On the Use of Non-Stationary Strategies for Solving Two-Player Zero-Sum Markov Games

Julien Pérolat[(1)] julien.perolat@ed.univ-lille1.fr
Bilal Piot[(1)] bilal.piot@univ-lille1.fr
Bruno Scherrer[(2)] bruno.scherrer@inria.fr
Olivier Pietquin[(1,3)] olivier.pietquin@univ-lille1.fr
[(1)]Univ. Lille, CNRS, Centrale Lille, Inria UMR 9189 - CRIStAL, F-59000 Lille, France
[(2)]Inria, Villers-lès-Nancy, F-54600, France
[(3)]Institut Universitaire de France (IUF), France

Abstract

The main contribution of this paper consists in extending 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 algorithms are shown to exhibit better performance bounds compared to their stationary counterparts. The obtained bounds are generically composed of three terms: 1) a dependency over γ (discount factor), 2) a concentrability coefficient and 3) a propagation error term. This error, depending on the algorithm, can be caused by a regression step, a policy evaluation step or a best-response evaluation step. As a second contribution, we empirically demonstrate, on generic MGs (called Garnets), that non-stationary algorithms outperform their stationary counterparts. In addition, it is shown that their 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 (even if they exhibit better concentrability coefficients and dependencies on γ) compared to those suffering from a regression error.

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 41.

1 Introduction

Because of its potential application to a wide range of complex problems, Multi-Agent Reinforcement Learning (MARL) [6] has recently been the subject of increased attention. Indeed, MARL aims at controlling systems composed of distributed autonomous agents, encompassing problems such as games (chess, checkers), human-computer interfaces design, load balancing in computer networks *etc.*. MARL extends, to a multi-agent setting, the Reinforcement Learning (RL) paradigm [4] in which single-agent control problems (or games against nature) are modeled as Markov Decision Processes (MDPs) [13]. Markov Games (MGs) (also called Stochastic Games (SGs)) [17] extend MDPs to the multi-agent setting and model MARL problems.

The focus of this work is on a special case of MGs, namely γ-discounted two-player zero-sum MGs, where the benefit of one agent is the loss of the other. In this case, the solution takes the form of a Nash equilibrium. As in the case of MDPs, Dynamic Programming (DP) is a family of methods relying on a sequence of policy evaluation and improvement steps that offers such solutions [5, 11]. When the scale of the game becomes too large or when its dynamics is unknown, DP becomes intractable or even inapplicable. In these situations, Approximate Dynamic Programming (ADP) [5, 8, 12] becomes more appropriate. ADP is inspired by DP but uses approximations (during the evaluation and/or the improvement step) at each iteration. Approximation errors thus accumulate over successive iterations.

Error propagation occurring in ADP has been first studied in the MDP framework in L_∞-norm [5]. However, the approximation steps in ADP are, in practice, implemented by Supervised Learning (SL) methods [7]. As SL errors (such as regression and classification errors) are not usually controlled in L_∞-norm

but rather in L_p-norm, the bounds from [5] have been extended to L_p-norm in [9, 10, 1]. It is well known, for γ-discounted MDPs, that the way errors propagate depends on $\frac{2\gamma C\epsilon}{(1-\gamma)^2}$. The final error is thus divided in three terms, a dependency over γ (in this case $\frac{2\gamma}{(1-\gamma)^2}$), a dependency over some concentrability coefficient (C) and a dependency over an error ϵ. This error ϵ comes from various sources: for the Value Iteration (VI) algorithm the error ϵ comes from a supervised learning problem for both MDPs and MGs. For the Policy Iteration (PI) algorithm, the error comes from a policy evaluation problem in the case of MDPs and from a full control problem in the case of MGs (namely evaluation of the best response of the opponent). Because γ is often close to 1 in practice, reducing algorithms sensitivity to γ is also crucial. Thus, various algorithms for MDPs intend to improve one or several terms of this error bound. The Conservative Policy Iteration (CPI) algorithm improves the concentrability coefficient [16], both Non-Stationary Value Iteration (NSVI) and Non-Stationary Policy Iteration (NSPI) improve the dependency over γ (from $\frac{1}{(1-\gamma)^2}$ to $\frac{1}{(1-\gamma)(1-\gamma^m)}$ where m is the length of the non-stationary policy considered) and Policy Search by Dynamic Programming (PSDP) improves the dependency over both.

This paper introduces generalizations to MGs of NSPI, NSVI and PSDP algorithms. CPI is not studied here since its generalization to MGs appears trickier[1]. The main contribution of the paper thus consists in generalizing several non-stationary RL algorithms known for γ-discounted MDPs to γ-discounted two-player zero-sum MGs. In addition, we extend the performance bounds of these algorithms to MGs. Thanks to those bounds, the effect of using non-stationary strategies on the error propagation control is demonstrated. However, analyses are conservative in the sense that they consider a worst case propagation of error and a best response of the opponent. Because such a theoretical worst case analysis does not account for the nature of the error, each algorithm is empirically tested on generic SGs (Garnets). Experiments show that non-stationary strategies always lead to improved average performance and standard deviation compared to their stationary counterparts. Finally, a comparison on randomly generated MGs between non-stationarity-based algorithms shows that, given a fixed budget of samples, NSVI outperforms all others schemes. Indeed, even if the other algorithms exhibit better concentrability coefficients and a better dependency w.r.t. γ, a simple regression empirically produces smaller errors than evaluating a policy or a best response. Therefore, as a second contribution, experiments suggest that the nature of the error does matter when choosing an algorithm, an issue that was ignored in previous research [16].

[1]In MDPs, CPI exploits the fact that the value function of some policy is a differentiable function of the (stochastic) policy. In a SG, differentiability does not hold anymore in general.

The rest of this paper is organized as follows: first standard notations for two-player zero-sum MGs are provided in Section 2. Then extensions of NSVI, NSPI and PSDP are described and analyzed in Section 3. The dependency w.r.t. γ, the concentrability coefficient and the error of each approximation scheme are discussed. Finally, results of experiments comparing algorithms on generic MDPs and MGs, are reported in Section 4. They highlight the importance of controlling the error at each iteration.

2 Background

This section reviews standard notations for two-players MGs. In MGs, unlike standard MDPs, players simultaneously choose an action at each step of the game. The joint action of both players generates a reward and a move to a next state according to the game dynamics.

A two-player zero-sum MG is a tuple $(S, (A^1(s))_{s\in S}, (A^2(s))_{s\in S}, p, r, \gamma)$ in which S is the state space, $(A^1(s))_{s\in S}$ and $(A^2(s))_{s\in S}$ are the sets of actions available to each player in state $s \in S$, $p(s'|s, a^1, a^2)$ is the Markov transition kernel which models the game dynamics, $r(s, a^1, a^2)$ is the reward function which represents the local benefit of doing actions (a^1, a^2) in state s and γ is the discount factor. A strategy μ (resp. ν) associates to each $s \in S$ a distribution over $A^1(s)$ (resp. $A^2(s)$). We note $\mu(.|s)$ (resp. $\nu(.|s)$) such a distribution. In the following, μ (resp ν) is thus the strategy of Player 1 (resp. Player 2). For any pair of stationary strategies μ and ν, let us define the corresponding stochastic transition kernel $\mathcal{P}_{\mu,\nu}(s'|s) = E_{a^1\sim\mu(.|s),a^2\sim\nu(.|s)}[p(s'|s, a^1, a^2)]$ and the reward function $r_{\mu,\nu} = E_{a^1\sim\mu(.|s),a^2\sim\nu(.|s)}[r(s, a^1, a^2)]$. The value function $v_{\mu,\nu}(s)$, measuring the quality of strategies μ and ν, is defined as the expected cumulative γ-discounted reward starting from s when players follow the pair of strategies (μ,ν).

$$v_{\mu,\nu}(s) = E[\sum_{t=0}^{+\infty} \gamma^t r_{\mu,\nu}(s_t)|s_0 = s,\ s_{t+1} \sim \mathcal{P}_{\mu,\nu}(.|s_t)].$$

The value $v_{\mu,\nu}$ is the unique fixed point of the following linear Bellman operator:

$$\mathcal{T}_{\mu,\nu} v = r_{\mu,\nu} + \gamma \mathcal{P}_{\mu,\nu} v.$$

In a two-player zero-sum MG, Player 1's goal is to maximize his value whereas Player 2 tries to

minimize it. In this setting, it is usual to define the following non-linear Bellman operators [11, 12]:

$$\mathcal{T}_\mu v = \min_\nu \mathcal{T}_{\mu,\nu} v \qquad \hat{\mathcal{T}}_\nu v = \max_\mu \mathcal{T}_{\mu,\nu} v$$
$$\mathcal{T} v = \max_\mu \mathcal{T}_\mu v \qquad \hat{\mathcal{T}} v = \min_\nu \hat{\mathcal{T}}_\nu v$$

The value $v_\mu = \min_\nu v_{\mu,\nu}$, fixed point of $\mathcal{T}_\mu$, is the value Player 1 can expect while playing strategy μ and when Player 2 plays optimally against it. Strategy ν is an optimal counter strategy when $v_\mu = v_{\mu,\nu}$. Player 1 will try to maximize value v_μ. In other words she will try to find $v^* = \max_\mu v_\mu = \max_\mu \min_\nu v_{\mu,\nu}$ the fixed point of operator $\mathcal{T}$. Von Neumann's minimax theorem [18] ensures that $\mathcal{T} = \hat{\mathcal{T}}$, thus $v^* = \max_\mu \min_\nu v_{\mu,\nu} = \min_\nu \max_\mu v_{\mu,\nu}$ which means that v^* is the value both players will reach by playing according to the Nash Equilibrium. We will also call v^* the *optimal value* of the game.

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

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

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 $t = 1$ s/he will play μ_1 and at time $t = Mj + i$ ($\forall i \in \{0, ..., M-1\}, \forall j \in N$) s/he will play strategy μ_i. This value is the fixed point of the operator $\mathcal{T}_{\mu_0} \ldots \mathcal{T}_{\mu_{M-1}}$ (proof in appendix A). Thus, we have:

$$v_{\mu_0,\mu_1,\ldots,\mu_{M-1}} = \mathcal{T}_{\mu_0} \ldots \mathcal{T}_{\mu_{M-1}} v_{\mu_0,\mu_1,\ldots,\mu_{M-1}}. \quad (1)$$

3 Algorithms

This section presents extensions to two-player zero-sum MGs of three non-stationary algorithms, namely PSDP, NSVI and NSPI, known for improving the error bounds for MDPs. For MDPs, PSDP is known to have the best concentrability coefficient [16], 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.

3.1 Value Iteration and Non-Stationary Value Iteration

Formally, VI consists in repetitively applying the optimal Bellman operator $\mathcal{T}$ starting from an initial value v_0 (usually set to 0). This process can be divided in two steps. First, a greedy step where the strategy of the maximizer is improved from μ_{k-1} to μ_k. Then, the algorithm updates the value function from v_{k-1} to v_k. Each of these two steps are prone to error, *i.e.* to a greedy error ϵ'_k and an evaluation error ϵ_k. For simplicity, in the main body of this paper we only consider an evaluation error ($\epsilon'_k = 0$). The general case is considered in appendix B:

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

Because those errors propagate from one iteration to the next, the final strategy may be far from optimal. To measure the performance of such an algorithm, one wants to bound (according to some norm) the distance between the optimal value and the value of the final strategy when the opponent is playing optimally. An upper bound for this error propagation has been computed in [11] in L_∞-norm and in [12] in L_p-norm for stationary strategies. Moreover, this bound has been shown to be tight for MDPs in [15]. Since MDPs are a subclass of MGs, the L_∞-norm bound is also tight for MGs.

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, simply consists in playing the m last strategies generated by VI for MGs. In the following, we provide a bound in L_p-norm for NSVI in the framework of zero-sum two-player MGs. To our knowledge, this is an original result. The goal is to bound the difference between the optimal value v^* and the value $v_{k,m} = v_{\mu_k,\mu_{k-1},\ldots,\mu_{k-m+1}}$ of the m last strategies generated by VI. 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 $\|.\|_{p,\rho}$ where $\|f\|_{p,\rho} = \left(\sum_{s\in S} |f(s)|^p \rho(s)\right)^{\frac{1}{p}}$.

The following theorem provides a performance guarantee in L_p-norm:

Theorem 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 strategy of size m and after k iterations we have:*

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

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:

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

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

Remark 1. *First, we can notice the full bound (appendix B) matches the bound on stationary strategies in L_p-norm of [12] for the first and second terms. It also matches the one in L_∞-norm for non-stationary policies of [15] in the case of MDPs.*

Remark 2. *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.*

3.2 Policy Search by Dynamic Programming (PSDP)

PSDP was first introduced in [3] 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 [16]. When applied to MDPs, this algorithm enjoys the best concentrability coefficient among several algorithms based on policy iteration, namely NSPI, CPI, API and NSPI(m) (see [16] 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.

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 $v_{\sigma_{k-1}} = \mathcal{T}_{\mu_{k-1}} ... \mathcal{T}_{\mu_0} 0 + \epsilon_{k-1}$. Following any non-stationary strategy that uses $\sigma_k (= \mu_k, ..., \mu_0)$ for the $k+1$ first steps, we have the following performance guarantee:

Theorem 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:*

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

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:

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

Proof. The full proof is left in appendix C.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 $\mathcal{T}_{\mu_{k-1}} ... \mathcal{T}_{\mu_0} 0$ 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.

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

Theorem 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:*

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

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 C.1. □

Remark 3. *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.*

Remark 4. *An issue with PSDP1 and PSDP2 is 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.*

Remark 5. *One can notice that* $\forall j \in N\ c_{\mu*,q}(j) \leq c_q(j)$ *and thus* $\mathcal{C}^{l,k,d}_{\mu^*,q} \leq \mathcal{C}^{l,k,d}_q$. *Then the concentrability coefficient of PSDP2 is worse than PSDP1's. Moreover,* $\mathcal{C}^{l,k,d,m}_q = (1-\gamma^m)\mathcal{C}^{l,k,d}_q + \gamma^m \mathcal{C}^{l,k,d+m,m}_q$ *meaning intuitively that* $\mathcal{C}^{l,k,d}_q$ *is* $\mathcal{C}^{l,k,d,m}_q$ *when* m *goes to infinity. This also means that if* $\mathcal{C}^{l,k,d}_q = \infty$, *then we have* $\mathcal{C}^{l,k,d,m}_q = \infty$. *In that sense, one can argue that PSDP2 offers a better concentrability coefficient than NSVI.*

3.3 Non Stationary Policy Iteration (NSPI(m))

Policy iteration (PI) is one of the most standard algorithms used for solving MDPs. Its approximate version 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 in [15]. 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 SGs, NSPI(m) estimates the value of the best response to the last m strategies.

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 $\mu_{k,m} = \mu_k, ..., \mu_{k-m+1}$. Then it moves to a new strategy μ_{k+1} satisfying $\mathcal{T} v_k = \mathcal{T}_{\mu_{k+1}} v_k$.

Theorem 4. *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:*

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

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

Remark 6. *The NSPI dependency over* γ *and the concentrability coefficient involved in the NSPI bound are the same as those found for NSVI. However, in the MDP case the policy evaluation error is responsible for the error* ϵ_k *and in the SG case the error comes from solving a full control problem.*

3.4 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.

4 Experiments

The previous section provided, for each new algorithm, a performance bound that assumes a worst case error propagation. Examples that suggest that the bound is tight were provided in [15] for MDPs (but as a lower bound, they also apply to SGs) in the case of L_∞ analysis ($p = \infty$). Those specific examples are pathological problems and, in practice, the bounds will generally be conservative. Furthermore, our analysis 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 [2].

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 $\{s_1, ..., s_{N_B}\}$ 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: $\forall i \in \{1, ..., N_B\},\ p(s_i|s,a) = p_i - p_{i-1}$. 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:

$$Q_{\mu,\nu}(s, a, b) = r(s, a, b) + \sum_{s' \in S} p(s'|s, a, b) v_{\mu,\nu}(s').$$

The analysis of previous section still holds since one can consider an equivalent (but larger) SG 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 $Q_0(s, a, b)$ where a and b are respectively actions of player 1 and player 2. At each iteration we draw uniformly over the state-actions space (s^i, a^i, b^i) then we compute $r^i = r(s^i, a^i, b^i)$ and draw $s'^i \sim p(.|s^i, a^i, b^i)$ for $i \in \{1, ..., N_k\}$. Then we compute $q^i = r^i + \gamma \min_b E_{a \sim \mu_k(.|s'^i)}[Q_k(s'^i, a, b)]$. The next state-actions value function Q_{k+1} is the best fit over the training dataset $\{(s^i, a^i, b^i), q^i\}_{i \in \{1,...,N_k\}}$. In all experiments on NSVI, all samples are refreshed after each iteration. The first advantage of using non-

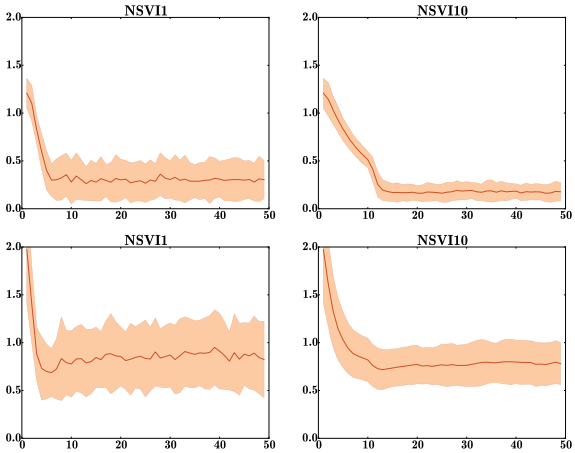

Figure 1: Performance (y-axis) of the strategy at step k (x-axis) for NSVI for a strategy of length 10 (right) and length 1 (left). Results are averaged over 70 Garnets $N_S = 100$, $N_A = 5$, $N_B = 1$ (top) and $N_B = 2$ (bottom). All Garnets have a sparsity of 0.5 and $\gamma = 0.9$. Each step of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

stationary strategies in VI is the reduction of the standard deviation of the value $v_{\mu_k,...,\mu_{k-m+1}}$. Figure 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 greater m is the more the performance concentrates (the parameter m is varied in figure 1, 4, 5, 6 and 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, since we are mixing last m strategies, the asymptotic performance is reached after more iterations (see Figure 1).

PSDP2 In practice PSDP2 builds N_k rollout $\{(s_i^j, a_i^j, b_i^j, r_i^j)\}_{i \in \{0,...,k+1\}}$ at iteration k. Where (s_0^j, a_0^j, b_0^j) are drawn uniformly over the state-actions space and where $s_{i+1}^j \sim p(.|s_i^j, a_i^j, b_i^j)$, $a_{i+1}^j \sim \mu_{k-i}(.|s_{i+1}^j)$, $b_{i+1}^j \sim \nu_{k-i}(.|s_{i+1}^j)$ and $r_{i+1}^j = r(s_{i+1}^j, a_{i+1}^j, b_{i+1}^j)$. Then we build the dataset $\{(s_0^j, a_0^j, b_0^j), \sum_{i=0}^{k+1} \gamma^i r_i^j\}_{j \in \{0,...,N_k\}}$. 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 min-max 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 2 and 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_0^j, b_0^j). In figure 2 the two top

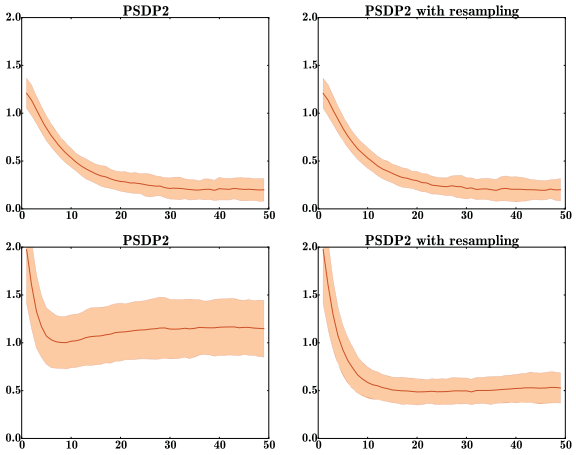

Figure 2: Performance (y-axis) of the strategy of length $k+1$ at step k (x-axis) for PSDP2 without resampling (left) with a 10 times resampling (right). Results are averaged over 70 Garnets $N_S = 100$, $N_A = 5$, $N_B = 1$ (top) and $N_B = 2$ (bottom). All Garnets have a sparsity of 0.5 and $\gamma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

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 $v_{\sigma_k} = \mathcal{T}_{\mu_k} ... \mathcal{T}_{\mu_0} 0$). 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 $\mu_k, ..., \mu_{k-m+1}$. 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 $\mu_k, ..., \mu_{k-m+1}$ is done by approximately applying in a cycle $\mathcal{T}_{\mu_k}$, ..., $\mathcal{T}_{\mu_{k-m+1}}$. For MG, the subroutine will contain $l \times m$ iterations. At iteration $p \in \{1, ..., l \times m\}$ 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 $\mu_k, ..., \mu_1$ (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 3 shows also an improvement of the average performance as m grows.

5 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

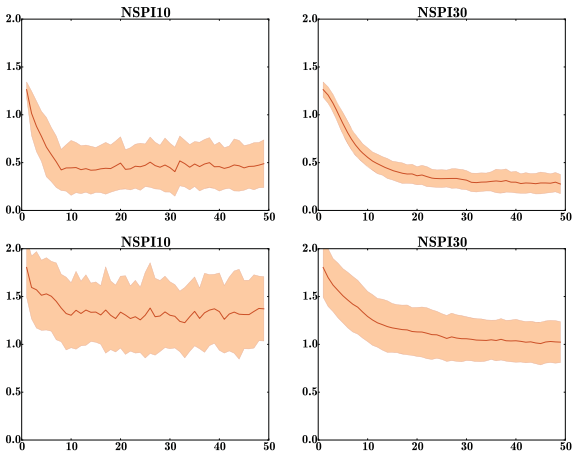

Figure 3: Performance (y-axis) of the strategy at step k (x-axis) for NSPI for a strategy of length 10 (left) and 30 (right). results are averaged over 70 Garnets $N_S = 100$, $N_A = 8$, $N_B = 1$ (top) and $N_B = 2$ (bottom). All Garnets have a sparsity of 0.5 and $\gamma = 0.9$. Each step of the algorithm uses $2.25 \times N_A \times N_S$ samples.

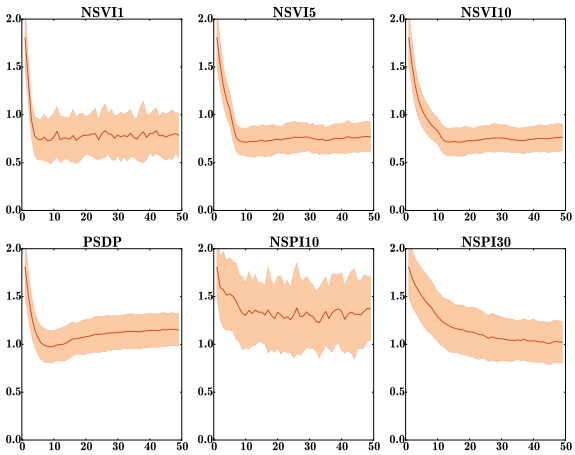

Figure 4: Performance (y-axis) of the strategy at step k (x-axis) for NSVI, PSDP and NSPI. Results are averaged over 70 Garnets $N_S = 100$, $N_A = 8$, $N_B = 1$. All Garnets have a sparsity of 0.5 and $\gamma = 0.9$. Each step of NSPI and NSVI uses $2.25 \times N_A \times N_S$ samples. Each step of PSDP2 uses $2.25 \times N_A \times N_S$ rollouts.

the memory issue of PSDP1 and PSDP2. The dependency to γ of the error bound is reduced from roughly $\frac{1}{1-\gamma}$ 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.

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 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 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 6 in the appendix).

6 Conclusion

This paper generalizes several algorithms using non-stationary strategies to the setting of γ-discounted zero-sum two-player MGs. 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 over 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.

In some sense, running a non-stationary strategy instead of a stationary one sounds like averaging over last strategies. Several techniques are based on averaging last policies, for instance CPI. But this generic idea is not new. For example in optimization, when using stochastic gradient descent one knows he has to return the average of the sequence of parameter outputted by the algorithm instead of last one. From a theoretical point of view, an interesting perspective for this work would be to figure out whether or not there is a general formulation of this intuitive idea. It would be an interesting start to go beyond the worst case analysis of each algorithm described in this paper.

From a practical point of view, trying to learn in large scale games with algorithms producing non-stationary strategies would be a nice perspective for this work especially in deterministic games like checkers or awalé.

Acknowledgements

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see `https://www.grid5000.fr`).

References

[1] A. Antos, C. Szepesvári, and R. Munos. Fitted-Q Iteration in Continuous Action-Space MDPs. In *Proc. of NIPS*, 2008.

[2] TW Archibald, KIM McKinnon, and LC Thomas. On the generation of markov decision processes. *Journal of the Operational Research Society*, pages 354–361, 1995.

[3] J Andrew Bagnell, Sham M Kakade, Jeff G Schneider, and Andrew Y Ng. Policy search by dynamic programming. In *Proc. of NIPS*, page None, 2003.

[4] A. G. Barto. *Reinforcement Learning: An Introduction*. MIT press, 1998.

[5] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific Belmont, MA, 1995.

[6] 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.

[7] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, pages 1107–1149, 2003.

[8] Michail G Lagoudakis and Ronald Parr. Value function approximation in zero-sum markov games. In *Proc. of UAI*, 2002.

[9] R. Munos. Performance bounds in L_p-norm for approximate value iteration. *SIAM Journal on Control and Optimization*, 46(2):541–561, 2007.

[10] Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *The Journal of Machine Learning Research*, pages 815–857, 2008.

[11] S. D. Patek. *Stochastic Shortest Path Games: Theory and Algorithms*. PhD thesis, Massachusetts Institute of Technology, Laboratory for Information and Decision Systems, 1997.

[12] Julien Perolat, Bruno Scherrer, Bilal Piot, and Olivier Pietquin. Approximate Dynamic Programming for Two-Player Zero-Sum Markov Games. In *Proc. of ICML*, 2015.

[13] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[14] B. Scherrer, M. Ghavamzadeh, V. Gabillon, and M. Geist. Approximate Modified Policy Iteration. In *Proc. of ICML*, 2012.

[15] B. Scherrer and B. Lesner. On the Use of Non-Stationary Policies for Stationary Infinite-Horizon Markov Decision Processes. In *Proc. of NIPS*, 2012.

[16] Bruno Scherrer. Approximate Policy Iteration Schemes: A Comparison. In *Proc. of ICML*, 2014.

[17] L. S. Shapley. Stochastic Games. *Proceedings of the National Academy of Sciences of the United States of America*, 1953.

[18] J. Von Neumann. Morgenstern, 0.(1944) theory of games and economic behavior. *Princeton: Princeton UP*, 1947.