Approximate Modified Policy Iteration

Bruno Scherrer, Mohammad Ghavamzadeh, Victor Gabillon, Matthieu Geist

Approximate Modified Policy Iteration

Bruno Scherrer BRUNO.SCHERRER@INRIA.FR
INRIA Nancy - Grand Est, Team Maia, FRANCE

Mohammad Ghavamzadeh MOHAMMAD.GHAVAMZADEH@INRIA.FR
Victor Gabillon VICTOR.GABILLON@INRIA.FR
INRIA Lille - Nord Europe, Team SequeL, FRANCE

Matthieu Geist MATTHIEU.GEIST@SUPELEC.FR
Suplec, IMS Research Group, Metz, FRANCE

Abstract

Modified policy iteration (MPI) is a dynamic programming (DP) algorithm that contains the two celebrated policy and value iteration methods. Despite its generality, MPI has not been thoroughly studied, especially its approximation form which is used when the state and/or action spaces are large or infinite. In this paper, we propose three implementations of approximate MPI (AMPI) that are extensions of well-known approximate DP algorithms: fitted-value iteration, fitted-Q iteration, and classification-based policy iteration. We provide error propagation analysis that unifies those for approximate policy and value iteration. For the classification-based implementation, we develop a finite-sample analysis that shows that MPI's main parameter allows to control the balance between the estimation error of the classifier and the overall value function approximation.

1. Introduction

Modified Policy Iteration (MPI) (Puterman & Shin, 1978) is an iterative algorithm to compute the optimal policy and value function of a Markov Decision Process (MDP). Starting from an arbitrary value function v_0, it generates a sequence of value-policy pairs

$$\pi_{k+1} = \mathcal{G}\, v_k \qquad \text{(greedy step)} \qquad (1)$$

$$v_{k+1} = (T_{\pi_{k+1}})^m v_k \qquad \text{(evaluation step)} \qquad (2)$$

where $\mathcal{G}\, v_k$ is a *greedy* policy w.r.t. v_k, T_{π_k} is the Bellman operator associated to the policy π_k, and $m \geq 1$ is a parameter. MPI generalizes the well-known dynamic programming algorithms Value Iteration (VI) and Policy Iteration (PI) for values $m = 1$ and $m = \infty$, respectively. MPI has less computation per iteration than PI (in a way similar to VI), while enjoys the faster convergence of the PI algorithm (Puterman & Shin, 1978). In problems with large state and/or action spaces, approximate versions of VI (AVI) and PI (API) have been the focus of a rich literature (see e.g., Bertsekas & Tsitsiklis 1996; Szepesvári 2010). The aim of this paper is to show that, similarly to its exact form, approximate MPI (AMPI) may represent an interesting alternative to AVI and API algorithms.

In this paper, we propose three implementations of AMPI (Sec. 3) that generalize the AVI implementations of Ernst et al. (2005); Antos et al. (2007); Munos & Szepesvári (2008) and the classification-based API algorithm of Lagoudakis & Parr (2003); Fern et al. (2006); Lazaric et al. (2010); Gabillon et al. (2011). We then provide an error propagation analysis of AMPI (Sec. 4), which shows how the L_p-norm of its performance loss can be controlled by the error at each iteration of the algorithm. We show that the error propagation analysis of AMPI is more involved than that of AVI and API. This is due to the fact that neither the contraction nor monotonicity arguments, that the error propagation analysis of these two algorithms rely on, hold for AMPI. The analysis of this section unifies those for AVI and API and is applied to the AMPI implementations presented in Sec. 3. We detail the analysis of the classification-based implementation of MPI (CBMPI) of Sec. 3 by providing its finite sample analysis in Sec. 5. Our analysis indicates that the parameter m allows us to balance the estimation error of the classifier with the overall quality of the value

Appearing in *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012.

approximation. We report some preliminary results of applying CBMPI to standard benchmark problems and comparing it with some existing algorithms in Appendix. G.

2. Background

We consider a discounted MDP $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is a finite action space, $P(ds'|s,a)$, for all (s,a), is a probability kernel on $\mathcal{S}$, the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is bounded by $R_{\max}$, and $\gamma \in (0,1)$ is a discount factor. A deterministic policy is defined as a mapping $\pi : \mathcal{S} \to \mathcal{A}$. For a policy π, we may write $r_\pi(s) = r\big(s, \pi(s)\big)$ and $P_\pi(ds'|s) = P\big(ds'|s, \pi(s)\big)$. The value of policy π in a state s is defined as the expected discounted sum of rewards received starting from state s and following the policy π, i.e., $v_\pi(s) = \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t r_\pi(s_t) |\, s_0 = s, s_{t+1} \sim P_\pi(\cdot|s_t)\big]$. Similarly, the action-value function of a policy π at a state-action pair (s,a), $Q_\pi(s,a)$, is the expected discounted sum of rewards received starting from state s, taking action a, and then following the policy. Since the rewards are bounded by $R_{\max}$, the values and action-values should be bounded by $V_{\max} = Q_{\max} = R_{\max}/(1-\gamma)$. The Bellman operator T_π of policy π takes a function f on $\mathcal{S}$ as input and returns the function $T_\pi f$ defined as $\forall s$, $[T_\pi f](s) = \mathbb{E}\big[r_\pi(s) + \gamma f(s') \mid s' \sim P_\pi(.|s)\big]$, or in compact form, $T_\pi f = r_\pi + \gamma P_\pi f$. It is known that v_π is the unique fixed-point of T_π. Given a function f on $\mathcal{S}$, we say that a policy π is greedy w.r.t. f, and write it as $\pi = \mathcal{G}\, f$, if $\forall s$, $(T_\pi f)(s) = \max_a (T_a f)(s)$, or equivalently $T_\pi f = \max_{\pi'}(T_{\pi'} f)$. We denote by v_* the optimal value function. It is also known that v_* is the unique fixed-point of the Bellman optimality operator $T : v \to \max_\pi T_\pi v = T_{\mathcal{G}(v)} v$, and that a policy π_* that is greedy w.r.t. v_* is optimal and its value satisfies $v_{\pi_*} = v_*$.

3. Approximate MPI Algorithms

In this section, we describe three approximate MPI (AMPI) algorithms. These algorithms rely on a function space $\mathcal{F}$ to approximate value functions, and in the third algorithm, also on a policy space Π to represent greedy policies. In what follows, we describe the iteration k of these iterative algorithms.

3.1. AMPI-V

For the first and simplest AMPI algorithm presented in the paper, we assume that the values v_k are represented in a function space $\mathcal{F} \subseteq \mathbb{R}^{|\mathcal{S}|}$. In any state s, the action $\pi_{k+1}(s)$ that is greedy w.r.t. v_k can be estimated as follows:

$$\pi_{k+1}(s) = \arg\max_{a \in \mathcal{A}} \frac{1}{M} \Big(\sum_{j=1}^{M} r_a^{(j)} + \gamma v_k(s_a^{(j)}) \Big), \qquad (3)$$

where $\forall a \in \mathcal{A}$ and $1 \le j \le M$, $r_a^{(j)}$ and $s_a^{(j)}$ are samples of rewards and next states when action a is taken in state s. Thus, approximating the greedy action in a state s requires $M|\mathcal{A}|$ samples. The algorithm works as follows. It first samples N states from a distribution μ, i.e., $\{s^{(i)}\}_{i=1}^N \sim \mu$. From each sampled state $s^{(i)}$, it generates a rollout of size m, i.e., $\big(s^{(i)}, a_0^{(i)}, r_0^{(i)}, s_1^{(i)}, \dots, a_{m-1}^{(i)}, r_{m-1}^{(i)}, s_m^{(i)}\big)$, where $a_t^{(i)}$ is the action suggested by π_{k+1} in state $s_t^{(i)}$, computed using Eq. 3, and $r_t^{(i)}$ and $s_{t+1}^{(i)}$ are the reward and next state induced by this choice of action. For each $s^{(i)}$, we then compute a rollout estimate $\widehat{v}_{k+1}(s^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m v_k(s_m^{(i)})$, which is an unbiased estimate of $\big[(T_{\pi_{k+1}})^m v_k\big](s^{(i)})$. Finally, v_{k+1} is computed as the best fit in $\mathcal{F}$ to these estimates, i.e.,

$$v_{k+1} = \text{Fit}_{\mathcal{F}} \left(\Big\{ \big(s^{(i)}, \widehat{v}_{k+1}(s^{(i)})\big) \Big\}_{i=1}^{N} \right).$$

Each iteration of AMPI-V requires N rollouts of size m, and in each rollout any of the $|\mathcal{A}|$ actions needs M samples to compute Eq. 3. This gives a total of $Nm(M|\mathcal{A}|+1)$ transition samples. Note that the fitted value iteration algorithm (Munos & Szepesvári, 2008) is a special case of AMPI-V when $m = 1$.

3.2. AMPI-Q

In AMPI-Q, we replace the value function $v : \mathcal{S} \to \mathbb{R}$ with an action-value function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The Bellman operator for a policy π at a state-action pair (s,a) can then be written as

$$[T_\pi Q](s,a) = \mathbb{E}\big[r_\pi(s,a) + \gamma Q(s', \pi(s')) | s' \sim P(\cdot|s,a)\big],$$

and the greedy operator is defined as

$$\pi = \mathcal{G}\, Q \quad \Leftrightarrow \quad \forall s \quad \pi(s) = \arg\max_{a \in \mathcal{A}} Q(s,a).$$

In AMPI-Q, action-value functions Q_k are represented in a function space $\mathcal{F} \subseteq \mathbb{R}^{|\mathcal{S} \times \mathcal{A}|}$, and the greedy action w.r.t. Q_k at a state s, i.e., $\pi_{k+1}(s)$, is computed as

$$\pi_{k+1}(s) \in \arg\max_{a \in \mathcal{A}} \; Q_k(s,a). \qquad (4)$$

The *evaluation step* is similar to that of AMPI-V, with the difference that now we work with state-action pairs. We sample N state-action pairs from a distribution μ on $\mathcal{S} \times \mathcal{A}$ and build a rollout set

```
Input: Value function space $\mathcal{F}$, policy space $\Pi$, state distribution $\mu$
Initialize: Let $\pi_1 \in \Pi$ be an arbitrary policy and $v_0 \in \mathcal{F}$ an arbitrary value function
for $k = 1, 2, \dots$ do
  • Perform rollouts:
  Construct the rollout set $\mathcal{D}_k = \{s^{(i)}\}_{i=1}^{n}$, $s^{(i)} \overset{\text{iid}}{\sim} \mu$
  for all states $s^{(i)} \in \mathcal{D}_k$ do
    Perform a rollout and return $\widehat{v}_k(s^{(i)})$
  end for
  Construct the rollout set $\mathcal{D}'_k = \{s^{(i)}\}_{i=1}^{N}$, $s^{(i)} \overset{\text{iid}}{\sim} \mu$
  for all states $s^{(i)} \in \mathcal{D}'_k$ and actions $a \in \mathcal{A}$ do
    for $j = 1$ to $M$ do
      Perform a rollout and return $R_k^j(s^{(i)}, a)$
    end for
    $\widehat{Q}_k(s^{(i)}, a) = \frac{1}{M}\sum_{j=1}^{M} R_k^j(s^{(i)}, a)$
  end for
  • Approximate value function:
  $v_k \in \operatorname{argmin}_{v \in \mathcal{F}} \widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; v)$        (regression)
  • Approximate greedy policy:
  $\pi_{k+1} \in \operatorname{argmin}_{\pi \in \Pi} \widehat{\mathcal{L}}_k^{\Pi}(\widehat{\mu}; \pi)$        (classification)
end for
```

Figure 1. The pseudo-code of the CBMPI algorithm.

$\mathcal{D}_k = \{(s^{(i)}, a^{(i)})\}_{i=1}^N$, $(s^{(i)}, a^{(i)}) \sim \mu$. For each $(s^{(i)}, a^{(i)}) \in \mathcal{D}_k$, we generate a rollout of size m, i.e., $\big(s^{(i)}, a^{(i)}, r_0^{(i)}, s_1^{(i)}, a_1^{(i)}, \cdots, s_m^{(i)}, a_m^{(i)}\big)$, where the first action is $a^{(i)}$, $a_t^{(i)}$ for $t \geq 1$ is the action suggested by π_{k+1} in state $s_t^{(i)}$ computed using Eq. 4, and $r_t^{(i)}$ and $s_{t+1}^{(i)}$ are the reward and next state induced by this choice of action. For each $(s^{(i)}, a^{(i)}) \in \mathcal{D}_k$, we then compute the rollout estimate

$$\widehat{Q}_{k+1}(s^{(i)}, a^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m Q_k(s_m^{(i)}, a_m^{(i)}),$$

which is an unbiased estimate of $\big[(T_{\pi_{k+1}})^m Q_k\big](s^{(i)}, a^{(i)})$. Finally, Q_{k+1} is the best fit to these estimates in $\mathcal{F}$, i.e.,

$$Q_{k+1} = \text{Fit}_{\mathcal{F}}\left(\Big\{\big((s^{(i)}, a^{(i)}), \widehat{Q}_{k+1}(s^{(i)}, a^{(i)})\big)\Big\}_{i=1}^N\right).$$

Each iteration of AMPI-Q requires Nm samples, which is less than that for AMPI-V. However, it uses a hypothesis space on state-action pairs instead of states. Note that the fitted-Q iteration algorithm (Ernst et al., 2005; Antos et al., 2007) is a special case of AMPI-Q when $m = 1$.

3.3. Classification-Based MPI

The third AMPI algorithm presented in this paper, called classification-based MPI (CBMPI), uses an explicit representation for the policies π_k, in addition to the one used for value functions v_k. The idea is similar to the classification-based PI algorithms (Lagoudakis & Parr, 2003; Fern et al., 2006; Lazaric et al., 2010; Gabillon et al., 2011) in which we search for the greedy policy in a policy space Π (defined by a classifier) instead of computing it from the estimated value or action-value function (like in AMPI-V and AMPI-Q).

In order to describe CBMPI, we first rewrite the MPI formulation (Eqs. 1 and 2) as

$$v_k = (T_{\pi_k})^m v_{k-1} \qquad \text{(evaluation step)} \tag{5}$$

$$\pi_{k+1} = \mathcal{G}\big[(T_{\pi_k})^m v_{k-1}\big] \qquad \text{(greedy step)} \tag{6}$$

Note that in the new formulation both v_k and π_{k+1} are functions of $(T_{\pi_k})^m v_{k-1}$. CBMPI is an approximate version of this new formulation. As described in Fig. 1, CBMPI begins with arbitrary initial policy $\pi_1 \in \Pi$ and value function $v_0 \in \mathcal{F}$.[1] At each iteration k, a new value function v_k is built as the best approximation of the m-step Bellman operator $(T_{\pi_k})^m v_{k-1}$ in $\mathcal{F}$ (*evaluation step*). This is done by solving a regression problem whose target function is $(T_{\pi_k})^m v_{k-1}$. To set up the regression problem, we build a rollout set $\mathcal{D}_k$ by sampling n states i.i.d. from a distribution μ.[2] For each state $s^{(i)} \in \mathcal{D}_k$, we generate a rollout $\big(s^{(i)}, a_0^{(i)}, r_0^{(i)}, s_1^{(i)}, \dots, a_{m-1}^{(i)}, r_{m-1}^{(i)}, s_m^{(i)}\big)$ of size m, where $a_t^{(i)} = \pi_k(s_t^{(i)})$, and $r_t^{(i)}$ and $s_{t+1}^{(i)}$ are the reward and next state induced by this choice of action. From this rollout, we compute an unbiased estimate $\widehat{v}_k(s^{(i)})$ of $\big[(T_{\pi_k})^m v_{k-1}\big](s^{(i)})$ as

$$\widehat{v}_k(s^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m v_{k-1}(s_m^{(i)}), \tag{7}$$

and use it to build a training set $\Big\{\big(s^{(i)}, \widehat{v}_k(s^{(i)})\big)\Big\}_{i=1}^n$. This training set is then used by the regressor to compute v_k as an estimate of $(T_{\pi_k})^m v_{k-1}$.

The *greedy step* at iteration k computes the policy π_{k+1} as the best approximation of $\mathcal{G}\big[(T_{\pi_k})^m v_{k-1}\big]$ by solving a cost-sensitive classification problem. From the definition of a greedy policy, if $\pi = \mathcal{G}\big[(T_{\pi_k})^m v_{k-1}\big]$, for each $s \in \mathcal{S}$, we have

$$\big[T_\pi (T_{\pi_k})^m v_{k-1}\big](s) = \max_{a \in \mathcal{A}} \big[T_a (T_{\pi_k})^m v_{k-1}\big](s). \tag{8}$$

By defining $Q_k(s, a) = \big[T_a (T_{\pi_k})^m v_{k-1}\big](s)$, we may

[1] Note that the function space $\mathcal{F}$ and policy space Π are automatically defined by the choice of the regressor and classifier, respectively.

[2] Here we used the same sampling distribution μ for both regressor and classifier, but in general different distributions may be used for these two components.

rewrite Eq. 8 as

$$Q_k\big(s,\pi(s)\big) = \max_{a\in\mathcal{A}} Q_k(s,a). \tag{9}$$

The cost-sensitive error function used by CBMPI is of the form

$$\mathcal{L}^{\Pi}_{\pi_k,v_{k-1}}(\mu;\pi) = \int_S \Big[\max_{a\in\mathcal{A}} Q_k(s,a) - Q_k\big(s,\pi(s)\big)\Big]\mu(ds).$$

To simplify the notation we use $\mathcal{L}^{\Pi}_k$ instead of $\mathcal{L}^{\Pi}_{\pi_k,v_{k-1}}$. To set up this cost-sensitive classification problem, we build a rollout set $\mathcal{D}'_k$ by sampling N states i.i.d. from a distribution μ. For each state $s^{(i)} \in \mathcal{D}'_k$ and each action $a \in \mathcal{A}$, we build M independent rollouts of size $m+1$, i.e.,[3]

$$\big(s^{(i)}, a, r_0^{(i,j)}, s_1^{(i,j)}, a_1^{(i,j)}, \dots, a_m^{(i,j)}, r_m^{(i,j)}, s_{m+1}^{(i,j)}\big)_{j=1}^M,$$

where for $t \geq 1$, $a_t^{(i,j)} = \pi_k(s_t^{(i,j)})$, and $r_t^{(i,j)}$ and $s_{t+1}^{(i,j)}$ are the reward and next state induced by this choice of action. From these rollouts, we compute an unbiased estimate of $Q_k(s^{(i)},a)$ as $\widehat{Q}_k(s^{(i)},a) = \frac{1}{M}\sum_{j=1}^M R_k^j(s^{(i)},a)$ where

$$R_k^j(s^{(i)},a) = \sum_{t=0}^{m} \gamma^t r_t^{(i,j)} + \gamma^{m+1} v_{k-1}(s_{m+1}^{(i,j)}).$$

Given the outcome of the rollouts, CBMPI uses a cost-sensitive classifier to return a policy π_{k+1} that minimizes the following *empirical error*

$$\widehat{\mathcal{L}}^{\Pi}_k(\widehat{\mu};\pi) = \frac{1}{N}\sum_{i=1}^{N}\Big[\max_{a\in\mathcal{A}} \widehat{Q}_k(s^{(i)},a) - \widehat{Q}_k\big(s^{(i)},\pi(s^{(i)})\big)\Big],$$

with the goal of minimizing the true error $\mathcal{L}^{\Pi}_k(\mu;\pi)$.

Each iteration of CBMPI requires $nm+M|\mathcal{A}|N(m+1)$ (or $M|\mathcal{A}|N(m+1)$ in case we reuse the rollouts, see Footnote 3) transition samples. Note that when m tends to ∞, we recover the DPI algorithm proposed and analyzed by Lazaric et al. (2010).

4. Error propagation

In this section, we derive a general formulation for propagation of error through the iterations of an AMPI algorithm. The line of analysis for error propagation is different in VI and PI algorithms. VI analysis is based on the fact that this algorithm computes the fixed point of the Bellman optimality operator, and this operator is a γ-contraction in max-norm (Bertsekas & Tsitsiklis, 1996; Munos, 2007). On the other hand, it can be shown that the operator by which PI updates the value from one iteration to the next is not a contraction in max-norm in general. Unfortunately, we can show that the same property holds for MPI when it does not reduce to VI (i.e., $m > 1$).

Proposition 1. *If $m > 1$, there exists no norm for which the operator that MPI uses to update the values from one iteration to the next is a contraction.*

Proof. Consider a deterministic MDP with two states $\{s_1, s_2\}$, two actions $\{change, stay\}$, rewards $r(s_1) = 0, r(s_2) = 1$, and transitions $P_{ch}(s_2|s_1) = P_{ch}(s_1|s_2) = P_{st}(s_1|s_1) = P_{st}(s_2|s_2) = 1$. Consider the following two value functions $v = (\epsilon, 0)$ and $v' = (0, \epsilon)$ with $\epsilon > 0$. Their corresponding greedy policies are $\pi = (st, ch)$ and $\pi' = (ch, st)$, and the next iterates of v and v' can be computed as $(T_\pi)^m v = \begin{pmatrix}\gamma^m\epsilon \\ 1+\gamma^m\epsilon\end{pmatrix}$ and $(T_{\pi'})^m v' = \begin{pmatrix}\frac{\gamma-\gamma^m}{1-\gamma} + \gamma^m\epsilon \\ \frac{1-\gamma^m}{1-\gamma} + \gamma^m\epsilon\end{pmatrix}$. Thus, $(T_{\pi'})^m v' - (T_\pi)^m v = \begin{pmatrix}\frac{\gamma-\gamma^m}{1-\gamma} \\ \frac{\gamma-\gamma^m}{1-\gamma}\end{pmatrix}$ while $v' - v = \begin{pmatrix}-\epsilon \\ \epsilon\end{pmatrix}$. Since ϵ can be arbitrarily small, the norm of $(T_{\pi'})^m v' - (T_\pi)^m v$ can be arbitrarily larger than the norm of $v - v'$ as long as $m > 1$. □

We also know that the analysis of PI usually relies on the fact that the sequence of the generated values is non-decreasing (Bertsekas & Tsitsiklis, 1996; Munos, 2003). Unfortunately, it can be easily shown that for m finite, the value functions generated by MPI may decrease (it suffices to take a very high initial value). It can be seen from what we just described and Proposition 1 that for $m \neq 1$ and ∞, MPI is neither contracting nor non-decreasing, and thus, a new line of proof is needed for the propagation of error in this algorithm.

To study error propagation in AMPI, we introduce an abstract algorithmic model that accounts for potential errors. AMPI starts with an arbitrary value v_0 and at each iteration $k \geq 1$ computes the greedy policy w.r.t. v_{k-1} with some error ϵ'_k, called the *greedy step error*. Thus, we write the new policy π_k as

$$\pi_k = \widehat{\mathcal{G}}_{\epsilon'_k} v_{k-1}. \tag{10}$$

Eq. 10 means that for any policy π',

$$T_{\pi'} v_{k-1} \leq T_{\pi_k} v_{k-1} + \epsilon'_k.$$

AMPI then generates the new value function v_k with some error ϵ_k, called the *evaluation step error*

$$v_k = (T_{\pi_k})^m v_{k-1} + \epsilon_k. \tag{11}$$

Before showing how these two errors are propagated through the iterations of AMPI, let us first define them

[3] We may implement CBMPI more sample efficient by reusing the rollouts generated for the greedy step in the evaluation step.

in the context of each of the algorithms presented in Section 3 separately.

AMPI-V: ϵ_k is the error in fitting the value function v_k. This error can be further decomposed into two parts: the one related to the approximation power of $\mathcal{F}$ and the one due to the finite number of samples/rollouts. ϵ'_k is the error due to using a finite number of samples M for estimating the greedy actions.

AMPI-Q: $\epsilon'_k = 0$ and ϵ_k is the error in fitting the state-action value function Q_k.

CBMPI: This algorithm iterates as follows:

$$v_k = (T_{\pi_k})^m v_{k-1} + \epsilon_k$$
$$\pi_{k+1} = \widehat{\mathcal{G}}_{\epsilon'_{k+1}}\left[(T_{\pi_k})^m v_{k-1}\right]$$

Unfortunately, this does not exactly match with the model described in Eqs. 10 and 11. By introducing the auxiliary variable $w_k \triangleq (T_{\pi_k})^m v_{k-1}$, we have $v_k = w_k + \epsilon_k$, and thus, we may write

$$\pi_{k+1} = \widehat{\mathcal{G}}_{\epsilon'_{k+1}}\left[w_k\right]. \tag{12}$$

Using $v_{k-1} = w_{k-1} + \epsilon_{k-1}$, we have

$$\begin{aligned} w_k &= (T_{\pi_k})^m v_{k-1} = (T_{\pi_k})^m (w_{k-1} + \epsilon_{k-1}) \\ &= (T_{\pi_k})^m w_{k-1} + (\gamma P_{\pi_k})^m \epsilon_{k-1}. \end{aligned} \tag{13}$$

Now, Eqs. 12 and 13 exactly match Eqs. 10 and 11 by replacing v_k with w_k and ϵ_k with $(\gamma P_{\pi_k})^m \epsilon_{k-1}$.

The rest of this section is devoted to show how the errors ϵ_k and ϵ'_k propagate through the iterations of an AMPI algorithm. We only outline the main arguments that will lead to the performance bound of Thm. 1 and report most proofs in (Scherrer et al., 2012). We follow the line of analysis developped by Thiery & Scherrer (2010). The results are obtained using the following three quantities:

1) The distance between the optimal value function and the value before approximation at the k^{th} iteration: $d_k \triangleq v_* - (T_{\pi_k})^m v_{k-1} = v_* - (v_k - \epsilon_k)$.

2) The shift between the value before approximation and the value of the policy at the k^{th} iteration: $s_k \triangleq (T_{\pi_k})^m v_{k-1} - v_{\pi_k} = (v_k - \epsilon_k) - v_{\pi_k}$.

3) The Bellman residual at the k^{th} iteration: $b_k \triangleq v_k - T_{\pi_{k+1}} v_k$.

We are interested in finding an upper bound on the **loss** $l_k \triangleq v_* - v_{\pi_k} = d_k + s_k$. To do so, we will upper bound d_k and s_k, which requires a bound on the Bellman residual b_k. More precisely, the core of our analysis is to prove the following point-wise inequalities for our three quantities of interest.

Lemma 1 (Proof in Appendix. A)**.** *Let $k \geq 1$, $x_k \triangleq (I - \gamma P_{\pi_k})\epsilon_k + \epsilon'_{k+1}$ and $y_k \triangleq -\gamma P_{\pi_*}\epsilon_k + \epsilon'_{k+1}$. We have:*

$$b_k \leq (\gamma P_{\pi_k})^m b_{k-1} + x_k,$$
$$d_{k+1} \leq \gamma P_{\pi_*} d_k + y_k + \sum_{j=1}^{m-1} (\gamma P_{\pi_{k+1}})^j b_k,$$
$$s_k = (\gamma P_{\pi_k})^m (I - \gamma P_{\pi_k})^{-1} b_{k-1}.$$

Since the stochastic kernels are non-negative, the bounds in Lemma 1 indicate that the loss l_k will be bounded if the errors ϵ_k and ϵ'_k are controlled. In fact, if we define ϵ as a uniform upper-bound on the errors $|\epsilon_k|$ and $|\epsilon'_k|$, the first inequality in Lemma 1 implies that $b_k \leq O(\epsilon)$, and as a result, the second and third inequalities give us $d_k \leq O(\epsilon)$ and $s_k \leq O(\epsilon)$. This means that the loss will also satisfy $l_k \leq O(\epsilon)$.

Our bound for the loss l_k is the result of careful expansion and combination of the three inequalities in Lemma 1. Before we state this result, we introduce some notations that will ease our formulation.

Definition 1. *For a positive integer n, we define $\mathbb{P}_n$ as the set of transition kernels that are defined as follows:*

1) *for any set of n policies $\{\pi_1, \ldots, \pi_n\}$, $(\gamma P_{\pi_1})(\gamma P_{\pi_2}) \ldots (\gamma P_{\pi_n}) \in \mathbb{P}_n$,*

2) *for any $\alpha \in (0,1)$ and $(P_1, P_2) \in \mathbb{P}_n \times \mathbb{P}_n$, $\alpha P_1 + (1-\alpha)P_2 \in \mathbb{P}_n$.*

Furthermore, we use the somewhat abusive notation Γ^n for denoting any element of $\mathbb{P}_n$. For example, if we write a transition kernel P as $P = \alpha_1 \Gamma^i + \alpha_2 \Gamma^j \Gamma^k = \alpha_1 \Gamma^i + \alpha_2 \Gamma^{j+k}$, it should be read as there exist $P_1 \in \mathbb{P}_i$, $P_2 \in \mathbb{P}_j$, $P_3 \in \mathbb{P}_k$, and $P_4 \in \mathbb{P}_{k+j}$ such that $P = \alpha_1 P_1 + \alpha_2 P_2 P_3 = \alpha_1 P_1 + \alpha_2 P_4$.

Using the notation introduced in Definition 1, we now derive a point-wise bound on the loss.

Lemma 2 (Proof in Appendix. B)**.** *After k iterations, the losses of AMPI-V and AMPI-Q satisfy*

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

while the loss of CBMPI satisfies

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

where $h(k) \triangleq 2\sum_{j=k}^{\infty} \Gamma^j |d_0|$ or $h(k) \triangleq 2\sum_{j=k}^{\infty} \Gamma^j |b_0|$.

Remark 1. A close look at the existing point-wise error bounds for AVI (Munos, 2007, Lemma 4.1) and API (Munos, 2003, Corollary 10) shows that they do not consider error in the greedy step (i.e., $\epsilon'_k = 0$) and that they have the following form:

$$\limsup_{k\to\infty} l_k \leq 2 \limsup_{k\to\infty} \sum_{i=1}^{k-1}\sum_{j=i}^{\infty} \Gamma^j |\epsilon_{k-i}|.$$

This indicates that the bound in Lemma 2 not only unifies the analysis of AVI and API, but it generalizes them to the case of error in the greedy step and to a finite horizon k. Moreover, our bound suggests that the way the errors are propagated in the whole family of algorithms VI/PI/MPI does not depend on m at the level of the abstraction suggested by Definition 1.[4]

The next step is to show how the point-wise bound of Lemma 2 can turn to a bound in weighted L_p-norm, which for any function $f : \mathcal{S} \to \mathbb{R}$ and any distribution μ on $\mathcal{S}$ is defined as $\|f\|_{p,\mu} \triangleq \left(\int |f(x)|^p \mu(dx)\right)^{1/p}$. Munos (2003; 2007); Munos & Szepesvári (2008), and the recent work of Farahmand et al. (2010), which provides the most refined bounds for API and AVI, show how to do this process through quantities, called *concentrability coefficients*, that measure how a distribution over states may concentrate through the dynamics of the MDP. We now state a lemma that generalizes the analysis of Farahmand et al. (2010) to a larger class of concentrability coefficients. We will discuss the potential advantage of this new class in Remark 4. We will also show through the proofs of Thms. 1 and 3, how the result of Lemma 3 provides us with a flexible tool for turning point-wise bounds into L_p-norm bounds. Thm. 3 in Appendix. D provides an alternative bound for the loss of AMPI, which in analogy with the results of Farahmand et al. (2010) shows that the last iterations have the highest impact on the loss (the influence exponentially decreases towards the initial iterations).

Lemma 3 (Proof in Appendix. C)**.** *Let $\mathcal{I}$ and $(\mathcal{J}_i)_{i\in\mathcal{I}}$ be sets of positive integers, $\{\mathcal{I}_1,\dots,\mathcal{I}_n\}$ be a partition of $\mathcal{I}$, and f and $(g_i)_{i\in\mathcal{I}}$ be functions satisfying*

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

Then for all p, q and q' such that $\frac{1}{q}+\frac{1}{q'}=1$, and for all distributions ρ and μ, we have

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

with the following concentrability coefficients

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

with the Radon-Nikodym derivative based quantity

$$c_q(j) \triangleq \max_{\pi_1,\cdots,\pi_j} \left\| \frac{d(\rho P_{\pi_1}P_{\pi_2}\cdots P_{\pi_j})}{d\mu}\right\|_{q,\mu}. \qquad (14)$$

We now derive a L_p-norm bound for the loss of the AMPI algorithm by applying Lemma 3 to the point-wise bound of Lemma 2.

Theorem 1 (Proof in Appendix. D)**.** *Let ρ and μ be distributions over states. Let p, q, and q' be such that $\frac{1}{q}+\frac{1}{q'}=1$. After k iterations, the loss of AMPI satisfies*

$$\|l_k\|_{p,\rho} \leq \frac{2(\gamma-\gamma^k)\left(\mathcal{C}_q^{1,k,0}\right)^{\frac{1}{p}}}{(1-\gamma)^2} \sup_{1\leq j\leq k-1} \|\epsilon_j\|_{pq',\mu} + \frac{(1-\gamma^k)\left(\mathcal{C}_q^{0,k,0}\right)^{\frac{1}{p}}}{(1-\gamma)^2} \sup_{1\leq j\leq k} \|\epsilon'_j\|_{pq',\mu} + g(k), \qquad (15)$$

while the loss of CBMPI satisfies

$$\|l_k\|_{p,\rho} \leq \frac{2\gamma^m(\gamma-\gamma^{k-1})\left(\mathcal{C}_q^{2,k,m}\right)^{\frac{1}{p}}}{(1-\gamma)^2} \sup_{1\leq j\leq k-2} \|\epsilon_j\|_{pq',\mu} + \frac{(1-\gamma^k)\left(\mathcal{C}_q^{1,k,0}\right)^{\frac{1}{p}}}{(1-\gamma)^2} \sup_{1\leq j\leq k} \|\epsilon'_j\|_{pq',\mu} + g(k), \qquad (16)$$

where for all q, l, k and d, the concentrability coefficients $\mathcal{C}_q^{l,k,d}$ are defined as

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

with $c_q(j)$ given by Eq. 14, and $g(k)$ is defined as $g(k) \triangleq \frac{2\gamma^k}{1-\gamma}\left(\mathcal{C}_q^{k,k+1}\right)^{\frac{1}{p}} \min\left(\|d_0\|_{pq',\mu}, \|b_0\|_{pq',\mu}\right)$.

Remark 2. When p tends to infinity, the first bound of Thm. 1 reduces to

$$\|l_k\|_\infty \leq \frac{2(\gamma-\gamma^k)}{(1-\gamma)^2} \sup_{1\leq j\leq k-1}\|\epsilon_j\|_\infty + \frac{1-\gamma^k}{(1-\gamma)^2}\sup_{1\leq j\leq k}\|\epsilon'_j\|_\infty + \frac{2\gamma^k}{1-\gamma}\min(\|d_0\|_\infty, \|b_0\|_\infty). \qquad (17)$$

When k goes to infinity, Eq. 17 gives us a generalization of the API ($m=\infty$) bound of Bertsekas & Tsitsiklis (1996, Prop. 6.2), i.e.,

$$\limsup_{k\to\infty}\|l_k\|_\infty \leq \frac{2\gamma\sup_j\|\epsilon_j\|_\infty + \sup_j\|\epsilon'_j\|_\infty}{(1-\gamma)^2}.$$

Moreover, since our point-wise analysis generalizes those of API and AVI (as noted in Remark 1), the L_p-bound of Eq. 15 unifies and generalizes those for API (Munos, 2003) and AVI (Munos, 2007).

[4] Note however that the dependence on m will reappear if we make explicit what is hidden in the terms Γ^j.

Remark 3. Canbolat & Rothblum (2012) recently (and independently) developped an analysis of an approximate form of MPI. Also, as mentioned, the proof technique that we used is mainly based on that in Thiery & Scherrer (2010). While Canbolat & Rothblum (2012) only consider the error in the greedy step and Thiery & Scherrer (2010) that in the value update, our work is more general in that we consider *both* sources of error – this is required for the analysis of CBMPI. Thiery & Scherrer (2010) and Canbolat & Rothblum (2012) provide bounds when the errors are controlled in max-norm, while we consider the more general L_p-norm. At a more technical level, Thm. 2 in Canbolat & Rothblum (2012) bounds the norm of the distance $v_* - v_k$, while we bound the loss $v_* - v_{\pi_k}$. If we derive a bound on the loss (using e.g., Thm. 1 in Canbolat & Rothblum 2012), this leads to a bound on the loss that is looser than ours. In particular, this does not allow to recover the standard bounds for AVI/API, as we managed to (c.f. Remark 2).

Remark 4. We can balance the influence of the concentrability coefficients (the bigger the q, the higher the influence) and the difficulty of controlling the errors (the bigger the q', the greater the difficulty in controlling the $L_{pq'}$-norms) by tuning the parameters q and q', given the condition that $\frac{1}{q} + \frac{1}{q'} = 1$. This potential leverage is an improvement over the existing bounds and concentrability results that only consider specific values of these two parameters: $q = \infty$ and $q' = 1$ in Munos (2007); Munos & Szepesvári (2008), and $q = q' = 2$ in Farahmand et al. (2010).

Remark 5. For CBMPI, the parameter m controls the influence of the value function approximator, cancelling it out in the limit when m tends to infinity (see Eq. 16). Assuming a fixed budget of sample transitions, increasing m reduces the number of rollouts used by the classifier, and thus, worsens its quality; in such a situation, m allows to make a trade-off between the estimation error of the classifier and the overall value function approximation.

5. Finite-Sample Analysis of CBMPI

In this section, we focus on CBMPI and detail the possible form of the error terms that appear in the bound of Thm. 1. We select CBMPI among the proposed algorithms because its analysis is more general than the others as we need to bound both greedy and evaluation step errors (in some norm), and also because it displays an interesting influence of the parameter m (see Remark 5). We first provide a bound on the *greedy step error*. From the definition of ϵ'_k for CBMPI (Eq. 12) and the description of the greedy step in CBMPI, we can easily observe that $\|\epsilon'_k\|_{1,\mu} = \mathcal{L}^{\Pi}_{k-1}(\mu;\pi_k)$.

Lemma 4 (Proof in Appendix. E). *Let Π be a policy space with finite VC-dimension $h = VC(\Pi)$ and μ be a distribution over the state space $\mathcal{S}$. Let N be the number of states in $\mathcal{D}'_{k-1}$ drawn i.i.d. from μ, M be the number of rollouts per state-action pair used in the estimation of $\widehat{Q}_{k-1}$, and $\pi_k = \operatorname{argmin}_{\pi\in\Pi} \widehat{\mathcal{L}}^{\Pi}_{k-1}(\widehat{\mu},\pi)$ be the policy computed at iteration $k-1$ of CBMPI. Then, for any $\delta > 0$,*

$$\|\epsilon'_k\|_{1,\mu} = \mathcal{L}^{\Pi}_{k-1}(\mu;\pi_k) \le \inf_{\pi\in\Pi} \mathcal{L}^{\Pi}_{k-1}(\mu;\pi) + 2(\epsilon'_1 + \epsilon'_2),$$

with probability at least $1-\delta$, where

$$\epsilon'_1(N,\delta) = 16 Q_{\max}\sqrt{\frac{2}{N}\Big(h\log\frac{eN}{h} + \log\frac{32}{\delta}\Big)}\,,$$

$$\epsilon'_2(N,M,\delta) = 8 Q_{\max}\sqrt{\frac{2}{MN}\Big(h\log\frac{eMN}{h} + \log\frac{32}{\delta}\Big)}\,.$$

We now consider the *evaluation step error*. The evaluation step at iteration k of CBMPI is a regression problem with the target $(T_{\pi_k})^m v_{k-1}$ and a training set $\big\{\big(s^{(i)}, \widehat{v}_k(s^{(i)})\big)\big\}_{i=1}^n$ in which the states $s^{(i)}$ are i.i.d. samples from μ and $\widehat{v}_k(s^{(i)})$ are unbiased estimates of the target computed according to Eq. 7. Different function spaces $\mathcal{F}$ (linear or non-linear) may be used to approximate $(T_{\pi_k})^m v_{k-1}$. Here we consider a linear architecture with parameters $\alpha \in \mathbb{R}^d$ and bounded (by L) basis functions $\{\varphi_j\}_{j=1}^d$, $\|\varphi_j\|_\infty \le L$. We denote by $\phi : \mathcal{X} \to \mathbb{R}^d$, $\phi(\cdot) = \big(\varphi_1(\cdot),\ldots,\varphi_d(\cdot)\big)^\top$ the feature vector, and by $\mathcal{F}$ the linear function space spanned by the features φ_j, i.e., $\mathcal{F} = \{f_\alpha(\cdot) = \phi(\cdot)^\top\alpha : \alpha \in \mathbb{R}^d\}$. Now if we define v_k as the truncation (by $V_{\max}$) of the solution of the above linear regression problem, we may bound the *evaluation step error* using the following lemma.

Lemma 5 (Proof in Appendix. F). *Consider the linear regression setting described above, then we have*

$$\|\epsilon_k\|_{2,\mu} \le 4\inf_{f\in\mathcal{F}} \|(T_{\pi_k})^m v_{k-1} - f\|_{2,\mu} + \epsilon_1 + \epsilon_2,$$

with probability at least $1-\delta$, where

$$\epsilon_1(n,\delta) = 32 V_{\max}\sqrt{\frac{2}{n}\log\Big(\frac{27(12e^2n)^{2(d+1)}}{\delta}\Big)}\,,$$

$$\epsilon_2(n,\delta) = 24\Big(V_{\max} + \|\alpha_*\|_2 \cdot \sup_x \|\phi(x)\|_2\Big)\sqrt{\frac{2}{n}\log\frac{9}{\delta}}\,,$$

and α_* is such that f_{α_*} is the best approximation (w.r.t. μ) of the target function $(T_{\pi_k})^m v_{k-1}$ in $\mathcal{F}$.

From Lemmas 4 and 5, we have bounds on $\|\epsilon'_k\|_{1,\mu}$ and $\|\epsilon_k\|_{1,\mu} \le \|\epsilon_k\|_{2,\mu}$. By a union bound argument, we thus control the r.h.s. of Eq. 16 in L_1-norm. In the context of Thm. 1, this means $p = 1$, $q' = 1$ and $q = \infty$, and we have the following bound for CBMPI:

Theorem 2. *Let* $d' = \sup_{g \in \mathcal{F}, \pi'} \inf_{\pi \in \Pi} \mathcal{L}^{\Pi}_{\pi', g}(\mu; \pi)$ *and* $d_m = \sup_{g \in \mathcal{F}, \pi} \inf_{f \in F} \|(T_\pi)^m g - f\|_{2,\mu}$. *With the notations of Thm. 1 and Lemmas 4 and 5, after k iterations, and with probability* $1 - \delta$*, the expected loss* $\mathbb{E}_\mu[l_k] = \|l_k\|_{1,\mu}$ *of CBMPI is bounded by*

$$\frac{2\gamma^m(\gamma - \gamma^{k-1})\mathcal{C}_\infty^{2,k,m}}{(1-\gamma)^2}\left(d_m + \epsilon_1(n, \frac{\delta}{2k}) + \epsilon_2(n, \frac{\delta}{2k})\right) + \frac{(1-\gamma^k)\mathcal{C}_\infty^{1,k,0}}{(1-\gamma)^2}\left(d' + \epsilon_1'(N, \frac{\delta}{2k}) + \epsilon_2'(N, M, \frac{\delta}{2k})\right) + g(k).$$

Remark 6. This result leads to a quantitative version of Remark 5. Assume that we have a fixed budget for the actor and the critic $B = nm = NM|A|m$. Then, up to constants and logarithmic factors, the bound has the form $\|l_k\|_{1,\mu} \leq O\left(\gamma^m\left(d_m + \sqrt{\frac{m}{B}}\right) + d' + \sqrt{\frac{M|A|m}{B}}\right)$. It shows the trade-off in the tuning of m: a big m can make the influence of the overall (approximation and estimation) value error small, but that of the estimation error of the classifier bigger.

6. Summary and Extensions

In this paper, we studied a DP algorithm, called modified policy iteration (MPI), that despite its generality that contains the celebrated policy and value iteration methods, has not been thoroughly investigated in the literature. We proposed three approximate MPI (AMPI) algorithms that are extensions of the well-known ADP algorithms: fitted-value iteration, fitted-Q iteration, and classification-based policy iteration. We reported an error propagation analysis for AMPI that unifies those for approximate policy and value iteration. We also provided a finite-sample analysis for the classification-based implementation of AMPI (CBMPI), whose analysis is more general than the other presented AMPI methods. Our results indicate that the parameter of MPI allows us to control the balance of errors (in value function approximation and estimation of the greedy policy) in the final performance of CBMPI. Although AMPI generalizes the existing AVI and classification-based API algorithms, additional experimental work and careful theoretical analysis are required to obtain a better understanding of the behaviour of its different implementations and their relation to the competitive methods. Extension of CBMPI to problems with continuous action space is another interesting direction to pursue.

Acknowledgments The second and third authors would like to thank French National Research Agency (ANR) under project LAMPADA n° ANR-09-EMER-007, European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 231495, and PASCAL2 European Network of Excellence for supporting their research.

References

Antos, A., Munos, R., and Szepesvári, Cs. Fitted Q-iteration in continuous action-space MDPs. In *Proceedings of NIPS*, pp. 9–16, 2007.

Bertsekas, D. and Tsitsiklis, J. *Neuro-Dynamic Programming.* Athena Scientific, 1996.

Canbolat, P. and Rothblum, U. (Approximate) iterated successive approximations algorithm for sequential decision processes. *Annals of Operations Research*, pp. 1–12, 2012.

Ernst, D., Geurts, P., and Wehenkel, L. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

Farahmand, A., Munos, R., and Szepesvári, Cs. Error propagation for approximate policy and value iteration. In *Proceedings of NIPS*, pp. 568–576, 2010.

Fern, A., Yoon, S., and Givan, R. Approximate policy iteration with a policy language bias: Solving relational markov decision processes. *Journal of Artificial Intelligence Research*, 25:75–118, 2006.

Gabillon, V., Lazaric, A., Ghavamzadeh, M., and Scherrer, B. Classification-based policy iteration with a critic. In *Proceedings of ICML*, pp. 1049–1056, 2011.

Lagoudakis, M. and Parr, R. Reinforcement learning as classification: Leveraging modern classifiers. In *Proceedings of ICML*, pp. 424–431, 2003.

Lazaric, A., Ghavamzadeh, M., and Munos, R. Analysis of a classification-based policy iteration algorithm. In *Proceedings of ICML*, pp. 607–614, 2010.

Munos, R. Error bounds for approximate policy iteration. In *Proceedings of ICML*, pp. 560–567, 2003.

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

Munos, R. and Szepesvári, Cs. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9:815–857, 2008.

Puterman, M. and Shin, M. Modified policy iteration algorithms for discounted Markov decision problems. *Management Science*, 24(11), 1978.

Scherrer, B., Gabillon, V., Ghavamzadeh, M., and Geist, M. Approximate modified policy iteration. Technical report, INRIA, 2012.

Szepesvári, Cs. Reinforcement learning algorithms for MDPs. In *Wiley Encyclopedia of Operations Research.* Wiley, 2010.

Thiery, C. and Scherrer, B. Performance bound for approximate optimistic policy iteration. Technical report, INRIA, 2010.