From Optimization to Control: Quasi Policy Iteration

Mohammad Amin Sharifi Kolarijani and Peyman Mohajerin Esfahani

Abstract. Recent control algorithms for Markov decision processes (MDPs) have been designed using an implicit analogy with well-established optimization algorithms. In this paper, we review this analogy across four problem classes with a unified solution characterization allowing for a systematic transformation of algorithms from one domain to the other. In particular, we identify equivalent optimization and control algorithms that have already been pointed out in the existing literature, but mostly in a scattered way. With this unifying framework in mind, we adopt the quasi-Newton method from convex optimization to introduce a novel control algorithm coined as quasi-policy iteration (QPI). In particular, QPI is based on a novel approximation of the "Hessian" matrix in the policy iteration algorithm by exploiting two linear structural constraints specific to MDPs and by allowing for the incorporation of prior information on the transition probability kernel. While the proposed algorithm has the same computational complexity as value iteration, it interestingly exhibits an empirical convergence behavior similar to policy iteration with a very low sensitivity to the discount factor.

Keywords: Dynamic programming, reinforcement learning, optimization algorithms, quasi-Newton methods, Markov decision processes.

1. Introduction

The problem of control, or the decision-making problem as it is also known within the operations research community, has been the subject of much research since the introduction of the Bellman principle of optimality in the late 1950s [4]. Apart from the fact that policy iteration (PI) is an instance of the Newton method, which has been known since the late 1970s [45], more recent works have made implicit use of the relationship between optimization and control problems to develop new control algorithms, with faster convergence and/or lower complexity, inspired by their counterparts for solving optimization problems. For instance, accelerated versions of value iteration (VI) in [21] are inspired by Polyak momentum and Nesterov acceleration in convex optimization, while the Q-learning combined with Polyak momentum and Nesterov acceleration produces momentum Q-learning [59]. In particular, more recently, Halpern's anchoring acceleration scheme [23] has been

Date: October 13, 2024.

The authors are with Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands. Email: {M.A.SharifiKolarijani, P.MohajerinEsfahani}@tudelft.nl.

This research is part of a project that has received funding from the European Research Council (ERC) under the grant TRUST-949796.

used to introduce the Anchored VI algorithm [34] with an improved convergence rate for large values of discount factor γ and even for $\gamma = 1$.

The implicit connection between optimization algorithms and control algorithms for Markov decision processes (MDPs) with a finite state-action space has also been studied more systematically. In [57], the authors look at the connection between *constrained* convex optimization algorithms and control algorithms such as Frank-Wolfe algorithm [16] and conservative PI [27]. A detailed comparison between *deterministic* optimization algorithms and *model-based*[1] control algorithms is also provided in [22], where the author looks at a wide range of optimization algorithms including gradient descent, accelerated gradient descent, Newton method, and quasi-Newton method and their counterparts for solving control problems.

When it comes to infinite (continuous) state-action spaces, except in special cases such as linear–quadratic regulators (LQR), one needs to resort to finite-dimensional approximation techniques for computational purposes. This approximation may be at the modeling level by aggregation (discretization) of the state and action spaces, which readily falls into the finite MDP setting mentioned above [6, 44]. Alternatively, one may directly approximate the value function via finite parametrization by minimizing (a proxy of) the residual of its fixed-point characterization based on the Bellman principle of optimality [7, 55]. Examples of such include linear parameterization [9, 56], or nonlinear parameterization with, for instance, neural network architectures [8, 52, 54] or max-plus approximation [5, 20, 32, 31, 38]. We also note that there is an alternative characterization of the original function as the solution to an infinite-dimensional linear program [24], paving the way for approximation techniques via finite tractable convex optimization [12, 25, 39]. With this view of the literature, it is worth noting that one can cast almost all of these approximation techniques as the solution to a finite-dimensional fixed-point or convex optimization problem.

Motivated by these observations, we exploit the well-known root-finding characterization of optimization problems and fixed-point characterization of control problems to provide a framework for explicit transformation of deterministic (resp. stochastic) convex optimization problems to model-based (resp. model-free) control problems, and vice versa (Table 1). These transformations, in turn, allow us to identify "equivalent" algorithms in the two domains that have already been pointed out in the existing literature, but mostly in a scattered way (Table 2).

The **main contribution** of this paper is that using the developed equivalence framework, we adopt the quasi-Newton method from convex optimization to introduce the quasi-policy iteration (QPI) algorithm with the following distinct features:

(1) **Hessian approximation via structural information**: QPI is based on a novel approximation of the "Hessian" matrix in the PI algorithm by exploiting two linear structural constraints specific to MDPs and by allowing for the incorporation of prior information on

[1] In this paper, the terminologies of "model-free" and "model-based" indicate the *available information (oracle)*, i.e., whether we have access to the model or only the system trajectory (samples); see Section 2.1 for more details. We note that this is different from the common terminologies in the RL literature where these terms refer to the *solution approach*, i.e., whether we identify the model along the way (model-based RL) or directly solve the Bellman equation to find the value function (model-free RL).

the transition probability kernel of the MDP (Theorem 4.1). In the special case of incorporating a uniform prior for the transition kernel, QPI can be viewed as a modification of the standard VI using two novel directions with adaptive step-sizes (Corollary 4.2).

(2) **Convergence rate and sensitivity to discount factor**: The per-iteration computational complexity of QPI is the same as VI (i.e., $\mathcal{O}(n^2)$ where n is the number of states), and its convergence can be guaranteed by safeguarding via standard VI (Theorem 4.1). However, in our numerical simulations with random and structured MDPs, QPI exhibits an empirical behavior similar to PI (which has a $\mathcal{O}(n^3)$ per-iteration complexity) concerning *sensitivity of convergence rate to discounted factor* (Figure 1).

(3) **Extension to model-free control (a.k.a. RL)**: We also introduce the quasi-policy learning (QPL) algorithm, the stochastic version of QPI, as a novel model-free algorithm, and guarantee its convergence by safeguarding via standard Q-learning (QL) algorithm (Theorem 4.4).

The paper is organized as follows. In Section 2, we describe the connection between optimization and control problems by providing the explicit transformations between them. We then use this framework to look at equivalent algorithms from the two domains in Section 3. In Section 4, we introduce and analyze the model-based QPI algorithm and its model-free extension, the QPL algorithm. All the technical proofs are provided in Section 5. The performance of these algorithms is then compared with multiple control algorithms via extensive numerical experiments in Section 6. Section 7 concludes the paper by providing some final remarks.

Notations. For a vector $v \in \mathbb{R}^n$, we use $v(i)$ and $[v](i)$ to denote its i-th element. Similarly, $M(i,j)$ and $[M](i,j)$ denote the element in row i and column j of the matrix $M \in \mathbb{R}^{m\times n}$. We use $\cdot^\top$ to denote the transpose of a vector/matrix. We use $\|\cdot\|_2$ and $\|\cdot\|_\infty$ to denote the 2-norm and ∞-norm of a vector, respectively. We use $\|\cdot\|_2$ and $\|\cdot\|_F$ for the induced 2-norm and the Frobenius norm of a matrix, respectively. Let $x \sim \mathbb{P}$ be a random variable with distribution $\mathbb{P}$. We particularly use $\hat{x} \sim \mathbb{P}$ to denote *a sample of the random variable* x drawn from the distribution $\mathbb{P}$. The identity operator is denoted by Id. We use $\mathbf{1}$ and $\mathbf{0}$ to denote the all-one and all-zero vectors, respectively. I and $E = \mathbf{1}\mathbf{1}^\top$ denote the identity and all-one matrices, respectively. We denote the i-th unit vector by e_i, that is, the vector with its i-th element equal to 1 and all other elements equal to 0.

2. Equivalence Transformations

In this section, we provide the generic framework that connects the optimization problems to the control problems. In particular, we provide the explicit transformations between specific characterizations of the solutions to these problems. Table 1 provides a condensed summary of this framework.

Domain	**Optimization**		**Control**	
Problem	Function $\widehat{f}:\mathbb{R}^\ell\times\boldsymbol{\Xi}\to\mathbb{R}$, Random variable $\xi\sim\mathbb{P}$, $\min_x\{f(x)\coloneqq\mathbb{E}_{\mathbb{P}}[\widehat{f}(x,\xi)]\}$		State $s\in\mathcal{S}$, Control $a\in\mathcal{A}$, Dynamics $s^+\sim\mathbb{P}(\cdot\|s,a)$, Cost $c:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$, $\min_{\pi:\mathcal{S}\to\mathcal{A}}\mathbb{E}_{\mathbb{P}}\left[\sum_{t=0}^{\infty}\gamma^t c(s_t,\pi(s_t))\middle\|s_0=s\right],\quad\forall s\in\mathcal{S}$	
Type	**Deterministic**	**Stochastic**	**Model-based**	**Model-free**
Equivalent characterization	$\nabla f(x^\star)=0$	$\mathbb{E}_{\mathbb{P}}[\nabla\widehat{f}(x^\star,\xi)]=0$	$v^\star=T(v^\star)$	$q^\star=\mathbb{E}_{\mathbb{P}}\big[\widehat{T}(q^\star,s^+)\big]$
Available oracle/info	$\nabla f(x)$ (Prob. distribution $\mathbb{P}$)	$\nabla\widehat{f}(x,\xi)$ (Samples $\hat{\xi}$)	$T(v)$ (Prob. kernel $\mathbb{P}$, Cost c)	$\widehat{T}(q,s^+)$ (Samples $(s,a,c(s,a),\hat{s}^+)$)
Transformation	x ⟷		v	
	∇f ⟶		$\mathrm{Id}-T$	
	$\mathrm{Id}-\nabla f$ ⟵		T	
	$\nabla^2 f$ ⟶		$I-\gamma P$	
	$I-\nabla^2 f$ ⟵		γP	
		(x,ξ) ⟷		(q,s^+)
		$\nabla\widehat{f}$ ⟶		$\mathrm{Id}-\widehat{T}$
		$\mathrm{Id}-\nabla\widehat{f}$ ⟵		$\widehat{T}$
		$\nabla^2\widehat{f}$ ⟶		$I-\gamma\widehat{P}$
		$I-\nabla^2\widehat{f}$ ⟵		$\gamma\widehat{P}$

TABLE 1. Equivalence transformations: The symbol Id denotes the identity operator, and T is the Bellman operator (3). The random operator $\widehat{T}$ is the sampled Bellman operator (5). The matrix $P=P(v)$ is the state transition probability matrix of the Markov chain under the greedy policy w.r.t. the value function v. The matrix $\widehat{P}=\widehat{P}(q,\hat{s}^+)$ is the sampled state-action transition probability matrix of the Markov chain under the greedy policy w.r.t. the Q-function q.

2.1. **Control problem**

A common formulation of the control problem relies on the concept of *Markov decision processes* (MDPs). MDPs are a powerful modeling framework for stochastic environments that can be controlled to minimize some measure of cost. An MDP is a tuple $(\mathcal{S},\mathcal{A},\mathbb{P},c,\gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state space and action space, respectively. The transition kernel $\mathbb{P}$ encapsulates the state dynamics: for each triplet $(s,a,s^+)\in\mathcal{S}\times\mathcal{A}\times\mathcal{S}$, it gives the probability $\mathbb{P}(s^+|s,a)$ of the transition to state s^+ given that the system is in state s and the chosen control is a. The cost function $c:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$, bounded from below, represents the cost $c(s,a)$ of taking the control action a while the system is in state s. The discount factor $\gamma\in(0,1)$ can be seen as a trade-off parameter between short- and long-term costs. *In this study, we consider tabular MDPs with a finite state-action space. In particular, we take* $\mathcal{S}=\{1,2,\dots,n\}$ *and* $\mathcal{A}=\{1,2,\dots,m\}$. This, in turn, allows us to treat functions $f:\mathcal{S}\to\mathbb{R}$ and $g:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ as vectors $f\in\mathbb{R}^{|\mathcal{S}|}=\mathbb{R}^n$ and $g\in\mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}=\mathbb{R}^{nm}$ – in the latter case, we are considering a proper 1-to-1 mapping $\mathcal{S}\times\mathcal{A}\to\{1,2,\dots,nm\}$.

Let us now fix a control policy $\pi:\mathcal{S}\to\mathcal{A}$, i.e., a mapping from states to actions. The stage cost of the policy π is denoted by $c^\pi\in\mathbb{R}^{|\mathcal{S}|}=\mathbb{R}^n$ with elements $c^\pi(s)=c\big(s,\pi(s)\big)$ for $s\in\mathcal{S}$. The transition (probability) kernel of the resulting Markov chain under the policy π is denoted by $\mathbb{P}^\pi$, where $\mathbb{P}^\pi(s^+|s)=\mathbb{P}\big(s^+|s,\pi(s)\big)$ for $s,s^+\in\mathcal{S}$. We also define the matrix $P^\pi\in\mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}=\mathbb{R}^{n\times n}$,

with elements $P^\pi(s, s^+) \coloneqq \mathbb{P}^\pi(s^+|s)$ for $s, s^+ \in \mathcal{S}$, to be the corresponding transition (probability) *matrix*. The value of a policy is the expected, discounted, accumulative cost of following this policy over an infinite-horizon trajectory: For the policy π, we define the value function $v^\pi \in \mathbb{R}^{|\mathcal{S}|} = \mathbb{R}^n$ with elements

$$v^\pi(s) \coloneqq \mathbb{E}_{s_{t+1}\sim\mathbb{P}^\pi(\cdot|s_t)}\big[\textstyle\sum_{t=0}^\infty \gamma^t c^\pi(s_t) \mid s_0 = s\big],$$

and the Q-function $q^\pi \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|} = \mathbb{R}^{nm}$ with elements

$$q^\pi(s, a) \coloneqq c(s, a) + \gamma\mathbb{E}_{s^+\sim\mathbb{P}(\cdot|s,a)}[v^\pi(s^+)],$$

so that we also have $v^\pi(s) = q^\pi\big((s, \pi(s)\big)$ for each $s \in \mathcal{S}$. Given a value function v, let us also define $\pi_v : \mathcal{S} \to \mathcal{A}$ by

$$\pi_v(s) \in \operatorname*{argmin}_{a\in\mathcal{A}} \left\{c(s, a) + \gamma\mathbb{E}_{s^+\sim\mathbb{P}(\cdot|s,a)}\left[v(s^+)\right]\right\},$$

to be the *greedy policy w.r.t. v*. Similarly, for a Q-function q, define $\pi_q : \mathcal{S} \to \mathcal{A}$ by

$$\pi_q(s) \in \operatorname*{argmin}_{a\in\mathcal{A}} q(s, a),$$

to be the *greedy policy w.r.t. q*. The problem of interest is to control the MDP optimally, that is, to find the optimal policy π^* with the optimal value/Q-function

$$v^\star = \min_\pi v^\pi, \quad q^\star = \min_\pi q^\pi, \tag{1}$$

so that the expected, discounted, infinite-horizon cost is minimized. Let us also note that the optimal policy, i.e., the minimizer of the preceding optimization problems, is the greedy policy w.r.t. $v^\star$ and $q^\star$, that is, $\pi^\star = \pi_{v^\star} = \pi_{q^\star}$.

Interestingly, the optimal value/Q-function introduced in (1) can be equivalently characterized as the fixed point of two different operators each of which is useful depending on the available information (oracle):

(i) *Model-based* control: When we have access to the transition kernel and the cost function, the problem is usually characterized by the fixed-point problem $v^\star = T(v^\star)$, i.e., for each $s \in \mathcal{S}$

$$v^\star(s) = [T(v^\star)](s), \tag{2}$$

where $T : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ is the *Bellman operator* given by

$$[T(v)](s) \coloneqq \min_{a\in\mathcal{A}} \left\{c(s, a) + \gamma\mathbb{E}_{s^+\sim\mathbb{P}(\cdot|s,a)}\left[v(s^+)\right]\right\}. \tag{3}$$

That is, $v^\star$ is the unique fixed-point of the Bellman operator T. The uniqueness follows from the fact that the operator T is a γ-contraction in ∞-norm. Observe that, in this case, T can be exactly computed given the model of the underlying MDP.

(ii) *Model-free* control: Alternatively, the model may not be known, and instead, one can generate samples. Examples of this are very large systems where identifying the model is prohibitively expensive but transitions between states can be observed and recorded, such as those in video games. This problem has been studied extensively in the reinforcement learning community and

is often characterized as the *expected* fixed-point problem $q^\star = \mathbb{E}_{s^+\sim\mathbb{P}}\left[\widehat{T}(q^\star, s^+)\right]$, i.e., for each $(s,a) \in \mathcal{S}\times\mathcal{A}$

$$q^\star(s,a) = \mathbb{E}_{s^+\sim\mathbb{P}(\cdot|s,a)}\left[[\widehat{T}(q^\star, s^+)](s,a)\right], \tag{4}$$

where $\widehat{T} : \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}\times\mathcal{S} \to \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$ is the *sampled* Bellman operator given by[2]

$$[\widehat{T}(q, \hat{s}^+)](s,a) \coloneqq c(s,a) + \gamma \min_{a^+\in\mathcal{A}} q(\hat{s}^+, a^+), \tag{5}$$

with $\hat{s}^+ \sim \mathbb{P}(\cdot|s,a)$ being a *sample* of the next state drawn from the distribution $\mathbb{P}(\cdot|s,a)$ for the pair (s,a).

2.2. **Optimization problem**

We now look at the root-finding characterization of the solution to convex optimization problems. Consider the minimization problem

$$\min_{x\in\mathbb{R}^\ell}\left\{f(x) = \mathbb{E}_{\xi\sim\mathbb{P}}[\widehat{f}(x,\xi)]\right\}, \tag{6}$$

where the function $\widehat{f} : \mathbb{R}^\ell \times \Xi \to \mathbb{R}$ and the probability distribution $\mathbb{P}$ over Ξ are such that the function $f : \mathbb{R}^\ell \to \mathbb{R}$ is twice continuously differentiable and strongly convex. Much like the control problem, this problem can be considered in two settings:

(i) *Deterministic* optimization: Assuming that $\mathbb{P}$ in (6) is known and the corresponding expectation can be computed, the unique minimizer $x^\star$ satisfies

$$\nabla f(x^\star) = 0. \tag{7}$$

(ii) *Stochastic* optimization: Now assume that $\mathbb{P}$ in (6) is unknown but can be sampled from. In this case, the minimizer $x^\star$ satisfies the expected root-finding problem

$$\mathbb{E}_{\xi\sim\mathbb{P}}[\nabla\widehat{f}(x^\star,\xi)] = 0, \tag{8}$$

where ∇ now denotes the partial derivative w.r.t. x. We note that, above, there is an underlying assumption that the differentiation w.r.t. x and expectation w.r.t. ξ can be operated in any order.

2.3. **Transformation**

Before providing the equivalence relations between optimization and control problems, let us provide an important result for the Bellman operator. For tabular MDPs, the Bellman operator is piece-wise affine. Indeed, we have $T(v) = \max_{\pi\in\Pi} c^\pi + \gamma P^\pi v$, where $\Pi = \{\pi : \mathcal{S}\to\mathcal{A}\}$ is the finite set of all deterministic control policies. Therefore, by Rademacher's Theorem, T is differentiable almost everywhere. In particular, we have (see Section 5.1 for the proof):

Lemma 2.1 (Jacobian of T)**.** *Let $\mathcal{S} = \{1,2,\dots,n\}$ and $\mathcal{A} = \{1,2,\dots,m\}$. If T is differentiable at v, then $\partial T(v) = \gamma P^{\pi_v}$, where π_v is the greedy policy w.r.t. v.*

[2]Strictly speaking, the provided sampled Bellman operator is the empirical version of the Bellman operator for the Q-function, given by $[T(q)](s,a) \coloneqq c(s,a) + \gamma\mathbb{E}_{s^+\sim\mathbb{P}(\cdot|s,a)}\left[\min_{a^+\in\mathcal{A}} q(s^+,a^+)\right]$ for $(s,a)\in\mathcal{S}\times\mathcal{A}$.

Using the preceding result, we can write $\partial(\mathrm{Id}-T)(v) = I - \gamma P(v)$, where $P(v) \coloneqq P^{\pi_v}$, i.e., the state transition matrix of the greedy policy π_{v_k} w.r.t. v_k and Id is the identity operator,. Then, comparing the characterizations (2) and (7), we can draw the following equivalence relations between deterministic optimization and model-based control:

$$
\begin{aligned}
&x \leftrightarrow v,\\
&\nabla f \to \mathrm{Id}-T, \quad T \to \mathrm{Id}-\nabla f,\\
&\nabla^2 f \to I - \gamma P, \quad P \to \gamma^{-1}(I - \nabla^2 f),
\end{aligned}
$$

where I is the identity matrix, and $P = P(v)$ is the transition matrix of the Markov chain under the greedy policy w.r.t. v. Similarly, for stochastic optimization and model-free control, the characterizations (4) and (8) point to the following equivalence relations:

$$
\begin{aligned}
&(x, \xi) \leftrightarrow (q, s^+),\\
&\nabla \widehat{f} \to \mathrm{Id}-\widehat{T}, \quad \widehat{T} \to \mathrm{Id}-\nabla \widehat{f},\\
&\nabla^2 \widehat{f} \to I - \gamma \widehat{P}, \quad \widehat{P} \to \gamma^{-1}(I - \nabla^2 \widehat{f}),
\end{aligned}
$$

where $\widehat{P} = \widehat{P}(q, \hat{s}^+) \in \mathbb{R}^{nm \times nm}$ is the synchronously *sampled* transition matrix of the Markov chain under π_q with elements[3]

$$
[\widehat{P}(q, \hat{s}^+)]\big((s,a),(s',a')\big) = \begin{cases} 1 & \text{if } s' = \hat{s}^+,\ a' = \pi_q(\hat{s}^+),\\ 0 & \text{otherwise,} \end{cases}
$$

for each $(s,a),(s',a') \in \mathcal{S} \times \mathcal{A}$, where $\hat{s}^+ \sim \mathbb{P}(\cdot|s,a)$ is again a *sample* of the next state drawn from the distribution $\mathbb{P}(\cdot|s,a)$ for the state-action pair (s,a).

3. **Equivalent Algorithms**

We now look at existing algorithms for optimization and control and their equivalence within the proposed framework. In particular, we show how the application of the proposed transformations on well-established optimization algorithms such as gradient descent, accelerated gradient descent, and Newton method leads to well-known control algorithms such as value iteration (VI), accelerated VI, and policy iteration (PI). We note that these equivalences have already been pointed out in the existing literature, however, mostly in a scattered way. An exception is [22] where the relation between *deterministic* optimization algorithms and *model-based* control algorithms are studied.

For tabular MDPs with $\mathcal{S} = \{1, \dots, n\}$ and $\mathcal{A} = \{1, \dots, m\}$, we have $v \in \mathbb{R}^n$ and $q \in \mathbb{R}^{nm}$ for the value function and the Q-function, respectively. Correspondingly, we have $T : \mathbb{R}^n \to \mathbb{R}^n$ with

$$
T(v) = \textstyle\sum_{s \in \mathcal{S}} [T(v)](s) \cdot e_s,
$$

[3]Once again, strictly speaking, $\widehat{P}(q, \hat{s}^+)$ is the empirical version of the *state-action* transition matrix $P(q) \in \mathbb{R}^{nm \times nm}$ of the Markov chain under the greedy policy π_q w.r.t. q, with elements $[P(q)]\big((s,a),(s',a')\big) = \mathbb{P}(s'|s,a)$ if $a' = \pi_q(s')$ and $= 0$ otherwise, for $(s,a),(s',a') \in \mathcal{S} \times \mathcal{A}$.

Deterministic optimization $(y = x)$	**Model-based control** $(y = v)$	**Stochastic optimization** $(y = x)$	**Model-free control** $(y = q)$
$g(x) \coloneqq \nabla f(x)$ $H(x) \coloneqq \nabla^2 f(x)$	$g(v) \coloneqq v - T(v)$ $H(v) \coloneqq I - \gamma P(v)$	$\widehat{g}_k(x) \coloneqq \nabla \widehat{f}(x, \hat{\xi}_k)$ $\widehat{H}_k(x) \coloneqq \nabla^2 \widehat{f}(x, \hat{\xi}_k)$	$\widehat{g}_k(q) \coloneqq q - \widehat{T}(q, \hat{s}_k^+)$ $\widehat{H}_k(q) \coloneqq I - \gamma \widehat{P}(q, \hat{s}_k^+)$
GD [35]	Relaxed VI [4, 33]	SGD [48]	QL [58]
$d_k = -\alpha_k g(y_k)$		$d_k = -\alpha_k \widehat{g}_k(y_k)$	
Polyak GD [42]	Momentum VI [21]	Momentum SGD [60]	Speedy QL [19], NeSA [13], Momentum QL [59]
$d_k = -\alpha_k g(y_k) + \beta_k d_{k-1}$		$\begin{cases} d'_{k-1} = \widehat{g}_k(y_{k-1}) - \widehat{g}_k(y_k) \\ d_k = -\alpha_k \widehat{g}_k(y_k) + \beta_k d'_{k-1} + \delta_k d_{k-1} \end{cases}$	
Nesterov GD [41]	Accelerated VI [21]		
$d_k = -\alpha_k g(y_k + \beta_k d_{k-1}) + \beta_k d_{k-1}$			
NM	PI [26]	SNR [49, 14]	Zap QL [14]
$d_k = -\left[H(y_k)\right]^{-1} g(y_k)$		$\begin{cases} D_k = (1-\beta_k) D_{k-1} + \beta_k \widehat{H}_k(y_k) \\ d_k = -\alpha_k D_k^{-1} \widehat{g}_k(y_k) \end{cases}$	

TABLE 2. Equivalent algorithms: The vector d_k is the update vector as in a generic iterative scheme $y_{k+1} = y_k + d_k$ for $k = 0, 1 \ldots$. The coefficients $\alpha_k, \beta_k, \delta_k > 0$ are step-sizes. The second row contains definitions of the mathematical objects used in the rows below. All the provided model-free control algorithms are synchronous, i.e., all the state-action pairs in the Q-function are updated at each iteration. (S)GD: (stochastic) gradient descent; VI: value iteration; NM: Newton method; PI: policy iteration; QL: Q-learning; SNR: stochastic Newton-Raphson.

where $e_s \in \mathbb{R}^n$ is the unit vector for the state $s \in \mathcal{S}$, and also $\widehat{T} : \mathbb{R}^{nm} \times \mathcal{S}^{nm} \to \mathbb{R}^{nm}$ with

$$\widehat{T}(q, \hat{s}^+) = \textstyle\sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} [\widehat{T}(q, \hat{s}^+)](s, a) \cdot e_{(s,a)},$$

where $e_{(s,a)} \in \mathbb{R}^{nm}$ is the unit vector for the state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$. Above, with some abuse of notation, $\hat{s}^+$ captures the dependence of the sampled Bellman operator $\widehat{T}$ on the specific samples $\hat{s}^+ \sim \mathbb{P}(\cdot|s, a)$, with one sample for each $(s, a) \in \mathcal{S} \times \mathcal{A}$.

Moreover, to ease the exposition, we see any iterative algorithm as

$$y_{k+1} = y_k + d_k, \quad k = 0, 1, \ldots$$

where $y_k = x_k$, v_k or q_k based on the context. In each setting, d_k represents the update vector between iterations k and $k + 1$. This form allows us to characterize algorithms in terms of d_k. A compact summary of this can be found in Table 2. Let us emphasize that these equivalences are merely in the update rules and correspond to the equivalent transformations of Table 1. In particular, they do not imply that these algorithms have the same convergence properties.

3.1. **First-order methods**

The celebrated gradient descent (GD) [35] method is characterized by $d_k = -\alpha_k \nabla f(x_k)$, where α_k is a properly chosen step-size. Applying the transformations of Table 1 on GD, we derive the so-called relaxed VI [33, 43, 21] with $d_k = -\alpha_k \big(v_k - T(v_k)\big)$, for model-based control. In particular, for

the constant step-size $\alpha_k = 1$, we have the standard VI algorithm $v_{k+1} = T(v_k)$ [4]. The stochastic counterpart of GD (SGD) [48] is characterized by $d_k = -\alpha_k \nabla \widehat{f}(x_k, \hat{\xi}_k)$. Under the transformations of Table 1, SGD leads to the *synchronous* Q-learning (QL) algorithm [58, 29] with[4]

$$d_k = -\alpha_k \big(q_k - \widehat{T}(q_k, \hat{s}_k^+)\big). \tag{9}$$

3.2. **Accelerated methods**

In the so-called momentum-based algorithms, the update vector d_k is specified by gradient oracles but also depends on d_{k-1}. One such algorithm is GD with Polyak momentum (Polyak GD) [42], a.k.a. heavy ball method, characterized by

$$d_k = -\alpha_k \nabla f(x_k) + \beta_k d_{k-1}.$$

Another well-known momentum-based algorithm is GD with Nesterov acceleration (Nesterov GD) [41] with update vector

$$d_k = -\alpha_k \nabla f(x_k + \beta_k d_{k-1}) + \beta_k d_{k-1}.$$

With a proper choice of the step-sizes α_k and β_k, these schemes can be shown to accelerate the convergence rate, compared to the standard GD, for particular classes of objective functions [42, 40]. The corresponding model-based control algorithms, using the transformations of Table 1, are *momentum VI* [21] with

$$d_k = -\alpha_k \big(v_k - T(v_k)\big) + \beta_k d_{k-1},$$

and *accelerated VI* [21] with

$$d_k = -\alpha_k \big(v_k + \beta_k d_{k-1} - T(v_k + \beta_k d_{k-1})\big) + \beta_k d_{k-1}.$$

However, the convergence of the preceding accelerated schemes is in general not guaranteed. In [21], the authors address this issue by *safeguarding*, i.e., combining the accelerated VI with the standard VI. For accelerating SGD, a direct combination of Polyak momentum or Nesterov acceleration with SGD has been shown to lead to no better (and even worse) performance in terms of convergence rate [60, 30]. At least, when it comes to almost sure convergence, [37] reports the same rate of convergence for SGD with Polyak momentum and SGD with Nesterov acceleration as for standard SGD. Nevertheless, modifications of momentum-based acceleration methods have led to a range of accelerated SGD algorithms with faster convergence rates with specific assumptions on the problem data [30, 36, 1]. The idea of using momentum to accelerate QL has also attracted some interest. In particular, applying the transformations of Table 1 on a generic momentum SGD [60] with

$$\begin{cases} d'_{k-1} = \nabla \widehat{f}(x_{k-1}, \hat{\xi}_k) - \nabla \widehat{f}(x_k, \hat{\xi}_k), \\ d_k = -\alpha_k \nabla \widehat{f}(x_k, \hat{\xi}_k) + \beta_k d'_{k-1} + \delta_k d_{k-1}, \end{cases}$$

[4] This is the so-called *synchronous* update of the Q-function in *all* state-action pairs in each iteration, corresponding to the *parallel sampling model* introduced by [29].

and step-sizes $\alpha_k, \beta_k, \delta_k > 0$, we obtain the *speedy QL* [19], *NeSA* [13], and *momentum QL* [59] algorithms with

$$\begin{cases} d'_{k-1} = \big(q_{k-1} - \widehat{T}(q_{k-1}, \hat{s}_k^+)\big) - \big(q_k - \widehat{T}(q_k, \hat{s}_k^+)\big), \\ d_k = -\alpha_k \big(q_k - \widehat{T}(q_k, \hat{s}_k^+)\big) + \beta_k d'_{k-1} + \delta_k d_{k-1}. \end{cases} \tag{10}$$

The difference between these three algorithms is in the choice of the step-sizes $\alpha_k, \beta_k, \delta_k > 0$.

3.3. **Second-order methods**

In second-order algorithms, d_k is specified by both the gradient and the Hessian oracles. The *damped* Newton method is one such algorithm with $d_k = -\alpha_k [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$. The *pure* Newton step with $\alpha_k = 1$, has a *local* quadratic convergence, if in addition to f being strongly convex, the Hessian is Lipschitz-continuous [11, Thm. 5.3]. Globally, however, the pure Newton method can lead to divergence. This is the reason behind introducing the step-size $\alpha_k < 1$ in the damped version, which can be used to guarantee a global linear convergence. We can use the transformations of Table 1 to transform the Newton method into a model-based control with $d_k = -\big(I - \gamma P(v_k)\big)^{-1}(v_k - T(v_k))$, where $P(v_k)$ is the transition matrix of the Markov chain under the greedy policy w.r.t. v_k. The derived model-based control algorithm then corresponds to the well-known PI algorithm [45] with $v_{k+1} = \big(I - \gamma P(v_k)\big)^{-1} c^{\pi_{v_k}}$, where $c^{\pi_{v_k}}$ is the vector of stage costs corresponding to the greedy policy π_{v_k} w.r.t. v_k. Indeed, the fact the Bellman operator is strongly semi-smooth everywhere has been used to show that the PI algorithm is an instance of the semi-smooth Newton method with a local quadratic convergence rate [17]. The second-order scheme has also been combined with the VI algorithm by using the *smooth* Bellman operation in which the maximization operation is approximated by a differentiable function, e.g., log-sum-exp [50]. This idea has been recently used to propose the generalized second-order VI with a quadratic convergence rate [28].

In the model-free case, the stochastic version of the Newton method [49] has been a source of inspiration for developing second-order-type Q-learning algorithms. In particular, the stochastic Newton-Raphson (SNR) [49] algorithm with

$$\begin{cases} D_k = (1 - \beta_k) D_{k-1} + \beta_k \nabla^2 \widehat{f}(x_k, \hat{\xi}_k), \\ d_k = -\alpha_k D_k^{-1} \nabla \widehat{f}(x_k, \hat{\xi}_k), \end{cases}$$

was used for developing the *zap QL* algorithm [14] with

$$\begin{cases} h_k = e_{(s_k, a_k)} - \gamma \, \mathbf{1}\big(\hat{s}_k^+, \pi_k(\hat{s}_k^+)\big), \\ \delta_k = q_k(s_k, a_k) - [\widehat{T}(q_k, \hat{s}_k^+)](s_k, a_k), \\ D_k = (1 - \beta_k) \, D_{k-1} + \beta_k \, e_{(s_k, a_k)} \, h_k^\top, \\ d_k = -\alpha_k \, D_k^{-1} \, \delta_k \, e_{(s_k, a_k)}, \end{cases}$$

where $\pi_k(\hat{s}_k^+) = \operatorname{argmin}_{a \in \mathcal{A}} q_k(\hat{s}_k^+, a)$ is the greedy action w.r.t. q_k evaluated at the sampled next sate $\hat{s}_k^+ \sim \mathbb{P}(\cdot | s_k, a_k)$. Note that the preceding algorithm involves updating one entry of the Q-function q_k at each iteration k, corresponding to the state-action pair (s_k, a_k) chosen at iteration k – recall that $e_{(s,a)} \in \mathbb{R}^{nm}$ is the unit vector corresponding to the state-action pair (s, a). The implementation

of zap QL algorithm with *synchronous* update of the Q-function in all state-action pairs in each iteration is then characterized by

$$\begin{cases} D_k = (1-\beta_k)D_{k-1} + \beta_k\big(I - \gamma\widehat{P}(q_k, \hat{s}_k^+)\big), \\ d_k = -\alpha_k D_k^{-1}\big(q_k - \widehat{T}(q_k, \hat{s}_k^+)\big), \end{cases} \tag{11}$$

where $\widehat{P}(q, \hat{s}^+)$ is the synchronously sampled state-action transition matrix of the Markov chain under the greedy policy w.r.t. q. Note that (11) is exactly the SNR algorithm under the transformations of Table 1.

4. **Quasi-Policy Iteration (QPI)**

While Newton method (NM) has a better convergence rate compared to gradient descent (GD), it suffers from a higher per-iteration computational cost. To be precise, consider again the unconstrained minimization problem $\min_{x\in\mathbb{R}^\ell} f(x)$, where f is twice continuously differentiable and strongly convex with a Lipschitz-continuous Hessian. Then, the GD update rule $x_{k+1} = x_k - \alpha_k\nabla f(x_k)$, with a proper choice of step-size α_k, converges *linearly* [11, Thm. 3.12] with $O(\ell)$ per-iteration complexity (disregarding the complexity of gradient oracle). On the other hand, the NM update rule $x_{k+1} = x_k - \alpha_k[\nabla^2 f(x_k)]^{-1}\nabla f(x_k)$, with a proper choice of step-size α_k, has a local *quadratic* convergence rate [11, Thm. 5.3] with $O(\ell^3)$ per-iteration complexity, assuming direct inversion (and disregarding the complexity of gradient and Hessian oracles).

Quasi-Newton methods (QNMs) are a class of methods that allow for a trade-off between computational complexity and (local) convergence rate. To do so, these methods use a Newton-type update rule

$$x_{k+1} = x_k - \alpha_k\widetilde{H}_k^{-1}\nabla f(x_k),$$

where $\widetilde{H}_k$ is an *approximation* of the true Hessian $\nabla^2 f(x_k)$ at iteration k. Different QNMs use different approximations of the Hessian. A generic approximation scheme in QNMs is

$$\widetilde{H}_k = \underset{H\in\mathbb{R}^{\ell\times\ell}}{\operatorname{argmin}} \|H - H_{\text{prior}}\|_F^2 \quad \text{s.t.} \quad Hr_i = b_i,\ i = 1,\dots,j, \tag{12}$$

which minimizes the distance (in Frobenius norm) to a given prior H_{prior} subject to j (≥ 1) linear constraints specified by $r_i, b_i \in \mathbb{R}^\ell$. This leads to the approximation $\widetilde{H}_k$ being a rank-j update of the prior H_{prior}, i.e.,

$$\widetilde{H}_k = H_{\text{prior}} + (B - H_{\text{prior}}R)(R^\top R)^{-1}R^\top,$$

where $R = (r_1,\dots,r_j)$, $B = (b_1,\dots,b_j) \in \mathbb{R}^{\ell\times j}$. Hence, $\widetilde{H}_k^{-1}$ can be easily computed based on H_{prior}^{-1} using the Woodbury formula. Different choices of the prior and the linear constraints in the generic approximation scheme above lead to different QNMs. For example, by choosing the so-called *secant conditions* with $r_i = x_{k-i+1} - x_{k-i}$ and $b_i = \nabla f(x_{k-i+1}) - \nabla f(x_{k-i})$ as linear constraints and $H_{\text{prior}} = I$ as the prior, we derive Anderson mixing with memory j [2], while by using a single secant condition with $j = 1$ and choosing $H_{\text{prior}} = \widetilde{H}_{k-1}$ as the prior, we derive QNM with Broyden approximation [10].

In this section, following a similar idea, we propose the *quasi-policy iteration (QPI)* algorithm by incorporating a computationally efficient approximation of the "Hessian" $H = I - \gamma P$ in the PI algorithm. We note that the authors in [18, 61, 53] also propose the combination of Anderson mixing with optimal control algorithms. However, the QPI algorithm is fundamentally different in the sense that it approximates the transition matrix P using a different set of constraints that are specific to the optimal control algorithms.

4.1. **QPI Algorithm**

For $k \in \{0, 1, 2, \dots\}$, let

$$c_k \coloneqq c^{\pi_{v_k}}, \quad P_k \coloneqq P^{\pi_{v_k}}, \quad T_k \coloneqq T(v_k).$$

(Recall that $c^{\pi_{v_k}}$ and $P^{\pi_{v_k}}$ are the stage cost and the state transition matrix of the greedy policy π_{v_k} w.r.t. v_k, respectively.) Recall the PI update rule

$$v_{k+1} = (I - \gamma P_k)^{-1} c_k = v_k - (I - \gamma P_k)^{-1}(v_k - T_k).$$

Inspired by the QNM approximation scheme (12), we propose the generic QPI update rule

$$v_{k+1} = v_k - (I - \gamma \widetilde{P}_k)^{-1}(v_k - T_k), \tag{13}$$

where

$$\widetilde{P}_k = \underset{P \in \mathbb{R}^{n \times n}}{\operatorname{argmin}} \|P - P_{\text{prior}}\|_F^2 \quad \text{s.t.} \quad Pr_i = b_i, \ i = 1, \dots, j. \tag{14}$$

Observe that instead of approximating the complete Hessian $H_k \coloneqq (I - \gamma P_k)^{-1}$ similar to standard QNMs, we are only approximating P_k. This choice particularly allows us to exploit the problem structure in order to form novel constraints and prior as we discuss next.

Regarding the constraints, the problem structure gives us two linear equality constraints: First, P_k is a row stochastic matrix, i.e.,

$$P_k \mathbf{1} = \mathbf{1}, \tag{15}$$

and hence we can set $r_1 = b_1 = \mathbf{1}$. Second, we can use the fact that the Bellman operator T is piece-wise affine. In particular, from the definition (3) of the Bellman operator, it follows that $T(v) = c^{\pi_v} + \gamma P^{\pi_v} v$. Thus,

$$T_k = c_k + \gamma P_k v_k \ \Rightarrow \ P_k v_k = \gamma^{-1}(T_k - c_k), \tag{16}$$

and we can set $r_2 = v_k$ and $b_2 = \gamma^{-1}(T_k - c_k)$. Note that, unlike the standard secant conditions in QNMs, the constraints (15) and (16) hold *exactly*. Incorporating these constraints, we propose the approximation

$$\widetilde{P}_k = \underset{P \in \mathbb{R}^{n \times n}}{\operatorname{argmin}} \|P - P_{\text{prior}}\|_F^2 \ \text{ s.t. } \ P\mathbf{1} = \mathbf{1}, \ Pv_k = \gamma^{-1}(T_k - c_k). \tag{17}$$

The update rule (13) using the approximation (17) is, however, not necessarily a contraction. The same problem also arises in similar algorithms such as Anderson accelerated VI [61] and Nesterov

accelerated VI [21]. Here, we follow the standard solution for this problem, that is, safeguarding the QPI update against the standard VI update based on the Bellman error

$$\theta_k := \|v_k - T_k\|_\infty .$$

To be precise, at each iteration $k = 0, 1, \ldots$, we consider the *safeguarded* QPI update rule as follows

$$\begin{array}{ll} \text{(QPI)} & \text{compute } v_{k+1} \text{ according to (13), (17);} \\ \text{(Safeguard)} & \textbf{if } \theta_{k+1} > \gamma^{k+1}\theta_0, \textbf{ then } v_{k+1} = T_k. \end{array} \tag{18}$$

The following theorem summarizes the discussion above by providing the QPI update rule explicitly (see Section 5.2 for the proof).

Theorem 4.1 (QPI convergence & complexity)**.** *Consider the update rule* (13) *using the approximation* (17) *where* $P_{\text{prior}}\mathbf{1} = \mathbf{1}$ *and let* $G_{\text{prior}} = (I - \gamma P_{\text{prior}})^{-1}$*. We have*

$$v_{k+1} = v_k - \widetilde{G}_k(v_k - T_k), \tag{19a}$$

where

$$\begin{gathered} w_k = T_k - c_k - \gamma P_{\text{prior}} v_k,\ \check{w}_k = G_{\text{prior}} w_k \in \mathbb{R}^n, \quad u_k = v_k - \tfrac{\mathbf{1}^\top v_k}{n}\mathbf{1},\ \check{u}_k = G_{\text{prior}}^\top u_k \in \mathbb{R}^n, \\ \tau_k = \begin{cases} 0 & \text{if } u_k^\top v_k = 0, \\ (u_k^\top v_k)^{-1} & \text{otherwise} \end{cases} \in \mathbb{R}, \quad \eta_k = \begin{cases} 0 & \text{if } u_k^\top v_k = 0, \\ \big(u_k^\top (v_k - \check{w}_k)\big)^{-1} & \text{otherwise} \end{cases} \in \mathbb{R}, \\ \widetilde{P}_k = P_{\text{prior}} + \gamma^{-1}\tau_k w_k u_k^\top, \quad \widetilde{G}_k = G_{\text{prior}} + \eta_k \check{w}_k \check{u}_k^\top. \end{gathered} \tag{19b}$$

Moreover, each iteration of the QPI update rule (19) *with the safeguarding* (18) *is a* γ*-contraction in the* ∞*-norm and has a time complexity of* $\mathcal{O}(n^2 m)$*.*

Observe that the safeguarded QPI update rule has the same per-iteration complexity as VI. Moreover, the convergence of QPI is ensured via safeguarding against VI, which leads to the same theoretically guaranteed linear convergence with rate γ as for VI. However, as we will show in the numerical examples below, we observe an empirically faster convergence for QPI with its rate showing less sensitivity to γ similar to PI. We also note that there is also a one-time computational cost of $O(n^3)$ in the QPI update rule (19) for computing G_{prior} (assuming direct inversion and if G_{prior} is not available in closed form). The pseudo-code for the safeguarded QPI is provided in Algorithm 1 in Appendix C.

Next to be addressed is the choice of the prior P_{prior}. First of all, note that for a fixed prior in all iterations, computation of τ_k and $\widetilde{P}_k$ in (19b) is not needed since the update (19a) only requires $\widetilde{G}_k$. The first choice for such a fixed prior is to exploit the available knowledge on the structure of the MDP. For instance, one can set $P_{\text{prior}} = P^\mu$ with μ being the stochastic policy choosing actions uniformly at random (so that P_{prior} is the average over actions of and has the same sparsity pattern as the true transition kernel of the MDP). A computationally advantageous choice for the prior is the uniform distribution $P_{\text{prior}} = \frac{1}{n}E = \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ for which the update rule can be simplified significantly (see Section 5.3 for the proof):

Corollary 4.2 (QPI with uniform prior)**.** *For the uniform prior* $P_{\text{prior}} = \frac{1}{n}E$*, the QPI update rule* (19) *equivalently reads as*

$$v_{k+1} = (1-\delta_k)T_k + \delta_k c_k + \lambda_k \mathbf{1}, \tag{20a}$$

where the scalar coefficients are given by

$$\begin{aligned} &z_k = c_k - \tfrac{\mathbf{1}^\top c_k}{n}\mathbf{1} \in \mathbb{R}^n, \quad g_k = v_k - T_k \in \mathbb{R}^n, \quad y_k = g_k - \tfrac{\mathbf{1}^\top g_k}{n}\mathbf{1} \in \mathbb{R}^n, \\ &\delta_k = \begin{cases} 0 & \text{if } v_k^\top(y_k + z_k) = 0, \\ \frac{v_k^\top y_k}{v_k^\top (y_k+z_k)} & \text{otherwise} \end{cases} \in \mathbb{R}, \quad \lambda_k = \tfrac{\gamma}{n(1-\gamma)}\mathbf{1}^\top\big((\delta_k - 1)g_k + \delta_k c_k\big) \in \mathbb{R}. \end{aligned} \tag{20b}$$

Observe that the QPI update rule (20a) is a modification of the standard VI update rule $v_{k+1} = T(v_k)$ using two new vectors, namely, $c_k - T(v_k)$ and the all-one vector $\mathbf{1}$, with adaptive coefficients δ_k and λ_k, respectively.

Another interesting choice for the prior in (17) is $P_{\text{prior}} = \widetilde{P}_{k-1}$, i.e., the previous approximation. This leads to a recursive scheme for approximating the transition matrix similar to QNM with Broyden approximation [10]. This can be achieved by choosing an initialization $\widetilde{P}_{-1}$ such that $\widetilde{P}_{-1}\mathbf{1} = \mathbf{1}$ and defining $\widetilde{G}_{-1} \coloneqq (1-\gamma\widetilde{P}_{-1})^{-1}$ (e.g., $\widetilde{P}_{-1} = \frac{1}{n}E$ and $\widetilde{G}_{-1} = I + \frac{\gamma}{n(1-\gamma)}E$).

We finish this section with the following remark.

Remark 4.3 (Other constraints)**.** *One can also add extra constraints to the minimization problem* (17) *to impose a particular structure on the approximate transition matrix* $\widetilde{P}_k$*. For instance, a natural constraint is to require this matrix to be entry-wise non-negative so that* $\widetilde{P}_k$ *is indeed a probability transition matrix; or, one can impose a sparsity pattern on* $\widetilde{P}_k$ *using the prior knowledge on the structure of the MDP. However, incorporating such information may lead to the problem* (17) *not having a closed-form and/or low-rank solution, undermining the computational efficiency of the proposed algorithm. In this regard, we note that the problem* (17) *has a closed-form solution which is a rank-one update of the prior; see* $\widetilde{P}_k$ *in* (19b)*.*

4.2. **Extension to model-free control: QPL algorithm**

We now introduce the quasi-policy learning (QPL) algorithm as the extension of QPI for model-free control problems with access to samples through a generative model. For simplicity, we limit the following discussion to the extension of the QPI algorithm (20) with a uniform prior. However, we note that the extension can be similarly applied to the QPI algorithm (19) with the generic prior.

The basic idea is to implement the stochastic version of the QPI update rule *for the Q-function* using the samples. In particular, similar to the approximation (17), we use an approximation of the state-action transition matrix under the greedy policy w.r.t. the Q-function q_k at each iteration k, where the second equality constraint is formed based on the *sampled* Bellman operator $\widehat{T}(\cdot, \hat{s}_k^+)$, evaluated at the sampled next states $\hat{s}_k^+$ at iteration k, as a surrogate for the Bellman operator $T(\cdot)$. To be precise, let

$$\widehat{T}_k \coloneqq \widehat{T}(q_k, \hat{s}_k^+),$$

at each iteration k. Also, let $c \in \mathbb{R}^{nm}$ be the vector of stage cost (with the same state-action ordering as the Q-function $q_k \in \mathbb{R}^{nm}$). We note that since the proposed QPL algorithm is synchronous with one sample for each state-action pair in each iteration, we have access to the complete stage cost c after the first iteration and can treat it as an input to the algorithm. The approximate state-action transition matrix $\widetilde{P}_k$ at each iteration k is then formed as follows

$$\widetilde{P}_k = \underset{P \in \mathbb{R}^{nm \times nm}}{\operatorname{argmin}} \ \|P - P_{\text{prior}}\|_F^2 \ \text{ s.t. } \ P\mathbf{1} = \mathbf{1}, \ Pq_k = \gamma^{-1}\big(\widehat{T}_k - c\big). \tag{21}$$

The minimization problem above also has a closed-form solution as a rank-one update of the prior, which allows us to compute $\widetilde{G}_k = (I - \gamma \widetilde{P}_k)^{-1}$ using Woodbury formula. In particular, by using the uniform prior $P_{\text{prior}} = \frac{1}{nm} E$, the update rule of the model-free QPL algorithm is

$$q_{k+1} = (1 - \alpha_k) q_k + \alpha_k \big((1 - \delta_k)\widehat{T}_k + \delta_k c + \lambda_k \mathbf{1}\big), \tag{22a}$$

where

$$\begin{aligned} &z = c - \tfrac{\mathbf{1}^\top c}{nm}\mathbf{1} \in \mathbb{R}^{nm}, \quad \widehat{g}_k = q_k - \widehat{T}_k \in \mathbb{R}^{nm}, \quad y_k = \widehat{g}_k - \tfrac{\mathbf{1}^\top \widehat{g}_k}{nm}\mathbf{1} \in \mathbb{R}^{nm}, \\ &\delta_k = \begin{cases} 0 & \text{if } q_k^\top (y_k + z) = 0, \\ \frac{q_k^\top y_k}{q_k^\top (y_k + z)} & \text{otherwise} \end{cases} \in \mathbb{R}, \quad \lambda_k = \tfrac{\gamma}{nm(1-\gamma)} \mathbf{1}^\top \big((\delta_k - 1)\widehat{g}_k + \delta_k c\big) \in \mathbb{R}. \end{aligned} \tag{22b}$$

and α_k is the diminishing learning rate of the algorithm, e.g., $\alpha_k = 1/(k+1)$. Compared to the standard Q-learning (QL) update rule, i.e., $q_{k+1} = (1 - \alpha_k) q_k + \alpha_k \widehat{T}_k$, QPL uses the two additional vectors $c - \widehat{T}_k$ and the all-one vector $\mathbf{1}$ with adaptive coefficients in its update rule.

Similar to the model-based case, the proposed QPL update rule is not necessarily convergent. To address this issue, we again use the basic idea of safeguarding. However, in this case, we safeguard the QPL update against the standard QL update based on the *sampled* Bellman error

$$\widehat{\theta}_k \coloneqq \|q_k - \widehat{T}_k\|_\infty.$$

To be precise, we run a QL algorithm

$$q_{k+1}^{\text{QL}} = (1 - \alpha_k) q_k^{\text{QL}} + \alpha_k \widehat{T}_k^{\text{QL}},$$

in parallel with the QPL algorithm using the same initialization $q_0^{\text{QL}} = q_0$ and the same samples for computing the corresponding sampled Bellman operator and error

$$\widehat{T}_k^{\text{QL}} \coloneqq \widehat{T}(q_k^{\text{QL}}, \hat{s}_k^+), \quad \widehat{\theta}_k^{\text{QL}} \coloneqq \|q_k^{\text{QL}} - \widehat{T}_k^{\text{QL}}\|_\infty.$$

We then use the sampled Bellman error of QL to *safeguard* the QPL update rule as follows

$$\begin{aligned} &\text{(QPL)} && \text{compute } q_{k+1} \text{ according to (22);} \\ &\text{(Safeguard)} && \textbf{if } k > K_{\textbf{sg}} \textbf{ and } \textstyle\sum_{i=0}^{k+1} \widehat{\theta}_i > \sum_{i=0}^{k+1} \widehat{\theta}_i^{\text{QL}} \textbf{ then } q_{k+1} = (1 - \alpha_k) q_k + \alpha_k \widehat{T}_k \ . \end{aligned} \tag{23}$$

Note that once the safeguard is activated, the QPL update follows a standard QL step, as opposed to the QPL step, using its own last iterate q_k (and not the last iterate q_k^{QL} of the QL algorithm that is running in parallel). Moreover, in order to increase the robustness against the stochasticity of the samples, the safeguard is activated (i) after K_{sg} iterations and (ii) based on the *accumulated*

sampled Bellman errors over the entire history of iterations. The following theorem summarizes properties of the proposed QPL algorithm (see Section 5.4 for the proof).

Theorem 4.4 (QPL convergence & complexity)**.** *The safeguarded QPL algorithm* (23) *has a per-iteration complexity of* $\mathcal{O}(nm^2)$ *and converges with at least the same rate as QL.*

Regarding the preceding result, we note that the per-iteration time complexity of QPL *with safeguard* is the same as that of the (synchronous) QL algorithm. Algorithm 2 in Appendix C provides the pseudo-code for the safeguarded QPL. We finish this section with the following remark on the *asynchronous* implementation of QPL.

Remark 4.5 (Asynchronous QPL)**.** *The proposed QPL update rule* (22) *can also be implemented in an asynchronous fashion. To be precise, this requires forming the approximate state-action transition matrix* $\widetilde{P}_k$ *based on a single sample* $(s_k, a_k, \hat{s}_k^+, c(s_k, a_k))$ *as follows*

$$\widetilde{P}_k = \operatorname*{argmin}_{P \in \mathbb{R}^{nm \times nm}} \|P - P_{\text{prior}}\|_F^2 \quad s.t. \quad P\mathbf{1} = \mathbf{1},\ e_{(s_k,a_k)}^\top P q_k = \gamma^{-1}\big(\widehat{T}_k(s_k, a_k) - c(s_k, a_k)\big).$$

Cf. approximation (21)*. In particular, by using the uniform prior* $P_{\text{prior}} = \frac{1}{nm}E$*, the corresponding update rule reads as*

$$q_{k+1} = q_k - \alpha_k(1 + \delta_k)\big(q_k(s_k, a_k) - \widehat{T}_k(s_k, a_k)\big)\big(e_{(s_k,a_k)} + \beta\mathbf{1}\big),$$

where the scalar coefficients are given by

$$\begin{aligned}
&\beta = \tfrac{\gamma}{nm(1-\gamma)}, \quad \rho_k = \tfrac{1}{nm}\mathbf{1}^\top q_k,\\
&\lambda_k = \big(\widehat{T}_k(s_k, a_k) - c(s_k, a_k) - \gamma\rho_k\big)\big(q_k(s_k, a_k) - \rho_k\big),\\
&\eta_k = \|q_k\|_2^2 - \rho_k^2 - \lambda_k, \quad \delta_k = \begin{cases} 0 & \text{if } \eta_k = 0,\\ \lambda_k/\eta_k & \text{otherwise.}\end{cases}
\end{aligned}$$

Note that the preceding update rule, as expected and similar to ZQL, leads to an update in all entries of the Q-function q_k *in each iteration.*

5. Technical Proofs

5.1. Proof of Lemma 2.1

Let us define the matrix $P^a \in \mathbb{R}^{n\times n}$ with entries $P^a(s, s^+) = \mathbb{P}(s^+|s, a)$, for every control action $a \in \mathcal{A}$. Fix $s \in \mathcal{S}$, and observe that

$$\partial_v\big([T(v)](s)\big) = \partial_v\left(\min_{a\in\mathcal{A}}\big\{c(s, a) + \gamma\mathbb{E}_{\mathbb{P}(\cdot|s,a)}[v(s^+)]\big\}\right) = \partial_v\left(\min_{a\in\mathcal{A}}\{c(s, a) + \gamma \cdot P^a(s, \cdot) \cdot v\}\right),$$

where $P^a(s, \cdot)$ is the s-th row of P^a. Then, using the envelope theorem [51], we have

$$\partial_v\big([T(v)](s)\big) = \partial_v\Big(c\big(s, \pi_v(s)\big) + \gamma \cdot P^{\pi_v}(s, \cdot) \cdot v\Big) = \gamma \cdot P^{\pi_v}(s, \cdot).$$

Hence, $\partial T(v) = \gamma P^{\pi_v}$.

5.2. **Proof of Theorem 4.1**

First, let us show that the two equality constraints in the minimization problem (17) are linearly dependent if and only if $u_k^\top v_k = 0$. In this regard, observe that the two equality constraints are linearly dependent if and only if $v_k = \rho\mathbf{1}$ for some $\rho \in \mathbb{R}$: For $\rho = 0$, the second equality constraint becomes trivial; and, for $\rho \neq 0$, the two constraints become equivalent. On the other hand, we have

$$u_k^\top v_k = \big(v_k - \frac{\mathbf{1}^\top v_k}{n}\mathbf{1}\big)^\top v_k = v_k^\top (I - \frac{1}{n}E)v_k.$$

Then, since $I - \frac{1}{n}E$ is positive semi-definite with one zero eigenvalue corresponding to the eigenvector $\mathbf{1}$, we have $u_k^\top v_k = 0$ if and only if $v_k = \rho\mathbf{1}$ for some $\rho \in \mathbb{R}$. Hence, the constraints in (17) are linearly dependent if and only if $u_k^\top v_k = 0$.

We first consider the update rule (19) for the case $u_k^\top v_k \neq 0$. Define $R \coloneqq (\mathbf{1}, v_k)$, $B \coloneqq \big(\mathbf{1}, \gamma^{-1}(T_k - c_k)\big) \in \mathbb{R}^{n\times 2}$ so that the minimization problem (17) can be written as

$$\widetilde{P}_k = \operatorname*{argmin}_{P\in\mathbb{R}^{n\times n}} \left\{ \|P - P_{\text{prior}}\|_F^2 \ : \ PR = B \right\}.$$

Note that, since $u_k^\top v_k \neq 0$ and hence v_k and $\mathbf{1}$ are linearly independent, R is of full column rank. The solution to the preceding problem is given by

$$\widetilde{P}_k = P_{\text{prior}} + (B - P_{\text{prior}}R)(R^\top R)^{-1}R^\top.$$

Now, observe that

$$\begin{aligned}(B - P_{\text{prior}}R) &= \begin{bmatrix} \mathbf{1} & \frac{1}{\gamma}(T_k - c_k) \end{bmatrix} - P_{\text{prior}} \begin{bmatrix} \mathbf{1} & v_k \end{bmatrix} = \begin{bmatrix} \mathbf{1} - P_{\text{prior}}\mathbf{1} & \frac{1}{\gamma}(T_k - c_k) - P_{\text{prior}}v_k \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0} & \frac{1}{\gamma}(T_k - c_k) - P_{\text{prior}}v_k \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \gamma^{-1}w_k \end{bmatrix},\end{aligned}$$

where we used the assumption $P_{\text{prior}}\mathbf{1} = \mathbf{1}$. Also,

$$(R^\top R)^{-1} = \left(\begin{bmatrix} \mathbf{1}^\top \\ v_k^\top \end{bmatrix} \begin{bmatrix} \mathbf{1} & v_k \end{bmatrix} \right)^{-1} = \begin{bmatrix} n & v_k^\top \mathbf{1} \\ v_k^\top \mathbf{1} & v_k^\top v_k \end{bmatrix}^{-1} = \frac{1}{n(u_k^\top v_k)} \begin{bmatrix} v_k^\top v_k & -\mathbf{1}^\top v_k \\ -\mathbf{1}^\top v_k & n \end{bmatrix}.$$

and

$$(R^\top R)^{-1}R^\top = \frac{1}{u_k^\top v_k} \begin{bmatrix} * \\ u_k^\top \end{bmatrix}.$$

Therefore, we have

$$\widetilde{P}_k = P_{\text{prior}} + \gamma^{-1}(u_k^\top v_k)^{-1} w_k u_k^\top.$$

For the case $u_k^\top v_k = 0$, as we discussed in the beginning of the proof, the two equality constraints in the minimization problem (17) become linearly dependent, and, in particular, the second constraint can be discarded. The solution to the problem (17) in this case is then $\widetilde{P}_k = P_{\text{prior}}$. Hence, the approximation (17) can be in general written as

$$\widetilde{P}_k = P_{\text{prior}} + \gamma^{-1}\tau_k w_k u_k^\top, \tag{24}$$

where

$$\tau_k = \begin{cases} 0 & \text{if } u_k^\top v_k = 0, \\ (u_k^\top v_k)^{-1} & \text{otherwise.} \end{cases}$$

That is, the approximation $\widetilde{P}_k$ is a rank-one update of the prior. Then, if $\tau_k = 0$, we clearly have

$$\widetilde{G}_k = (I - \gamma \widetilde{P}_k)^{-1} = (I - \gamma P_{\text{prior}})^{-1} = G_{\text{prior}},$$

and, for the case $\tau_k \neq 0$, we can use the the Woodbury formula to write

$$\widetilde{G}_k = (G_{\text{prior}}^{-1} - \tau_k w_k u_k^\top)^{-1} = G_{\text{prior}} + \frac{1}{\tau_k^{-1} - u_k^\top (G_{\text{prior}} w_k)} (G_{\text{prior}} w_k)(G_{\text{prior}}^\top u_k)^\top = G_{\text{prior}} + \eta_k \check{w}_k \check{u}_k^\top.$$

What remains to be shown is that η_k is well-defined for $u_k^\top v_k \neq 0$ (i.e., $\tau_k \neq 0$). First, we use $T_k = c_k + \gamma P_k v_k$ to write

$$\eta_k^{-1} = u_k^\top (v_k - \check{w}_k) = u_k^\top \big(v_k - G_{\text{prior}}(T_k - c_k - \gamma P_{\text{prior}} v_k)\big) = u_k^\top \big(v_k - G_{\text{prior}}(\gamma P_k v_k - \gamma P_{\text{prior}} v_k)\big),$$

Next, since $P_{\text{prior}} = \gamma^{-1}(I - G_{\text{prior}}^{-1})$ and using the fact that $v_k = u_k + \frac{\mathbf{1}^\top v_k}{n}\mathbf{1}$, we have

$$\begin{aligned} \eta_k^{-1} &= u_k^\top \big(I - \gamma G_{\text{prior}}(P_k - P_{\text{prior}})\big) v_k = u_k^\top G_{\text{prior}}(I - \gamma P_k) v_k \\ &= u_k^\top G_{\text{prior}}(I - \gamma P_k)(u_k + \alpha_k \mathbf{1}) = u_k^\top G_{\text{prior}}(I - \gamma P_k) u_k + \frac{\mathbf{1}^\top v_k}{n} u_k^\top G_{\text{prior}}(I - \gamma P_k)\mathbf{1}. \end{aligned}$$

Now, note that $P_k \mathbf{1} = \mathbf{1}$ and $G_{\text{prior}}\mathbf{1} = (1-\gamma)^{-1}\mathbf{1}$. Hence,

$$\eta_k^{-1} = u_k^\top G_{\text{prior}}(I - \gamma P_k) u_k + \frac{\mathbf{1}^\top v_k}{n} u_k^\top \mathbf{1} = u_k^\top G_{\text{prior}}(I - \gamma P_k) u_k,$$

where we also used the fact that $u_k^\top \mathbf{1} = 0$. Then, since the matrices G_{prior} and $(I - \gamma P_k)$ are non-singular with their eigenvalues having strictly positive real parts (because the eigenvalues of P_k all reside within the unit disc), we have

$$u_k^\top G_{\text{prior}}(I - \gamma P_k) u_k = 0 \Leftrightarrow u_k = \mathbf{0} \Leftrightarrow v_k = \rho \mathbf{1} \;\text{ for some }\; \rho \in \mathbb{R} \Leftrightarrow u_k^\top v_k = 0.$$

That is, η_k is well-defined for $u_k^\top v_k \neq 0$.

Next, we consider the rate of convergence of the safeguarded QPI update rule. Observe that since T is a γ-contraction in ∞-norm, the safeguarding using standard VI as in (18) implies that

$$\|v_{k+1} - T_{k+1}\|_\infty \leq \max\{\gamma^{k+1} \|v_0 - T_0\|_\infty,\ \gamma \|v_k - T_k\|_\infty\}$$

for all $k \geq 0$. This ensures a linear convergence with rate γ.

Finally, the per-iteration time complexity of each iteration of the safeguarded QPI update rule: The update rule (19a) requires $\mathcal{O}(n^2 m)$ operations for computing the vectors $T_k = T(v_k)$, $\mathcal{O}(n^2)$ operations for the matrix-vector multiplication, and $\mathcal{O}(n)$ operations for the vector additions. Computing the objects in (19b) involves vector/matrix additions and matrix-vector multiplications (all of size n) and hence requires $\mathcal{O}(n^2)$ operations. For the safeguarding (18), we need to compute $T_{k+1} = T(v_{k+1})$ which again requires $\mathcal{O}(n^2 m)$ operations. Summing up the aforementioned complexities, we derive the total time complexity to be $\mathcal{O}(n^2 m)$.

5.3. **Proof of Corollary 4.2**

The result follows from Theorem 4.1 by plugging in $P_{\text{prior}} = \frac{1}{n}E$ and $G_{\text{prior}} = I + \frac{\gamma}{n(1-\gamma)}E$ and simplifying the expression. In particular, we note that the condition $v_k^\top(y_k + z_k) = 0$ in (20b) is equivalent to the condition $u_k^\top v_k = 0$ in (19b). To see this, recall that $u_k^\top v_k = 0$ if and only if $v_k = \rho\mathbf{1}$ for some $\rho \in \mathbb{R}$; see the first part of the proof of Theorem 4.1 in Section 5.2. Also, observe that

$$\begin{aligned} v_k^\top(y_k + z_k) &= v_k^\top\big(g_k + c_k - \frac{\mathbf{1}^\top(g_k + c_k)}{n}\mathbf{1}\big) = v_k^\top\big(v_k - T_k + c_k - \frac{\mathbf{1}^\top(v_k - T_k + c_k)}{n}\mathbf{1}\big) \\ &= v_k^\top(I - \frac{1}{n}E)(v_k - T_k + c_k) = v_k^\top(I - \frac{1}{n}E)(I - \gamma P_k)v_k, \end{aligned}$$

where, for the last equality, we used $T_k = c_k + \gamma P_k v_k$. Then, since $u_k = (I - \frac{1}{n}E)v_k = v_k - \frac{\mathbf{1}^\top v_k}{n}\mathbf{1}$, we have

$$\begin{aligned} v_k^\top(y_k + z_k) &= u_k^\top(I - \gamma P_k)(u_k + \frac{\mathbf{1}^\top v_k}{n}\mathbf{1}) = u_k^\top(I - \gamma P_k)u_k + \frac{\mathbf{1}^\top v_k}{n}u_k^\top(I - \gamma P_k)\mathbf{1} \\ &= u_k^\top(I - \gamma P_k)u_k + \frac{\mathbf{1}^\top v_k}{n}(1 - \gamma)u_k^\top\mathbf{1} = u_k^\top(I - \gamma P_k)u_k, \end{aligned}$$

where, for the last equality, we used the fact that $u_k^\top\mathbf{1} = 0$. Finally, since $(I - \gamma P_k)$ is non-singular with its eigenvalues having strictly positive real parts (because the eigenvalues of P_k all reside within the unit disc), we have

$$v_k^\top(y_k + z_k) = 0 \Leftrightarrow u_k = \mathbf{0} \Leftrightarrow v_k = \rho\mathbf{1} \quad \text{for some} \quad \rho \in \mathbb{R}.$$

This completes the proof.

5.4. **Proof of Theorem 4.4**

The per-iteration time complexity of each iteration of the safeguarded QPL update rule: The update rule (22) requires $\mathcal{O}(nm^2)$ operations for computing the vectors $\widehat{T}_k = \widehat{T}(q_k, \hat{s}_k^+)$, and $\mathcal{O}(nm)$ operations for computing the step-sizes δ_k and λ_k and the vector additions. For the safeguarding (23), we need to run a QL algorithm in parallel which also has a $\mathcal{O}(nm^2)$ per-iteration complexity. Moreover, we need to compute $\widehat{T}_{k+1}$ (for computing $\widehat{\theta}_{k+1}$) which again requires $\mathcal{O}(nm^2)$ operations. Summing up the aforementioned complexities, the total time complexity is $\mathcal{O}(nm^2)$.

Regarding the convergence, observe that the safeguarding ensures that the accumulated sampled Bellman for QPL is dominated by that of QL ran in parallel, that is, $\sum_{i=0}^k \widehat{\theta}_i \leq \sum_{i=0}^k \widehat{\theta}_i^{\text{QL}}$ for all $k > K_{\text{sg}}$, and, if not, it replaces the QPL update by a QL update. This ensures the convergence of QPL with at least the same rate as QL.

6. **Numerical Simulations**

We now compare the performance of the proposed algorithms with that of the standard existing algorithms for the optimal control of different MDPs. See Appendix A for a description of the considered MDPs. To this end, we first focus on the proposed algorithms with uniform priors

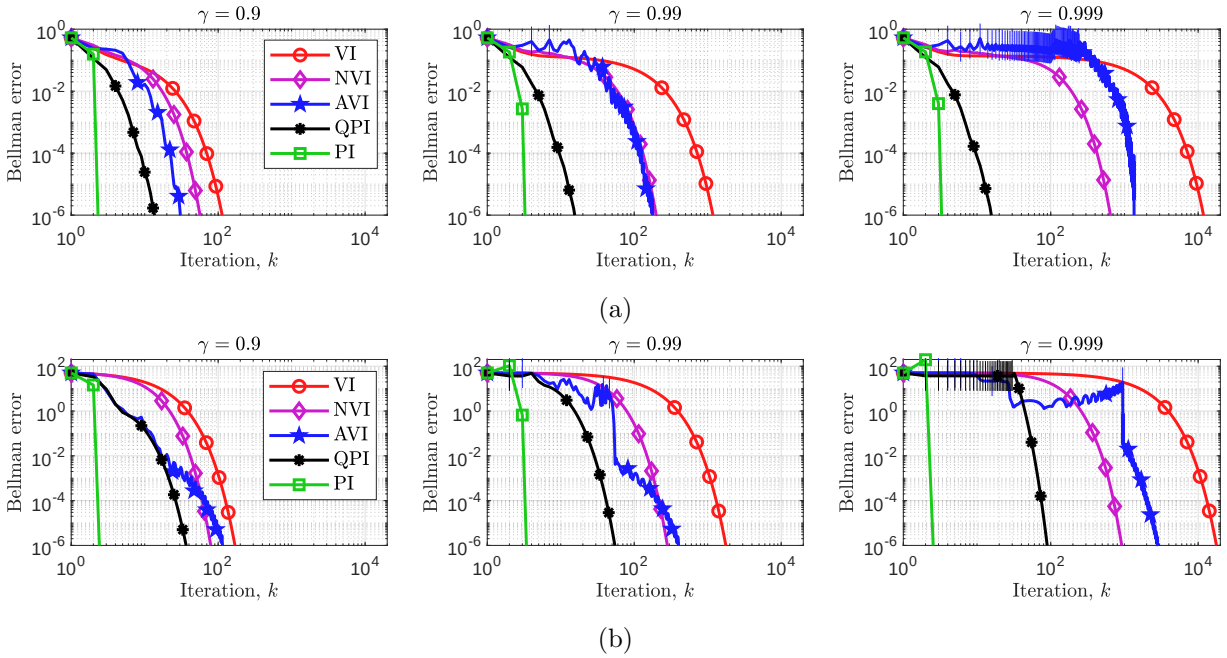

FIGURE 1. Performance of model-based algorithms for three values of γ: (a) Garnet MDP; (b) Healthcare MDP. The bars indicate the iterations at which the safeguard is activated (for NVI, AVI, and QPI).

corresponding to update rules (20) and (22) in Sections 6.1 and 6.2, respectively. The results of numerical experiments with alternative priors are then reported in Section 6.3.

6.1. **Model-based algorithms**

For model-based algorithms we consider two MPDs: a randomly generated Garnet MDPs [3] and the Healthcare MDP [21] with an *absorbing* state. The proposed QPI algorithm (20) is compared with the following algorithms: VI (value iteration); NVI (VI with Nesterov acceleration) [21]; AVI (VI with Anderson acceleration) [18]; and, PI (policy iteration). For AVI, we use a memory of one leading to a rank-one update (of the identity matrix) for approximating the Hessian so that it is comparable with the rank-one update of the uniform distribution for approximating the transition matrix in QPI. See Appendix B for the exact update rules of NVI and AVI. We note that since NVI and AVI are not guaranteed to converge, we safeguard them using VI (using the same safeguarding rule (18) used for QPI). All the algorithms are initialized by $v_0 = \mathbf{0}$ with termination condition $\|v_k - T(v_k)\|_\infty \leq \epsilon = 10^{-6}$. The results of the simulations are provided in Figures 1 and 2.

In Figure 1, VI, NVI, and AVI show a linear convergence with a rate depending on γ in both MDPs. In particular, as we increase γ from 0.9 to 0.999, we observe more than a tenfold increase in the number of iterations required for these algorithms to terminate. This is expected since these algorithms only use first-order information and their convergence rate is determined by γ.

Figure 1 also shows that for both MDPs, PI converges with a quadratic rate in 3 to 5 iterations, independent of γ. Now, observe that QPI is the only algorithm showing a similar behavior as

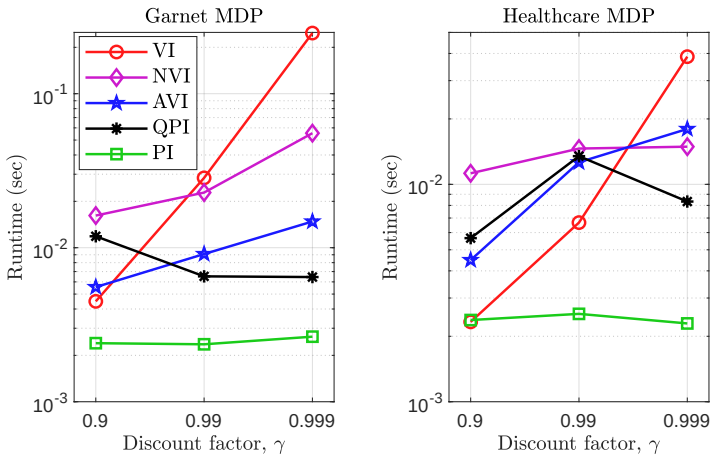

FIGURE 2. The running time of the model-based algorithms for three values of γ corresponding to Figure 1.

PI and terminating in approximately the same number of iterations, independent of γ, in both MDPs. Moreover, comparing the performance of QPI with AVI (its counterpart in the class of quasi-Newton methods), we also see the importance of newly introduced linear constraints and prior in the approximation of the transition matrix. Moreover, observe that QPI's safeguard is activated for Healthcare MDP as shown in Figure 1b (one instance for $\gamma = 0.99$ and multiple instances for $\gamma = 0.999$). In this regard, we note that for QPI, we have observed that the activation of the safeguard is particularly due to the existence of *absorbing states* in the MDP as is the case for the Healthcare MDP.

Figure 2 reports the corresponding running times of the algorithms. The reported running times are in line with convergence behaviors seen in Figure 1 and the theoretical time complexity of these algorithms. In particular, PI and QPI are the only algorithms with running time less sensitive to γ for both of the considered MDPs. In this regard, we note that since the size of the MDPs considered in our numerical simulations is relatively small, PI is the fastest algorithm despite the fact that it requires a matrix inversion.

6.2. **Model-free algorithms**

For model-free algorithms we also consider two MPDs: again a randomly generated Garnet MDPs [3] and the Graph MDP [14]. The proposed QPL algorithm (22) is compared with the following algorithms: QL (Q-learning) as in (9) with $\alpha_k = \frac{1}{(k+1)}$; SQL (speedy QL) as in (10) with $\alpha_k = \frac{1}{(k+1)}$ and $\beta_k = \delta_k = 1 - \frac{2}{(k+1)}$ [19]; and, ZQL (zap QL) as in (11) with $\alpha_k = \beta_k = \frac{1}{(k+1)}$ [14]. All the algorithms are initialized by $q_0 = \mathbf{0}_{nm}$ and terminated after $K = 10^4$ iterations with a synchronous sampling of all state-action pairs at each iteration. For QPL, the safeguard is activated after $K_{\text{sg}} = 100$ iterations. For each algorithm, we report the *average* of the Bellman error $\|q_k - T(q_k)\|_\infty$ over 20 runs of the algorithm. The results of the simulations are provided in Figure 3 and Table 3.

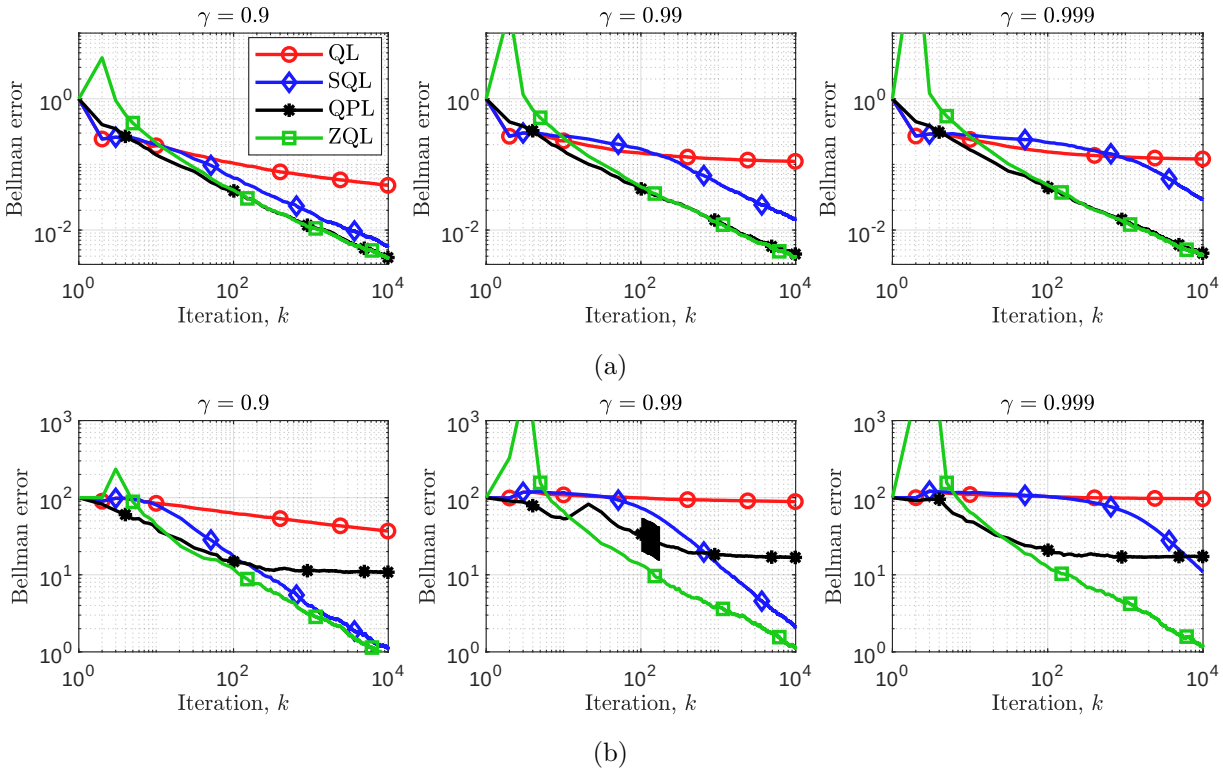

FIGURE 3. Performance of model-free algorithms (averaged over 20 runs) for three values of γ: (a) Garnet MDP; (b) Graph MDP. The bars indicate the iterations at which the safeguard is activated (for QPL).

TABLE 3. The running time (in seconds) of the model-free algorithms over $K = 10^4$ iterations (averaged over 20 runs) for $\gamma = 0.9$ corresponding to Figure 3.

	QL	SQL	QPL	ZQL	Sampling
Garnet	1.4	2.6	2.7	11	61
Graph	0.089	0.16	0.19	0.28	2.9

As can be seen in Figures 3a and 3b, the performance of QL and SQL (the first-order methods) deteriorates as γ increases for both MDPs. However, for these MDPs, ZQL (the second-order method that estimates the transition matrix by averaging over the samples) leads to almost the same error level after a fixed number of iterations for different values of γ.

Figure 3 shows that the performance of QPL is not as consistent as its model-based counterpart: QPL has the same rate of convergence as ZQL for Garnet MDP (Figure 3a), while it is showing the same rate of convergence as QL for Graph MDP (Figures 3b). This means that for structured MDPs, QPL may not lead to a better performance compared to SQL or ZQL. Moreover, Figure 3b shows that the safeguard of QPL is activated for Graph MDP and $\gamma = 0.99$. In this regard, we note that the model-free QPL algorithm uses an approximation of the transition matrix which is

constructed based on sampled data; see (21). This use of sampling on top of approximation can be the reason behind the poor performance of the model-free QPL algorithm for structured MDPs.

Finally, we note that the running times reported in Table 3 also align with the corresponding theoretical time complexities of these algorithms. In particular, QPL and SQL require almost the same amount of time, which is slightly more than QL and less than ZQL. (We report the runtime only for $\gamma = 0.9$ because it is independent of γ). Note that the time required for generating the samples is reported separately in Table 3, which is indeed the dominating factor in the actual runtime of the model-free algorithms.

6.3. **QPI and QPL with different priors**

Figures 4 and 5 report the result of our numerical simulations for the QPI and QPL algorithms, respectively, with the three choices of the prior: (i) QPI/L-A with a uniform prior $P_{\text{prior}} = \frac{1}{n}\mathbf{1}\mathbf{1}^\top$, (ii) QPI/L-B with recursive prior $P_{\text{prior}} = \widetilde{P}_{k-1}$, and (iii) QPI/L-$\mu$ with prior $P_{\text{prior}} = P^\mu$ and μ being the stochastic policy choosing (state-)actions uniformly at random so that the prior has the same sparsity pattern as the true transition probability matrix.

As depicted in Figures 4a and 5a, the experiments with alternative priors shows no improvement in the performance of the QPI and QPL algorithms in comparison with the uniform prior for random Garnet MDPs. For structured MDPs, however, we observe contradictory results as shown in Figures 4b and 5b: Using a structured prior leads to a significant improvement in the performance of the (model-based) QPI algorithm for Healthcare MDP, while using a structured or recursive prior significantly deteriorates the performance of the (model-free) QPL algorithm for Graph MDP.

7. **Limitations and Future Research**

In this paper, we exploited the well-known root-finding characterization of the optimal solution to the optimization problems and the fixed-point characterization of the optimal value function in control problems in order to look at existing equivalent algorithms for solving these problems in a more systematic way. We then used this framework to propose the model-based quasi-policy iteration (QPI) algorithm and its model-free counterpart, the quasi-policy learning (QPL) algorithm. The proposed algorithms were particularly inspired by the quasi-Newton methods and employed a novel approximation of the "Hessian" by using two new linear constraints specific to MDPs.

Safeguarding. The main drawback of the proposed algorithms, similar to other accelerated VI schemes in the literature, is the need for safeguarding to ensure convergence. Our experiments in Section 6 showed examples of MDPs in which the safeguard is activated. First, we note that an alternative way for implementing the safeguard in a model-based QPI algorithm is to scale the step-sizes in the update rule (20a) using backtracking, similar to the safeguarding of Anderson acceleration in [61]. Second, a possible approach to guarantee convergence without the need for safeguarding is the use of the operator splitting method introduced in [47] for policy evaluation. In this regard, we note that the proposed QPI algorithm is essentially the PI algorithm in which the policy evaluation step uses the approximation $\widetilde{P}_k$ in (17) instead of the true transition matrix P_k

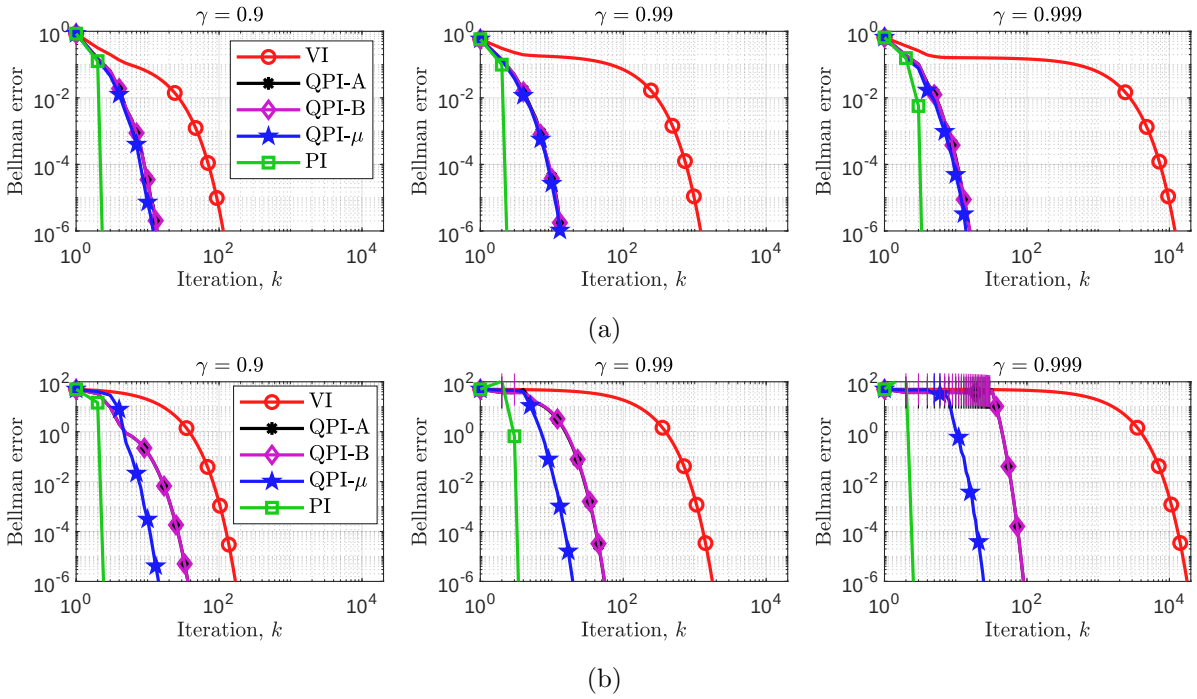

FIGURE 4. Performance of model-based algorithms for three values of γ and three different priors: (a) Garnet MDP; (b) Healthcare MDP. The bars indicate the iterations at which the safeguard is activated in QPI.

and the cost $\widetilde{c}_k = c_k + \gamma(P_k - \widetilde{P}_k)v_k$ instead of the true cost c_k. However, the convergence requires $\widetilde{P}_k$ to be close to P_k. To be precise, a sufficient condition is $\|P_k - \widetilde{P}_k\|_\infty \leq 1-\gamma$ [47, Thm. 1], which is difficult to achieve for a low-rank approximation $\widetilde{P}_k$ of P_k.

Convergence rate. Another limitation of the current work is the lack of a theoretical guarantee for the empirically observed improvement in the convergence rate, particularly for the model-based QPI algorithm. A promising approach for establishing a local super-linear convergence rate is to use similar results for semi-smooth QNMs [46] with a Broyden-type approximation [10], i.e., setting $P_{\text{prior}} = \widetilde{P}_{k-1}$ in the approximation (17). Another possibility is to use the results for Anderson acceleration in [15] to establish an improved linear rate for convergence. To that end, similar to what is done in [53], one needs to use a smoothed version of the Bellman operator, e.g., by replacing the *max* operation with a *soft-max* operation in the Bellman operator. However, in both cases, the main difficulty to be addressed is the fact that the linear constraints in (17) are not the standard secant conditions used in QNMs.

Approximation of transition matrix. The proposed algorithms in this study heavily rely on the approximation (17) of the transition matrix. As we discussed, this approximation easily allows for incorporation of different priors, e.g., a prior with the same sparsity pattern as the true transition matrix, or, the recursive prior. Our numerical simulations with these alternative priors however did not show a definitive improvement in the the performance of the proposed algorithms and hence needs further investigation with other MDPs. In this regard, we also note that the main drawback

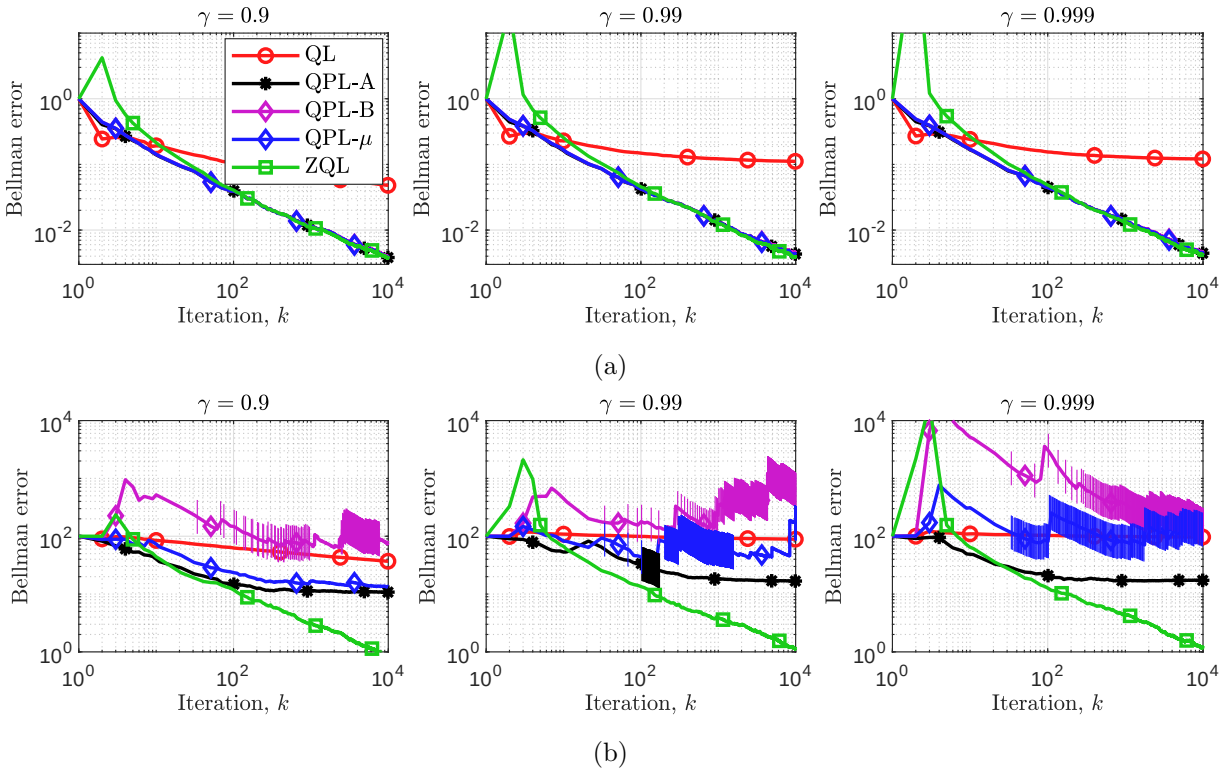

FIGURE 5. Performance of model-free algorithms (averaged over 20 runs) for three values of γ and three different priors: (a) Garnet MDP; (b) Graph MDP. The bars indicate the iterations at which the safeguard is activated in QPL.

of the approximation (17) is that it does not allow for a computationally efficient incorporation of other constraints, such as non-negativity constraints. A promising future research direction is the development of alternative approximation schemes that allow such constraints to be included at a reasonable computational cost.

Appendix A. MDPs of the numerical simulations

Garnet MDP. The considered Garnet MDP [3] is generated randomly with $n = 50$ states, $m = 5$ actions, and the branching parameter $n_b = 10$. For each state-action pair (s, a), we first form the set of reachable next states $\{s_1^+, \dots, s_{n_b}^+\}$ chosen uniformly at random from the state space $\{1, \dots, n\}$. Then, the corresponding probabilities are formed by choosing the points $p_i \in [0, 1], i = 1, \dots, n_b - 1$, uniformly at random, and setting $\mathbb{P}(s_i^+|s, a) = p_i - p_{i-1}$ with $p_0 = 0$ and $p_{n_b} = 1$. The stage cost $c(s, a)$ for each state-action pair (s, a) is also chosen uniformly at random from the interval $[0, 1]$.

Healthcare MDP. The considered Healthcare MDP is borrowed from [21]. The MPD has 6 states corresponding to the deteriorating health condition of a patient with the last state $n = 6$ being an absorbing state representing the mortality terminal state. For each of the first five states, one can choose three inputs $m \in \{1, 2, 3\}$ corresponding to increasing levels of drug dosage for treatment. The goal is to minimize the invasiveness of the treatment while avoiding the terminal

state. For the transition probabilities, we refer the reader to [21, Fig. D.1]. The cost function is chosen to be $c(n,m) = \sum_{n^+=1}^{6} n^+ \mathbb{P}(n^+|n,m) + m$ for each $n \in \{1,2,3,4,5\}$ and $m \in \{1,2,3\}$ and $c(6,1) = 50$.

Graph MDP. The considered Graph MDP is borrowed from [14]. The MDP has 18 state-action pairs in total and corresponds to a simple path-finding problem. We refer the reader to [14, Sec. 3] for the description of the MDP.

Appendix B. Accelerated VI algorithms

The update rule of accelerated VI algorithms is as follows: For $k \geq 0$ (with initialization $v_{-1} = v_0 = \mathbf{0}$)

- Nesterov accelerated VI (NVI) algorithm [21]:

$$\begin{aligned} y_k &= v_k + \gamma^{-1}(1-\sqrt{1-\gamma^2})(v_k - v_{k-1}), \\ v_{k+1} &= y_k - (1+\gamma)^{-1}\big(y_k - T(y_k)\big). \end{aligned}$$

- Anderson accelerated VI (AVI) algorithm [18]:

$$\begin{aligned} y_k &= v_k - v_{k-1}, \\ z_k &= T(v_k) - T(v_{k-1}), \\ \delta_k &= \begin{cases} 0 & \text{if } y_k^\top (y_k - z_k) = 0, \\ \frac{y_k^\top \big(v_k - T(v_k)\big)}{y_k^\top (y_k - z_k)} & \text{otherwise,} \end{cases} \\ v_{k+1} &= (1-\delta_k)T(v_k) + \delta_k T(v_{k-1}). \end{aligned}$$

Appendix C. QPI and QPL pseudo-codes

Algorithm 1 provides the pseudo-code of the safeguarded QPI algorithm with arbitrary initialization v_0. We note that the output of Algorithm 1 satisfies $\|v^\epsilon - v^\star\|_\infty \leq \epsilon/(1-\gamma)$, where $v^\star$ is the optimal value function.

The pseudo-code for the safeguarded QPL algorithm and arbitrary initialization q_0 is provided in Algorithm 2. We note that lines 4 and 10-13 of Algorithm 2 are related to the QL algorithm running in parallel for the proposed safeguarding.

Algorithm 1 Quasi-Policy Iteration (QPI)

Input: cost $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$; probability kernel $\mathbb{P}$; discount factor $\gamma \in (0,1)$; termination constant $\epsilon > 0$;
Output: sub-optimal value function v^ϵ;

initialize $v_0 \in \mathbb{R}^n$;
compute $T_0 = T(v_0)$ and $c_0 = c^{\pi_{v_0}}$; $\theta_0 = \|v_0 - T_0\|_\infty$;
for $k = 1, 2, \ldots$ **do**
  if $\|v_{k-1} - T_{k-1}\|_\infty \le \epsilon$, **then** `terminate`;
  compute v_k according to (19) or (20);
  compute $T_k = T(v_k)$ and $c_k = c^{\pi_{v_k}}$;
  if $\|v_k - T_k\|_\infty > \gamma^k \theta_0$, **then** `safeguard`:

$$v_k = T_{k-1}; \text{recompute } T_k = T(v_k) \text{ and } c_k = c^{\pi_{v_k}};$$

end for
$v^\epsilon = v_k$;

Algorithm 2 Quasi-Policy Learning (QPL)

Input: cost $c \in \mathbb{R}^{nm}$; discount factor $\gamma \in (0,1)$; maximum number K of iterations; safeguard activation iteration number K_{sg};
Output: sub-optimal Q-function q^{out};

initialize $q_0 \in \mathbb{R}^{nm}$; $z = c - \frac{\mathbf{1}^\top c}{nm}\mathbf{1}$;
generate samples $\hat{s}_0^+ \sim \mathbb{P}(\cdot|s,a)$ for each $(s,a) \in \mathcal{S} \times \mathcal{A}$;
compute $\widehat{T}_0 = [\widehat{T}(q_0, \hat{s}_0^+)]$; $\widehat{\Theta}_0 = \|q_0 - \widehat{T}_0\|_\infty$;
$q_0^{\text{QL}} = q_0$; $\widehat{T}_0^{\text{QL}} = \widehat{T}_0$; $\widehat{\Theta}_0^{\text{QL}} = \widehat{\Theta}_0$;
for $k = 1, 2, \ldots, K$ **do**
  $\alpha_k = 1/k$;
  compute q_k according to (22);
  generate samples $\hat{s}_k^+ \sim \mathbb{P}(\cdot|s,a)$ for each $(s,a) \in \mathcal{S} \times \mathcal{A}$;
  compute $\widehat{T}_k = \widehat{T}(q_k, \hat{s}_k^+)$; $\widehat{\theta}_k = \|q_k - \widehat{T}_k\|_\infty$;
  $q_k^{\text{QL}} = (1-\alpha_k) q_{k-1}^{\text{QL}} + \alpha_k \widehat{T}_{k-1}^{\text{QL}}$;
  compute $\widehat{T}_k^{\text{QL}} = \widehat{T}(q_k^{\text{QL}}, \hat{s}_k^+)$; $\widehat{\theta}_k^{\text{QL}} = \|q_k^{\text{QL}} - \widehat{T}_k^{\text{QL}}\|_\infty$;
  if $k > K_{\text{sg}}$ & $\widehat{\Theta}_{k-1} + \widehat{\theta}_k > \widehat{\Theta}_{k-1}^{\text{QL}} + \widehat{\theta}_k^{\text{QL}}$, **then** `safeguard`:

$$q_k = (1-\alpha_k) q_{k-1} + \alpha_k \widehat{T}_{k-1}; \text{ recompute } \widehat{T}_k = \widehat{T}(q_k, \hat{s}_k^+); \widehat{\theta}_k = \|q_k - \widehat{T}_k\|_\infty;$$

  $\widehat{\Theta}_k = \widehat{\Theta}_{k-1} + \widehat{\theta}_k$; $\widehat{\Theta}_k^{\text{QL}} = \widehat{\Theta}_{k-1}^{\text{QL}} + \widehat{\theta}_k^{\text{QL}}$;
end for
$q^{\text{out}} = q_K$;

References

[1] Allen-Zhu, Z. (2017). Katyusha: The first direct acceleration of stochastic gradient methods. *The Journal of Machine Learning Research*, 18(1):8194–8244.

[2] Anderson, D. G. (1965). Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM)*, 12(4):547–560.

[3] Archibald, T., McKinnon, K., and Thomas, L. (1995). On the generation of Markov decision processes. *Journal of the Operational Research Society*, 46(3):354–361.

[4] Bellman, R. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684.

[5] Berthier, E. and Bach, F. (2020). Max-plus linear approximations for deterministic continuous-state markov decision processes. *IEEE Control Systems Letters*, 4(3):767–772.

[6] Bertsekas, D. (1975). Convergence of discretization procedures in dynamic programming. *IEEE Transactions on Automatic Control*, 20(3):415–419.

[7] Bertsekas, D. (2022). *Abstract dynamic programming*. Athena Scientific.

[8] Bertsekas, D. and Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Athena Scientific.

[9] Bertsekas, D. P. (2011). Temporal difference methods for general projected equations. *IEEE Transactions on Automatic Control*, 56(9):2128–2139.

[10] Broyden, C. G. (1965). A class of methods for solving nonlinear simultaneous equations. *Mathematics of computation*, 19(92):577–593.

[11] Bubeck, S. (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357.

[12] De Farias, D. P. and Van Roy, B. (2004). On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3):462–478.

[13] Devraj, A. M., Bušić, A., and Meyn, S. (2019). On matrix momentum stochastic approximation and applications to Q-learning. In *57th Annual Allerton Conference on Communication, Control, and Computing*, pages 749–756.

[14] Devraj, A. M. and Meyn, S. (2017). Zap Q-learning. In *Advances in Neural Information Processing Systems*, volume 30.

[15] Evans, C., Pollock, S., Rebholz, L. G., and Xiao, M. (2020). A proof that anderson acceleration improves the convergence rate in linearly converging fixed-point methods (but not in those converging quadratically). *SIAM Journal on Numerical Analysis*, 58(1):788–810.

[16] Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110.

[17] Gargiani, M., Zanelli, A., Liao-McPherson, D., Summers, T., and Lygeros, J. (2022). Dynamic programming through the lens of semismooth Newton-type methods. *IEEE Control Systems Letters*, 6:2996–3001.

[18] Geist, M. and Scherrer, B. (2018). Anderson acceleration for reinforcement learning. *preprint arXiv:1809.09501*.

[19] Ghavamzadeh, M., Kappen, H., Azar, M., and Munos, R. (2011). Speedy Q-learning. In *Advances in Neural Information Processing Systems*, volume 24.

[20] Gonçalves, V. M. (2021). Max-plus approximation for reinforcement learning. *Automatica*, 129:109623.

[21] Goyal, V. and Grand-Clément, J. (2022). A first-order approach to accelerated value iteration. *Operations Research*, 71(2):517–535.

[22] Grand-Clément, J. (2021). From convex optimization to MDPs: A review of first-order, second-order and quasi-Newton methods for MDPs. *preprint arXiv:2104.10677*.

[23] Halpern, B. (1967). Fixed Points of Nonexpanding Maps. *Bulletin of the American Mathematical Society*, 73(6):957–961.

[24] Hernández-Lerma, O. and Lasserre, J. B. (2012a). *Discrete-time Markov control processes: basic optimality criteria*, volume 30. Springer Science & Business Media.

[25] Hernández-Lerma, O. and Lasserre, J. B. (2012b). *Further topics on discrete-time Markov control processes*, volume 42. Springer Science & Business Media.

[26] Howard, R. A. (1960). *Dynamic programming and Markov processes*. John Wiley.

[27] Kakade, S. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *19th International Conference on Machine Learning*, pages 267–274.

[28] Kamanchi, C., Diddigi, R. B., and Bhatnagar, S. (2022). Generalized second order value iteration in Markov decision processes. *IEEE Transactions on Automatic Control*, 67(8):4241–4247.

[29] Kearns, M. and Singh, S. (1998). Finite-sample convergence rates for Q-learning and indirect algorithms. In *Advances in neural information processing systems*, volume 11.

[30] Kidambi, R., Netrapalli, P., Jain, P., and Kakade, S. (2018). On the insufficiency of existing momentum schemes for stochastic optimization. In *Information Theory and Applications Workshop*, pages 1–9.

[31] Kolarijani, M. A. S., Max, G. F., and Mohajerin Esfahani, P. (2021). Fast approximate dynamic programming for infinite-horizon markov decision processes. In *Advances in Neural Information Processing Systems*, volume 34, pages 23652–23663.

[32] Kolarijani, M. A. S. and Mohajerin Esfahani, P. (2023). Fast approximate dynamic programming for input-affine dynamics. *IEEE Transactions on Automatic Control*, 68(10):6315–6322.

[33] Kushner, H. and Kleinman, A. (1971). Accelerated procedures for the solution of discrete Markov control problems. *IEEE Transactions on Automatic Control*, 16(2):147–152.

[34] Lee, J. and Ryu, E. (2024). Accelerating Value Iteration with Anchoring. In *Advances in Neural Information Processing Systems*, volume 36.

[35] Lemaréchal, C. (2012). Cauchy and the gradient method. *Documenta Mathematica Extra*, pages 251–254.

[36] Liu, C. and Belkin, M. (2018). Accelerating SGD with momentum for over-parameterized learning. *preprint arXiv:1810.13395*.

[37] Liu, J. and Yuan, Y. (2022). On almost sure convergence rates of stochastic gradient methods. In *35th Conference on Learning Theory*, pages 2963–2983.

[38] McEneaney, W. M. (2006). *Max-plus methods for nonlinear control and estimation*. Springer Science & Business Media.

[39] Mohajerin Esfahani, P., Sutter, T., Kuhn, D., and Lygeros, J. (2018). From infinite to finite programs: Explicit error bounds with applications to approximate dynamic programming. *SIAM Journal on Optimization*, 28(3):1968–1998.

[40] Nesterov, Y. (2018). *Lectures on convex optimization*. Springer.

[41] Nesterov, Y. E. (1983). A method for solving the convex programming problem with convergence rate $o(1/k^2)$. In *Doklady Akademii Nauk SSSR*, volume 269, pages 543–547.

[42] Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17.

[43] Porteus, E. L. and Totten, J. C. (1978). Accelerated computation of the expected discounted return in a Markov chain. *Operations Research*, 26(2):350–358.

[44] Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons.

[45] Puterman, M. L. and Brumelle, S. L. (1979). On the convergence of policy iteration in stationary dynamic programming. *Mathematics of Operations Research*, 4(1):60–69.

[46] Qi, L. and Sun, J. (1993). A nonsmooth version of newton's method. *Mathematical programming*, 58(1):353–367.

[47] Rakhsha, A., Wang, A., Ghavamzadeh, M., and Farahmand, A.-m. (2022). Operator splitting value iteration. In *Advances in Neural Information Processing Systems*, volume 35, pages 38373–38385.

[48] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407.

[49] Ruppert, D. (1985). A Newton-Raphson version of the multivariate Robbins-Monro procedure. *The Annals of Statistics*, 13(1):236–245.

[50] Rust, J. (1994). Structural estimation of Markov decision processes. *Handbook of Econometrics*, 4:3081–3143.

[51] Samuelson, P. A. (1948). Foundations of economic analysis. *Science and Society*, 13(1).

[52] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

[53] Sun, K., Wang, Y., Liu, Y., Pan, B., Jui, S., Jiang, B., Kong, L., et al. (2021). Damped anderson mixing for deep reinforcement learning: Acceleration, convergence, and stabilization. *Advances in Neural Information Processing Systems*, 34:3732–3743.

[54] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[55] Szepesvári, C. (2010). *Algorithms for reinforcement learning*. Morgan & Claypool.

[56] Tsitsiklis, J. and Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690.

[57] Vieillard, N., Pietquin, O., and Geist, M. (2019). On connections between constrained optimization and reinforcement learning. *preprint arXiv:1910.08476*.

[58] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

[59] Weng, B., Xiong, H., Zhao, L., Liang, Y., and Zhang, W. (2021). Finite-time theory for momentum Q-learning. In *37th Conference on Uncertainty in Artificial Intelligence*, pages 665–674.

[60] Yang, T., Lin, Q., and Li, Z. (2016). Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. *preprint arXiv:1604.03257*.

[61] Zhang, J., O'Donoghue, B., and Boyd, S. (2020). Globally convergent type-I Anderson acceleration for nonsmooth fixed-point iterations. *SIAM Journal on Optimization*, 30(4):3170–3197.