymbol{\pi_{k-m+1}}}\tilde{v}^i_{k-m} - \mathcal{T}^i_{\boldsymbol{\pi_k}}\dots\mathcal{T}^i_{\boldsymbol{\pi_{k-m+1}}} v^i_{\boldsymbol{\pi_{k,m}}}, && (5.36)\\
&\leq \frac{4\gamma^m R_{\max}}{1-\gamma}. && (5.37)
\end{aligned}
$$

And the three sums (5.32), (5.33) and (5.34) are bounded by:

$$
2\frac{\gamma-\gamma^k}{1-\gamma}\sup_{l\in\{1,\dots,k-1\}}\|\epsilon^i_{k-l}\|_{i,\infty} + \frac{1-\gamma^k}{1-\gamma}\sup_{l\in\{0,\dots,k-1\}}\|\epsilon'^i_{k-l}\|_{i,\infty}, \tag{5.38}
$$

which concludes the proof.

Part III

Learning in Games : A Bellman Residual Approach

Chapter 6

Bellman Residual Minimization in Zero-Sum Games

The first part of this dissertation explored how ADP methods could be used to produce sample based algorithms for games. In MDPs, the Policy Iteration (PI) algorithm can also be seen as a Newton's method on the Optimal Bellman Residual (OBR) $\|v - \mathcal{T}^* v\|_2$. However, in zero-sum two player MGs, the Newton's method on the OBR (Pollatschek and Avi-Itzhak's algorithm) does not reduce to the policy iteration for zero-sum two-player MGs (Hoffman and Karp's algorithm). In a nutshell, this chapter shows that two sample-based approximate PI algorithms for MDPs (Least-Squares Policy Iteration (LSPI) and Bellman Residual Minimizing Policy Iteration (BRMPI)) and one algorithm for two player MGs (LSPI for MGs) can be seen as Newton's methods on some form of the OBR.

This simple observation leads to several novel contributions. First it reveals that the LSPI algorithm for zero-sum two-player MGs (which is a sample-based algorithm using function approximation) is the Pollatschek and Avi-Itzhak's algorithm when the model is known and when no function approximation is used. Thus, the LSPI algorithm for MGs does not converge even when the model is known since the Pollatschek and Avi-Itzhak's algorithm can oscillate. All these algorithms (LSPI and BRMPI) suffers from oscillation issues. Our second set of contributions is to consider the use of the quasi-Newton methods instead of Newton's method to address this oscillation problem. This solution was introduced to solve the oscillation issue of the (Pollatschek and Avi-Itzhak's algorithm in (Filar and Tolwinski, 1991). The modification of Filar and Tolwinski amounts to use a quasi-Newton method on the $\mathcal{L}_2$-norm of the OBR instead of a Newton's Method. Their proof of convergence is based on the assumption that the $\mathcal{L}_2$-norm of the OBR is smooth, which is untrue in general (as we show at the end of Section 1). Indeed, the derivative of the $\mathcal{L}_2$-norm of the OBR might be discontinuous and thus a quasi-Newton method is not anymore guaranteed to converge to a local minimum (counterexamples exist Lewis and Overton (2013)). But, in practice (Lewis and Overton, 2013), using quasi-Newton methods often leads to pretty good solutions even in the non-smooth case.

The resulting algorithms are more thrifty than other approximate PI like algorithms and do not suffer from the oscillation problem.

1 Background

First let us recall quickly the PI algorithm for MDP and the (Pollatschek and Avi-Itzhak's algorithm. The two algorithms perform similar updates are detailed in the following table (table 6.1):

Table 6.1 – Policy Iteration and (Pollatschek and Avi-Itzhak's algorithm

Policy Iteration	Pollatschek and Avi-Itzhak
π_k such as $\mathcal{T}_{\pi_k} v_{k-1} = \mathcal{T}^* v_{k-1}$	μ_k, ν_k such as $\mathcal{T} v_{k-1} = \mathcal{T}_{\mu_k} v_{k-1} = \mathcal{T}_{\mu_k,\nu_k} v_{k-1} = \hat{\mathcal{T}}_{\nu_k} v_{k-1}$
$v_k = v_{\pi_k}$	$v_k = v_{\mu_k,\nu_k}$

The algorithm of Pollatschek and Avi-Itzhak does not compute the best response at each iteration as the PI algorithm of Hoffman and Karp. Instead, it computes the value of a joint strategy. The complexity of computing the fixed point of $\mathcal{T}_{\mu_{k+1},\nu_{k+1}}$ is just inverting a matrix of size $card(S)$ instead of solving an MDP (as in the algorithm of Hoffman and Karp). However, this scheme doesn't work in general (Van Der Wal, 1978).

Let us recall some part of the related work described in Section 3.4. Van Der Wal (1978) shows that Shapley's algorithm is slower in practice than the (Pollatschek and Avi-Itzhak's algorithm and the Hoffman and Karp's PI. It thus motivates the use of PI schemes. Between the two extensions of PI to games, the Pollatschek and Avi-Itzhak's algorithm is the less computationally intensive. In (Filar and Tolwinski, 1991), this algorithm is slightly modified to introduce a learning rate. The update is $v_{k+1} = \alpha v_{\mu_{k+1},\nu_{k+1}} + (1-\alpha) v_k$ where α is chosen according to Armijo's Rule. This comes from the fact that the Pollatschek and Avi-Itzhak's algorithm is a Newton's method on the $\mathcal{L}_2$-norm of the Bellman residual $||v - \mathcal{T}^* v||_2^2$ (proof in Section 3.4.5). The adaptation of the Pollatschek and Avi-Itzhak's algorithm introduced by Filar and Tolwinski (1991) uses a quasi-Newton method instead of a Newton's method on the objective $\mathcal{J}(v) = ||v - \mathcal{T}^* v||_2^2$. To argue for the convergence of their algorithm towards the optimal value of the game, Filar and Tolwinski show first that a local minimum of the objective is also a global minimum and then they assure that their quasi-Newton method converges to a local minimum. However, they use a version of the Zoutendijk theorem Nocedal and Wright (2006) to prove the convergence to a local minimum. This theorem requires the gradient of the objective function ($\mathcal{L}_2$-norm of the Bellman residual) to exists (in the Fréchet sense) and to be a Lipschitz function. This assumption does not hold in the case of MDPs or MGs because of the max and minmax operators respectively. The question whether there is convergence to a local minimum or not when the gradient is not Lipschitz is still an open problem in optimization. However, empirical evidence suggests quasi-Newton methods always converge to a Clarke stationary point even when the gradient is not Lipschitz Lewis and Overton (2009, 2013). This means that, despite Pollatschek and Avi-Itzhak's proof does not stand, there is good evidence that using quasi-Newton methods will empirically perform well. Again, there are no theoretical guarantees that quasi-Newton methods converge to a local optimum in the case of

non-convex and non-smooth functions Lewis and Overton (2013) (which is the case of the Bellman residual). But, as written by Lemaréchal it can be "good practice to use a quasi-Newton method in nonsmooth optimization" in his opinion it "is essentially due to the fact that inaccurate line-searches are made". He also indicates that there are "no theoretical possibility to prove convergence to the right point (in fact counterexamples exist)" (see Lewis and Overton (2009)).

Newton's method and quasi-Newton methods: Newton's method is an optimization technique aiming at minimizing a function $f : \mathcal{R}^n \to \mathcal{R}$. Starting from $x_0 \in \mathbb{R}^n$, it computes the sequence $(x_n)_{n\in\mathbb{N}}$ such that:

$$x_{n+1} = x_n - [Hf(x_n)]^{-1}\nabla f(x_n), \tag{6.1}$$

where $Hf(x_n)$ and $\nabla f(x_n)$ are respectively the Hessian matrix and the gradient in x_n. This method might be unstable Nocedal and Wright (2006) and one way to soften it is to introduce a learning rate α_k meeting the Wolf conditions and to compute the sequence:

$$x_{n+1} = x_n - \alpha_k[Hf(x_n)]^{-1}\nabla f(x_n). \tag{6.2}$$

This is a quasi-Newton method. When the function f is not differentiable (in the classical Fréchet sense), one can use more general definitions of differential (or gradient) such as the Clarke differential cla for instance. In Section 2, we discuss our choice of differential for our specific objectives which are not differentiable everywhere due to the use of the minmax operator.

2 Newton's Method on the OBR with Linear Function Approximation

As in Section 3.5.1 and in Section 1.2.2, in Chapter 2 Q-functions are represented as a linear combination of d linear independent features $\Phi = [\phi_1, \phi_2, ..., \phi_d]$, where $\phi_i \in \mathbb{R}^{S\times A^2}$. Thus, for each $\omega \in \mathbb{R}^d$, we can define a Q-function $Q_\omega = \sum_{i=1}^d \omega_i\phi_i = \Phi\omega$. We also define the linear span corresponding to Φ as $\text{Span}(\Phi) = \{Q_\omega\}_{\omega\in\mathbb{R}^d}$. Moreover, it is usual to see Φ as a matrix where $\Phi[(s, a, b), i] = \phi_i(s, a, b)$. Thus, we have $Q_\omega = \Phi\omega$. In addition, we recall that for any element Q of $\mathbb{R}^{S\times A^2}$, its orthogonal projection $\Pi_{\rho,\Phi}Q$ under a non-null measure ρ over $S \times A^2$ is defined as:

$$\Pi_{\rho,\Phi}Q = \underset{u\in\text{Span}(\Phi)}{\text{argmin}} \|Q - u\|_{2,\rho} = \Phi(\Phi^\top\Delta_\rho\Phi)^{-1}\Phi^\top\Delta_\rho Q, \tag{6.3}$$

Two canonical objectives to minimize can be defined in order to search for a Q-function within $\text{Span}(\Phi)$: the OBR $\mathcal{J}_{OBR}(\omega)$ and the POBR $\mathcal{J}_{POBR}(\omega)$:

$$\mathcal{J}_{OBR}(\omega) = \frac{1}{2}||\Phi\omega - \mathcal{T}^*\Phi\omega||^2_{2,\rho}, \tag{6.4}$$

$$\mathcal{J}_{POBR}(\omega) = \frac{1}{2}||\Phi\omega - \Pi_{\rho,\Phi}\mathcal{T}^*\Phi\omega||^2_{2,\rho}, \tag{6.5}$$

$$= \frac{1}{2}||\Phi^\top(\Phi^\top\Delta_\rho\Phi)^{-1}\Phi^\top\Delta_\rho(\Phi\omega - \mathcal{T}^*\Phi\omega)||^2_{2,\rho}. \tag{6.6}$$

Notice that explicit minimization of the OBR through gradient descent is not new (Piot et al., 2014a, Baird et al., 1995), as for the POBR with approximate stochastic gradient descent (Maei et al., 2010). However, few works, at our knowledge, focus on the use of Newton's or quasi-Newton techniques to minimize those objectives. The Newton's method requires computing a gradient and a Hessian matrix. This means that one needs to compute the derivative of $\mathcal{T}^*\Phi\omega$ with respect to ω.

Computing the gradient of $\mathcal{T}^*\Phi\omega$ with respect to ω As shown by Correa and Seeger (1985), when for a fixed ω, there exists a unique pair of strategies (μ_ω, ν_ω) such that $\mathcal{T}^*\Phi\omega = \mathcal{T}_{\mu_\omega,\nu_\omega}\Phi\omega = \mathcal{T}_{\mu_\omega}\Phi\omega = \tilde{\mathcal{T}}_{\nu_\omega}\Phi\omega$, the gradient of $\mathcal{T}^*\Phi$ on ω, is simply the gradient of the linearized operator $\mathcal{T}_{\mu_\omega,\nu_\omega}\Phi = r + \gamma P_{\mu_\omega,\nu_\omega}\Phi$. From now on, we will note $P_{\mu_\omega,\nu_\omega} = P_\omega$ and $\mathcal{T}_{\mu_\omega,\nu_\omega} = \mathcal{T}_\omega$.

When there exist U_ω and V_ω such that, for all μ in U_ω and for all ν in V_ω, $\mathcal{T}^*\Phi\omega = \mathcal{T}_{\mu,\nu}\Phi\omega = \mathcal{T}_\mu\Phi\omega = \tilde{\mathcal{T}}_\nu\Phi\omega$, then the directional derivative of $\mathcal{T}^*\Phi\omega$ (written $\partial_d\mathcal{T}^*\Phi\omega$) is $\min_{\nu\in V_\omega}\max_{\mu\in U_\omega}\partial_d\mathcal{T}_{\mu,\nu}\Phi$ (Correa and Seeger, 1985). In practice, when the minmax is not unique, a pair of strategies (μ_ω, ν_ω) is chosen in $U_\omega \times V_\omega$ and $\mathcal{T}^*\Phi$ is linearized with $\mathcal{T}_\omega\Phi$ to compute the gradient: $\nabla\mathcal{T}^*\Phi\omega = \nabla\mathcal{T}_\omega\Phi\omega = \gamma P_\omega\Phi$.

For example, in the Pollatschek and Avi-Itzhak's algorithm ($\Phi = \mathcal{I}$), a couple (μ_ω, ν_ω) is chosen such that $\mathcal{T}^*\Phi\omega = \mathcal{T}_\omega\Phi\omega = \mathcal{T}_{\mu_\omega}\Phi\omega = \tilde{\mathcal{T}}_{\nu_\omega}\Phi\omega$ which is the pair of strategies for which the chosen gradient is $\gamma P_\omega\Phi$. In LSPI for games, the gradient is not the gradient of the linearized operator $\mathcal{T}_\omega$. Rather, they choose $\mathcal{T}_{\mu,\nu}$ such that $\mathcal{T}^*\Phi\omega = \mathcal{T}_{\mu,\nu}\Phi\omega = \mathcal{T}_\mu\Phi\omega$ and with ν a deterministic strategy. Thus, the LSPI algorithm as it has been described by Lagoudakis and Parr (2002) does not follow the gradient $\gamma P_\omega\Phi$. From now, we will call the Newton-LSPI algorithm the one with the Newton's gradient $\gamma P_\omega\Phi$. The reader should notice that this difference only appears when it comes to MGs and that Newton-LSPI is exactly LSPI when applied to MDPs since ν is not considered in MDPs. This also proves that LSPI for MGs is almost the (Pollatschek and Avi-Itzhak's algorithm when $\Phi = \mathcal{I}$ as it almost perform the same greedy step. Interestingly, the example of Van Der Wal also applies and thus LSPI is not guaranteed to converge in MGs whether the function approximation is stable or not.

2.1 Newton's Method on the POBR

As mentioned in the previous section, the Newton's method on the POBR reduces to the LSPI algorithm but with a slightly different choice of the gradient. Here, we will consider the POBR with the linearized gradient ($\nabla\mathcal{T}^*\Phi\omega = \nabla\mathcal{T}_\omega\Phi\omega = \gamma P_\omega\Phi$) described previously. Let's write $A_\omega = \Phi^\top\Delta_\rho(I - \gamma P_{\omega_k})\Phi$ and $b = \Phi^\top\Delta_\rho r$

The POBR gradient is (for details see appendix 8):

$$\nabla J_{POBR}(\omega) = A_\omega^\top(\Phi^\top\Delta_\rho\Phi)^{-1}(A_\omega\omega - b). \tag{6.7}$$

The Hessian matrix is (for details see appendix 8):

$$HJ_{POBR}(\omega) = A_\omega^\top(\Phi^\top\Delta_\rho\Phi)^{-1}A_\omega. \tag{6.8}$$

If the matrix A_ω is invertible and if ρ is a non-null measure on $S \times A^2$, the Newton's method direction is:

$$[HJ_{POBR}(\omega)]^{-1}\nabla J_{POBR}(\omega) = (A_\omega)^{-1}(A_\omega\omega - b). \tag{6.9}$$

The update of ω_k of the Newton's method is thus:

$$\omega_{k+1} = \omega_k - (A_{\omega_k})^{-1}(A_{\omega_k}\omega_k - b) = (A_{\omega_k})^{-1}b,$$

which is the update of the Newton-LSPI algorithm (Lagoudakis and Parr, 2003). The Newton-LSPI algorithm optimizes $J_{POBR}(\omega)$ as a cost function, which might never be zero (see Remark 7.1 below). Moreover, to our knowledge, controlling the POBR does not allow controlling the quality of the final policy. The main benefit of the Newton-LSPI algorithm is that the matrix A_{ω_k} is easy to estimate from batch data whether the state space is continuous or not. This makes Newton-LSPI a very practical algorithm Lagoudakis and Parr (2002).

It is well known that, for a fixed policy, the minimum of the projected Bellman residual $\frac{1}{2}||\Phi^\top(\Phi^\top\Delta_\rho\Phi)^{-1}\Phi^\top\Delta_\rho(\Phi\omega - \mathcal{T}_\pi\Phi\omega)||^2_{2,\rho}$ is equal to 0 Koller and Parr (2000) for all but finitely many number of γ. Nonetheless, we can show that it is not the case for the POBR. In other words, the minimum of $\frac{1}{2}||\Phi^\top(\Phi^\top\Delta_\rho\Phi)^{-1}\Phi^\top\Delta_\rho(\Phi\omega - \mathcal{T}^*\Phi\omega)||^2_{2,\rho}$ might be positive for a continuous interval of γ (see appendix 7.1).

2.2 Newton's Method on the OBR

Here we will note $C_\omega = \Phi^\top(I - \gamma P_\omega)^\top\Delta_\rho(I - \gamma P_\omega)\Phi$ and $e_\omega = \Phi^\top(I - \gamma P_\omega)^\top\Delta_\rho r$. Applying the Newton's method to the residual $\mathcal{J}_{OBR}(\omega)$ gives the following algorithm:

$$\omega_0 = 0, \tag{6.10}$$
$$\omega_{k+1} = \omega_k + (H\mathcal{J}_{OBR}(\omega_k))^{-1}\nabla\mathcal{J}_{OBR}(\omega_k). \tag{6.11}$$

The gradient is (for details see appendix 8):

$$\nabla\mathcal{J}_{OBR}(\omega) = C_\omega\omega - e_\omega.$$

The Hessian matrix is (for details see appendix 8):

$$H\mathcal{J}_{OBR}(\omega) = C_\omega.$$

The update of ω_k of the Newton's method is:

$$\begin{aligned}\omega_{k+1} &= \omega_k - (C_{\omega_k})^{-1}(C_\omega\omega_k - e_{\omega_k}),\\ &= (C_{\omega_k})^{-1}e_{\omega_k}.\end{aligned}$$

Lagoudakis and Parr (2003) propose this algorithm as an alternative method to learn an optimal policy in MDPs and call it Bellman Residual Minimization Policy Iteration (BRMPI). Extension to games is easy as shown previously.

2.3 Comparison of BRMPI and Newton-LSPI

In MGs, the two algorithms previously described derive from the Pollatschek and Avi-Itzhak algorithm and are thus not guaranteed to converge (Van Der Wal, 1978). They both follow the Newton's direction of either $\mathcal{J}_{OBR}$ or $\mathcal{J}_{POBR}$. On the one hand, the cost function $\mathcal{J}_{OBR}(\omega)$ controls the norm $||Q^* - Q_\mu||_2$ where μ is the maximizer's greedy strategy with respect to $\Phi\omega$ and Q_μ the fixed point of $\mathcal{T}_\mu$ (see Piot et al. (2014a) in the case of an MDP). In consequence, minimizing the OBR gives a nearly optimal strategy. However, there are no easy ways to estimate the matrix C_ω and the vector e_ω from data, especially when the state space is continuous. One possible way to obtain such estimates is to use embeddings in RKHS (Grunewalder et al., 2012).

Theorem 6.1. *Let Q be a value function and let μ be a greedy strategy on Q (i.e. $\mathcal{T}^*Q = \mathcal{T}_\mu Q$). Then we have:*

$$||Q^* - Q_\mu||_{2,\rho} \leq \frac{2}{1-\gamma}\left(\frac{C_2(\rho,\mu,\nu) + C_2(\rho,\mu^*,\tilde{\nu})}{2}\right)^{\frac{1}{2}} ||\mathcal{T}^*Q - Q||_{2,\rho}$$

with μ, ν, μ^* and $\tilde{\nu}$ strategies such that $\mathcal{T}^*Q^* = \mathcal{T}_{\mu^*}Q^*$, $\mathcal{T}_{\mu^*,\tilde{\nu}}Q = \mathcal{T}_{\mu^*}Q$, $\mathcal{T}_{\mu,\bar{\nu}}Q_\mu = \mathcal{T}_\mu Q_\mu$ and $\mathcal{T}_{\mu,\nu}Q = \mathcal{T}_\mu Q$. And with $C_2(\rho,\mu,\nu) = ||\frac{\partial \rho^\top (1-\gamma)(I-\gamma P_{\mu,\nu})^{-1}}{\partial \rho^\top}||_{2,\rho}$ $\mathcal{L}_\in$-norm of the Radon-Nikodym derivative of the Kernel $(1-\gamma)(I-\gamma P_{\mu,\nu})^{-1}$.

Proof. Proof in appendix 7. □

On the other hand, applying the Newton's method to $\mathcal{J}_{POBR}(\omega)$ is easy. Estimating A_ω and b_ω is computationally cheap. However, minimizing $\mathcal{J}_{POBR}(\omega)$ does not guarantee to find a good strategy.

Concerning the stability, Newton-LSPI is clearly the less stable one. The Hessian matrix $H\mathcal{J}_{POBR}(\omega)$ might not even be invertible since A_ω is not invertible for a finite number of values of γ (Koller and Parr, 2000). For BRMPI however the Hessian $H\mathcal{J}_{OBR}(\omega)$ is always invertible and the cosine between the Newton's update and the gradient is bounded away from zero.

Theorem 6.2. *For an MG with finite state space, there exists $\delta > 0$ such that:*

$$-\frac{\mathcal{J}_{OBR}(\omega)^\top (H\mathcal{J}_{OBR}(\omega))^{-1}\mathcal{J}_{OBR}(\omega)}{||\mathcal{J}_{OBR}(\omega)||_2 ||(H\mathcal{J}_{OBR}(\omega))^{-1}\mathcal{J}_{OBR}(\omega)||_2} \leq -\delta.$$

Proof. Proof in appendix 7 □

This means that, when on a point ω where the derivative is well defined, the Newton's update is guaranteed to decrease the OBR in a neighborhood of ω.

3 Batch Learning in Games

In the batch scenario, the dynamics and the reward function are known only through logs of interactions between the two players and the environment. These logs are a

collection of tuple $\{(s_i, a_i^1, a_i^2, r_i, s_i')\}_{i\in\{1,...,N\}}$ where (s_i, a_i^1, a_i^2) is drawn from a distribution ρ where $r_i = r(s_i, a_i^1, a_i^2)$ and where $s_i' \sim p(.|s_i, a_i^1, a_i^2)$. Thus, at each iteration of Newton-LSPI or of BRMPI, the objective will be to estimate one matrix and one vector.

3.1 Newton-LSPI with Batch Data

Newton-LSPI with batch data proceeds as follows. The goal is to estimate the matrix A_ω and the vector b (the estimates will be noted $\hat{A}_\omega$ and $\hat{b}$). The procedure is described in Algorithm 24 where μ_ω and ν_ω are the minmax strategies with respect to the approximate Q-function $\Phi\omega$.

The update of $\hat{A}_\omega$ is convenient since it allows computing $(\hat{A}_\omega)^{-1}$ very efficiently (Lagoudakis and Parr, 2003) with the Sherman–Morrison formula. Then doing Newton-LSPI with batch data corresponds simply in updating our parameter ω as follows:

$$\omega_{k+1} = (\hat{A}_{\omega_k})^{-1}\hat{b}. \tag{6.12}$$

Algorithm 24 LSPI-Matrix update

Input: $D_N = ((x_j, a_j^1, a_j^2), r_j, x_j')_{j=1,...,N}$ some samples, Φ a d dimensional feature space and ω.
for $i \in \{1, ..., N\}$ **do**
$\hat{A}_\omega += \phi(s_i, a_i^1, a_i^2)[\phi(s_i, a_i^1, a_i^2) - \gamma \sum_{b^1\in A^1(s_i')} \sum_{b^2\in A^2(s_i')} \mu_\omega(s_i', b^1)\nu_\omega(s_i', b^2)\phi(s_i', b^1, b^2)]^\top$
$\hat{b} \ += \phi(s_i, a_i^1, a_i^2)r_i$
end for
output: $\hat{A}_\omega$, $\hat{b}$

3.2 BRMPI with Batch Data

A natural extension of BRMPI to batch data when the state space is finite is to find estimates of C_ω and e_ω. To do so we first estimate $(I - \gamma P_\omega)$ (noted $\hat{B}_\omega$) and r (noted $\hat{r}$). The procedure is described in Algorithm-25.

Then, estimates of C_ω and e_ω would be respectively $\hat{C}_\omega = \Phi^\top \hat{B}_\omega^\top \hat{B}_\omega \Phi$ and $\hat{e}_\omega = \Phi^\top \hat{B}_\omega^\top \hat{r}$. Those estimates are clearly biased estimates of C_ω and e_ω. Thus, the Newton's update is:

$$\omega_{k+1} = (\hat{C}_{\omega_k})^{-1}\hat{e}_{\omega_k}. \tag{6.13}$$

4 Quasi-Newton's Method on the OBR and on the POBR

In this section, a quasi-Newton's method is applied to the norm of the OBR and of the PORBR. This corresponds to the introduction of a learning rate in Newton-LSPI and

Algorithm 25 BRMPI-Matrix update

Input: $D_N = ((x_j, a_j^1, a_j^2), r_j, x'_j)_{j=1,...,N}$ some samples, Φ a d dimensional feature space and ω.
for $i \in \{1, ..., N\}$ **do**
  $\hat{B}_\omega(s_i, a_i^1, a_i^2, s_i, a_i^1, a_i^2) \ += 1$
  for $b^1 \in A^1(s'_i)$ and $b^2 \in A^2(s'_i)$ **do**
    $\hat{B}_\omega(s_i, a_i^1, a_i^2, s'_i, b^1, b^2) += -\gamma\mu_\omega(s'_i, b^1)\nu_\omega(s'_i, b^2)$
  end for
  $\hat{r}(s_i, a_i^1, a_i^2) \leftarrow \hat{r}(s_i, a_i^1, a_i^2) + r_i$
end for

BRMPI and their batch versions described in the herein-before section. In (Filar and Tolwinski, 1991) the learning rate is chosen to verify Amijo's rule for the exact case (Amijo's rule is satisfied when the first Wolf condition is satisfied eq. (6.14)). We choose to use a learning rate according to the Wolf conditions for the case with linear function approximation. There exist theorems to guarantee convergence to a local minimum when the gradient is a Lipschitz function which is not the case here. However, there are empirical evidences that using inexact line search improves convergence to a local minimum of the objective function (Lewis and Overton, 2009).

In both Newton-LSPI and BRMPI, when the dynamics is known and in the batch scenario, the update takes the following shape:

$$\omega_{k+1} = (\Xi_{\omega_k})^{-1}\psi_{\omega_k}$$

and the objective is to minimize a $\mathcal{L}_2$ norm of the following shape (the corresponding notations is given in table 6.2):

$$f(\omega) = \|\Psi_\omega\omega - \upsilon\|_{\rho,2}.$$

Let's note the Newton's update:

$$g_{\omega_k} = (\Xi_{\omega_k})^{-1}\psi_{\omega_k} - \omega_k,$$

The quasi-Newton method will instead perform the following update:

$$\omega_{k+1} = (1 - \alpha_k)\omega_k + \alpha_k(\Xi_{\omega_k})^{-1}\psi_{\omega_k},$$

where the learning rate α is chosen to satisfy the Wolf conditions ($0 < c_1 < c_2 < 1$):

$$f(\omega + \alpha g_\omega) \leq f(\omega) + c_1\alpha g_\omega^\top.\nabla f(\omega) \tag{6.14}$$
$$f(\omega + \alpha g_\omega) \geq f(\omega) + c_2\alpha g_\omega^\top.\nabla f(\omega) \tag{6.15}$$

Constants c_1 is typically chosen around 10^{-4} and c_2 is usually chosen close to 0.9. Condition (6.14) ensures that the learning rate is not too large whereas condition (6.15) guarantees a large enough learning rate.

Table 6.2 – Parameters for LSPI, BRMPI and their Batch version

	LSPI	BRMPI	Batch LSPI	Batch BRMPI
Ξ_{ω_k}	A_{ω_k}	$C_\omega = \Phi^\top B_\omega^\top B_\omega \Phi$	$\hat{A}_{\omega_k}$	$\hat{C}_\omega = \Phi^\top \hat{B}_\omega^\top \hat{B}_\omega \Phi$
ψ_{ω_k}	b	$e_\omega = \Phi^\top B_\omega^\top r$	$\hat{b}$	$\hat{e}_\omega = \Phi^\top \hat{B}_\omega^\top \hat{r}$
Ψ_ω	$\Phi(\Phi^\top \Delta_\rho \Phi)^{-1} A_\omega$	$B_\omega \Phi$	$\hat{A}_\omega$	$\hat{B}_\omega \Phi$
υ	$\Phi(\Phi^\top \Delta_\rho \Phi)^{-1} b$	r	$\hat{b}$	$\hat{r}$
ρ	ρ	ρ	uniform	uniform

5 Experiments

To empirically illustrate the application of quasi-Newton methods to both MDPs and MGs, we ran experiments on Garnets (defined in Chapter 4). A Garnet is an abstract class of MDPs. It is described by a triplet (N_S, N_A, N_B). The parameter N_B, the branching factor, denotes the number of reachable states from any state-action pair. In one of our experiment, we enforced some regularity in the dynamics. Usually (as described in Chapter 4), for all (s, a) we generate the following transition distribution $p(.|s, a)$ according to the following procedure: first one draws uniformly $N_B - 1$ points over $[0, 1]$. Let's note those numbers $(p_i)_{1 \leq i \leq N_B - 1}$ (in increasing order) and $p_0 = 0$ and $p_{N_B} = 1$. Then we draw a subset of size N_B of $\{1, ..., N_S\}$ written $\{s'_1, ..., s'_{N_B}\}$. The second type of Garnet, $\{s'_1, ..., s'_{N_B}\}$ is drawn in a subset of $\{1, ..., N_S\}$ centered around s (meaning we will draw states s' such that $|s - s'| \leq \zeta$ in the experiment $\zeta = N_B$). Finally, we define the kernel as follows, $\forall i \in \{1, ..., N_B\},\ p(s'_i|s, a) = p_i - p_{i-1}$. The reward function will depend on the experiment.

For simultaneous two-player zero-sum MGs the N_A parameter will describe the number of actions both players will be allowed to play at each state in the MG. The kernel $p(.|a^1, a^2, s)$ is generated according to the same procedure as for MDPs.

Results always show the performance of the strategy with respect to the iteration number. The performance of a strategy μ is quantified as the following ratio: $\frac{||Q^* - Q_\mu||_2}{||Q^*||_2}$. This is the normalized norm of the difference between the optimal Q-function of the MG and the Q-function of strategy μ considering the opponent plays his best response. Results are averaged over experiments and the envelope is proportional to the standard deviation.

The main purpose of the two following subsections is to emphasize the effects of quasi-Newton methods both on the stability and on the performance of the algorithms. In those experiments, we can notice the importance of tuning constants c_1 and c_2 from equations (6.14) and (6.15). Here, $c_1 = 10^{-4}$ and $c_2 = 0.9$. We performed the line search as follows: if condition (6.14) was not checked we decreased the learning rate geometrically $\alpha_k \leftarrow \eta \times \alpha_k$ and, if condition (6.15) was not checked, we increased it geometrically $\alpha_k \leftarrow \frac{1}{\eta} \times \alpha_k$ ($\eta = 0.9$). This was done until both conditions were checked.

Algorithm 26 quasi-Newton LSPI

Input: $D_N = ((x_j, a_j^1, a_j^2), r_j, x'_j)_{j=1,...,N}$ some samples, Φ a d dimensional feature space and $\omega_0 = 0$. Constants $c_1(= 10^{-4})$,$c_2(= 0.9)$, $\eta(= 0.9)$, $\alpha_0 = 1$, $\alpha_{min}(= 10^{-10})$ and $\alpha_{min}(= 1.0)$.
let's suppose there is an algorithm f such that $\hat{A}_\omega, \hat{b} = f(D_N, \omega, \Phi)$ (the procedure is described in Section 3.1)
for k=1,2,...,K **do**
  $\hat{A}_{\omega_k}, \hat{b} = f(D_N, \omega_k, \Phi)$
  $g_{\omega_k} = (\hat{A}_{\omega_k})^{-1}\hat{b} - \omega_k$
  $p_{\omega_k} = \hat{A}_{\omega_k}^\top(\hat{A}_{\omega_k}\omega_k - \hat{b})$
  $\alpha_k = \alpha_{k-1}$
  $l_{\omega_k} = ||\hat{A}_{\omega_k}\omega_k - \hat{b}||_2^2$
  $\hat{A}_{\omega_k+\alpha_k g_{\omega_k}}, \hat{b} = f(D_N, \omega_k + \alpha_k g_{\omega_k}, \Phi)$
  $l = ||\hat{A}_{\omega_k+\alpha_k g_{\omega_k}}(\omega_k + \alpha_k g_{\omega_k}) - \hat{b}||_2^2$
  if $l > l_{\omega_k} + c_1\alpha_k g_{\omega_k}^\top p_{\omega_k}$ **then**
    while $l > l_{\omega_k} + c_1\alpha_k g_{\omega_k}^\top p_{\omega_k}$ **and** $\alpha_k \geq \alpha_{min}$ **do**
      $\alpha_k \leftarrow \alpha_k \times \eta$
      $\hat{A}_{\omega_k+\alpha_k g_{\omega_k}}, \hat{b} = f(D_N, \omega_k + \alpha_k g_{\omega_k}, \Phi)$
      $l = ||\hat{A}_{\omega_k+\alpha_k g_{\omega_k}}(\omega_k + \alpha_k g_{\omega_k}) - \hat{b}||_2^2$
    end while
  end if
  if $l < l_{\omega_k} + c_1\alpha_k g_{\omega_k}^\top p_{\omega_k}$ **then**
    while $l < l_{\omega_k} + c_1\alpha_k g_{\omega_k}^\top p_{\omega_k}$ **and** $\alpha_k \leq \alpha_{max}$ **do**
      $\alpha_k \leftarrow \frac{\alpha_k}{\eta}$
      $\hat{A}_{\omega_k+\alpha_k g_{\omega_k}}, \hat{b} = f(D_N, \omega_k + \alpha_k g_{\omega_k}, \Phi)$
      $l = ||\hat{A}_{\omega_k+\alpha_k g_{\omega_k}}(\omega_k + \alpha_k g_{\omega_k}) - \hat{b}||_2^2$
    end while
  end if
  $\omega_{k+1} = \omega_k + \alpha_k g_{\omega_k}$
end for
output: $\Phi\omega_{K+1}$

5.1 Experiments on Markov Decision Processes

The PI algorithm for MDPs is known to converge because it builds a sequence of increasing values while searching in an underlying finite policy space Puterman (1994). This does not stand anymore when it comes to function approximation, since the argument of increasing values is not verified. As a result, the LSPI algorithm is not proven to converge in general for MDPs.

We ran experiments on Garnets to compare LSPI and BRMPI (Newton's methods) with their quasi-Newton counterparts (Softened LSPI described in Algorithm-26). Here, the reward function depends on states and actions, and only a ratio of the rewards

were non-zero (this ratio is called the sparsity). Experiments of figure 6.1 was done on type 2 Garnets (with a regular dynamic). The feature space Φ has $d = 0.2 \times N_S \times N_A$ features. Those d features are vectors randomly generated according to a Gaussian law.

First, one can notice that quasi-Newton version of LSPI and respectively BRMPI did not under-perform LSPI and respectively BRMPI. Figure 6.1 illustrates that quasi-Newton methods do not under perform their respective counterparts. We noticed that the BRMPI algorithm often oscillates from one iteration to another. The use of quasi-Newton method usually eliminates that instability.

In Figure 6.1 the Garnet is drawn with $N_B = 10$. Usually, MDPs with high branching factors are easier to optimize since a high branching factor has a tendency to smooth the optimal value function. Figure 6.1 shows the effect of the learning rate on the learning curve. Especially in the case of Newton-LSPI, introducing a learning rate improves both the stability and the performance of the algorithm. Also, we can notice that the BRMPI algorithm oscillates on MDPs.

5.2 Experiments on Markov Games

We ran experiments on MG Garnets to compare Newton's method and quasi-Newton method. In the hereafter experiment, the reward function only depends on the state. Only a ratio of rewards are non-zero and are drawn according to a normal law. Again, we compared Newton-LSPI and BRMPI with their quasi-Newton counterparts. The feature space is also randomly generated according to a Gaussian law. But, we needed more features compared to the MDP case to learn a strategy. In the following experiment, we used $d = 0.8 \times N_A \times N_A \times N_S$ features.

Compared to experiments on MDPs, one can notice that the variance of the performance is higher even if we use more features, more samples on MGs with less actions. We do not fully understand why it is significantly harder to learn a good strategy in an MG compared to an MDP. The reason might be that simultaneous games are simply more complex to optimize or that comparing with the value of a best response is too conservative. Anyway, the use of quasi-Newton method appears to be useful even with this conservative performance criterion.

Here again, one can notice that quasi-Newton methods did not under-perform their counterparts. Although, we could not notice huge gaps in the performance (Figure 6.2 and 6.3) as in experiments on MDPs, quasi-Newton methods did reduce the unstable behavior of the strategy (Figure 6.2).

6 Conclusion

To sum up this chapter, we first pointed out the fact that the proof of convergence of Filar and Tolwinski's algorithm is based on assumptions that do not hold. Second, we found out that, as a consequence of the instability of the Pollatschek and Avi-Itzhak algorithm, LSPI for games does not converge whether the linear approximation

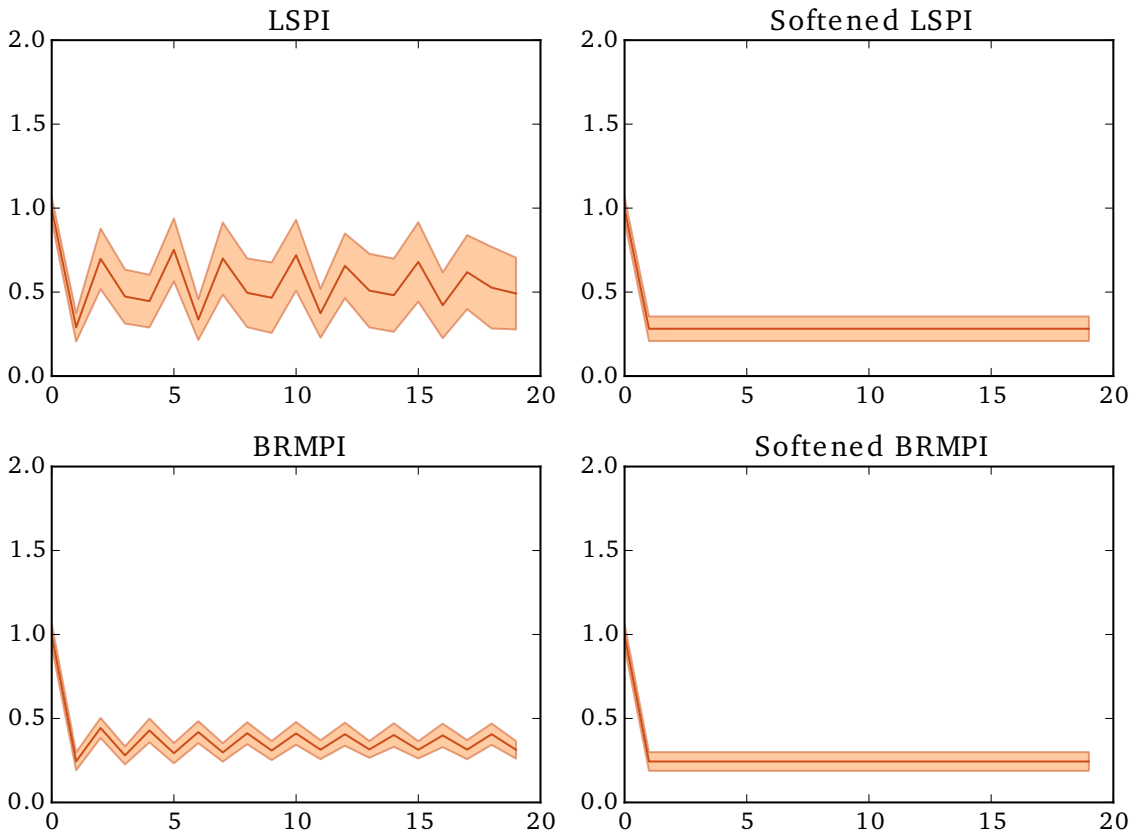


Figure 6.1 – Performance (y-axis) of the strategy at step k (x-axis) for LSPI, BRMPI and the corresponding quasi-Newton method. Results are averaged over 50 Garnets $N_S = 100$, $N_A = 8$, $N_B = 10$. All Garnets have a sparsity of 0.5 and $\gamma = 0.99$. The number of batch samples used is $0.5 \times N_A \times N_S$.

is stable or not. Third, we showed that LSPI and BRMPI can be regarded as Newton's methods. Fourth, this naturally led to the use of quasi-Newton methods instead of Newton's method as they result in more stable solutions. And, finally, these slight modifications on algorithms for games dramatically improve the stability of one of the most popular model-free algorithms to solve MDPs on synthetic problems, namely LSPI. We make novel connections between approximate policy iteration schemes on both MDPs and MGs and optimization methods. It describes three algorithms as Newton's method on different Bellman residuals. This unified picture of three popular algorithms naturally led to the use of quasi-Newton methods which is believed, in the optimization community, to be steady and to perform better in practice (Nocedal and Wright, 2006).

Yet, this chapter rises three open questions: first, since LSPI and BRMPI with batch data are minimizing some empirical residual, a natural question is whether those estimators are good estimators of the residual or not. Few works exist on direct min-

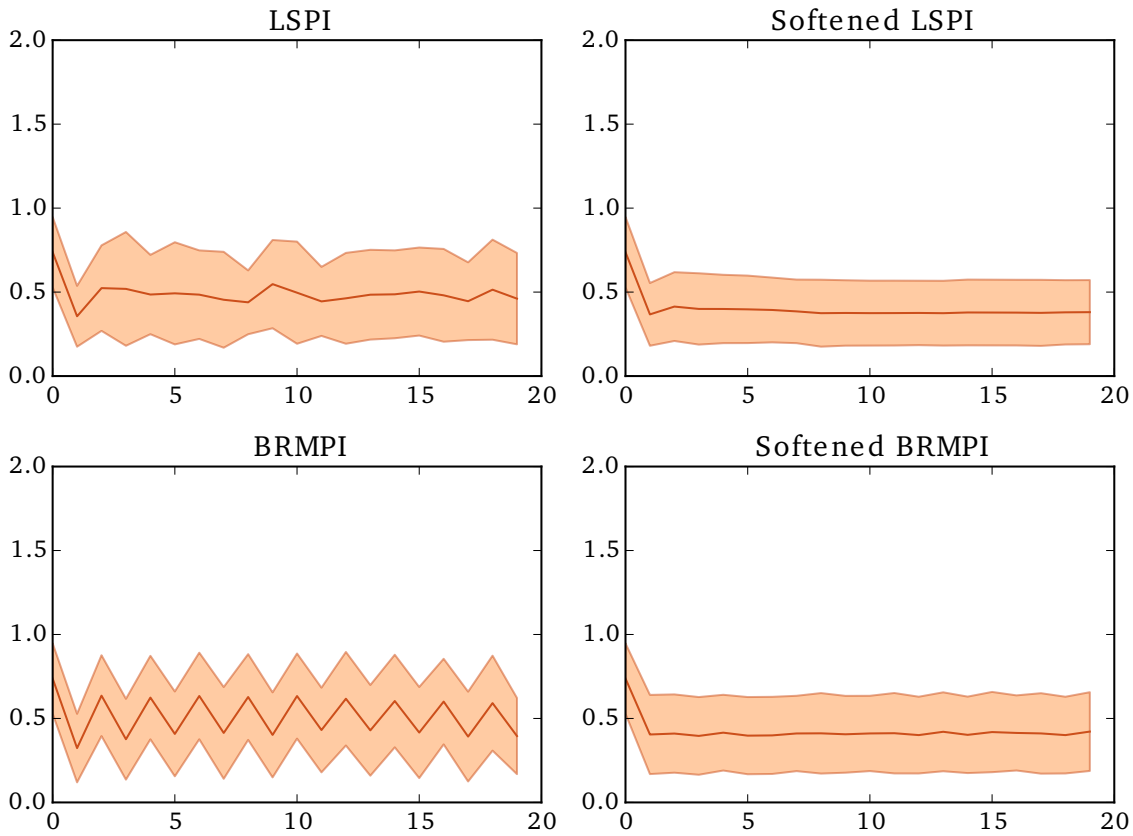


Figure 6.2 – Performance (y-axis) of the strategy at step k (x-axis) for Newton-LSPI, BRMPI and the corresponding quasi-Newton method. Results are averaged over 50 Garnets $N_S = 50$, $N_A = 2$, $N_B = 1$. All Garnets have a sparsity of 0.3 and $\gamma = 0.9$. The number of batch samples used is $1.0 \times N_A \times N_A \times N_S$.

imization of the OBR (Baird et al., 1995, Piot et al., 2014a). Piot et al. proved that minimizing a specific estimate of the Optimal Bellman residual on MDPs is consistent in the Vapnik sense. A second question is whether it makes sense theoretically to minimize the POBR or not. Little is known on the quality of the solution given by the minimum of the POBR and research should be conducted in that sense. Finally, since the goal is to minimize some Bellman residual, a third question is whether there exist smarter methods than quasi-Newton ones to minimize the considered residual. There exists a trove of Newton-like methods that are known to perform well on non-smooth functions. One interesting perspective would be to study how the BFGS method (Nocedal and Wright, 2006, Lewis and Overton, 2013) would perform, as it does not require to compute and invert the Hessian matrix at each iteration.

But in the end, minimizing the Bellman residual can only be used in zero-sum games as there exists no known Bellman operator for general sum MGs and as no value function based method can find a Nash equilibrium Zinkevich et al. (2006).

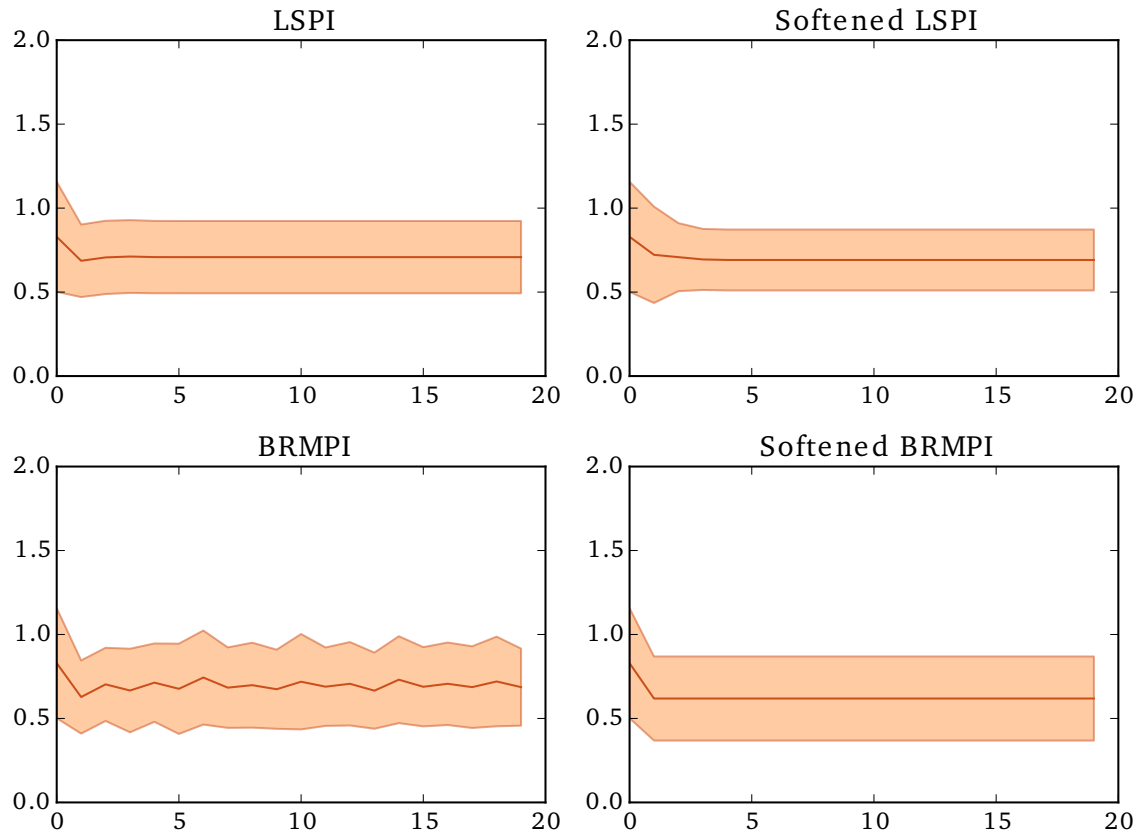


Figure 6.3 – Performance (y-axis) of the strategy at step k (x-axis) for Newton-LSPI, BRMPI and the corresponding quasi-Newton method. Results are averaged over 50 Garnets $N_S = 50$, $N_A = 2$, $N_B = 4$. All Garnets have a sparsity of 0.3 and $\gamma = 0.9$. The number of batch samples used is $2.0 \times N_A \times N_A \times N_S$.

7 Appendix

7.1 Remark

The minimum of POBR $J_{POBR}(\omega)$ might not be 0. Indeed, let us consider the following simple example on an MDP with only two actions:

$$r = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \; P_{a_1} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \end{pmatrix}, \; P_{a_2} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 1 & 0 \end{pmatrix}, \Phi = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

One can notice that, if $\omega \geq 0$, then a_1 is greedy with respect to $\Phi\omega$. If $\omega \leq 0$, then a_2 is greedy with respect to $\Phi\omega$.

$$(\Phi^\top \Phi)^{-1} \Phi^\top (T_{a_1} \Phi\omega - \Phi\omega) = \frac{1}{5}(2 + (5.5\gamma - 5)\omega)$$

$$(\Phi^\top \Phi)^{-1} \Phi^\top (T_{a_2} \Phi\omega - \Phi\omega) = \frac{1}{5}(2 + (3.5\gamma - 5)\omega)$$

Then, for $\gamma \in [\frac{10}{11}, 1]$, the value of $(\Phi^\top \Phi)^{-1}\Phi^\top(T^*\Phi\omega - \Phi\omega)$ is a piecewise linear function. It is decreasing when $\omega \leq 0$ and increasing when $\omega \geq 0$. For $\omega = 0$ it is equal to $\frac{2}{5}$. Thus $J_{POBR}(\omega)$ is positive ($J_{POBR}(\omega) \geq \frac{4}{25}$).

7.2 Proof of Theorem 6.1

Let us recall theorem 6.1

Let Q be a value function and let μ be a greedy strategy on v (meaning $\mathcal{T}^*v = \mathcal{T}_\mu v$). Then we have:

$$||Q^* - Q_\mu||_2 \leq \frac{C}{1-\gamma}||Q - \mathcal{T}^*Q||_2$$

Proof.

$$\begin{aligned} Q^* - Q_\mu &= \mathcal{T}^*Q^* - \mathcal{T}^*Q + \mathcal{T}_\mu Q - \mathcal{T}_\mu Q_\mu && (6.16)\\ &\leq \mathcal{T}_{\mu^*}Q^* - \mathcal{T}_{\mu^*}Q + \mathcal{T}_{\mu,\bar{\nu}}Q - \mathcal{T}_{\mu,\bar{\nu}}Q_\mu && (6.17)\\ &\leq \mathcal{T}_{\mu^*,\tilde{\nu}}Q^* - \mathcal{T}_{\mu^*,\tilde{\nu}}Q + \mathcal{T}_{\mu,\bar{\nu}}Q - \mathcal{T}_{\mu,\bar{\nu}}Q_\mu && (6.18)\\ & && (6.19)\end{aligned}$$

with $\mathcal{T}^*Q^* = \mathcal{T}_{\mu^*}Q^*$, $\mathcal{T}_{\mu^*,\tilde{\nu}}Q = \mathcal{T}_{\mu^*}Q$ and $\mathcal{T}_{\mu,\bar{\nu}}Q_\mu = \mathcal{T}_\mu Q_\mu$. We will note μ the strategy such that $\mathcal{T}_{\mu,\nu}Q = \mathcal{T}_\mu Q$

$$(I - \gamma P_{\mu^*,\tilde{\nu}})(Q^* - Q_\mu) \leq (\gamma P_{\mu^*,\tilde{\nu}} - \gamma P_{\mu,\bar{\nu}})(Q - Q_\mu) \tag{6.20}$$

Now let's bound $Q - Q_\mu$

$$\begin{aligned} Q - Q_\mu = Q - Q_{\mu,\bar{\nu}} &= Q - (I - \gamma P_{\mu,\bar{\nu}})^{-1}r && (6.21)\\ &= (I - \gamma P_{\mu,\bar{\nu}})^{-1}(Q - \gamma P_{\mu,\bar{\nu}}Q - r) && (6.22)\\ &= (I - \gamma P_{\mu,\bar{\nu}})^{-1}(Q - \mathcal{T}_{\mu,\bar{\nu}}Q) && (6.23)\\ &\leq (I - \gamma P_{\mu,\bar{\nu}})^{-1}(Q - \mathcal{T}_\mu Q) && (6.24)\\ &\leq (I - \gamma P_{\mu,\bar{\nu}})^{-1}(Q - \mathcal{T}^*Q) && (6.25)\end{aligned}$$

and:

$$\begin{aligned} Q_\mu - Q \leq Q_{\mu,\nu} - Q &\leq (I - \gamma P_{\mu,\nu})^{-1}(\mathcal{T}_{\mu,\nu}Q - Q) && (6.26)\\ &\leq (I - \gamma P_{\mu,\nu})^{-1}(\mathcal{T}^*Q - Q) && (6.27)\end{aligned}$$

Finally:

$$Q^* - Q_\mu \leq [(I - \gamma P_{\mu,\nu})^{-1} + (I - \gamma P_{\mu^*,\tilde{\nu}})^{-1}]|\mathcal{T}^*Q - Q|$$

with $C_2(\rho,\mu,\nu) = ||\frac{\partial\rho^\top(1-\gamma)(I-\gamma P_{\mu,\nu})^{-1}}{\partial\rho^\top}||_{2,\rho}$ the Radon-Nikodym derivative of the Kernel $(1-\gamma)(I-\gamma P_{\mu,\nu})^{-1}$.

Then we have:

$||Q^* - Q_\mu||_{2,\rho} \leq \frac{2}{1-\gamma}\left(\frac{C_2(\rho,\mu,\nu)+C_2(\rho,\mu^*,\tilde{\nu})}{2}\right)^{\frac{1}{2}}||\mathcal{T}^*Q - Q||_{2,\rho}$

□

7.3 Proof of Theorem 6.2

Let us recall theorem 6.2:

For a MG with a finite state space, there exists $\delta > 0$ such that:

$$-\frac{\mathcal{J}_{OBR}(\omega)^\top (H\mathcal{J}_{OBR}(\omega))^{-1}\mathcal{J}_{OBR}(\omega)}{||\mathcal{J}_{OBR}(\omega)||_2||(H\mathcal{J}_{OBR}(\omega))^{-1}\mathcal{J}_{OBR}(\omega)||_2} \leq -\delta$$

Proof. If $\gamma \in [0,1[$, then $H\mathcal{J}(\omega)$ is invertible and thus it is a definite positive symmetric matrix. Let's note $L_\omega^{\frac{1}{2}}$ the root square of $(H\mathcal{J}_{OBR}(\omega))^{-1}$ (then $(H\mathcal{J}_{OBR}(\omega))^{-1} = L_\omega^{\frac{1}{2}} L_\omega^{\frac{1}{2}} = L_\omega$). Then:

$$-\frac{\mathcal{J}_{OBR}(\omega)^\top (H\mathcal{J}_{OBR}(\omega))^{-1}\mathcal{J}_{OBR}(\omega)}{||\mathcal{J}_{OBR}(\omega)||_2||(H\mathcal{J}_{OBR}(\omega))^{-1}\mathcal{J}_{OBR}(\omega)||_2} = -\frac{||L_\omega^{\frac{1}{2}}\mathcal{J}_{OBR}(\omega)||_2^2}{||\mathcal{J}_{OBR}(\omega)||_2||L_\omega\mathcal{J}_{OBR}(\omega)||_2} \tag{6.28}$$

We have:

$$||\mathcal{J}_{OBR}(\omega)||_2 = ||L_\omega^{-\frac{1}{2}} L_\omega^{\frac{1}{2}}\mathcal{J}_{OBR}(\omega)||_2 \leq ||L_\omega^{-\frac{1}{2}}||_2||L_\omega^{\frac{1}{2}}\mathcal{J}_{OBR}(\omega)||_2 \tag{6.29}$$

And:

$$||L_\omega\mathcal{J}_{OBR}(\omega)||_2 = ||L_\omega^{\frac{1}{2}} L_\omega^{\frac{1}{2}}\mathcal{J}_{OBR}(\omega)||_2 \leq ||L_\omega^{\frac{1}{2}}||_2||L_\omega^{\frac{1}{2}}\mathcal{J}_{OBR}(\omega)||_2 \tag{6.30}$$

Then:

$$-\frac{||L_\omega^{\frac{1}{2}}\mathcal{J}_{OBR}(\omega)||_2^2}{||\mathcal{J}_{OBR}(\omega)||_2||L_\omega\mathcal{J}_{OBR}(\omega)||_2} \leq \frac{-1}{||L_\omega^{\frac{1}{2}}||_2||L_\omega^{-\frac{1}{2}}||_2}$$

Furthermore with $||.||_2 \leq c_1||.||_\infty$ (here $c_1 = \sqrt[2]{|S| \times |A|^2}$) and with noticing $1-\gamma \leq ||(I-\gamma P_\omega)||_\infty \leq 1+\gamma$:

$$||L_\omega^{-\frac{1}{2}}||_2 = ||\Delta_\rho^{\frac{1}{2}}(I-\gamma P_\omega)\Phi||_2 \leq ||\Delta_\rho^{\frac{1}{2}}||_2||(I-\gamma P_\omega)||_2||\Phi||_2 \leq c_1(1+\gamma)\sqrt[2]{\rho_{\max}\lambda_{\max}^\Phi}$$

Where $\lambda_{\max}^\Phi$ is the largest eigenvalue of $\Phi^\top\Phi$, and $\rho_{\max}$ is the maximum of ρ

$$||L_\omega^{\frac{1}{2}}||_2 = \sup_\omega \sqrt{\frac{\omega^\top(\Phi^\top(I-\gamma P_\omega)^\top\Delta_\rho(I-\gamma P_\omega)\Phi)^{-1}\omega}{\omega^\top\omega}} = \frac{1}{\inf_\omega \sqrt{\frac{\omega^\top\Phi^\top(I-\gamma P_\omega)^\top\Delta_\rho(I-\gamma P_\omega)\Phi\omega}{\omega^\top\omega}}}$$

But:

$$\begin{aligned}\inf_\omega \sqrt{\frac{\omega^\top\Phi^\top(I-\gamma P_\omega)^\top\Delta_\rho(I-\gamma P_\omega)\Phi\omega}{\omega^\top\omega}} &\geq \inf_X \sqrt{\frac{X^\top(I-\gamma P_\omega)^\top\Delta_\rho(I-\gamma P_\omega)X}{X^\top X}} \inf_\omega \sqrt{\frac{\omega^\top\Phi^\top\Phi\omega}{\omega^\top\omega}} \\ &\geq \frac{1}{||(\Delta_\rho^{\frac{1}{2}}(I-\gamma P_\omega))^{-1}||_2}\sqrt[2]{\lambda_{\min}^\Phi}\end{aligned}$$

WithFilar and Tolwinski (1991) we have that $||\Delta_\rho^{-\frac{1}{2}}(I-\gamma P_\omega)^{-1}||_2 \leq \frac{c_1}{(1-\gamma)\sqrt[2]{\rho_{\min}}}$

Then:

$$||L_\omega^{\frac{1}{2}}||_2 \leq \frac{c1}{(1-\gamma)\sqrt[2]{\rho_{\min}\lambda_{\min}^{\Phi}}}$$

Finally we have:

$$-\frac{\mathcal{J}_{OBR}(\omega)^\top (H\mathcal{J}_{OBR}(\omega))^{-1}\mathcal{J}_{OBR}(\omega)}{||\mathcal{J}_{OBR}(\omega)||_2||(H\mathcal{J}_{OBR}(\omega))^{-1}\mathcal{J}_{OBR}(\omega)||_2} \leq -\frac{1-\gamma}{c_1^2(1+\gamma)}\sqrt[2]{\frac{\rho_{\min}\lambda_{\min}^{\Phi}}{\rho_{\max}\lambda_{\max}^{\Phi}}}$$

□

8 Computation of the Gradient and of the Hessian

Let's remind the reader:

$$\mathcal{J}_{OBR}(\omega) = \frac{1}{2}||\Phi\omega - \mathcal{T}^*\Phi\omega||_{2,\rho}^2, \tag{6.31}$$

$$= \frac{1}{2}(\Phi\omega - \mathcal{T}^*\Phi\omega)^\top \Delta_\rho(\Phi\omega - \mathcal{T}^*\Phi\omega), \tag{6.32}$$

$$\mathcal{J}_{POBR}(\omega) = \frac{1}{2}||\Phi\omega - \Pi_{\rho,\Phi}\mathcal{T}^*\Phi\omega||_{2,\rho}^2, \tag{6.33}$$

$$= \frac{1}{2}||\Phi(\Phi^\top\Delta_\rho\Phi)^{-1}\Phi^\top\Delta_\rho(\Phi\omega - \mathcal{T}^*\Phi\omega)||_{2,\rho}^2, \tag{6.34}$$

$$= \frac{1}{2}\left[\Phi(\Phi^\top\Delta_\rho\Phi)^{-1}\Delta_\rho\Phi^\top\Delta_\rho(\Phi\omega - \mathcal{T}^*\Phi\omega)\right]^\top \Delta_\rho\Phi(\Phi^\top\Delta_\rho\Phi)^{-1}\Phi^\top\Delta_\rho(\Phi\omega - \mathcal{T}^*\Phi\omega) \tag{6.35}$$

$$\nabla\mathcal{J}_{POBR}(\omega) = \frac{\partial}{\partial\omega}\left[\Phi(\Phi^\top\Delta_\rho\Phi)^{-1}\Delta_\rho\Phi^\top\Delta_\rho(\Phi\omega - \mathcal{T}^*\Phi\omega)\right]^\top \Delta_\rho\Phi(\Phi^\top\Delta_\rho\Phi)^{-1}\Phi^\top\Delta_\rho(\Phi\omega - \mathcal{T}^*\Phi\omega) \tag{6.36}$$

$$= \Phi^\top(I-\gamma P_\omega)^\top\Delta_\rho\Phi(\Phi^\top\Delta_\rho\Phi)^{-1}\Phi^\top\Delta_\rho\Phi(\Phi^\top\Delta_\rho\Phi)^{-1}\Phi^\top\Delta_\rho(\Phi\omega - \mathcal{T}^*\Phi\omega) \tag{6.37}$$

$$= \Phi^\top(I-\gamma P_\omega)^\top\Delta_\rho\Phi(\Phi^\top\Delta_\rho\Phi)^{-1}\Phi^\top\Delta_\rho(\Phi\omega - \mathcal{T}^*\Phi\omega) \tag{6.38}$$

$$= \Phi^\top(I-\gamma P_\omega)^\top\Delta_\rho\Phi(\Phi^\top\Delta_\rho\Phi)^{-1}\Phi^\top\Delta_\rho(\Phi\omega - \mathcal{T}_\omega\Phi\omega) \tag{6.39}$$

$$= \Phi^\top(I-\gamma P_\omega)^\top\Delta_\rho\Phi(\Phi^\top\Delta_\rho\Phi)^{-1}\Phi^\top\Delta_\rho((I-\gamma P_\omega)\Phi\omega - r) \tag{6.40}$$

Then obviously:

$$H\mathcal{J}_{POBR}(\omega) = \frac{\partial}{\partial\omega^\top}\Phi^\top(I-\gamma P_\omega)^\top\Delta_\rho\Phi(\Phi^\top\Delta_\rho\Phi)^{-1}\Phi^\top\Delta_\rho(\Phi\omega - \mathcal{T}^*\Phi\omega) \tag{6.41}$$

$$= \Phi^\top(I-\gamma P_\omega)^\top\Delta_\rho\Phi(\Phi^\top\Delta_\rho\Phi)^{-1}\Phi^\top\Delta_\rho(I-\gamma P_\omega)\Phi \tag{6.42}$$

And for $\mathcal{J}_{OBR}$

$$\nabla\mathcal{J}_{OBR}(\omega) = \frac{\partial}{\partial\omega}\left[\Phi\omega - \mathcal{T}^*\Phi\omega\right]^\top\Delta_\rho(\Phi\omega - \mathcal{T}^*\Phi\omega) \tag{6.43}$$

$$= \Phi^\top(I-\gamma P_\omega)^\top\Delta_\rho(\Phi\omega - \mathcal{T}^*\Phi\omega) \tag{6.44}$$

$$= \Phi^\top(I-\gamma P_\omega)^\top\Delta_\rho((I-\gamma P_\omega)\Phi\omega - r) \tag{6.45}$$

Then obviously:

$$H\mathcal{J}_{OBR}(\omega) = \frac{\partial}{\partial \omega^\top}\Phi^\top(I-\gamma P_\omega)^\top\Delta_\rho(\Phi\omega - \mathcal{T}^*\Phi\omega) \tag{6.46}$$

$$= \Phi^\top(I-\gamma P_\omega)^\top\Delta_\rho(I-\gamma P_\omega)\Phi \tag{6.47}$$

Chapter 7

Bellman Residual Minimization in General-Sum Games

None of the approaches described in the last four chapters are meant to find a stationary Nash equilibrium in general-sum MGs. The very reason why all the previous approaches can't find a stationary Nash equilibrium is that they are all value function based methods. Approximate dynamic programming algorithms iterate over a value function or a state-action value function and the Bellman residual approach developed in the previous chapter also estimates a value function. In all these methods, the policy is implicitly defined by the value function. In general-sum MGs, value function based methods can't find a stationary Nash equilibrium. Zinkevich et al. found an example of turn-taking general-sum MGs for which no value function based method can find a stationary Nash equilibrium.

Exact methods to solve general-sum MGs have been developed in the past two decades. These methods are generally hopelessly difficult when more than two players are interacting. Thus, extending exact algorithms to the batch scenario and to function approximation seems to be a risky path as all exact algorithms require too much computation. Our approach is to reduce the problem to the minimization of a loss function over the polices and over the state action value functions. This loss is a sum over players of two Bellman residuals. This loss is non convex and thus, the method is not guaranteed to converge when the model is known and when no function approximations are used. However, our method can be efficiently estimated with samples and support approximation both on the policy and on the state-actions value function.

The first key ingredient of our method is to define as a norm a weaker notion of an ϵ-Nash equilibrium than the one in 3.1. Since this norm can not be directly optimized and estimated with batch data, we define a surrogate loss (a sum of Bellman residuals) that can be estimated with batch data and supports function approximation. As there are no theoretical guarantees of convergence of these methods, we evaluate its soundness on randomly generated MDPs and general-sum MGs.

1 Nash, ϵ-Nash and Weak ϵ-Nash Equilibrium

We have defined the Notion of Nash equilibrium in Section 3.1. It states that one player cannot improve his own value by switching his strategy if the other players do

not vary their own one Filar and Vrieze (2012). The goal here is to find one strategy for players which is as close as possible to a Nash equilibrium. The definition of Section 3.1 definition can be rewritten with Bellman operators:

Definition 7.1. *In an MG, a strategy* $\boldsymbol{\pi}$ *is a Nash equilibrium if* $\exists \boldsymbol{v}$ *such as* $\forall i \in \{1, ..., N\}, \mathcal{T}^i_{\boldsymbol{\pi}} v^i = v^i$ *and* $\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i = v^i$.

Proof. The proof is left in appendix 7. □

We will use the definition of ϵ-Nash equilibrium in to quantify the performance of a joint strategy in MGs. As in MDP the criterion to evaluate a policy is the distance of it's value v_π to the optimal value v^*, in MGs, we will use the maximum over player of the distance between a joint value $v^i_{\boldsymbol{\pi}}$ to the value of a best response $v^{*i}_{\boldsymbol{\pi}^{-i}}$. In the case of a single player MG (or MDP), a Nash equilibrium is simply the optimal strategy. Formally Filar and Vrieze (2012):

Definition 7.2. *In an MG, a strategy* $\boldsymbol{\pi}$ *is an* ϵ*-Nash equilibrium if:*
$\forall i \in \{1, ..., N\},\ v^i_{\boldsymbol{\pi}} + \epsilon \geq v^{*i}_{\boldsymbol{\pi}^{-i}}$
or $\forall i \in \{1, ..., N\},\ v^{*i}_{\boldsymbol{\pi}^{-i}} - v^i_{\boldsymbol{\pi}} \leq \epsilon$,
which is equivalent to: $\left\| \left\| v^{*i}_{\boldsymbol{\pi}^{-i}} - v^i_{\boldsymbol{\pi}} \right\|_{s,\infty} \right\|_{i,\infty} \leq \epsilon$.

An ϵ-Nash equilibrium is a relaxed solution concept in game theory. When all players play an ϵ-Nash equilibrium the value they will receive is at most ϵ sub-optimal compared to a best response. Interestingly, when considering an MDP, the definition of an ϵ-Nash equilibrium is reduced to control the $\mathcal{L}_{+\infty}$-norm between the value of the players' strategy and the optimal value. However, it is known that approximate dynamic programming algorithms do not control a $\mathcal{L}_{+\infty}$-norm but rather an $\mathcal{L}_p$-norm Munos and Szepesvári (2008) (we take the definition of the $\mathcal{L}_p$-norm of Piot et al. (2014a)). Using $\mathcal{L}_p$-norm is necessary for approximate dynamic programming algorithms to use complexity bounds from learning theory Piot et al. (2014a). The convergence of these algorithms was analyzed using supervised learning bounds in $\mathcal{L}_p$-norm and thus guaranties are given in $\mathcal{L}_p$-norm Scherrer et al. (2012). In addition, Bellman residual approaches on MDPs also give guaranties in $\mathcal{L}_p$-norm Maillard et al. (2010), Piot et al. (2014a). Thus, we define a natural relaxation of the previous definition of the ϵ-Nash equilibrium in $\mathcal{L}_p$-norm which is consistent with the existing work on MDPs.

Definition 7.3. *In an MG,* $\boldsymbol{\pi}$ *is a weak* ϵ*-Nash equilibrium if:* $\left\| \left\| v^{*i}_{\boldsymbol{\pi}^{-i}} - v^i_{\boldsymbol{\pi}} \right\|_{\mu(s),p} \right\|_{\rho(i),p} \leq \epsilon$.

One should notice that an ϵ-Nash equilibrium is a weak ϵ-Nash equilibrium (as an $\mathcal{L}_{+\infty}$-norm is an upper bound of any $\mathcal{L}_p$-norm with respect to any probability measure). Conversely, a weak ϵ-Nash equilibrium is not always an ϵ-Nash equilibrium. Furthermore, both ϵ do not need to be equal. The notion of weak ϵ-Nash equilibrium defines a performance criterion to evaluate a strategy while seeking for a Nash equilibrium. Thus, this definition has the great advantage to provide a convincing way to evaluate

the final strategy. In the case of an MDP, it states that a weak ϵ-Nash equilibrium only consists in controlling the difference in $\mathcal{L}_p$-norm between the optimal value and the value of the learned strategy. For an MG, this criterion is an $\mathcal{L}_p$-norm over players ($i \in \{1, \dots, N\}$) of an $\mathcal{L}_p$-norm over states ($s \in S$) of the difference between the value of the joint strategy $\boldsymbol{\pi}$ for player i in state s and of his best response against the joint strategy $\boldsymbol{\pi}^{-i}$. In the following, we consider that learning a Nash equilibrium should result in minimizing the loss $\left\| \left\| v^{*i}_{\boldsymbol{\pi}^{-i}} - v^i_{\boldsymbol{\pi}} \right\|_{\mu(s),p} \right\|_{\rho(i),p}$. For each player, we want to minimize the difference between the value of his strategy and a best response considering the strategy of the other players is fixed. However, a direct minimization of that norm is not possible in the batch setting even for MDPs. Indeed, $v^{*i}_{\boldsymbol{\pi}^{-i}}$ cannot be directly observed and be used as a target. A common strategy to alleviate this problem is to minimize a surrogate loss. In MDPs, a possible surrogate loss is the optimal Bellman residual $\|v - T^* v\|_{\mu,p}$ (where μ is a distribution over states and T^* is the optimal Bellman operator for MDPs) Piot et al. (2014a), Baird et al. (1995). The optimal policy is then extracted from the learnt optimal value (or Q-value in general). In the following section, we extend this Bellman residual approach to MGs.

2 Bellman Residual Minimization in MGs

Optimal Bellman residual minimization can't be generalized to general-sum MGs as it is because multi-player strategies cannot be directly extracted from value functions as shown in (Zinkevich et al., 2006). Yet, from Definition 7.1, we know that a joint strategy $\boldsymbol{\pi}$ is a Nash equilibrium if there exists $\boldsymbol{v}$ such that, for any player i, v^i is the value of the joint strategy $\boldsymbol{\pi}$ for player i (i.e. $\mathcal{T}^i_{\boldsymbol{\pi}} v^i = v^i$) and v^i is the value of the best response player i can achieve regarding the opponent's strategy $\boldsymbol{\pi}^{-i}$ (i.e. $\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i = v^i$). We thus propose to build a second Bellman-like residual optimizing over the set of strategies so as to directly learn a weak ϵ-Nash equilibrium. The first (traditional) residual (i.e. $\left\| \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i - v^i \right\|_{\rho(i),p}$) forces the value of each player to be close to their respective best response to every other player while the second residual (i.e. $\|\mathcal{T}^i_{\boldsymbol{\pi}} v^i - v^i\|_{\rho(i),p}$) will force every player to play the strategy corresponding to that value.

One can thus wonder how close from a Nash-Equilibrium $\boldsymbol{\pi}$ would be if there existed $\boldsymbol{v}$ such that $\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i \approx v^i$ and $\mathcal{T}^i_{\boldsymbol{\pi}} v^i \approx v^i$. In this section, we prove that, if we are able to control over $(\boldsymbol{v}, \boldsymbol{\pi})$ a sum of the $\mathcal{L}_p$-norm of the associated Bellman residuals ($\left\| \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i - v^i \right\|_{\mu,p}$ and $\|\mathcal{T}^i_{\boldsymbol{\pi}} v^i - v^i\|_{\mu,p}$), then we are able to control $\left\| \left\| v^{*i}_{\boldsymbol{\pi}^{-i}} - v^i_{\boldsymbol{\pi}} \right\|_{\mu(s),p} \right\|_{\rho(i),p}$.

Theorem 7.1. *$\forall p, p'$ positive reals such that $\frac{1}{p} + \frac{1}{p'} = 1$ and $\forall (v^1, \dots, v^N)$:*

$$\left\| \left\| v^{*i}_{\boldsymbol{\pi}^{-i}} - v^i_{\boldsymbol{\pi}} \right\|_{\mu(s),p} \right\|_{\rho(i),p} \leq \frac{2^{\frac{1}{p'}} C_\infty(\mu,\nu)^{\frac{1}{p}}}{1-\gamma} \tag{7.1}$$

$$\times \left[\sum_{i=1}^{N} \rho(i) \left(\left\| \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i - v^i \right\|^p_{\nu,p} + \left\| \mathcal{T}^i_{\boldsymbol{\pi}} v^i - v^i \right\|^p_{\nu,p} \right) \right]^{\frac{1}{p}}, \tag{7.2}$$

with the following concentrability coefficient (the norm of a Radon-Nikodym derivative). $C_\infty(\mu,\nu,\pi^i,\boldsymbol{\pi}^{-i}) = \left\| \frac{\partial \mu^T (1-\gamma)(\mathcal{I}-\gamma P_{\pi^i,\boldsymbol{\pi}^{-i}})^{-1}}{\partial \nu^T} \right\|_{\nu,\infty}$ *and* $C_\infty(\mu,\nu) = \sup_{\boldsymbol{\pi}} C_\infty(\mu,\nu,\pi^i,\boldsymbol{\pi}^{-i})$.

Proof. The proof is left in appendix 8. □

This theorem shows that an ϵ-Nash equilibrium can be controlled by the sum over the players of the sum of the norm of two Bellman-like residuals: the Bellman Residual of the best response of each player and the Bellman residual of the joint strategy. If the residual of the best response of player i ($\left\|\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i - v^i\right\|^p_{\nu,p}$) is small, then the value v^i is close to the value of the best response $v^{*i}_{\boldsymbol{\pi}^{-i}}$ and if the Bellman residual of the joint strategy $\|\mathcal{T}^i_{\boldsymbol{\pi}} v^i - v^i\|^p_{\nu,p}$ for player i is small, then v^i is close to $v^i_{\boldsymbol{\pi}}$. In the end, if all those residuals are small, the joint strategy is an ϵ-Nash equilibrium with ϵ small since $v^{*i}_{\boldsymbol{\pi}^{-i}} \simeq v^i \simeq v^i_{\boldsymbol{\pi}}$.

Theorem 7.1 also emphasizes the necessity of a weakened notion of an ϵ-Nash equilibrium. It is much easier to control a $\mathcal{L}_p$-norm than a $\mathcal{L}_\infty$-norm with samples. In the following, the weighted sum of the norms of the two Bellman residuals will be noted as: $f_{\nu,\rho,p}(\boldsymbol{\pi},\boldsymbol{v}) = \sum_{i=1}^{N} \rho(i) \left(\left\|\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i - v^i\right\|^p_{\nu,p} + \|\mathcal{T}^i_{\boldsymbol{\pi}} v^i - v^i\|^p_{\nu,p} \right)$.

Finding a Nash equilibrium is then reduced to a non-convex optimization problem. If we can find a $(\boldsymbol{\pi},\boldsymbol{v})$ such that $f_{\nu,\rho,p}(\boldsymbol{\pi},\boldsymbol{v}) = 0$, then the joint strategy $\boldsymbol{\pi}$ is a Nash equilibrium. This procedure relies on a search over the joint value function space and the joint strategy space. Besides, if the state space or the number of joint actions is too large, the search over values and strategies might be intractable. We addressed the issue by making use of approximate value functions and strategies. Actually, Theorem 7.1 can be extended with function approximation. A good joint strategy $\boldsymbol{\pi}_\theta$ within an approximate strategy space Π can still be found by computing $\boldsymbol{\pi}_{\boldsymbol{\theta}}, \boldsymbol{v}_{\boldsymbol{\eta}} \in \underset{\boldsymbol{\theta},\,\boldsymbol{\eta}}{\operatorname{argmin}} f_{\nu,\rho,p}(\boldsymbol{\pi}_{\boldsymbol{\theta}}, \boldsymbol{v}_{\boldsymbol{\eta}})$ (where $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ respectively parameterize $\boldsymbol{\pi}_{\boldsymbol{\theta}}, \boldsymbol{v}_{\boldsymbol{\eta}}$). Even with function approximation, the learned joint strategy $\boldsymbol{\pi}_\theta$ would be at least a weak ϵ-Nash equilibrium (with $\epsilon \leq \frac{2^{\frac{1}{p'}} C_\infty(\mu,\nu)^{\frac{1}{p}}}{1-\gamma} f_{\nu,\rho,p}(\boldsymbol{\pi}_{\boldsymbol{\theta}}, \boldsymbol{v}_{\boldsymbol{\eta}})$). This is, to our knowledge, the first approach to solve MGs within an approximate strategy space and an approximate value function space in the batch setting.

3 The Batch Scenario

In this section, we explain how to learn a weak ϵ-Nash equilibrium from Theorem 7.1 with approximate strategies and an approximate value functions. As said previously, we will focus on the batch setting where only a finite number of historical data sampled from an MG are available. As for previous batch algorithms, we will work on state-actions value functions (the definition of the Q-function can be found in Section 3 and the one of Bellman operators can be found in Section 3.2).

Thus, we will have to minimize the following function depending on strategies and Q-functions:

$$f(\mathbf{Q}, \boldsymbol{\pi}) = \sum_{i=1}^{N} \rho(i) \left(\left\| \mathcal{B}^{*i}_{\boldsymbol{\pi}^{-i}} Q^i - Q^i \right\|^p_{\nu,p} + \left\| \mathcal{B}^{i}_{\boldsymbol{\pi}} Q^i - Q^i \right\|^p_{\nu,p} \right) \tag{7.3}$$

The batch scenario consists in having a set of k samples $(s_j, (a^1_j, ..., a^N_j), (r^1_j, ..., r^N_j), s'_j)_{j \in \{1,...,k\}}$ where $r^i_j = r^i(s_j, a^1_j, ..., a^N_j)$ and where the next state is sampled according to $p(.|s_j, a^1_j, ..., a^N_j)$. From Equation (7.3), we can minimize the empirical-norm by using the k samples to obtain the empirical estimator of the Bellman residual error.

$$\tilde{f}_k(\mathbf{Q}, \boldsymbol{\pi}) = \sum_{j=1}^{k} \sum_{i=1}^{N} \rho(i) \left[\left| \mathcal{B}^{*i}_{\boldsymbol{\pi}^{-i}} Q^i(s_j, \mathbf{a}_j) - Q^i(s_j, \mathbf{a}_j) \right|^p + \left| \mathcal{B}^{i}_{\boldsymbol{\pi}} Q^i(s_j, \mathbf{a}_j) - Q^i(s_j, \mathbf{a}_j) \right|^p \right], \tag{7.4}$$

For more details, an extensive analysis beyond the minimization of the Bellman residual in MDPs can be found in Piot et al. (2014a). In the following we discuss the estimation of the two Bellman residuals $\left| \mathcal{B}^{*i}_{\boldsymbol{\pi}^{-i}} Q^i(s_j, \mathbf{a}_j) - Q^i(s_j, \mathbf{a}_j) \right|^p$ and $|\mathcal{B}^{i}_{\boldsymbol{\pi}} Q^i(s_j, \mathbf{a}_j) - Q^i(s_j, \mathbf{a}_j)|^p$ in different cases.

Deterministic Dynamics: With deterministic dynamics, the estimation is straightforward. We estimate $\mathcal{B}^{i}_{\boldsymbol{\pi}} Q^i(s_j, \mathbf{a}_j)$ with $r^i_j + \gamma E_{\mathbf{b} \sim \boldsymbol{\pi}}[Q^i(s'_j, \mathbf{b})]$ and $\mathcal{B}^{*i}_{\boldsymbol{\pi}^{-i}} Q^i(s_j, \mathbf{a}_j)$ with $r^i_j + \gamma \max_{b^i} \left[E_{\boldsymbol{b}^{-i} \sim \boldsymbol{\pi}^{-i}}[Q^i(s'_j, b^i, \boldsymbol{b}^{-i})] \right]$, where the expectation are:

$$E_{\mathbf{b} \sim \boldsymbol{\pi}}[Q^i(s', \mathbf{b})] = \sum_{b^1 \in A^1} \dots \sum_{b^N \in A^N} \pi^1(b^1|s') \dots \pi^N(b^N|s') Q^i(s', b^1, \dots, b^N)$$

and where

$$\begin{aligned} E_{\boldsymbol{b}^{-i} \sim \boldsymbol{\pi}^{-i}}[Q^i(s', b^i, \boldsymbol{b}^{-i})] = & \sum_{b^1 \in A^1} \dots \sum_{b^{i-1} \in A^{i-1}} \sum_{b^{i+1} \in A^{i+1}} \dots \sum_{b^N \in A^N} \pi^1(b^1|s') \dots \\ & \dots \quad \pi^{i-1}(b^{i-1}|s') \pi^{i+1}(b^{i+1}|s') \dots \pi^N(b^N|s') Q^i(s', b^1, \dots, b^N) \end{aligned}$$

Note that this computation can be turned into tensor operations as described in Appendix-9 and thus the architecture can be implemented using relevant libraries.

Stochastic Dynamics: In the case of stochastic dynamics, the previous estimator (7.3) is known to be biased Maillard et al. (2010), Piot et al. (2014a). If the dynamic is known, one can use the following unbiased estimator: $\mathcal{B}^{i}_{\boldsymbol{\pi}} Q^i(s_j, \mathbf{a}_j)$ is estimated with $r^i_j + \gamma \sum_{s' \in S} p(s'|s_j, \mathbf{a}_j) E_{\mathbf{b} \sim \boldsymbol{\pi}}[Q^i(s', \mathbf{b})]$ and $\mathcal{B}^{*i}_{\boldsymbol{\pi}^{-i}} Q^i(s_j, \mathbf{a}_j)$ with $r^i_j + \gamma \sum_{s' \in S} p(s'|s_j, \mathbf{a}_j) \max_{b^i} \left[E_{\boldsymbol{b}^{-i} \sim \boldsymbol{\pi}^{-i}}[Q^i(s', b^i, \boldsymbol{b}^{-i})] \right]$.

If the dynamic of the game is not known (e.g. batch scenario), the unbiased estimator cannot be used since the kernel of the MG is required and other techniques must be applied. One idea would be to first learn an approximation of this kernel. For instance, one could extend the MDP techniques to MGs such as embedding a kernel in a Reproducing Kernel Hilbert Space (RKHS) Grunewalder et al. (2012), Piot

et al. (2014b,a) or using kernel estimators Taylor and Parr (2012), Piot et al. (2014a). Therefore, $p(s'|s_j, \boldsymbol{a}_j)$ would be replaced by an estimator of the dynamics $\tilde{p}(s'|s_j, \boldsymbol{a}_j)$ in the previous equation. If a generative model is available, the issue can be addressed with double sampling as discussed in (Maillard et al., 2010, Piot et al., 2014a).

4 Neural Network Architecture

Minimizing the sum of Bellman residuals is a challenging problem as the objective function is not convex. It is all the more difficult as both the Q-function and the strategy of every player must be learned independently. Nevertheless, neural networks have been able to provide good solutions to problems that require minimizing non-convex objective such as image classification or speech recognition LeCun et al. (2015). Furthermore, neural networks were successfully applied to reinforcement learning to approximate the Q-function Mnih et al. (2015) in one agent MGs with eclectic state representation.

Here, we introduce a novel neural architecture[1] that implements our Bellman residual approach. For every player, a two-fold network is defined: the Q-network that learns a Q-function and a π-network that learns the stochastic strategy of the players. The Q-network is a multilayer perceptron which takes the state representation as input. It outputs the predicted Q-values of the individual action such as the network used by Mnih et al. (2015). Identically, the π-network is also a multilayer perceptron which takes the state representation as input. It outputs a probability distribution over the action space by using a softmax. We then compute the two Bellman residuals for every player following Equation 7.5. Finally, we back-propagate the error by using classic gradient descent operations.

In our experiments, we focus on deterministic turn-based games. It entails a specific neural architecture that is fully described in Figure 7.6. During the training phase, all the Q-networks and the π-networks of the players are used to minimize the Bellman residual. Once the training is over, only the π-network is kept to retrieve the strategy of each player. Note that this architecture differs from classic actor-critic networks Lillicrap et al. (2016) for several reasons. Although Q-network is an intermediate support to the computation of π-network, neither policy gradient nor advantage functions are used in our model. Besides, the Q-network is simply discarded at the end of the training. Our method directly searches in the strategy space by minimizing the Bellman residual.

5 Experiments

In this section, we report an empirical evaluation of our method on randomly generated MGs. This class of problems has been first described by Archibald et al. (1995) for MDPs and has been studied for the minimization of the optimal Bellman residual Piot

[1] https://github.com/fstrub95/nash_network

et al. (2014a) and in zero-sum two-player MGs Perolat et al. (2016). First, we extend the class of randomly generated MDPs to general-sum MGs, then we describe the training setting, finally we analyze the results. Without loss of generality we focus on deterministic turn-based MGs for practical reasons. Indeed, in simultaneous games the complexity of the state actions space grows exponentially with the number of player whereas in turn-based MGs it only grows linearly. Besides, as in the case of simultaneous actions, a Nash equilibrium in a turn-based MG (even deterministic) might be a stochastic strategy Zinkevich et al. (2006). In a turn-based MG, only one player can choose an action in each state. Finally, we run our experiments on deterministic MGs to avoid bias in the estimator as discussed in Section 3.

To sum up, we use turn-based games to avoid the exponential growth of the number of actions and deterministic games to avoid bias in the estimator. The techniques described in Section 3 could be implemented with a slight modification of the architecture described in Section 4.

Benchmark: Again we use artificially generated MGs to test our approach. The general procedure to generate a Garnet is described in Chapter 4. We chose this benchmark for two reasons. First, we chose them for the repeatability of our result on several instances of the same class of MGs. Second, the underlying model of the dynamics of the Garnet is fully known. Thus, it is possible to investigate whether an ϵ-Nash equilibrium have been reached during the training and to quantify the ϵ. It would have been impossible to do so with more complex games. We only investigate $N_B = 1$ to avoid the bias issue and the dynamics is defined as follow. For each state and action (s, a), we randomly pick the next state s' in the neighbourhood of indices of s where s' follow a rounded normal distribution $\mathcal{N}(s, \hat{\sigma}_{s'})$. We then build the transition matrix such as $p(s'|s, a) = 1$. Next, we enforce an arbitrary reward structure into the Garnet to enable the emergence of a Nash equilibrium. Each player i has a random critical state $\hat{s}^i$ and the reward of this player linearly decreases by the distance (encoded by indices) to his critical state. Therefore, the goal of the player is to get as close as possible from his critical state. Some players may have close critical states and may act together while other players have to follow opposite directions. A Gaussian noise of standard deviation $\hat{\sigma}_{noise}$ is added to the reward, then we sparsify it with a ratio m_{noise} to harden the task. Finally, a player is selected uniformly over $\{1, \dots, N\}$ for every state once for all. The resulting vector v_c encodes which player plays at each state as it is a turn-based game. Concretely, a Garnet works as follows: Given a state s, a player is first selected according the vector $v_c(s)$. This player then chooses an action according its strategy π^i. Once the action is picked, every player i receives its individual reward $r^i(s, a)$. Finally, the game moves to a new state according $p(s'|s, a)$ and a new state's player $(v_c(s'))$. Again, the goal of each player is to move as close as possible to its critical state. Thus, players benefit from choosing the action that maximizes its cumulative reward and leads to a state controlled by a non-adversarial player.

Finally, samples $(s, (a^1, \dots, a^N), (r^1, \dots, r^N), s')$ are generated by randomly select-

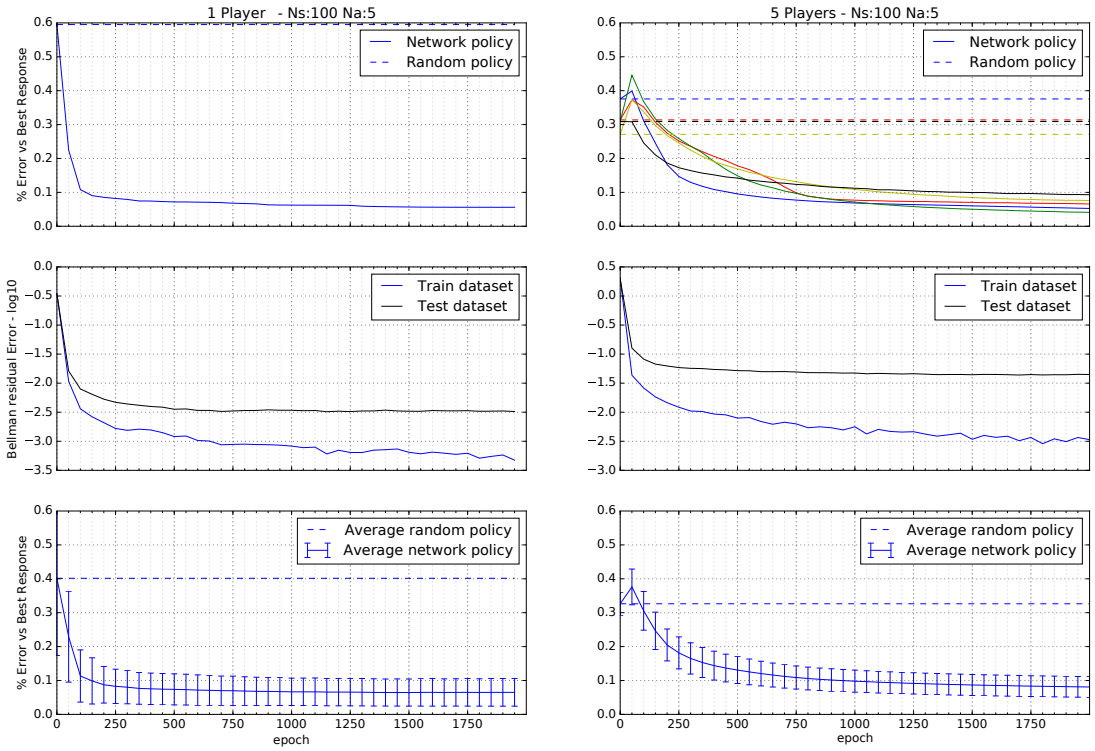


Figure 7.1 – (top) Evolution of the Error vs Best Response during the training for a given Garnet. In both cases, players manage to iteratively improve their strategies. A strong drop may occur when one of the player finds a important move. (middle) Empirical Bellman residual for the training dataset and the testing dataset. It highlights the link between minimizing the Bellman residual and the quality of the learned strategy. (bottom) Average Error vs Best Response averaged over every players, Garnets and every batch. It highlights the robustness of minimizing the Bellman residual over several games. Experiments are run on 5 Garnets which is re-sampled 5 times to average the metrics.

ing a state and an action uniformly. The reward, the next state, and the next player are selected according to the model described above.

Evaluation: Our goal is to verify whether the-joint strategy of the players is an ϵ-Nash equilibrium. To do so, we first retrieve the strategy of every player by evaluating the π^i-networks over the state space. Then, given $\boldsymbol{\pi}$, we can exactly evaluate the value of the joint strategy $v^i_{\boldsymbol{\pi}}$ for each player i and the value of the best response to the strategy of the others $v^{*i}_{\boldsymbol{\pi}\text{-}\boldsymbol{i}}$. The value $v^i_{\boldsymbol{\pi}}$ is computed by inverting the linear system $v^i_{\boldsymbol{\pi}} = (\mathcal{I} - \gamma \mathcal{P}_{\boldsymbol{\pi}})^{-1} r^i_{\boldsymbol{\pi}}$. The value $v^{*i}_{\boldsymbol{\pi}\text{-}\boldsymbol{i}}$ is computed with the policy iteration algorithm. Finally, we compute the *Error vs Best Response* for every player defined as $\frac{\|v^i_{\boldsymbol{\pi}} - v^{*i}_{\boldsymbol{\pi}\text{-}\boldsymbol{i}}\|_2}{\|v^{*i}_{\boldsymbol{\pi}\text{-}\boldsymbol{i}}\|_2}$. If this metric is close to zero for all players, the players reach a weak ϵ-Nash equilibrium with ϵ close to zero. Actually, *Error vs Best Response* is a normalized quantification

of how sub-optimal the player's strategy is compared to his best response. It indicates the sub-optimality margin of the policy of a player compared to his best response while other players keep playing the same strategies. If this metric is close to zero for all players, then they have little incentive to switch from their current strategy. It is an ϵ-Nash equilibrium. In addition, we keep track of the empirical norm of the Bellman residual on both the training dataset and the test dataset as it is our training objective.

Training parameters: We use N-player Garnet with 1, 2 or 5 players. The state space and the action space are respectively of size 100 and 5. The state is encoded by a binary vector. The transition kernel is built with a standard deviation $\hat{\sigma}_{s'}$ of 1. The reward function is $r^i(s) = \frac{2\min(|s-\hat{s}^i|,N_S)-|s-\hat{s}^i|)}{N_S}$ (it is a circular reward function). The reward sparsity m_{noise} is set to 0.5 and the reward white noise $\hat{\sigma}_{noise}$ has a standard deviation of 0.05. The discount factor γ is set to 0.9. The Q-networks and π-networks have one hidden layers of size 80 with RELUs Goodfellow et al. (2016). Q-networks have no output transfer function while π-networks have a softmax. The gradient descent is performed with AdamGrad Goodfellow et al. (2016) with an initial learning rate of 1e-3 for the Q-network and 5e-5 for the π-networks. We use a weight decay of 1e-6. The training set is composed of $5N_SN_A$ samples split into random minibatch of size 20 while the testing dataset contains N_SN_A samples. The neural network is implemented by using the python framework Tensorflow Abadi et al. (2015). The source code is available on Github (Hidden for blind review) to run the experiments.

Results: Results for 1 player (MDP) and 5 players are reported in Figure 7.1. Additional settings are reported in the Appendix-9 such as the two-player case and the tabular cases. In those scenarios, the quality of the learned strategies converges in parallel with the empirical Bellman residual. Once the training is over, the players can increase their cumulative reward by no more than 8% on average over the state space. Therefore, neural networks succeed in learning a weak ϵ-Nash equilibrium. Note that it is impossible to reach a zero error as (i) we are in the batch setting, (ii) we use function approximation and (iii) we only control a weak ϵ-Nash equilibrium. Moreover, the quality of the strategies are well-balanced among the players as the standard deviation is below 5 points.

Our neural architecture has good scaling properties. First, scaling from 2 players to 5 results in the same strategy quality. Furthermore, it can be adapted to a wide variety of problems by only changing the bottom of each network to fit with the state representation of the problem.

Discussions: The neural architecture faces some over-fitting issues. It requires a high number of samples to converge as described on Figure 7.2. Lillicrap et al. (2016) introduces several tricks that may improve the training. Furthermore, we run additional experiments on non-deterministic Garnets but the result remains less conclusive. Indeed, the estimators of the Bellman residuals are biased for stochastic dynamics. As discussed, embedding the kernel or using kernel estimators may help to estimate prop-

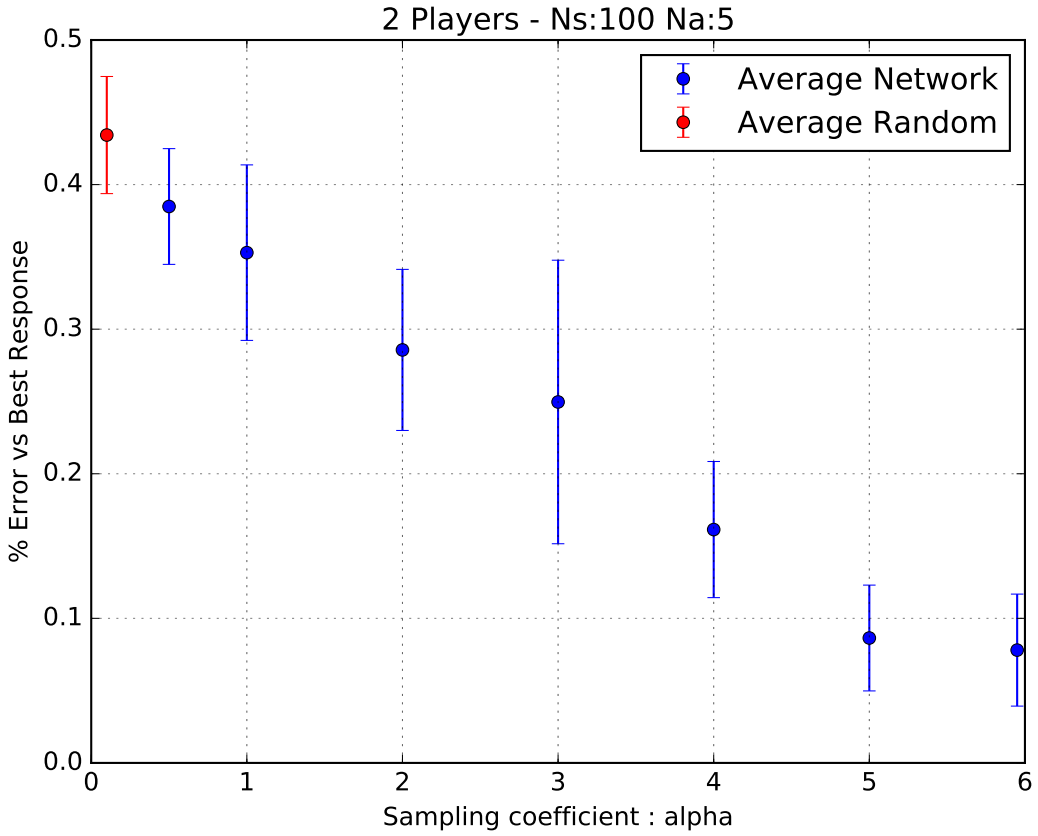


Figure 7.2 – Impact of the number of samples on the quality of the learned strategy. The number of samples per batch is computed by $N_{samples} = \alpha N_A N_S$. Experiments are run on 3 different Garnets which are re-sampled 3 times to average the metrics.

erly the cost function (Equation (7.4)). Finally, our algorithm only seeks for a single Nash-Equilibrium when several equilibria might exist. Finding a specific equilibrium among others is out of the scope of this thesis.

6 Conclusion

In this chapter, we present a novel approach to learn a Nash equilibrium in MGs. The contributions are both theoretical and empirical. First, we define a new (weaker) concept of an ϵ-Nash equilibrium. We prove that minimizing the sum of different Bellman residuals is sufficient to learn a weak ϵ-Nash equilibrium. From this result, we provide empirical estimators of these Bellman residuals with batch data. Finally, we describe a novel neural network architecture to learn a Nash equilibrium from batch data. This architecture does not rely on a specific MG. It also scales to a high number of players for turn taking games. Thus it can be applied to a trove of applications.

This chapter concludes this part on Bellman residual minimization. Whilst approximate dynamic programming was not a convincing approach to learn Nash equilibria, Bellman residual minimization seems to be a reasonable approach for general sum

MGs. This approach does not reduce to the existing ones in MDPs and in zero-sum two-player MGs. Indeed, general-sum MGs lack key properties of MDPs and zero-sum two-player MGs. In MDPs and in zero-sum two-player MGs only require finding the solution of a single fixed point equation whilst an N-player general sum MG requires solving $2N$ fixed point equations simultaneously.

7 Proof of the Equivalence of Definition 2.4 and 7.1

Proof of the equivalence between definition 7.1 and 2.4.

(7.1) $\Rightarrow$ (2.4):

If $\exists \boldsymbol{v}$ such as $\forall i \in \{1, ..., N\}, \mathcal{T}^i_{\boldsymbol{\pi}} v^i = v^i$ and $\mathcal{T}^{*i}_{\boldsymbol{\pi^{-i}}} v^i = v^i$, then $\forall i \in \{1, ..., N\}$, $v^i = v^i_{\pi^i, \boldsymbol{\pi^{-i}}}$ and $v^i = \max_{\tilde{\pi}^i} v^i_{\tilde{\pi}^i, \boldsymbol{\pi^{-i}}}$.

(2.4) $\Rightarrow$ (7.1):

if $\forall i \in \{1, ..., N\}$, $v^i_{\pi^i, \boldsymbol{\pi^{-i}}} = \max_{\tilde{\pi}^i} v^i_{\tilde{\pi}^i, \boldsymbol{\pi^{-i}}}.$, then $\forall i \in \{1, ..., N\}$, the value $v^i = v^i_{\pi^i, \boldsymbol{\pi^{-i}}}$ is such as $\mathcal{T}^i_{\boldsymbol{\pi}} v^i = v^i$ and $\mathcal{T}^{*i}_{\boldsymbol{\pi^{-i}}} v^i = v^i$.

8 Proof of Theorem 7.1

First we will prove the following lemma. The proof is strongly inspired by previous work on the minimization of the Bellman residual for MDPs Piot et al. (2014a).

Lemma 7.1. *let p and p' be a real numbers such that $\frac{1}{p} + \frac{1}{p'} = 1$, then $\forall \mathbf{v}, \boldsymbol{\pi}$ and $\forall i \in \{1, ..., N\}$:*

$$\left\| v^i_{\pi^i_*, \boldsymbol{\pi^{-i}}} - v^i_{\pi^i, \boldsymbol{\pi^{-i}}} \right\|_{\mu, p} \tag{7.5}$$

$$\leq \frac{1}{1-\gamma} \left(C_\infty(\mu, \nu, \pi^i_*, \boldsymbol{\pi^{-i}})^{\frac{p'}{p}} + C_\infty(\mu, \nu, \pi^i, \boldsymbol{\pi^{-i}})^{\frac{p'}{p}} \right)^{\frac{1}{p'}} \left[\left\| \mathcal{T}^{*i}_{\boldsymbol{\pi^{-i}}} v^i - v^i \right\|^p_{\mu,p} + \left\| \mathcal{T}^i_{\boldsymbol{\pi}} v^i - v^i \right\|^p_{\mu,p} \right]^{\frac{1}{p}}, \tag{7.6}$$

where π^i_* is the best response to $\boldsymbol{\pi^{-i}}$. Meaning $v^i_{\pi^i_*, \boldsymbol{\pi^{-i}}}$ is the fixed point of $\mathcal{T}^{*i}_{\boldsymbol{\pi^{-i}}}$. And with the following concentrability coefficient $C_\infty(\mu, \nu, \pi^i, \boldsymbol{\pi^{-i}}) = \left\| \frac{\partial \mu^T (1-\gamma)(\mathcal{I} - \gamma \mathcal{P}_{\pi^i, \boldsymbol{\pi^{-i}}})^{-1}}{\partial \nu^T} \right\|_{\nu, \infty}$.

Proof. The proof uses similar techniques as in Piot et al. (2014a). First we have:

$$v^i_{\pi^i, \boldsymbol{\pi^{-i}}} - v^i = (\mathcal{I} - \gamma \mathcal{P}_{\pi^i, \boldsymbol{\pi^{-i}}})^{-1} (r^i_{\pi^i, \boldsymbol{\pi^{-i}}} - (\mathcal{I} - \gamma \mathcal{P}_{\pi^i, \boldsymbol{\pi^{-i}}}) v^i), \tag{7.7}$$

$$= (\mathcal{I} - \gamma \mathcal{P}_{\pi^i, \boldsymbol{\pi^{-i}}})^{-1} (\mathcal{T}^i_{\pi^i, \boldsymbol{\pi^{-i}}} v^i - v^i). \tag{7.8}$$

But we also have:

$$v^i_{\pi^i_*, \boldsymbol{\pi^{-i}}} - v^i = (\mathcal{I} - \gamma \mathcal{P}_{\pi^i_*, \boldsymbol{\pi^{-i}}})^{-1} (\mathcal{T}^i_{\pi^i_*, \boldsymbol{\pi^{-i}}} v^i - v^i),$$

then:

$$v^i_{\pi^i_*, \boldsymbol{\pi^{-i}}} - v^i_{\pi^i, \boldsymbol{\pi^{-i}}} = v^i_{\pi^i_*, \boldsymbol{\pi^{-i}}} - v^i + v^i - v^i_{\pi^i, \boldsymbol{\pi^{-i}}}, \tag{7.9}$$

$$= (\mathcal{I} - \gamma \mathcal{P}_{\pi^i_*, \boldsymbol{\pi^{-i}}})^{-1} (\mathcal{T}^i_{\pi^i_*, \boldsymbol{\pi^{-i}}} v^i - v^i) - (\mathcal{I} - \gamma \mathcal{P}_{\pi^i, \boldsymbol{\pi^{-i}}})^{-1} (\mathcal{T}^i_{\pi^i, \boldsymbol{\pi^{-i}}} v^i - v^i), \tag{7.10}$$

$$\leq (\mathcal{I} - \gamma \mathcal{P}_{\pi^i_*, \boldsymbol{\pi^{-i}}})^{-1} (\mathcal{T}^{*i}_{\boldsymbol{\pi^{-i}}} v^i - v^i) - (\mathcal{I} - \gamma \mathcal{P}_{\pi^i, \boldsymbol{\pi^{-i}}})^{-1} (\mathcal{T}^i_{\pi^i, \boldsymbol{\pi^{-i}}} v^i - v^i), \tag{7.11}$$

$$\leq (\mathcal{I} - \gamma \mathcal{P}_{\pi^i_*, \boldsymbol{\pi^{-i}}})^{-1} \left| \mathcal{T}^{*i}_{\boldsymbol{\pi^{-i}}} v^i - v^i \right| + (\mathcal{I} - \gamma \mathcal{P}_{\pi^i, \boldsymbol{\pi^{-i}}})^{-1} \left| \mathcal{T}^i_{\pi^i, \boldsymbol{\pi^{-i}}} v^i - v^i \right|. \tag{7.12}$$

Finally, using the same technique as the one in Piot et al. (2014a), we get:

$$\left\|v^i_{\pi^i_*,\boldsymbol{\pi}^{-i}} - v^i_{\pi^i,\boldsymbol{\pi}^{-i}}\right\|_{\mu,p} \tag{7.13}$$

$$\leq \left\|(\mathcal{I}-\gamma\mathcal{P}_{\pi^i_*,\boldsymbol{\pi}^{-i}})^{-1}\left|\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}}v^i - v^i\right|\right\|_{\mu,p} + \left\|(\mathcal{I}-\gamma\mathcal{P}_{\pi^i,\boldsymbol{\pi}^{-i}})^{-1}\left|\mathcal{T}^{i}_{\pi^i,\boldsymbol{\pi}^{-i}}v^i - v^i\right|\right\|_{\mu,p}, \tag{7.14}$$

$$\leq \frac{1}{1-\gamma}\left[C_\infty(\mu,\nu,\pi^i_*,\boldsymbol{\pi}^{-i})^{\frac{1}{p}}\left\|\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}}v^i - v^i\right\|_{\nu,p} + C_\infty(\mu,\nu,\pi^i,\boldsymbol{\pi}^{-i})^{\frac{1}{p}}\left\|\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}}v^i - v^i\right\|_{\nu,p}\right], \tag{7.15}$$

$$\leq \frac{1}{1-\gamma}\left(C_\infty(\mu,\nu,\pi^i_*,\boldsymbol{\pi}^{-i})^{\frac{p'}{p}} + C_\infty(\mu,\nu,\pi^i,\boldsymbol{\pi}^{-i})^{\frac{p'}{p}}\right)^{\frac{1}{p'}}\left[\left\|\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}}v^i - v^i\right\|^p_{\nu,p} + \left\|\mathcal{T}^{i}_{\boldsymbol{\pi}}v^i - v^i\right\|^p_{\nu,p}\right]^{\frac{1}{p}}. \tag{7.16}$$

□

Theorem 7.1 falls in two steps:

$$\left\|\left\|\max_{\tilde{\pi}^i} v_{\tilde{\pi}^i,\boldsymbol{\pi}^{-i}} - v^i_\pi\right\|_{\mu(s),p}\right\|_{\rho(i),p} \leq \frac{1}{1-\gamma}\left[\max_{i\in\{1,\dots,N\}}\left(C_\infty(\mu,\nu,\pi^i_*,\boldsymbol{\pi}^{-i})^{\frac{p'}{p}} + C_\infty(\mu,\nu,\pi^i,\boldsymbol{\pi}^{-i})^{\frac{p'}{p}}\right)^{\frac{1}{p'}}\right] \tag{7.17}$$

$$\times\left[\sum_{i=1}^N \rho(i)\left(\left\|\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}}v^i - v^i\right\|^p_{\nu,p} + \left\|\mathcal{T}^{i}_{\boldsymbol{\pi}}v^i - v^i\right\|^p_{\nu,p}\right)\right]^{\frac{1}{p}}, \tag{7.18}$$

$$\leq \frac{2^{\frac{1}{p'}}C_\infty(\mu,\nu)^{\frac{1}{p}}}{1-\gamma}\left[\sum_{i=1}^N \rho(i)\left(\left\|\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}}v^i - v^i\right\|^p_{\nu,p} + \left\|\mathcal{T}^{i}_{\boldsymbol{\pi}}v^i - v^i\right\|^p_{\nu,p}\right)\right]^{\frac{1}{p}}, \tag{7.19}$$

with $C_\infty(\mu,\nu) = \left(\sup_{\pi^i,\boldsymbol{\pi}^{-i}} C_\infty(\mu,\nu,\pi^i,\boldsymbol{\pi}^{-i})\right)$

The first inequality is proven using lemma 1 and Holder inequality. The second inequality falls noticing $\forall \pi^i, \boldsymbol{\pi}^{-i},\ C_\infty(\mu,\nu,\pi^i,\boldsymbol{\pi}^{-i}) \leq \sup_{\pi^i,\boldsymbol{\pi}^{-i}} C_\infty(\mu,\nu,\pi^i,\boldsymbol{\pi}^{-i})$.

9 Additional curves

This section provides additional curves regarding the training of the Network.

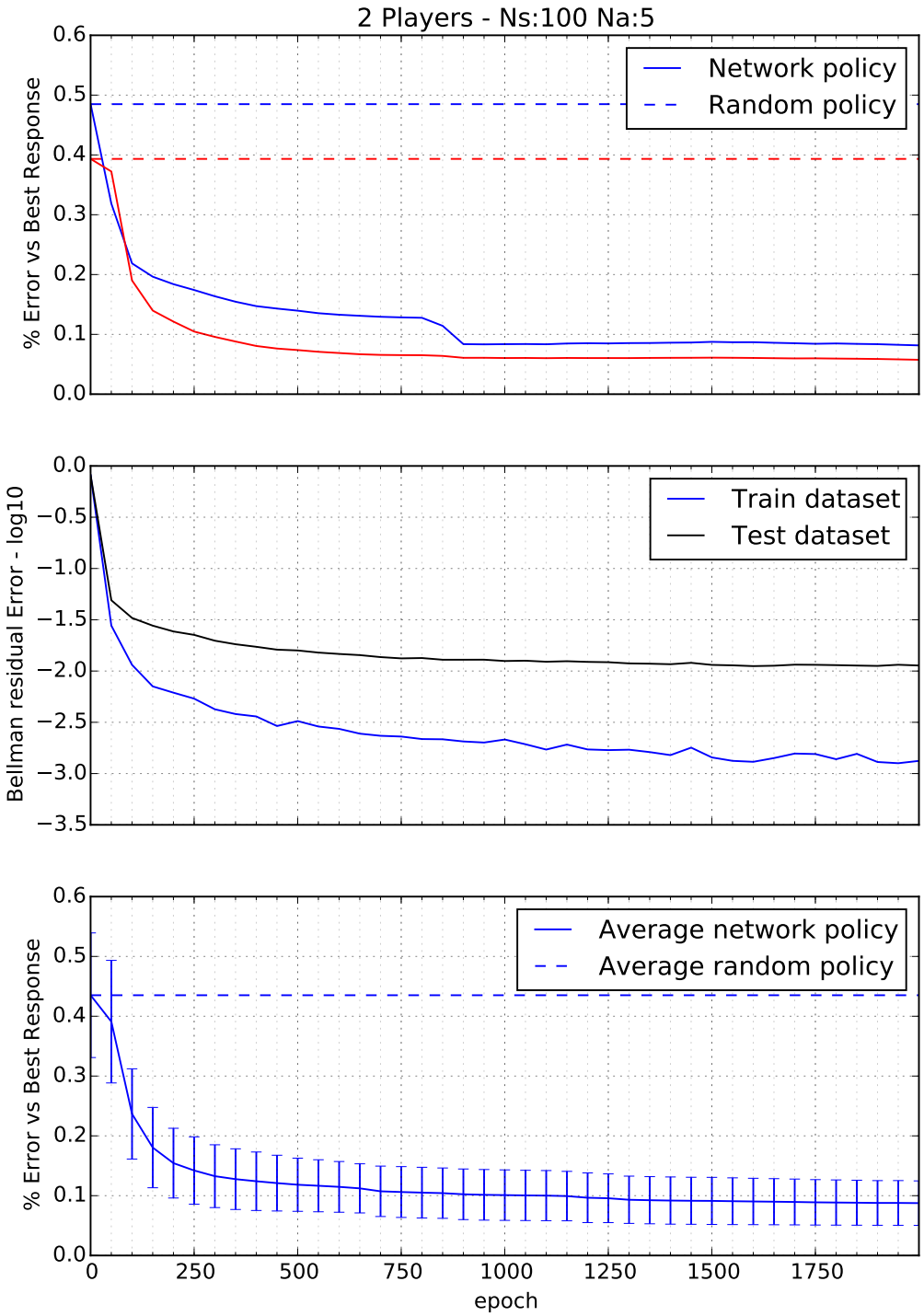


Figure 7.3 – (top) Evolution of the Error vs Best Response during the training for a given Garnet. (middle) Empirical Bellman residual for the training dataset and the testing dataset. (bottom) Average Error vs Best Response averaged over every players, Garnets and every batch. Experiments are run on 5 Garnets which is re-sampled 5 times to average the metrics.

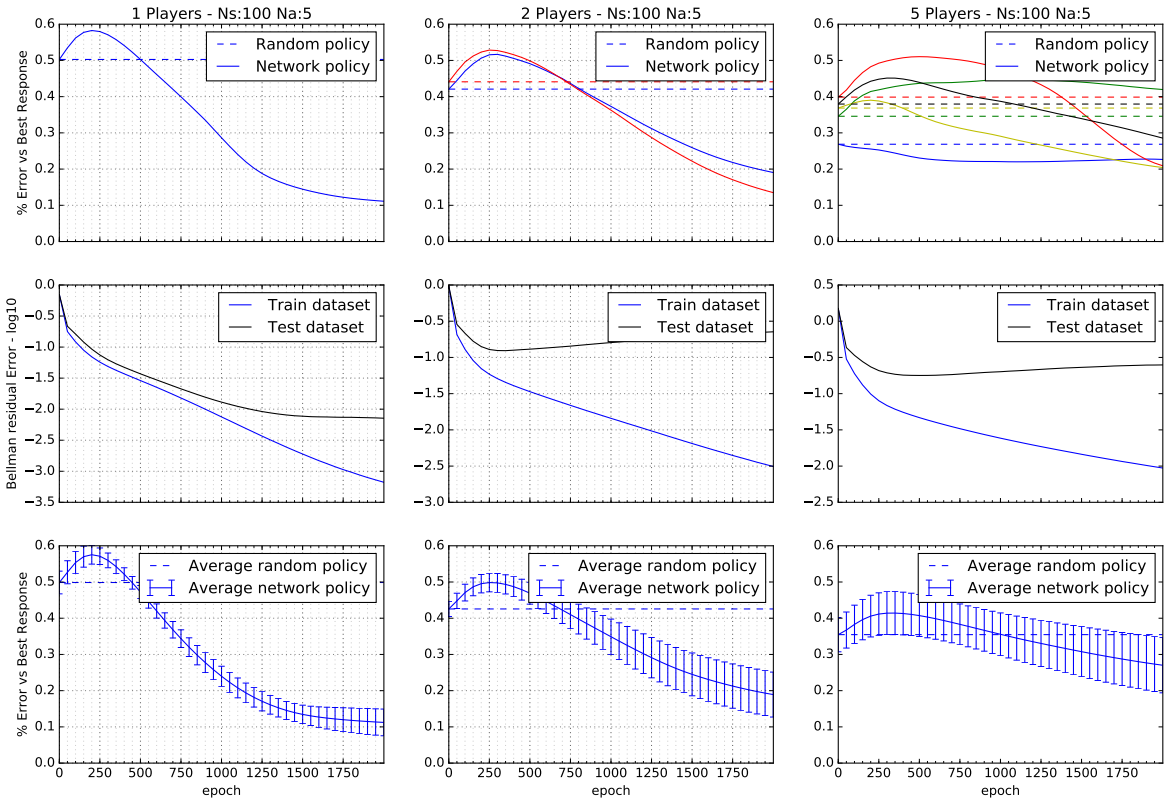


Figure 7.4 – Tabular basis. One may notice that the tabular case works well for a 1 player game (MDP). Yet, the more players there are, the worth it performs. Experiments are run on 5 Garnets which is re-sampled 5 times to average the metrics.

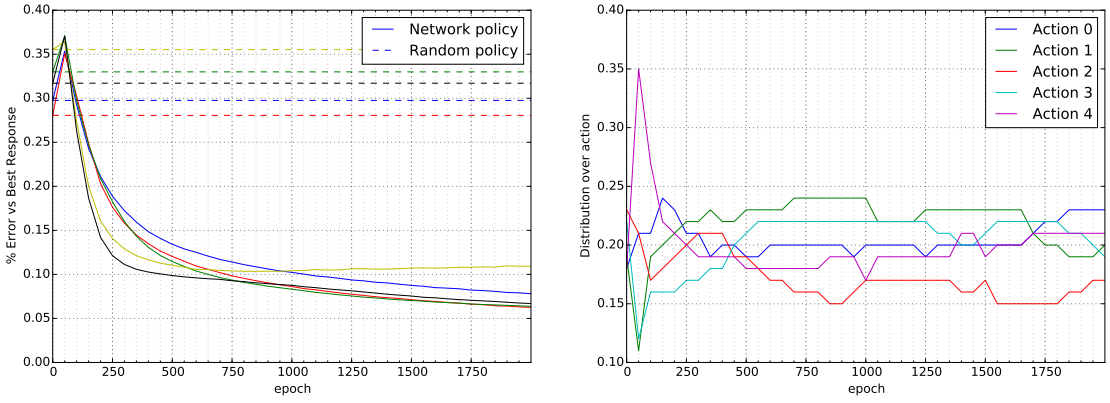


Figure 7.5 – (left)Evolution of the Error vs Best Response during the training for a given Garnet. When the Garnet has a complex structure or the batch is badly distributed, one player sometimes fails to learn a good strategy. (right) Distribution of actions in the strategy among the players with the highest probability. This plots highlights that π-networks do modify the strategy during the training

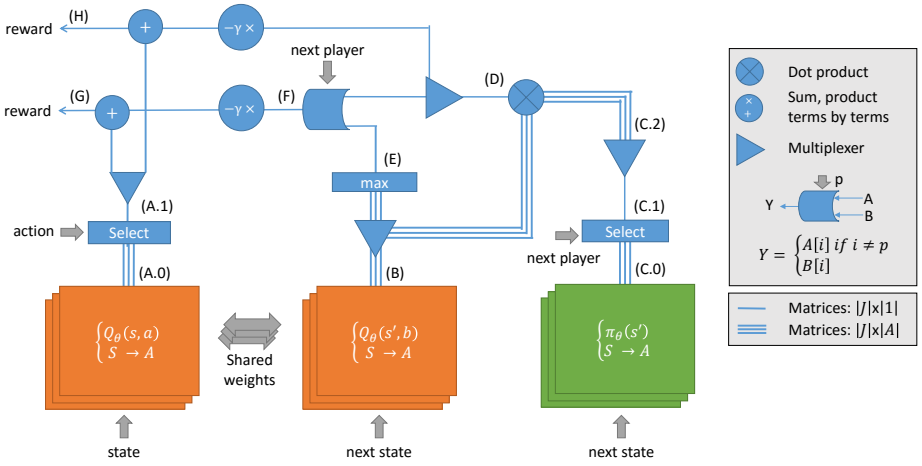


Figure 7.6 – This scheme summarizes the neural architecture that learns a Nash equilibrium by minimizing the Bellman residual for turn-based games. Each player has two networks : a Q-network that learns a state-action value function and a π-network that learns the strategy of the player. The final loss is an empirical estimate of the sum of Bellman residuals given a batch of data of the following shape $(s, (a^1, \dots, a^N), (r^1, \dots, r^N), s')$. The equation (7.4) can be divided into key operations that are described below. For each player i : first of all, in **(A.0)** we compute $Q^i(s, a^1, \dots, a^N)$ by computing the tensor $Q^i(s, \boldsymbol{a})$ and then by selecting the value of $Q^i(s, \boldsymbol{a})$ **(A.1)** given the action of the batch. Step **(B)** computes the tensor $Q^i(s', \boldsymbol{b})$ and step **(C.0)** computes the strategy $\pi^i(.|s')$. In all Bellman residuals we need to average over the strategy of players $Q^i(s', \boldsymbol{b})$. Since we focus on turn-based MGs, we will only average over the strategy of the player controlling next state s' (in the following, this player is called the next player). In **(C.1)** we select the strategy $\pi(.|s')$ of the next player given the batch. In **(C.2)** we duplicate the strategy of the next player for all other players. In **(D)** we compute the dot product between the $Q^i(s', \boldsymbol{b})$ and the strategy of the next player to obtain $E_{\mathbf{b}\sim\boldsymbol{\pi}}[Q^i(s', \mathbf{b})]$ and in **(E)** we pick the highest expected rewards and obtain $\max_{\boldsymbol{b}} Q^i(s', \boldsymbol{b})$. Step **(F)** aims at computing $\max_{b^i} \left[E_{\boldsymbol{b}^{-i}\sim\boldsymbol{\pi}^{-i}}[Q^i(s', b^i, \boldsymbol{b}^{-i})]\right]$ and, since we deal with a turn-based MG, we need to select between the output of **(D)** or **(E)** according to the next player. For all i, we either select the one coming from **(E)** if the next player is i or the one from **(D)** otherwise. In **(G)** we compute the error between the reward r^i to $Q^i(s, \mathbf{a}) - \gamma \max_{b^i} \left[E_{\boldsymbol{b}^{-i}\sim\boldsymbol{\pi}^{-i}}[Q^i(s', b^i, \boldsymbol{b}^{-i})]\right]$ and in **(H)** between r^i and $\gamma E_{\mathbf{b}\sim\boldsymbol{\pi}}[Q^i(s', \mathbf{b})] - Q^i(s, \mathbf{a})$. The final loss is the sum of all the residuals.

Part IV

Independent Learning in Games

Chapter 8

Actor-Critic Fictitious Play

This chapter contributes to filling a gap in the MARL literature by providing two online independent RL algorithms converging to a Nash equilibrium in multi-stage games both in the cooperative case and the zero-sum two-player case. Those two cases used to be treated as different agendas since the seminal paper of Shoham et al. (2007) and we expect our work to serve as a milestone to reconcile them going further than normal form games (Leslie and Collins, 2006, Hofbauer and Sandholm, 2002).

We proposed two novel on-line and decentralized algorithms inspired by actor-critic architectures for Markov Games, each of them working on two time-scales. Those two algorithms perform the same actors' update but use different methods for the critic. The first one performs an off-policy control step whilst the second relies on a policy evaluation step. Although the actor-critic architecture is popular for its success in solving continuous action RL domains, we chose this architecture for a different reason. Our framework requires handling non-stationarity (because of adaptation of the other players) which is another nice property of actor-critic architectures. Our algorithms are stochastic approximations of two dynamical systems that generalize the work of Leslie and Collins (2006) and Hofbauer and Sandholm (2002) on the fictitious play process from normal form games to multistage games Bošanský et al. (2016).

In the following, we first describe the necessary background in both game theory and RL (Section 1) to introduce our first contribution, the two-timescale algorithms (Section 2). These algorithms are stochastic approximations of two continuous-time processes defined in Section 3. Then, we study (in Section 3) the asymptotic behavior of these continuous-time processes and show, as a second contribution, that they converge in self-play in cooperative games and in zero-sum two-player games. In Section 4, our third contribution proves that the algorithms are stochastic approximations of the two continuous-time processes. Finally, we perform an empirical evaluation (in Section 5).

1 Specific Background

Multistage Game: We consider games that can be modeled as trees (see Fig. 8.1 in appendix). The state $\tilde{s}$ is the root of the game tree and each state can be reached with some non-zero probability with at least a deterministic joint strategy. Furthermore, we enforce the fact that once a state is visited, it can never be visited again except state Ω, the end of the game. Formally, a multistage game is an MG with an initial

state $\tilde{s} \in S$ and an absorbing state $\Omega \in S$. To enforce that a state can be only visited once, we define an ordering bijection $\phi : S \to \{1, \dots, |S|\}$ where $|S|$ is the cardinal of S, $\phi(\tilde{s}) = |S|$, $\phi(\Omega) = 1$ and such that $\forall s, s' \in S \times S \backslash \{(\Omega, \Omega)\}$, $\phi(s) \leq \phi(s') \Rightarrow \forall \boldsymbol{\pi}$, $\mathcal{P}_{\boldsymbol{\pi}}(s'|s) = 0$. Moreover the reward in state Ω is null (meaning $\boldsymbol{r}(\Omega, .) = 0$) and no player has more than one action available in that absorbing state. Furthermore, for any state $s \in S \backslash \{\tilde{s}, \Omega\}$ there exists a deterministic strategy $\boldsymbol{\pi}$ and a time $t \leq |S|$ such that $\mathcal{P}^t_{\boldsymbol{\pi}}(s|\tilde{s}) > 0$. This condition means that every states can be reached from state $\tilde{s}$ with at least a deterministic strategy.

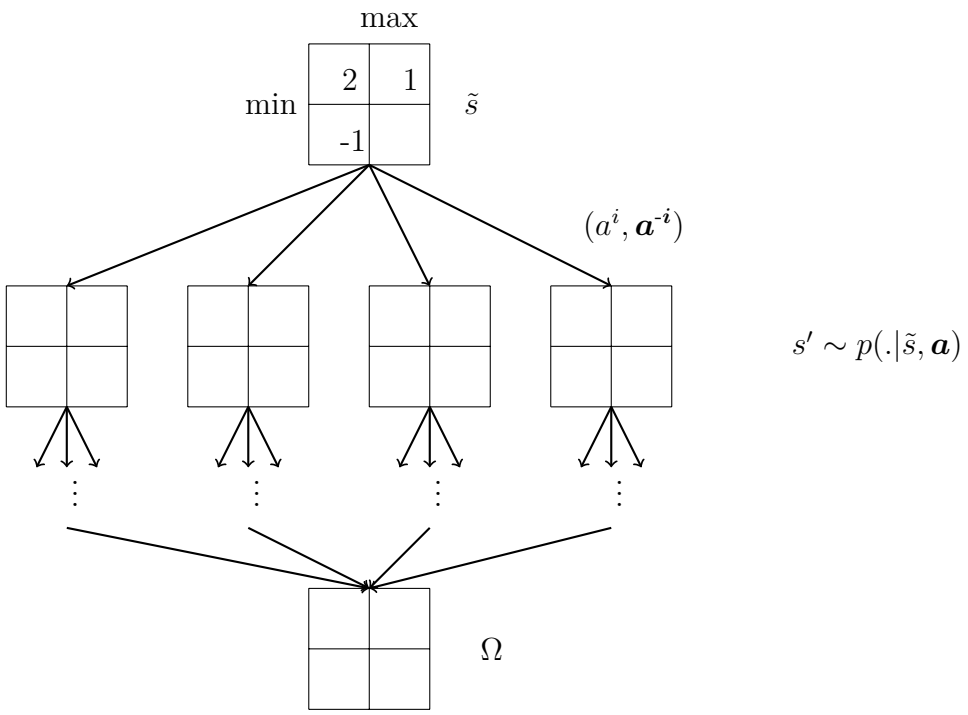


Figure 8.1 – Example of a two-player zero-sum multistage game with deterministic dynamics.

1.1 Specific Operators and Value Functions

We define here specific best responses operators and values. Instead of using argmax as a greedy strategy, we will use a smooth version of the argmax (i.e. a Logit choice function). In this subsection, we review notations introduced in Chapter 2 and define ones specific to this chapter.

Value Function and State-Action Value Function: In a γ-discounted multistage game, (with $\gamma \in (0, 1)$), the value is defined as in MGs (see Section 3 of Chapter 2):

$$\begin{aligned} v^i_{\boldsymbol{\pi}}(s) &= E[\sum_{t=0}^{+\infty} \gamma^t r^i_{\boldsymbol{\pi}}(s_t) | s_0 = s, s_{t+1} \sim \mathcal{P}_{\boldsymbol{\pi}}(.|s_t)], \\ &= \left(\mathcal{I} + \gamma \mathcal{P}_{\boldsymbol{\pi}} + \cdots + \gamma^{|S|} \mathcal{P}^{|S|}_{\boldsymbol{\pi}}\right) r^i_{\boldsymbol{\pi}}. \end{aligned}$$

Note that for a multistage game, the value function is well defined even for $\gamma = 1$ since the process has an absorbing state Ω where the reward function is null. For $t > |S|$, we have that $\mathcal{P}_{\boldsymbol{\pi}}^t r_{\boldsymbol{\pi}}^i = 0$ since after $|S|$ steps (at most) the process ends up in the absorbing state Ω. We can also define a state-action value function per player, that is the value of performing action a^i in state s and then following its strategy: $Q_{\boldsymbol{\pi}}^i(s, a^i) = r_{\boldsymbol{\pi}^{-i}}^i(s, a^i) + \gamma \sum_{s'} p_{\boldsymbol{\pi}^{-i}}(s'|s, a^i) v_{\boldsymbol{\pi}}^i(s')$ (8.1).

Logit Choice Function: when given a choice between actions $a \in A$ based on an expected value $Q(a)$ (in the following $Q(a)$ will be the state-action value function $Q_{\boldsymbol{\pi}}^i(s, a^i)$ or $Q_{\sigma,\boldsymbol{\pi}^{-i}}^{*i}(s, a^i)$ defined below), it is common not to take the action that maximizes $Q(a)$ but to choose suboptimal actions so as to favor exploration. One common choice of suboptimal actions is to pick them according to the logit choice function $B_\eta(Q)(a) = \frac{\exp(\eta^{-1}Q(a))}{\sum_{\tilde{a}} \exp(\eta^{-1}Q(\tilde{a}))}$. We will write $C_\eta(Q) = \sum_{\tilde{a} \in A} Q(\tilde{a}) B_\eta(Q)(\tilde{a})$ the expected outcome if actions are taken according to the logit choice function. This definition can be generalized to the notion of choice probability function (see remark 8.1) and will be written $B_\sigma(Q)$ and $C_\sigma(Q)$ where σ is a function in $\mathbb{R}^{\Delta A}$. Here ΔA is the set of distributions over actions. In the case of the logit choice function, η is linked to σ through the following formula $\sigma(\pi) = \sum_a \eta \pi(a) \ln(\pi(a))$.

Best Responses: If the strategy $\boldsymbol{\pi}^{-i}$ of all opponents is fixed, the value of the best response is $v_{\sigma,\boldsymbol{\pi}^{-i}}^{*i}$ and is recursively defined as follows (starting from Ω and going in increasing order with respect to ϕ): $v_{\sigma,\boldsymbol{\pi}^{-i}}^{*i}(s) = C_\sigma(r_{\boldsymbol{\pi}^{-i}}^i(s, a^i) + \sum_{s' \in S} p_{\boldsymbol{\pi}^{-i}}(s'|s, a^i) v_{\sigma,\boldsymbol{\pi}^{-i}}^{*i}(s'))$. From the definition of that value function, we define the corresponding Q-function: $Q_{\sigma,\boldsymbol{\pi}^{-i}}^{*i}(s, a^i) = r_{\boldsymbol{\pi}^{-i}}^i(s, a^i) + \gamma \sum_{s'} p_{\boldsymbol{\pi}^{-i}}(s'|s, a^i) v_{\sigma,\boldsymbol{\pi}^{-i}}^{*i}(s')$. (8.2)

Bellman Operators: We define the two following Bellman operators on the Q-function: $\left[T_{\boldsymbol{\pi}}^i Q^i\right](s, a^i) = r_{\boldsymbol{\pi}^{-i}}^i(s, a^i) + \gamma \sum_{s'} p_{\boldsymbol{\pi}^{-i}}(s'|s, a^i) E_{b^i \sim \pi^i(.|s')} \left[Q^i(s', b^i)\right]$, and $[T_{\sigma,\boldsymbol{\pi}^{-i}}^{*i} Q^i](s, a^i) = r_{\boldsymbol{\pi}^{-i}}^i(s, a^i) + \gamma \sum_{s'} p_{\boldsymbol{\pi}^{-i}}(s'|s, a^i) C_\sigma\left(Q^i(s', .)\right)$. The value $Q_{\boldsymbol{\pi}}^i$ is the fixed point of the operator $T_{\boldsymbol{\pi}}^i$ and $Q_{\sigma,\boldsymbol{\pi}^{-i}}^{*i}$ is the fixed point of the operator $T_{\sigma,\boldsymbol{\pi}^{-i}}^{*i}$. Furthermore, a strategy $\tilde{\pi}^i$ is greedy with respect to a Q-function Q^i if $T_{\sigma,\boldsymbol{\pi}^{-i}}^{*i} Q^i = T_{\tilde{\pi}^i,\boldsymbol{\pi}^{-i}}^i Q^i$.

Operators on the Value Function: we define the counterparts of those operators on value functions as follows: $\left[T_{\boldsymbol{\pi}}^i v^i\right](s) = r_{\boldsymbol{\pi}}^i(s) + \gamma \sum_{s'} p_{\boldsymbol{\pi}}(s'|s) v^i(s')$, $[T_{\boldsymbol{\pi}}^i v^i] = r_{\boldsymbol{\pi}}^i + \gamma \mathcal{P}_{\boldsymbol{\pi}} v^i$ and $[T_{\sigma,\boldsymbol{\pi}^{-i}}^{*i} v^i] = C_\sigma(r_{\boldsymbol{\pi}^{-i}}^i(s, .) + \gamma \sum_{s'} p_{\boldsymbol{\pi}^{-i}}(s'|s, .) v^i(s'))$. (8.3) From these definitions, we have the two following value functions: $v_{\sigma,\boldsymbol{\pi}^{-i}}^{*i}(s) = C_\sigma(Q_{\sigma,\boldsymbol{\pi}^{-i}}^{*i}(s, .))$ and $v_{\boldsymbol{\pi}}^i(s) = E_{b^i \sim \pi^i(.|s)} \left[Q_{\boldsymbol{\pi}}^i(s, b^i)\right]$.

Smooth Nash Equilibrium: The goal in this setting is to find a strategy π^i for each player that recursively (in increasing order with respect to function $\phi(.)$) fulfills the following condition: $\forall i$, $E_{a^i \sim \pi^i(.|s)} \left[Q_{\boldsymbol{\pi}}^i(s, .)\right] = C_\sigma(Q_{\sigma,\boldsymbol{\pi}}^i(s, .))$. As a consequence, we have that $Q_{\sigma,\boldsymbol{\pi}^{-i}}^{*i} = Q_{\boldsymbol{\pi}}^i$ and $v_{\boldsymbol{\pi}}^i = v_{\sigma,\boldsymbol{\pi}^{-i}}^{*i}$. Furthermore, in the case of a zero-sum two-player game, we have $v_{\boldsymbol{\pi}}^i = -v_{\boldsymbol{\pi}}^{-i}$ (where $v_{\boldsymbol{\pi}}^{-i}$ is the value function of the opponent)

2 Actor-Critic Fictitious Play

We present here our first contribution: the actor-critic fictitious play algorithm for MGs. This is an on-line and decentralized process. At each time-step n, all players are in state s_n, players choose independently an action a_n^i according to their current strategy $\pi_n^i(.|s_n)$ and observe independently one from another a reward signal $r_n^i = r^i(s_n, \boldsymbol{a}_n)$. The process is decentralized, meaning players do not observe others' actions nor their rewards. Then, the game moves to the following state s_{n+1}. If the process reaches the absorbing state Ω, we simply restart from the beginning ($\tilde{s}$).

Algorithm 27 On-line Actor Critic Fictitious Play

Input: An initial strategy π_0^i and an initial value $v_0^i = 0$. Two learning rates $\{\alpha_n\}_{n\geq 0}$, $\{\beta_n\}_{n\geq 0}$ satisfying assumption **A 3** and an initial state $s_0 = \tilde{s}$.
for n=1,2,... **do**
  Agent i draws action $a_n^i \sim \pi^i(.|s_n)$.
  Agent i observes reward $r_n^i = r^i(s_n, \boldsymbol{a}_n)$.
  Every player observes the next state $s_{n+1} \sim p(.|s_n, \boldsymbol{a}_n)$.
  actor step
  $\pi_{n+1}^i(s_n, .) = (1-\beta_n)\pi_n^i(s_n, .) + \beta_n B_\sigma(Q_n^i(s_n, .))$
  critic step
  Either an off-policy control step:
  $Q_{n+1}^i(s_n, a_n^i) = (1-\alpha_n)Q_n^i(s_n, a_n^i) + \alpha_n \left(r_n^i + C_\sigma\left(Q_n^i(s_{n+1}, .)\right)\right)$
  Or a policy evaluation step:
  $Q_{n+1}^i(s_n, a_n^i) = (1-\alpha_n)Q_n^i(s_n, a_n^i) + \alpha_n \left(r_n^i + E_{b\sim\pi_n^i(.|s_{n+1})}\left(Q_n^i(s_{n+1}, b)\right)\right)$
  if $s_{n+1} = \Omega$ **then**
    $s_{n+1} = \tilde{s}$.
  end if
end for
Return The joint strategy $\boldsymbol{\pi}$ and values v^i for all i.

The learning algorithm performs two updates. First, it updates the players' strategy (actors' update). The strategy π_{n+1}^i is a mixture between the current strategy π_n^i and either a local best response $B_\sigma(Q_{\boldsymbol{\pi}_n}^i(s_n, .))$ or a global best response $B_\sigma(Q_{\sigma,\boldsymbol{\pi}_n^{-i}}^{*i}(s_n, .))$. The actors' update is performed according to a slow timescale β_n. Second, it performs the critics' update which evaluates the current strategy. It happens on a fast timescale α_n on which we can consider that the strategy of every player is stationary. If at the actors' step we want to act according to a local best response dynamics, the critic step will perform a policy evaluation step. If we want to perform a global best response dynamics, the critic will perform an off-policy evaluation step. Thus, at the slow timescale, the Q-function Q_n^i has almost converged to $Q_{\sigma,\boldsymbol{\pi}_n^{-i}}^{*i}$ or to $Q_{\boldsymbol{\pi}_n}^i$. Therefore, we obtain canonically two algorithms.

In the next section, as a second contribution, we introduce the two dynamical processes corresponding to these algorithms and we prove that they possess desirable

properties (*i.e.* rationality and convergence in self play for zero-sum two-player and cooperative games). The proofs rely non trivial techniques to propagate the Lyapunov stability property of the Fictitious play process of these ODE on the tree structure of the multistage games. Then in Section 4, as a third contribution, we show formally that these two algorithms (Algorithm 27) are stochastic approximations of those dynamical processes. This is again non-trivial as we need to prove the convergence of a two-time scale discrete scheme depending on a Markov chain.

Remark 8.1. In the previous section, we defined the logit choice function. One can generalize this notion with the concept of choice probability function (see Hofbauer and Sandholm (2002)). Imagine a player is given a choice of an action $a \in A$ based on the outcome $Q(a)$. Instead of choosing based on the outcome $Q(a)$, he also obtains an additive random payoff ϵ_a and chooses with respect to $Q(a) + \epsilon_a$. The vector $(\epsilon_a)_{a\in A}$ is a positive random variable taking it's value in $\mathbb{R}^{|A|}$ and does not depend on $Q(a)$. These choice probability functions allow us to get smooth Bellman operators and ease the analysis of our algorithms. A choice probability function is defined as follows: $B_\sigma(Q)(a) = P(\text{argmax}_{\tilde{a}\in A}[Q(\tilde{a}) + \epsilon_{\tilde{a}}] = a)$. This definition is equivalent to the following one where $\sigma(.)$ is a deterministic perturbation: $B_\sigma(Q) = \text{argmax}_{\tilde{\pi}\in\Delta(A)}[E_{\tilde{a}\sim\tilde{\pi}}[Q(\tilde{a})] - \sigma(\tilde{\pi})]$. The function $C_\sigma(Q)$ is the average value considering the choice probability function $B_\sigma(Q)$: $C_\sigma(Q) = \sum_{\tilde{a}\in A} Q(\tilde{a})B_\sigma(Q)(\tilde{a})$. This perturbation is said admissible Hofbauer and Sandholm (2002) if for all y, $\nabla^2\sigma(y)$ is positive definite on $\mathbb{R}_0 = \{z \in \mathbb{R}^n \ : \sum_j z_j = 0\}$, and if $\|\nabla\sigma(y)\|$ goes to infinity as y approaches the boundary of ΔA. For example, $B_\sigma(Q)(a) = \frac{\exp(\eta^{-1}Q(a))}{\sum_{\tilde{a}}\exp(\eta^{-1}Q(\tilde{a}))}$ if the $\sigma(.)$ perturbation is defined as $\sigma(\pi) = \sum_a \eta\pi(a)\ln(\pi(a))$. In that case, the noise $\epsilon_{\tilde{a}}$ follows a Gumbel distribution. From now on, we consider that $\sigma(.)$ is fixed.

3 Fictitious play in Markov Games

In this section, we propose novel definitions for two perturbed best response dynamics in the case of MGs. These dynamics are defined as a set of Ordinary Differential Equations (ODE) that generalizes the continuous time Fictitious play process to MSGs Hofbauer and Sandholm (2002). Then, we prove the convergence of these processes in multistage games. To do so, we build on the work of Hofbauer and Sandholm (2002) on the stability of the Fictitious play process in normal form games. By induction on the tree structure of multistage games, we prove that our processes have stable attractors in zero-sum two-player and in cooperative multistage games. Later in Section 4, we will prove that our actor-critic algorithms track the solutions of that ODE and thus are guaranteed to converge in both settings. The first one considers a local best response dynamics: $\dot{\pi}^i_t(.|s) = d_{\boldsymbol{\pi}_t}(s)[B_\sigma(Q^i_{\boldsymbol{\pi}_t}(s,.)) - \pi^i_t(.|s)]$ (8.4)

The second process considers a global best response: $\dot{\pi}^i_t(.|s) = d_{\boldsymbol{\pi}_t}(s)[B_\sigma(Q^{*i}_{\sigma,\boldsymbol{\pi}^{-i}_t}(s,.)) - \pi^i_t(.|s)]$ (8.5)

Remark 8.2. The distribution $d_{\boldsymbol{\pi}}$ is the stationary distribution of the Markov process defined by $s' \sim p_{\boldsymbol{\pi}}(.|s)$ and if $s' = \Omega$, we restart the process from $\tilde{s}$. One can show that

if we start the process with a strategy $\boldsymbol{\pi_0}$ that gives a non-null probability for each action and each player, the distribution over states $d_{\boldsymbol{\pi_t}}(s)$ is never null since we consider that there is at least a deterministic strategy that reaches any state. Furthermore, since we consider smooth best response dynamics, the smooth best response will assign to each action some probability which is bounded away from zero (since the Q-function are bounded). Thus, we can always consider that we will visit each state with some minimal probability (i.e. $d_{\boldsymbol{\pi_t}}(s) \geq \delta > 0$).

Two properties are usually desirable for such a stochastic process which are *rationality* and *convergence* in self-play Bowling and Veloso (2001). Rationality implies that if other players converge to a stationary strategy, the learning algorithm will converge to a best response strategy. The convergence property received many definitions in the MARL literature and usually ensures convergence of the algorithm against a class of other algorithms or convergence in self-play. Here, we choose the later.

Rationality: For the rest of this paragraph, let us study a fixed player i. First, if we consider the case where other players are stationary (*i.e.* $\boldsymbol{\pi}_t^{-i} = \boldsymbol{\pi}^{-i}$), the strategy of player i will converge to $B_\sigma(Q^{*i}_{\sigma,\boldsymbol{\pi}^{-i}}(s,.))$ for the second process Eq. (8.5) (since $B_\sigma(Q^{*i}_{\sigma,\boldsymbol{\pi}^{-i}}(s,.))$does not depend on π_t^i the solution of the second process converges exponentially toward the best response strategy). For the first process, let us show that if the strategy π^i follows the dynamics described by (8.4), it converges to a best response strategy. The proof of this property requires the following two technical lemmas.

(T,δ)-perturbation: Let us consider the following ODE where $h(.) : \mathbb{R}^n \to \mathbb{R}^n$ is a Lipschitz continuous function: $\dot{z}(t) = h(z(t))$ (8.6)

We consider the case where ODE (8.6) has an asymptotically stable attractor set J. A bounded measurable function $y(.) : \mathbb{R}^+ \to \mathbb{R}^N$ is a (T,δ)-perturbation of (8.6) $(T,\delta > 0)$ if there exist $0 = T_0 < T_1 < T_2 < \cdots < T_n$ with $T_{i+1} - T_i \geq T$ and solutions $z^j(t),\ t \in [T_j, T_{j+1}]$ of (8.6) for all $j \geq 0$ such that $\sup_{t\in[T_j,T_{j+1}]} \|z^j(t) - y(t)\| < \delta$. The following technical lemma can be proved Borkar (1997b).

Lemma 8.1. *Given $\epsilon, T > 0$, there exists a $\bar{\delta} > 0$ such that for $\delta \in (0,\bar{\delta})$, every (T,δ)-perturbation of Eq. (8.6) converges to J^ϵ (def: the ϵ-neighborhood of J).*

The next lemma is used all over this section. It proves that if, instead of following a given process $\dot{x}(t) = f(x(t))$, one follows a perturbed version of that process $\dot{y}(t) = f_t(y(t))$ one converges to the set of stable equilibria of the unperturbed process if $f_t(.)$ converges toward $f(.)$.

Lemma 8.2. *Let's study the two following ODE $\dot{x}(t) = f(x(t))$ and $\dot{y}(t) = f_t(y(t))$ and let's assume that $\limsup \|f(.) - f_t(.)\| \to 0$ uniformly. Furthermore, let's assume that f is Lipschitz continuous. Then, any bounded solution of $\dot{y}(t) = f_t(y(t))$ converges to the set of attractors of $\dot{x}(t) = f(x(t))$.*

Proof. The full proof of this lemma is left in appendix 7. It shows that all bounded solutions of the time dependent ODE is a (T, δ)-perturbation of the other ODE. □

Now, we show that if player i follows the direction given by the local best response dynamics Eq. (8.4), it converges to a best response to $\boldsymbol{\pi}^{-i}$.

Proposition 1. In any game, if the strategy $\boldsymbol{\pi}^{-i}$ of the opponents is fixed, the process (8.4) converges to a best response.

Proof. The proof of that property is left in appendix 10 □

Now, let's consider the case where the strategy of other players $\boldsymbol{\pi}_t^{-i}$ converges to a stationary strategy $\boldsymbol{\pi}^{-i}$. In that case, one can show that, for a fixed i, the solutions of ODE (8.4) and (8.5) are the same as if $\boldsymbol{\pi}_t^{-i}$ was fixed.

Proposition 2. If $\boldsymbol{\pi}_t^{-i} \to \boldsymbol{\pi}^{-i}$, then the solutions of ODE (8.4) and (8.5) are the same as the one of:

$$\dot{\pi}_t^i(.|s) = d_{\pi_t^i, \boldsymbol{\pi}^{-i}}(s)[B_\sigma(Q^i_{\pi_t^i, \boldsymbol{\pi}^{-i}}(s, .)) - \pi_t^i(.|s)] \text{ and } \dot{\pi}_t^i(.|s) = d_{\pi_t^i, \boldsymbol{\pi}^{-i}}(s)[B_\sigma(Q^{*i}_{\sigma, \boldsymbol{\pi}^{-i}}(s, .)) - \pi_t^i(.|s)]$$

Proof. The proof of that property is left in appendix 11 □

Thus, if all other players converge to a stationary strategy, the strategy of player i converges to a best response to $\boldsymbol{\pi}^{-i}$.

Convergence in Multistage Games: Concerning the convergence in self-play of those two fictitious play processes, one must first be aware that they will not converge in all multistage games. Even in normal form games, the fictitious play process is known to be unstable in some cases. For example, the fictitious play process is known to oscillate in some two-player games Shapley (1964). More surprisingly, Hart and Mas-Colell (2003) proved in the case of normal form games that there exists no uncoupled dynamics which guarantees Nash convergence. In their paper, they consider dynamics for each player which only depends on its own reward and possibly on the strategy of all players. They present an example in which this class of dynamical systems has no stable equilibrium. We were able to prove the convergence of our process in the case of zero-sum two-player and cooperative multistage games.

Proposition 3. In a zero-sum two player multistage game, the process (8.4) and the process (8.5) converge to a smooth Nash equilibrium $\boldsymbol{\pi}$.

Proof. The proof of that property is left in appendix 12 □

Remark 8.3. Proposition 3 can be adapted to study the cooperative case (i.e. when players receive the same reward $\forall i, j\ r^i(s, \boldsymbol{a}) = r^j(s, \boldsymbol{a})$). The proof in the cooperative case also works by induction. The corresponding proposition and proof can be found in the appendix 9.

4 Stochastic Approximation with Two-Timescale

In this section, we provide a novel stochastic approximation theorem taking advantage of two independent techniques. First, we use two timescales because the process we are studying (defined on the strategy of each player called the *actor*) has a complex dependency on the strategy of all players through the Q-function (the *critic*). In an on-line setting, one can't have access to the Q-function of a given strategy (either $Q^i_{\boldsymbol{\pi}}$ or $Q^{*i}_{\sigma,\boldsymbol{\pi}^{-i}}$) as it is an asymptotic solution of the critic process. Thus formally, we look for an asymptotically stable solution of $\dot{y}(t) = g(\mu(y(t)), y(t))$ (actors' update) where $\mu(y)$ is the stable solution of another process $\dot{x}(t) = f(x(t), y)$(critics' update). In our case $\mu(y)$ is either $Q^i_{\boldsymbol{\pi}}$ or $Q^{*i}_{\sigma,\boldsymbol{\pi}^{-i}}$, y is the strategy and x the action-value function. Instead of waiting until the subroutine converges (the critic part) to iterate over the main routine (the actor part), Borkar (1997b) gives an elegant solution to that class of problems by using two timescales to update simultaneously x_n and y_n. The process x_n will be updated according to a "fast" timescale α_n. On that timescale, y_n behaves as if it was stationary and thus, x_n is an estimate of $\mu(y_n)$ (i.e. $\|\mu(y_n) - x_n\| \to 0$). The process y_n will move on a "slower" timescale β_n. On that slower timescale, one can treat the process x_n as $\mu(y_n)$ and thus, y_n will converge to a stable solution of $\dot{y}(t) = g(\mu(y(t)), y(t))$.

Second we use an averaging technique. Our process is on-line and follows a Markovian dynamics controlled by the policy $\boldsymbol{\pi}_n$. All policies and Q-functions are only updated on state s_n. Thus, the two processes x_n and y_n also depend on a Markov process Z_n controlled by y_n. Again, Borkar (2006) shows that, in the case of a simple timescale, the process $y_{n+1} = y_n + \alpha_n f(y_n, Z_n)$ tracks the solution of $\dot{y}(t) = \bar{f}(y(t), d_{y(t)})$ where $\bar{f}(y(t), d_{y(t)})$ is the average of f over the stationary distribution of Z_n (the distribution d_y). Formally, $\bar{f}(y, d_y) = \sum_{z \in \mathcal{Z}} f(y, z) d_y(z)$. In our case, we need to study the following recursion:

$$x_{n+1} = x_n + \alpha_n f(x_n, y_n, Z_n), \tag{8.7}$$
$$y_{n+1} = y_n + \beta_n g(x_n, y_n, Z_n), \tag{8.8}$$

Where:

Assumption 1. *Functions f and g are jointly continuous in their arguments and Lipschitz in their two first arguments uniformly with respect to the third.*

Assumption 2. *The controlled Markov process Z_n takes its value in a discrete space $\mathcal{Z}$ controlled by variable y_n. The variable Z_{n+1} follows the kernel $p(.|Z_n, y_n)$ which is uniformly continuous in y_n. Furthermore, let us suppose that if $y_n = y$, the Markov chain Z_n has a unique invariant distribution $d_y(.)$.*

We note $\bar{f}(x, y, d_y) = \sum_{z \in \mathcal{Z}} f(x, y, z) d_y(z)$ the function f averaged over the stationary distribution of the Markov chain defined in the previous assumption. Similarly, $\bar{g}(x, y, d_y) = \sum_{z \in \mathcal{Z}} g(x, y, z) d_y(z)$.

Assumption 3. *The sequences* $\{\alpha_n\}_{n\geq 0}$ *and* $\{\beta_n\}_{n\geq 0}$ *are two positives decreasing step-size sequences satisfying:* $\sum_{n\geq 0}\alpha_n = \sum_{n\geq 0}\beta_n = \infty$, $\sum_{n\geq 0}\alpha_n^2 < \infty$, $\sum_{n\geq 0}\beta_n^2 < \infty$ *and* $\beta_n = o(\alpha_n)$

Assumption 4. *We need* $\sup_n \|x_n\| < \infty$ *and* $\sup_n \|y_n\| < \infty$

Assumption 5. *For any constant* y*, the ODE:* $\dot{x}(t) = \bar{f}(x(t), y, d_y)$*, has a globally asymptotically stable equilibrium* $\mu(y)$*. That stable equilibrium* $\mu(.)$ *is a Lipschitz continuous function.*

Assumption 6. *The ODE* $\dot{y}(t) = \bar{g}\left(\mu\left(y(t)\right), y(t), d_{y(t)}\right)$ *has a globally asymptotically stable equilibrium* y^*

Theorem 8.1. $(x_n, y_n) \to (\mu(y^*), y^*)$ *almost surely (a.s.).*

Analysis of Actor-Critic Fictitious Play: The analysis of our two algorithms is a direct application of the previous theorem and is left in appendix (App. 13). In a nutshell, the critic Q_n is updated on a fast timescale (i.e. x_n) and we will prove that it's recursion satisfy assumption A 1, A 2, A 3, A 4, and A 5. The actor π_n is updated on a slow timescale (i.e. y_n) and satisfy assumption A 1, A 2, A 3, A 4, and A 6. Thanks to the use of choice probability function, assumption A 1 is satisfied on the slow timescale since a choice probability function is lipschitz with respect to the Q-function. The two on-line algorithms we present here (Algo 27) are stochastic approximations of the two dynamical systems (8.4) and (8.5). In an on-line setting, the interaction between players proceeds as follows. At a step n, players are in state s_n and individually take an action $a_n^i \sim \pi_n^i(.|s_n)$. Each of them receives a reward $r^i(s_n, a_n^1, \dots, a_n^N)$ and the process moves from state s_n to state $s_{n+1} \sim p(.|s_n, a_n^1, \dots, a_n^N)$. In addition, we consider the controlled Markov process $Z_n = (s_n, a_n^1, \dots, a_n^N, s_{n+1})$ (controlled by the strategy $\boldsymbol{\pi}_n$). Finally, if $s_{n+1} = \Omega$, we restart from state $\tilde{s}$ and $Z_{n+1} = (\tilde{s}, \dots)$. We define the following two-timescale processes (with α_n and β_n defined as in A 3):

$$\pi_{n+1}^i = \pi_n^i + \beta_n \mathbf{1}_{s_n} \mathbf{1}_{s_n}^\top \left[B_\sigma(Q_n^i(s_n, .)) - \pi_n^i(.|s_n)\right] \tag{8.9}$$

And:

$$Q_{n+1}^i = Q_n^i + \alpha_n \mathbf{1}_{(s_n, a_n^i)} \mathbf{1}_{(s_n, a_n^i)}^\top \left[r^i(s_n, \boldsymbol{a}_n) + C_\sigma\left(Q_n^i(s_{n+1}, .)\right) - Q_n^i(s_n, a_n^i)\right] \tag{8.10}$$

Or:

$$Q_{n+1}^i = Q_n^i + \alpha_n \mathbf{1}_{(s_n, a_n^i)} \mathbf{1}_{(s_n, a_n^i)}^\top \left[r^i(s_n, \boldsymbol{a}_n) + E_{b^i \sim \pi^i(.|s_{n+1})}\left(Q_n^i(s_{n+1}, b^i)\right) - Q_n^i(s_n, a_n^i)\right] \tag{8.11}$$

Assumption A 1 and assumption A 2 are verified. Regarding assumption A 4, the strategy π_n^i is obviously bounded since it remains in the simplex. If $Q_0^i = 0$, the state-action value function is bounded as follows $\|Q_n^i(s,a)\| \leq \phi(s) R_{\max}$ (this can be shown by recursion). The two faster processes(Eq. (8.22) and (8.23)) track the two following ODE:

$$\dot{Q}_t^i(s, a^i) = d_{\boldsymbol{\pi}}(s)\pi^i(.|s)\left(\left[T_{\sigma, \boldsymbol{\pi}^{-i}}^{*i} Q_t^i\right](s, a^i) - Q_t^i(s, a^i)\right)$$

And

$$\dot{Q}_t^i(s,a^i) = d_{\boldsymbol{\pi}}(s)\pi^i(.|s)\left(\left[T_{\boldsymbol{\pi}}^i Q_t^i\right](s,a^i) - Q_t^i(s,a^i)\right)$$

Those two equations admit as an attractor $Q^{*i}_{\sigma,\boldsymbol{\pi}^{-i}}$ and $Q^i_{\boldsymbol{\pi}}$. Then, the strategy recursion follows either ODE (8.4) (if the subroutine is defined by Eq. (8.23)) or ODE (8.5) (if the subroutine is defined by (8.22)).

5 Experiment on Alesia

The game of Alesia is described in Chapter 3. This game is challenging in many aspects. First, rewards are sparse and are received at the end of the game. Furthermore, strategies need to be stochastic Buro (2004), Bošanský et al. (2016). The performance criterion used to compare algorithms is the difference of the value of the joint strategy $v^i_{\boldsymbol{\pi}}(\tilde{s})$ and the value of the best response $v^i_{0,\boldsymbol{\pi}^{-i}}(\tilde{s})$ without any perturbation σ (*i.e.* $\sigma = 0$)

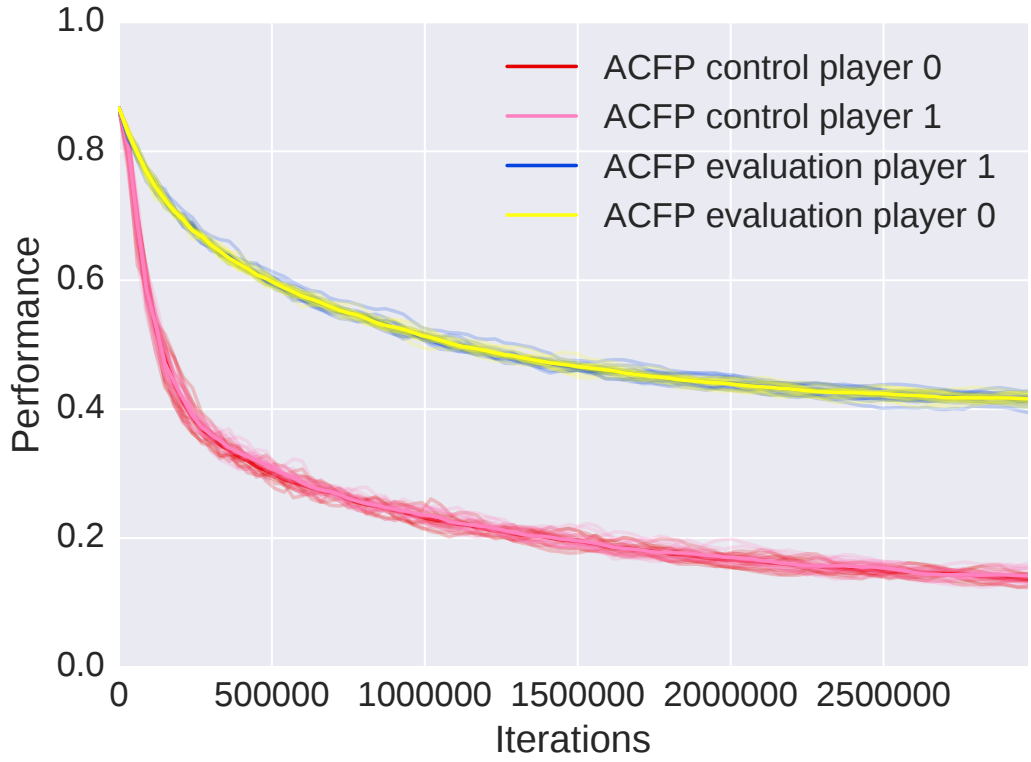


Figure 8.2 – Performance of the strategy π^i (y-axis) along iterations (x-axis).

We ran experiments for a game with learning rates of the form $\alpha_n = \frac{\alpha_0 a_\alpha}{a_\alpha + n^{b_\alpha}}$ and $\beta_n = \frac{\beta_0 a_\beta}{a_\beta + n^{b_\beta}}$. We used a logit choice function with η decaying with the number of iterations $\eta_n = \frac{\eta_0 a_\eta}{a_\eta + n}$. The two step-sizes α_n and β_n were chosen to satisfy the conditions of Theorem 8.1. The experiments were ran with an initial budget of $N = 10$ and a board of size $K = 3$. The step-size parameters were $(\alpha_0, a_\alpha, b_\alpha) = (0.1, 10^4, 0.8)$,

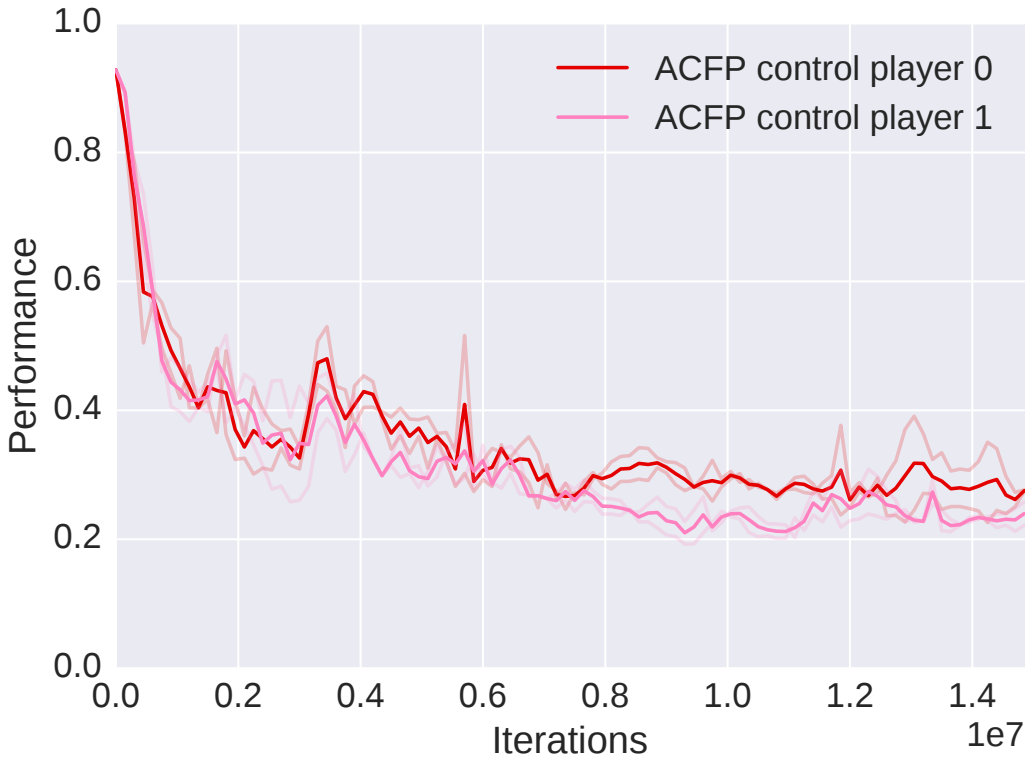


Figure 8.3 – Performance of the strategy π^i (y-axis) along iterations (x-axis).

$(\beta_0, a_\beta, b_\beta) = (0.01, 10^4, 1.0)$ and $(\eta_0, a_\eta) = (0.1, 10^6)$. The results are displayed in figure 8.2 and shows that both algorithms learn a strategy that tend to be closer to its own best response over learning (meaning that both algorithms learn a Nash equilibrium). An empirical finding is that ACFP with a control critic converges faster than the one with a policy evaluation critic on that game.

Function approximation: We also ran these experiments with function approximations both on the strategy and on the Q-function. The main issue in Alesia is that the number of actions available to each player depends on the remaining budget of the player. For each player, we aggregated states for which the budget of the player is fixed. The aggregated states are consecutive states with respect to the budget of the other players'. The results reported in Figure 8.3 show that our method is robust to this form of function approximation. We ran experiment with an initial budget of $N = 15$ on which the algorithm was able to learn a minimax strategy. The step-size parameters were $(\alpha_0, a_\alpha, b_\alpha) = (0.1, 10^5, 0.9)$, $(\beta_0, a_\beta, b_\beta) = (0.01, 10^5, 1.0)$ and $(\eta_0, a_\eta) = (0.05, 10^8)$.

6 Conclusion

This chapter explores two independent RL algorithms that provably converge to Nash equilibria in zero-sum MSGs and in cooperative MSGs. This work studies extensions of the Fictitious play process from normal form games to MSGs. Compared to previous families of methods (Bowling and Veloso, 2001, Littman, 1994), these algorithms can be applied to a larger number of settings without heuristics and without knowing the opponents actions. The main contributions were algorithmic and theoretical but we also proposed an empirical evaluation that provides good evidence that our methods are applicable to real problems and could scale up thanks to function approximation.

Several interesting research paths remain to be explored. The first is the study of the stability of the two dynamical processes in MGs. The proof given for MSGs propagates the convergence property from the end of the game to the initial state. Since MGs have a graph structure, this technique can't be applied. Thus, other stability arguments must be found to ensure convergence in MGs. The second research path could be to further study the use of function approximation. Whilst there is a wide literature dealing with the off-policy evaluation part or the off-policy control (Watkins and Dayan, 1992), the update of the strategy using general function approximation remains challenging. The main issue is that the strategy space should support convex combination with a best response. The convex combination of the strategy and the best response could be fitted at each iteration using a classification neural network.

7 Proof of Lemma 8.2

Lemma 8.2: *Let's study the two following ODE* $\dot{x}(t) = f(x(t))$ *and* $\dot{y}(t) = f_t(y(t))$ *and let's assume that* $\limsup \|f(.) - f_t(.)\| \to 0$ *uniformly. Furthermore, let's assume that* f *is Lipschitz continuous. Then, any solution of* $\dot{y}(t) = f_t(y(t))$ *converges to the set of attractors of* $\dot{x}(t) = f(x(t))$.

Proof. let us fix t such as for all $T > 0, \|f(.) - f_{t+T}(.)\| \leq \epsilon$. if $x(t) = y(t)$, then:

$$x(t+T) - y(t+T) = \int_t^{t+T} [f(x(\tau)) - f(y(\tau))]\, d\tau + \int_t^{t+T} [f(y(\tau)) - f_\tau(y(\tau))]\, d\tau \tag{8.12}$$

$$\|(x-y)(t+T)\| \leq \int_t^{t+T} \|f(x(\tau)) - f(y(\tau))\| d\tau + \int_t^{t+T} \|f(y(\tau)) - f_\tau(y(\tau))\| d\tau \tag{8.13}$$

$$\leq K \int_t^{t+T} \|(x-y)(\tau)\| d\tau + T\epsilon \tag{8.14}$$

Let's write $g(T) = \|(x-y)(t+T)\|$. We have $g(T) \leq K \int_0^T g(\tau) d\tau + T\epsilon$

Let's write $h(T) = T\epsilon + K \int_0^T g(\tau) d\tau + \frac{\epsilon}{K}$ and $h'(T) = \epsilon + Kg(T) \leq \epsilon + K(h(T) - \frac{\epsilon}{K}) \leq Kh(T)$

With the differential form of the Grönwall lemma:

$$h(\delta) \leq h(0)\exp(KT) = \frac{\epsilon}{K}\exp(KT)$$

And we have $\|(x-y)(t+T)\| = g(T) \leq \frac{\epsilon}{K}(\exp(KT) - 1)$

This inequality means that, given T, for all δ we can choose t_0 large enough such that any trajectory of $\dot{y}(t) = f_{t_0+t}(y(t))$ is a (T, δ)-perturbation of $\dot{x}(t) = f(x(t))$. Then, lemma 8.1 concludes the proof. □

8 Proof of Theorem 8.1

$$x_{n+1} = x_n + \alpha_n f(x_n, y_n, Z_n), \tag{8.15}$$

$$y_{n+1} = y_n + \beta_n g(x_n, y_n, Z_n), \tag{8.16}$$

Where:

A 1. *Functions* f *and* g *are jointly continuous in their arguments and Lipschitz in their two first arguments uniformly with respect to the third,*

A 2. *The controlled Markov process* Z_n *takes its value in a discrete space* $\mathcal{Z}$ *controlled by variable* y_n. *The variable* Z_{n+1} *follows the transition kernel* $p(.|Z_n, y_n)$ *which is uniformly continuous in* y_n. *Furthermore, let us suppose that if* $y_n = y$, *the Markov chain* Z_n *has a unique invariant distribution* $d_y(.)$.

From now on, we define $\bar{f}(x, y, d_y) = \sum_{z \in \mathcal{Z}} f(x, y, z) d_y(z)$ which is the function f averaged over the stationary distribution of the Markov chain defined in the previous assumption. Similarly, $\bar{g}(x, y, d_y) = \sum_{z \in \mathcal{Z}} g(x, y, z) d_y(z)$.

A 3. *The sequences* $\{\alpha_n\}_{n \geq 0}$ *and* $\{\beta_n\}_{n \geq 0}$ *are two positives decreasing step-size sequences satisfying:* $\sum_{n \geq 0} \alpha_n = \sum_{n \geq 0} \beta_n = \infty$, $\sum_{n \geq 0} \alpha_n^2 < \infty$, $\sum_{n \geq 0} \beta_n^2 < \infty$ *and* $\beta_n = o(\alpha_n)$

A 4. *We need* $\sup_n \|x_n\| < \infty$ *and* $\sup_n \|y_n\| < \infty$

A 5. *For any constant* y*, the ODE:*

$$\frac{dx(t)}{dt} = \bar{f}(x(t), y, d_y), \tag{8.17}$$

has a globally asymptotically stable equilibrium $\mu(y)$*. That stable equilibrium* $\mu(.)$ *is a Lipschitz continuous function.*

A 6. *The ODE:*

$$\frac{dy(t)}{dt} = \bar{g}\left(\mu\left(y(t)\right), y(t), d_{y(t)}\right) \tag{8.18}$$

has a globally asymptotically stable equilibrium y^*

Theorem 8.1. $(x_n, y_n) \to (\mu(y^*), y^*)$ *almost surely (a.s.).*

Proof. First, rewrite Eq. (8.8) as:

$$y_{n+1} = y_n + \alpha_n \left[\frac{\beta_n}{\alpha_n} g(x_n, y_n, Z_n)\right].$$

Since the function g is Lipschitz in the two first arguments (A 1), since $\mathcal{Z}$ is discrete (A 2) and since x_n and y_n are bounded (A 4), then we have that $\frac{\beta_n}{\alpha_n} g(x_n, y_n, Z_n) \to 0$ a.s.. Then from corollary 8[1] of chapter 6.3 of Borkar (2009) (first presented in (Borkar, 2006)) it follows that (x_n, y_n) converges to an internally chain transitive invariant set of the ODE $\dot{y}(t) = 0$ and $\dot{x}(t) = \bar{f}(x(t), y(t), d_{y(t)})$. In other words, $\|x_n - \mu(y_n)\| \to 0$ a.s..

Second, let's write (8.8) as:

$$y_{n+1} = y_n + \alpha_n \Big[g(\mu(y_n), y_n, Z_n) + \left(g(x_n, y_n, Z_n) - g(\mu(y_n), y_n, Z_n)\right)\Big]. \tag{8.19}$$

Since $g(.,.,.)$ is Lipschitz in the two first variables uniformly with respect to the third one, we have that $\|g(x_n, y_n, Z_n) - g(\mu(y_n), y_n, Z_n)\| \leq K \|x_n - \mu(y_n)\|$ (for some K). Then Eq. (8.8) can be rewritten as:

$$y_{n+1} = y_n + \alpha_n \Big[g(\mu(y_n), y_n, Z_n) + \epsilon_n\Big], \tag{8.20}$$

where $\epsilon_n \to 0$ a.s.. Again, from corollary 8 of chapter 6.3 of Borkar (2009), we have that $y_n \to y^*$ and $x_n \to \mu(y^*)$ □

[1] As written in chapter 2 Borkar (2009) in all stochastic approximation scheme studied in the book (except those in chapter 9) can be added a noise ϵ_n that converges to 0 a.s.

9 Convergence in Cooperative Multistage Games

In a cooperative game (as defined in (Busoniu et al., 2008)), the reward signal is as follows: $\forall s, i, j \ r^i(s, \boldsymbol{a}) = r^j(s, \boldsymbol{a})$). Thus we have the property that $\forall s, i, j \ v^i_{\boldsymbol{\pi}}(s) = v^j_{\boldsymbol{\pi}}(s)$

Proposition 4. In a cooperative two player multistage game, the process (8.4) and the process (8.5) converge to a smooth Nash equilibrium $\boldsymbol{\pi}$.

Proof. The proof of this result works again by induction on the set $S_n = \{s \in S | \phi(s) \leq n\}$. Let's suppose that for all states s in S_n, the process converges to a smooth Nash equilibrium. This means that for all states in $s \in S_n$ and for all players i, the strategies $\pi^i_t(.|s)$ converge to $\pi^i(.|s)$ such as $v^i_{\boldsymbol{\pi}}(s) = v^{*i}_{\sigma,\boldsymbol{\pi}^{-i}}(s)$. We also have that $\forall i, j \ v^i_{\boldsymbol{\pi}}(s) = v^j_{\boldsymbol{\pi}}(s)$.

Let $\hat{s}$ be the state such that $\phi(\hat{s}) = n + 1$. Then, we define:

$$M^i(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) = r^i(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) + \sum_{s' \in S} p(s'|\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) v^i_{\boldsymbol{\pi}}(s')$$

$$M^i_t(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) = r^i(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) + \sum_{s' \in S} p(s'|\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) v^i_{\boldsymbol{\pi}_t}(s')$$

And:

$$M^{*i}_t(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) = r^i(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) + \sum_{s' \in S} p(s'|\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) v^{*i}_{\sigma,\boldsymbol{\pi}^{-i}_t}(s')$$

Since the strategy $\pi^i_t(.|s)$ converges to $\pi^i(.|s)$, we have $v^i_{\boldsymbol{\pi}_t}(s)$ and $v^{*i}_{\sigma,\boldsymbol{\pi}^{-i}_t}(s)$ that converges to $v^i_{\boldsymbol{\pi}}(s)$ for all $s \in S_n$ and finally $M^i_t(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}})$ and $M^{*i}_t(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}})$ converges to $M^i(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}})$. Then, lemma 8.2 and results of convergence of stochastic fictitious play for N-player potential games from Hofbauer and Sandholm (2002) guarantees that the process in state $\hat{s}$ will converge to a smooth Nash equilibrium of the normal form game defined by $M^i(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}})$. Finally, we have $v^i_{\boldsymbol{\pi}}(\hat{s}) = v^{*i}_{\sigma,\boldsymbol{\pi}^{-i}}(\hat{s})$ and $\forall i, j \ v^i_{\boldsymbol{\pi}}(\hat{s}) = v^j_{\boldsymbol{\pi}}(\hat{s})$. □

10 Proof of proposition 1

Proof. We prove the result by induction on the set $S_n = \{s \in S | \phi(s) \leq n\}$. First, the property is true in state Ω since by definition, there is only one action available per player. Suppose that for all $s \in S_n$, the process converges to a best response (i.e. $v^i_{\boldsymbol{\pi}}(s) = v^{*i}_{\sigma,\boldsymbol{\pi}^{-i}}$). Let $\hat{s}$ be the state of order $n+1$ (i.e. $\phi(\hat{s}) = n+1$). From the definition of the Q-function (Eq. (8.1) and (8.2)), we have that in $\hat{s}$ the $Q^i_{\pi^i_t,\boldsymbol{\pi}^{-i}}(\hat{s}, .)$ converges to $Q^i_{\boldsymbol{\pi}}(\hat{s}, .)$ uniformly over the actions. Moreover, since for all $s \in S_n$, $v^i_{\boldsymbol{\pi}}(s) = v^{*i}_{\sigma,\boldsymbol{\pi}^{-i}}$ we have that $Q^i_{\boldsymbol{\pi}}(\hat{s}, .) = Q^{*i}_{\sigma,\boldsymbol{\pi}^{-i}}(\hat{s}, .)$. From lemma 8.2, we get the convergence of $\pi^i(.|\hat{s})$ to a best response if the distribution $d_{\boldsymbol{\pi}}$ is non-null in all states (see remark 8.2). □

11 Proof of Proposition 2

Proof. The proof comes from the fact that $B_\sigma\left(Q^i_{\pi^i,\boldsymbol{\pi}_t^{-i}}(s,.)\right)$ converges to $B_\sigma\left(Q^i_{\pi^i,\boldsymbol{\pi}^{-i}}(s,.)\right)$ uniformly with respect to π^i (since the Q-function is polynomial in $\boldsymbol{\pi}$ and $B_\sigma(.)$ is Lipschitz with respect to the Q-function. Moreover, since the state space is finite, simple convergence imply the uniform convergence) and the fact that $B_\sigma\left(Q^i_{\pi^i,\boldsymbol{\pi}^{-i}}(s,.)\right)$ is Lipschitz with respect to π^i. □

12 Proof of Proposition 3

Proof. The proof of this result works by induction on the set $S_n = \{s \in S | \phi(s) \leq n\}$. Again, the property is true in state Ω. Let's suppose that for all states s in S_n, the process converges to a smooth Nash equilibrium. This means that for all states in $s \in S_n$ and for all players i, the strategies $\pi^i_t(.|s)$ converge to $\pi^i(.|s)$ such as $v^i_{\boldsymbol{\pi}}(s) = v^{*i}_{\sigma,\boldsymbol{\pi}^{-i}}(s)$. We also have that $v^i_{\boldsymbol{\pi}}(s) = -v^{-i}_{\boldsymbol{\pi}}(s)$.

Let $\hat{s}$ be the state such that $\phi(\hat{s}) = n+1$. Then, we define:

$$M^i(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) = r^i(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) + \sum_{s' \in S} p(s'|\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) v^i_{\boldsymbol{\pi}}(s')$$

$$M^i_t(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) = r^i(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) + \sum_{s' \in S} p(s'|\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) v^i_{\boldsymbol{\pi}_t}(s')$$

And:

$$M^{*i}_t(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) = r^i(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) + \sum_{s' \in S} p(s'|\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}}) v^{*i}_{\sigma,\boldsymbol{\pi}_t^{-i}}(s')$$

Since the strategy $\pi^i_t(.|s)$ converges to $\pi^i(.|s)$, we have $v^i_{\boldsymbol{\pi}_t}(s)$ and $v^{*i}_{\sigma,\boldsymbol{\pi}_t^{-i}}(s)$ that converges to $v^i_{\boldsymbol{\pi}}(s)$ for all $s \in S_n$ and finally $M^i_t(\hat{s}, a^i, a^-)$ and $M^{*i}_t(\hat{s}, a^i, a^-)$ converges to $M^i(\hat{s}, a^i, a^-)$. Then, lemma 8.2 and results of convergence of stochastic fictitious play from Hofbauer and Sandholm (2002) guarantees that the process in state $\hat{s}$ will converge to a smooth Nash equilibrium of the normal form game defined by $M^i(\hat{s}, a^i, \boldsymbol{a}^{-\boldsymbol{i}})$. Finally, we have $v^i_{\boldsymbol{\pi}}(\hat{s}) = v^{*i}_{\sigma,\boldsymbol{\pi}^{-i}}(\hat{s})$ and $v^i_{\boldsymbol{\pi}}(\hat{s}) = -v^{-i}_{\boldsymbol{\pi}}(\hat{s})$. □

13 Analysis of Actor-Critic Fictitious Play

The two on-line algorithms we present here (Algo 27) are stochastic approximations of the two dynamical systems (8.4) and (8.5). In an on-line setting, the interaction between players proceeds as follows. At a step n, players are in state s_n and individually take an action $a^i_n \sim \pi^i_n(.|s_n)$. Each of them receives a reward $r^i(s_n, a^1_n, \dots, a^N_n)$ and the process moves from state s_n to state $s_{n+1} \sim p(.|s_n, a^1_n, \dots, a^N_n)$. In addition, we consider the controlled Markov process $Z_n = (s_n, a^1_n, \dots, a^N_n, s_{n+1})$ (controlled by the strategy $\boldsymbol{\pi}_n$). Finally, if $s_{n+1} = \Omega$, we restart from state $\tilde{s}$ and $Z_{n+1} = (\tilde{s}, \dots)$. We define the following two-timescale processes (with α_n and β_n defined as in A 3):

$$\pi^i_{n+1} = \pi^i_n + \beta_n \mathbf{1}_{s_n} \mathbf{1}^\top_{s_n} \left[B_\sigma(Q^i_n(s_n, .)) - \pi^i_n(.|s_n)\right] \tag{8.21}$$

And:

$$Q^i_{n+1} = Q^i_n + \alpha_n \mathbf{1}_{(s_n,a^i_n)} \mathbf{1}^\top_{(s_n,a^i_n)} \left[r^i(s_n, \boldsymbol{a}_n) + C_\sigma \left(Q^i_n(s_{n+1}, .) \right) - Q^i_n(s_n, a^i_n) \right] \tag{8.22}$$

Or:

$$Q^i_{n+1} = Q^i_n + \alpha_n \mathbf{1}_{(s_n,a^i_n)} \mathbf{1}^\top_{(s_n,a^i_n)} \left[r^i(s_n, \boldsymbol{a}_n) + E_{b^i \sim \pi^i(.|s_{n+1})} \left(Q^i_n(s_{n+1}, b^i) \right) - Q^i_n(s_n, a^i_n) \right] \tag{8.23}$$

Assumption A 1 and assumption A 2 are verified. Regarding assumption A 4, the strategy π^i_n is obviously bounded since it remains in the simplex. If $Q^i_0 = 0$, the state-action value function is bounded as follows $\|Q^i_n(s,a)\| \leq \phi(s) R_{\max}$ (this can be shown by recursion). The two faster processes(Eq. (8.22) and (8.23)) track the two following ODE:

$$\dot{Q}^i_t(s,a^i) = d_{\boldsymbol{\pi}}(s)\pi^i(.|s) \left(\left[T^{*i}_{\sigma,\boldsymbol{\pi}^{-i}} Q^i_t \right](s,a^i) - Q^i_t(s,a^i) \right)$$

And

$$\dot{Q}^i_t(s,a^i) = d_{\boldsymbol{\pi}}(s)\pi^i(.|s) \left(\left[T^i_{\boldsymbol{\pi}} Q^i_t \right](s,a^i) - Q^i_t(s,a^i) \right)$$

Those two equations admit as an attractor $Q^{*i}_{\sigma,\boldsymbol{\pi}^{-i}}$ and $Q^i_{\boldsymbol{\pi}}$. Then, the strategy recursion follows either ODE (8.4) (if the subroutine is defined by Eq. (8.23)) or ODE (8.5) (if the subroutine is defined by (8.22)).

14 On the Guarantees of Convergence of OFF-SGSP and ON-SGSP

The main contribution to the field of MARL related to our work is the paper of Prasad & al Prasad et al. (2015). These algorithms are not decentralized but one of them, ON-SGSP, is an on-line and model-free algorithm. This paper proposes two algorithms OFF-SGSP and ON-SGSP which are stochastic approximations of a dynamical system described in section 8. The authors claims that these algorithms converges to a Nash equilibrium of the game. The proof of the stability given in lemma 11 is wrong. In the following we point out the issue with this proof.

Using their notations:

- G is the set of Nash equilibrium (the feasible set of the optimization problem),
- K is the limit set of the dynamical system (and $G \subset K$)
- K_1 is the set of limits points of the dynamical system which are Nash equilibrium $K \cap G$
- K_2 is the complementary of K_1 in K (i.e. $K_2 = K \backslash K_2$)

Lemma 11 shows that K_2 contains only unstable equilibrium and conclude that both processes converges to K_1 and thus to a Nash equilibrium since K_1 is not empty (this fact is proven early in the paper).

Unfortunately, that proof contains a mistake. The proof proceeds as follows: They show that if $\pi^* \in K_2$ then there exists a^i,x and i such that $g^i_{x,a^i}(\boldsymbol{v}^i_\pi, \pi^{-i}) > 0$ and they conclude that consequently $\frac{\partial f(\boldsymbol{v}_\pi,\pi)}{\partial \pi^i} < 0$. However, there is no direct link between the sine of $g^i_{x,a^i}(\boldsymbol{v}^i_\pi, \pi^{-i})$ and $\frac{\partial f(\boldsymbol{v}_\pi,\pi)}{\partial \pi^i}$ since $f(\boldsymbol{v}_\pi, \pi) = \sum_{i=1}^{N} \sum_{x\in S} \sum_{z\in A^i(x)} \pi^i(x,z) g^i_{x,z}(\boldsymbol{v}^i_\pi, \pi^{-i})$. This imply that both processes might converge in K_2 (i.e. not to a Nash equilibrium).

Part V

Conclusions and Future Work

1 Conclusion

This thesis is devoted to the study of learning in games from interaction data. We consider that the interaction can be modelled as a Markov Game (MG). This model is a generalization of the Markov Decision Process to the multi-agent setting and a generalization of normal form games to multi-state models. We studied two kinds of problems. The first one is the problem of learning from batch data, meaning that there is no way to collect new interaction data from the model. The second one is independent reinforcement learning where each agent plays independently one from an other without knowing the others behaviour.

In part II, we extend several ADP techniques from MDPs to zero-sum two-player MGs. In Chapter 3 we extend known ADP bounds for zero-sum two-player MGs from the $\mathcal{L}_{+\infty}$-norm to the $\mathcal{L}_p$-norm. Those ADP bounds are consistent with previous analysis of modified policy iteration in MDP. Based on these results, we propose a non-parametric batch algorithm to learn Nash equilibria in zero-sum two-player MGs. We tested this approach on the game of Alesia and Chapter 4 explores the use of non-stationary strategies to improve the guarantees of Chapter 3. To do so, we defined extensions of most ADP algorithms for zero-sum two-player MGs to the use of non-stationary strategies. These non-stationary algorithms improve both theoretically and empirically their stationary counterpart. The main improvement is to reduce the γ-sensitivity of the contribution of the error from $O\left(\frac{1}{(1-\gamma)^2}\right)$ to $O\left(\frac{1}{(1-\gamma)(1-\gamma^m)}\right)$. Furthermore, whilst the value iteration algorithm enjoys the worst guarantees, it is the one that performs the best in practice on randomly generated zero-sum two-player MGs. Chapter 5 investigates the use of non-stationary strategies in general-sum MGs as was suggested in previous work (Zinkevich et al., 2006). We prove that the γ-sensitivity of value iteration in general-sum MGs is improved by the use of non-stationary strategies but that it can't match the performance of value iteration in MDPs or in zero-sum two-player MGS. In the case of general-sum games, the only factor that counts is the length of the cycle and not the number of iterations.

Part III explores the Bellman residual approach to solve MDPs, zero-sum two-player MGs and general-sum MGs. In Chapter 6, we proved that a lot of existing batch algorithms to solve MDPs and zero-sum two player MGs could be seen as a Newton's method on some Bellman residuals. We leveraged this link and proposed the use of quasi-Newton methods to improve convergence at a small computation cost. Empirical evaluation on randomly generated MDPs and zero-sum two-player MGs show improvement both in stability and performance of our method compared to existing ones. However, this method can't be applied directly to general sum games. The next chapter (Chapter 7) generalizes this Bellman residual approach to general-sum MGs. This method relies on the minimization of two Bellman residuals per player and does not reduce to the one for MDPs and zero-sum two-player MGs. The minimization of the Bellman residual is not guaranteed. However, having a small Bellman residual guarantees having an ϵ-Nash equilibrium. This approach was empirically investigated on randomly generated MGs and shows promising results.

Part IV (Chapter 8) studies another scenario. Instead of being given a fixed amount of data we study an online learning problem. Here players must learn while they play in an independent manner. We proposed an independent RL algorithm that provably converges to zero-sum two-player Multi-Stage Games (MSGs) and cooperative MSGs and tested with function approximation.

2 Future Work

This thesis mainly contributes to the problem of learning from batch data and the algorithms we created and studied are based on Approximate Dynamic Programming and Bellman Residual Minimization. These approaches could be extended in other settings as in the MDP literature.

The first part of this dissertation studies batch learning using Approximate dynamic programming. But when a generative model of the game is provided, many other approaches can be used to perform each iteration of the inner loop of AGPI-Q. This inner loop is required to provide an approximate solution of a finite horizon MDP of length m. Using batch data restricts the number of possible methods (we only used approximate value iteration to solve that part). When a generative model of the game is provided, one can use a vast amount of methods. For instance, one could use Monte Carlo tree search to evaluate the best response of the computation.

We have seen in Chapter 3 that the use of value iteration gave the best asymptotic performance but that early convergence could be improved using large m. In that chapter, we suggest using large m at the beginning and reducing it after a while. Chapter 4 suggests using non-stationary strategies to improve the asymptotic performance of ADP methods. The value iteration algorithm using non-stationary strategies that had the best asymptotic performance but the early convergence is slowed down compared to stationary algorithms. Adapting AGPI-Q to the use of non-stationary strategies could improve early convergence while getting the same asymptotic performance.

In Chapter 6 we propose using a Bellman residual approach to learn from batch data. We proved that using quasi Newton methods on a Bellman residual is more stable and more accurate than using LSPI or BRMPI. These methods are hardly comparable to approximate dynamic programming methods in theory but an exhaustive empirical comparison can still be done and building a large benchmark of simultaneous move MGs where using stochastic strategies is compulsory, would definitely be a valuable contribution to that field of research.

We proved that LSPI could be improved using quasi Newton methods. And LSPI has been adapted to the online setting. In MDPs and in zero-sum two-player MGs, it could be interesting to investigate whether or not the use of second order methods could be used online to solve MGs. Many advances in stochastic second order methods have been made in recent years and it could be interesting to investigate its application to online learning in MDPs and in zero-sum two-player MGs.

In Chapter 7 we have generalized the Bellman residual approach to general-sum MGs but this method hardly scales when the actions are taken simultaneously. Future

work in this direction should focus on whether or not it is possible to only get a dependency on the action of one player instead of the joint action. One idea could be to use importance sampling if the policy used to collect the data is known.

Finally, the last part of this dissertation focused on independent learning in Multi Stage Games (MSGs). We could provide an algorithm that is guaranteed to converge to a smooth Nash equilibrium in the case of a zero-sum two-player MSG and in the case of a cooperative MSG. In future work, we plan to investigate whether or not, this process converges in MGs (without assuming a tree structure). Another way to improve that work would be to propose different kinds of algorithms using more general function approximations than state aggregation.

As a long term perspective, many areas of MARL remain to explore. In this thesis we assumed that the reward function was a parameter of the problem but this assumption is not always realistic. In RL, this problem is addressed with different approaches such as inverse reinforcement learning , imitation learning or apprenticeship learning. These three approaches remain superficially explored in the MARL literature compared to the state of the art in RL. In this manuscript, we only studied the batch case and the online learning case. We could explore how to leverage batch data or demonstrations of an optimal policy to accelerate the learning process.

Bibliography

Optimization and Nonsmooth Analysis. (→ page 107.)

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org. (→ page 131.)

N. Akchurina. Multiagent Reinforcement Learning: Algorithm Converging to Nash Equilibrium in General-Sum Discounted Stochastic Games. In *Proc. of AAMAS*, 2009. (→ page 38.)

A. Antos, C. Szepesvári, and R. Munos. Fitted-Q Iteration in Continuous Action-Space MDPs. In *Proc. of NIPS*, 2008a. (→ pages 18 and 21.)

A. Antos, C. Szepesvári, and R. Munos. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71, 2008b. (→ page 21.)

T. Archibald, K. McKinnon, and L. Thomas. On the Generation of Markov Decision Processes. *Journal of the Operational Research Society*, 46:354–361, 1995. (→ pages 71 and 128.)

J. A. Bagnell, S. M. Kakade, J. G. Schneider, and A. Y. Ng. Policy Search by Dynamic Programming. In *Proc. of NIPS*, page None, 2003. (→ page 68.)

L. Baird et al. Residual Algorithms: Reinforcement Learning with Function Approximation. In *Proc. of ICML*, 1995. (→ pages 17, 108, 117, and 125.)

B. Banerjee and J. Peng. Adaptive policy gradient in multiagent learning. In *Proc. AAMAS*. ACM, 2003. (→ page 39.)

R. Bellman. *Dynamic Programming*. Princeton University Press, 1957. (→ page 11.)

R. Bellman, R. Kalaba, and B. Kotkin. Polynomial approximation–a new computational technique in dynamic programming: Allocation processes. *Mathematics of Computation*, 17, 1963. (→ page 18.)

D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific Belmont, MA, 1995. (→ pages 91 and 93.)

V. Borkar. Stochastic Approximation with Two Time Scales. *Systems & Control Letters*, 29(5):291–294, 1997a. (→ page 38.)

V. Borkar. Stochastic approximation with two time scales. *Systems & Control Letters*, 1997b. (→ pages 146 and 148.)

V. S. Borkar. Stochastic approximation with 'controlled markov'noise. *Systems & control letters*, 2006. (→ pages 148 and 154.)

V. S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. 2009. (→ page 154.)

R. N. Borkovsky, U. Doraszelski, and Y. Kryukov. A user's guide to solving dynamic stochastic games using the homotopy method. *Operations Research*, 58, 2010. (→ page 37.)

B. Bošanský, V. Lisý, M. Lanctot, J. Čermák, and M. Winands. Algorithms for computing strategies in two-player simultaneous move games. *Artificial Intelligence*, 237: 1 – 40, 2016. (→ pages 38, 51, 141, and 150.)

M. Bowling and M. Veloso. Rational and Convergent Learning in Stochastic Games. In *Proc. of IJCAI*, 2001. (→ pages 39, 146, and 152.)

L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. CRC press, 1984. (→ page 52.)

S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012. (→ page 39.)

M. Buro. *Solving the Oshi-Zumo Game*, pages 361–366. Springer US, 2004. (→ pages 51 and 150.)

L. Busoniu, R. Babuska, and B. De Schutter. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 2008. (→ page 155.)

L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška. Online least-squares policy iteration for reinforcement learning control. In *American Control Conference (ACC), 2010*, 2010. (→ page 21.)

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA, 2006. (→ page 39.)

R. Correa and A. Seeger. Directional Derivative of a Minimax Function. *Nonlinear Analysis: Theory, Methods & Applications*, 9(1):13–22, 1985. (→ page 108.)

C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009. (→ page 26.)

L. M. Dermed and C. L. Isbell. Solving stochastic games. In *Advances in Neural Information Processing Systems*, 2009. (→ page 38.)

D. Ernst, P. Geurts, and L. Wehenkel. Tree-Based Batch Mode Reinforcement Learning. In *Journal of Machine Learning Research*, pages 503–556, 2005. (→ pages 18 and 91.)

A.-M. Farahmand, C. Szepesvári, and R. Munos. Error Propagation for Approximate Policy and Value Iteration. In *Proc. of NIPS*, 2010. (→ page 21.)

J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer Science & Business Media, 2012. (→ pages 29 and 124.)

J. A. Filar and B. Tolwinski. *On the Algorithm of Pollatschek and Avi-ltzhak*. Springer, 1991. (→ pages 32, 35, 105, 106, 112, and 120.)

V. Gabillon, A. Lazaric, M. Ghavamzadeh, and B. Scherrer. Classification-Based Policy Iteration with a Critic. In *Proc. of ICML*, pages 1049–1056, 2011. (→ page 21.)

I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. Book in preparation for MIT Press, 2016. (→ page 131.)

A. Greenwald, K. Hall, and R. Serrano. Correlated Q-learning. In *Proc. of ICML*, 2003. (→ page 39.)

S. Grunewalder, G. Lever, L. Baldassarre, M. Pontil, and A. Gretton. Modelling Transition Dynamics in MDPs With RKHS Embeddings. In *Proc. of ICML*, 2012. (→ pages 24, 110, and 127.)

T. D. Hansen, P. B. Miltersen, and U. Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *J. ACM*, 60(1), 2013a. (→ page 16.)

T. D. Hansen, P. B. Miltersen, and U. Zwick. Strategy Iteration is Strongly Polynomial for 2-Player Turn-Based Stochastic Games with a Constant Discount Factor. *JACM*, 60(1):1, 2013b. (→ pages 34 and 44.)

S. Hart and A. Mas-Colell. Uncoupled dynamics do not lead to nash equilibrium. *The American Economic Review*, 2003. (→ page 147.)

J. Heinrich, M. Lanctot, and D. Silver. Fictitious self-play in extensive-form games. In *Proc. of ICML*, 2015. (→ page 39.)

J.-J. Herings and R. J. Peeters. Stationary equilibria in stochastic games: Structure, selection, and computation. *Journal of Economic Theory*, 118, 2004. (→ page 37.)

P. J.-J. Herings and R. Peeters. Homotopy methods to compute equilibria in game theory. *Economic Theory*, 42, 2010. (→ page 37.)

J. Hofbauer and W. Sandholm. On the global convergence of stochastic fictitious play. *Econometrica*, 70(6):2265–2294, 2002. (→ pages 39, 40, 141, 145, 155, and 156.)

A. J. Hoffman and R. M. Karp. On nonterminating stochastic games. *Management Science*, 12(5):359–370, 1966. (→ page 32.)

J. Hu and M. P. Wellman. Nash Q-Learning for General-Sum Stochastic Games. *Journal of Machine Learning Research*, 4:1039–1069, 2003. (→ page 39.)

E. Kalai and E. Lehrer. Rational learning leads to nash equilibrium. *Econometrica: Journal of the Econometric Society*, 1993. (→ page 38.)

N. Karmarkar. A New Polynomial-time Algorithm for Linear Programming. In *Proc. of ACM Symposium on Theory of Computing*, 1984. (→ pages 27 and 51.)

M. Kearns, Y. Mansour, and S. Singh. Fast Planning in Stochastic Games. In *Proc. of UAI*, 2000. (→ pages 93 and 99.)

D. Koller and R. Parr. Policy Iteration for Factored MDPs. In *Proc. of UAI*, pages 326–334, 2000. (→ pages 109 and 110.)

D. Koller, N. Megiddo, and B. Von Stengel. Fast algorithms for finding randomized strategies in game trees. In *Proc. of STOC*. ACM, 1994. (→ page 27.)

M. G. Lagoudakis and R. Parr. Value Function Approximation in Zero-Sum Markov Games. In *Proc. of UAI*, 2002. (→ pages 36, 108, and 109.)

M. G. Lagoudakis and R. Parr. Least-Squares Policy Iteration. *Journal of Machine Learning Research*, pages 1107–1149, 2003. (→ pages 18, 19, 20, 109, and 111.)

M. Lanctot, K. Waugh, M. Zinkevich, and M. Bowling. Monte carlo sampling for regret minimization in extensive games. In *Proc. of NIPS*, 2009. (→ page 38.)

G. J. Laurent, L. Matignon, and N. Le Fort-Piat. The world of Independent learners is not Markovian. *International Journal of Knowledge-Based and Intelligent Engineering Systems*, 2011. (→ page 38.)

A. Lazaric, M. Ghavamzadeh, and R. Munos. Finite-sample analysis of least-squares policy iteration. *Journal of Machine Learning Research*, 13, 2012. (→ page 21.)

Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, 521:436–444, 2015. (→ page 128.)

D. Leslie and E. Collins. Generalised weakened fictitious play. *Games and Economic Behavior*, 56(2):285–298, 2006. (→ pages 39 and 141.)

B. Lesner and B. Scherrer. Non-stationary approximate modified policy iteration. 2015. (→ pages 16 and 23.)

A. S. Lewis and M. L. Overton. Nonsmooth Optimization via BFGS. *Submitted to SIAM J. Optimiz*, 2009. (→ pages 106, 107, and 112.)

A. S. Lewis and M. L. Overton. Nonsmooth Optimization via Quasi-Newton Methods. *Mathematical Programming*, 141(1-2):135–163, 2013. (→ pages 105, 106, 107, and 117.)

L. Li, M. L. Littman, and C. R. Mansley. Online exploration in least-squares policy iteration. In *Proc. of AAMAS*, 2009. (→ page 21.)

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous Control with Deep Reinforcement Learning. In *Proc. of ICLR*, 2016. (→ pages 128 and 131.)

M. L. Littman. Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *Proc. of ICML*, 1994. (→ pages 39 and 152.)

H. R. Maei, C. Szepesvári, S. Bhatnagar, and R. S. Sutton. Toward Off-Policy Learning Control with Function Approximation. In *Proc. of ICML*, pages 719–726, 2010. (→ page 108.)

O.-A. Maillard, R. Munos, A. Lazaric, and M. Ghavamzadeh. Finite-Sample Analysis of Bellman Residual Minimization. In *Proc. of ACML*, 2010. (→ pages 21, 24, 124, 127, and 128.)

C. Meyer, J. Ganascia, and J. Zucker. Learning Strategies in Games by Anticipation. In *Proc. of IJCAI 97, August 23-29, 1997, 2 Volumes*, pages 698–707, 1997. (→ page 51.)

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-Level Control Through Deep Reinforcement Learning. *Nature*, 518:529–533, 2015. (→ page 128.)

R. Munos. Performance Bounds in L_p-norm for Approximate Value Iteration. *SIAM Journal on Control and Optimization*, 46(2):541–561, 2007. (→ pages 21 and 91.)

R. Munos and C. Szepesvári. Finite-Time Bounds for Fitted Value Iteration. *The Journal of Machine Learning Research*, 9:815–857, 2008. (→ page 124.)

N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*, volume 1. Cambridge University Press Cambridge, 2007. (→ page 26.)

J. Nocedal and S. Wright. *Numerical Optimization*. Springer Science & Business Media, 2006. (→ pages 106, 107, 116, and 117.)

S. D. Patek. *Stochastic Shortest Path Games: Theory and Algorithms*. PhD thesis, Massachusetts Institute of Technology, Laboratory for Information and Decision Systems, 1997. (→ pages 31, 32, 33, 49, and 53.)

J. Perolat, B. Scherrer, B. Piot, and O. Pietquin. Approximate Dynamic Programming for Two-Player Zero-Sum Markov Games. In *Proc. of ICML*, 2015. (→ pages 43, 51, 91, and 93.)

J. Perolat, B. Piot, B. Scherrer, and O. Pietquin. On the use of non-stationary strategies for solving two-player zero-sum markov games. In *Proc. of AISTATS*, 2016. (→ pages 63 and 129.)

B. Piot, M. Geist, and O. Pietquin. Difference of convex functions programming for reinforcement learning. In *Proc. of NIPS*, 2014a. (→ pages 23, 24, 108, 110, 117, 124, 125, 127, 128, 134, and 135.)

B. Piot, M. Geist, and O. Pietquin. Boosted Bellman Residual Minimization Handling Expert Demonstrations. In *Proc. of ECML*, 2014b. (→ pages 24 and 127.)

M. Pollatschek and B. Avi-Itzhak. Algorithms for Stochastic Games with Geometrical Interpretation. *Management Science*, 15(7):399–415, 1969. (→ page 32.)

H. Prasad, P. LA, and S. Bhatnagar. Two-Timescale Algorithms for Learning Nash Equilibria in General-Sum Stochastic Games. In *Proc. of AAMAS*, 2015. (→ pages 38 and 157.)

M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994. (→ pages 11, 12, 14, 15, 17, 53, and 114.)

M. Riedmiller. Neural Fitted Q Iteration–First Experiences with a Data Efficient Neural Reinforcement Learning Method. In *Proc. of ECML*. 2005. (→ page 18.)

J. Robinson. An iterative method of solving a game. *Annals of mathematics*, pages 296–301, 1951. (→ page 39.)

B. Scherrer. Approximate Policy Iteration Schemes: A Comparison. In *Proc. of ICML*, 2014. (→ pages 23, 66, 67, and 68.)

B. Scherrer. Improved and generalized upper bounds on the complexity of policy iteration. *Mathematics of Operations Research*, 41(3), 2016. (→ page 16.)

B. Scherrer and B. Lesner. On the Use of Non-Stationary Policies for Stationary Infinite-Horizon Markov Decision Processes. In *Proc. of NIPS*, 2012. (→ pages 12, 23, 68, 70, 71, 92, and 99.)

B. Scherrer, M. Ghavamzadeh, V. Gabillon, and M. Geist. Approximate Modified Policy Iteration. In *Proc. of ICML*, 2012. (→ pages 21, 45, 46, 48, 53, 55, 56, 58, 81, 82, and 124.)

L. S. Shapley. Stochastic Games. *In Proc. of the National Academy of Sciences of the United States of America*, 1953. (→ pages 11, 32, and 93.)

L. S. Shapley. Some topics in two-person games. *Advances in game theory*, 1964. (→ page 147.)

Y. Shoham and K. Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008. (→ page 26.)

Y. Shoham, R. Powers, and T. Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365 – 377, 2007. (→ pages 39 and 141.)

G. Taylor and R. Parr. Value Function Approximation in Noisy Environments Using Locally Smoothed Regularized Approximate Linear Programs. In *Proc. of UAI*, 2012. (→ pages 24 and 128.)

J. Van Der Wal. Discounted Markov Games: Generalized Policy Iteration Method. *Journal of Optimization Theory and Applications*, 25(1):125–138, 1978. (→ pages 32, 33, 34, 35, 106, and 110.)

C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992. (→ pages 24, 39, and 152.)

M. Zinkevich, A. Greenwald, and M. Littman. Cyclic Equilibria in Markov Games. In *Proc. of NIPS*, 2006. (→ pages 78, 91, 92, 93, 117, 123, 125, 129, and 161.)

M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. In *Proc. of NIPS*, 2008. (→ page 27.)