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 adopt the quasi-Newton method (QNM) 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, which exploits two linear structural constraints specific to MDPs and allows for the incorporation of prior information on the transition probability kernel. While the proposed algorithm has the same computational complexity as value iteration, it exhibits an empirical convergence behavior similar to that of QNM with a 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]. In particular, the connection between control algorithms for Markov decision processes (MDPs) and optimization algorithms has been noticed since the late 1970s [43]: Value iteration (VI) [4] can be seen as an instance of gradient descent (GD) algorithm, and policy iteration (PI) [27] is an instance of the Newton method (NM).

More recent works have used the relationship mentioned above to develop new control algorithms, with faster convergence and/or lower complexity, inspired by their counterparts for solving optimization problems [23, 52]. In case of *model-based* (a.k.a. planning, with access to the model of the MDP) algorithms, the combination of the VI algorithm with Polyak momentum [40], Nesterov acceleration [38], and Anderson acceleration [2] have been explored in [22] and [56]. More recently, Halpern anchoring acceleration [24] has been used to introduce the Anchored VI algorithm [33], which in particular exhibits an $\mathcal{O}(1/k)$-rate for large values of discount factor and even for $\gamma = 1$. Also of notice is the Generalized Second-Order VI algorithm [28] which applies NM on a *smoothed* version of the Bellman operator. For *model-free* (a.k.a. learning, with access to samples from the MDP)

Date: August 26, 2025.

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.

algorithms, Speedy Q-Learning [20], Momentum Q-Learning [54], and Nesterov Stochastic Approximation [14] are among the algorithms that use the idea of momentum for accelerating stochastic GD algorithm [55, 30, 34, 1] in order to achieve a better rate of convergence compared to standard Q-learning (QL). Moreover, the Zap Q-Learning algorithm [15] can be thought of as a second-order learning algorithm which was inspired by the stochastic Newton-Raphson (SNR) algorithm [45].

We note that all of the aforementioned connections have been focused on *finite* state-action MDPs. 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 state-action MDP setting [6, 41]. Alternatively, one may directly approximate the value function via finite parametrization and minimizing (a proxy of) the residual of its fixed-point characterization based on the Bellman principle of optimality [7, 50]. Examples of such include linear parameterization [10, 51], or nonlinear parameterization with, for instance, neural network architectures [9, 46, 49] or max-plus approximation [5, 21, 32, 31, 36, 35]. We also note that there is an alternative characterization of the original function as the solution to an infinite-dimensional linear program [25], paving the way for approximation techniques via finite tractable convex optimization [13, 26, 37]. 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.

Main contribution. Motivated by the connection between control and optimization algorithms, we adopt the quasi-Newton method (QNM) 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 the transition probability kernel of the MDP (Theorem 3.1). 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 3.2).
2) **Convergence and sensitivity to discount factor**: The per-iteration computational complexity of QPI is the same as VI, and its linear convergence can be guaranteed by safeguarding against standard VI (Theorem 3.1) or backtracking (Lemma 3.3). However, in our numerical simulations with random and structured MDPs, QPI exhibits an empirical behavior similar to QNMs with the convergence rate being less sensitive to the discount factor (Figure 1).
3) **Local superlinear convergence**: We provide a modified implementation of QPI which also incorporates the secant-type constraints in approximation of the Hessian to guarantee the local superlinear convergence (Theorem 3.4).
4) **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 with guaranteed convergence and the same per-iteration complexity as standard Q-learning (QL) algorithm (Theorem 3.5).

The paper is organized as follows. In Section 2, we describe the optimal control problem of MDPs. In Section 3, 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 4. The performance of these algorithms is then compared with multiple control algorithms via extensive numerical experiments in Section 5. Section 6 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 $\mathbb{P}$. 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. Optimal control of MDPs

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* $\mathcal{S} = \{1, 2, \ldots, n\}$ *and* $\mathcal{A} = \{1, 2, \ldots, m\}$*.* 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 $\mathcal{S} \times \mathcal{A} \to \{1, 2, \ldots, nm\}$.

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 $P^\pi(s, s^+) := \mathbb{P}^\pi(s^+|s)$ 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

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

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

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

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 the corresponding Bellman operators. This fixed-point characterization is the basis for the class of value iteration algorithms. To be precise, we have $v^\star = T(v^\star)$, i.e.,

$$v^\star(s) = [T(v^\star)](s), \quad \forall s \in \mathcal{S}, \tag{2}$$

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

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

Similarly, we have $q^\star = \mathbb{E}[\widehat{T}(q^\star)]$, i.e.,

$$q^\star(s,a) = \mathbb{E}\left[[\widehat{T}(q^\star)](s,a)\right], \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A} \tag{4}$$

where $\widehat{T} : \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$ is the *sampled* Bellman operator given by[1]

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

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). We note that the expectation in (4) is w.r.t. $\hat{s}^+$.

3. Quasi-Policy Iteration (QPI)

In this section, we develop and analyze the quasi-policy iteration (QPI) algorithm for model-based control problems and its extension, the quasi-policy learning (QPL) algorithm, for model-free control problems.[2] These algorithms, as the name suggests, are inspired by quasi-Newton methods (QNMs)

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

[2] 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). 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).

from convex optimization. For this reason, we begin with a brief overview of the connection between optimization and control algorithms, as well as QNMs, to motivate the algorithms developed in this study.

We note that 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

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

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

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

where $e_{(s,a)} \in \mathbb{R}^{nm}$ is the unit vector for the state-action pair $(s,a) \in \mathcal{S} \times \mathcal{A}$.

3.1. **Optimization vs. Control**

Two classical algorithms for solving the optimal control problem of MDPs are value iteration (VI) and policy iteration (PI), which are instances of gradient descent (GD) and Newton method (NM), respectively, as we mentioned above. To be precise, consider the unconstrained minimization problem

$$\min_{x \in \mathbb{R}^\ell} f(x),$$

where $f : \mathbb{R}^\ell \to \mathbb{R}$ is twice continuously differentiable and strongly convex, and let $g(x) = \nabla f(x)$ and $H(x) = \nabla^2 f(x)$ be the gradient and Hessian of f evaluated at x. Then, by defining $g(v) := v - T(v)$, one can see that the VI update rule [4]

$$v_{k+1} = T(v_k) = v_k - g(v_k),$$

resembles the GD update rule

$$x_{k+1} = x_k - \alpha_k g(x_k),$$

where α_k is the step-size. Similarly, by defining $H(v) := I - \gamma P^{\pi_v}$, the PI update rule [42, Prop. 6.5.1]

$$v_{k+1} = \left(I - \gamma P^{\pi_{v_k}}\right)^{-1} c^{\pi_{v_k}} = v_k - [H(v_k)]^{-1} g(v_k),$$

resembles the NM update rule

$$x_{k+1} = x_k - \alpha_k [H(x_k)]^{-1} g(x_k),$$

where α_k is again the step-size.

The VI algorithm converges linearly with rate γ owing to the fact that the operator T is a γ-contraction in the ∞-norm [42, Prop. 6.2.4]. The PI algorithm, on the other hand, outputs the optimal policy in a finite number of iterations [42, Thm. 6.4.2]. Moreover, the algorithm has a local *quadratic* rate of convergence when initiated in a small enough neighborhood around the optimal solution [8, 18]. The faster convergence of PI compared to VI, however, comes with a higher per-iteration computational complexity: The per-iteration complexities of VI and PI are $\mathcal{O}(n^2 m)$ and $\mathcal{O}(n^2 m + n^3)$, respectively. The extra $\mathcal{O}(n^3)$ complexity is due to the policy evaluation step, i.e., solving a linear system of equations; see also the matrix inversion in the characterization above. Not surprisingly, we see the same convergence-complexity trade-off between GD and NM. While NM has a better convergence rate compared to GD, it suffers from a higher per-iteration computational cost. GD, with a proper choice of step-size, converges *linearly* [12, Thm. 3.12] with $O(\ell)$ per-iteration complexity (disregarding the complexity of gradient oracle). On the other hand, NM, with a proper choice of step-size, has a local *quadratic* convergence rate [12, Thm. 5.3] with $O(\ell^3)$ per-iteration complexity, assuming direct inversion (and disregarding the complexity of gradient and Hessian oracles).

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

$$\begin{aligned} \widetilde{H}_k = \underset{H \in \mathbb{R}^{\ell \times \ell}}{\operatorname{argmin}} \quad & \|H - H_{\text{prior}}\|_F^2 \\ \text{s.t.} \quad & H r_i = b_i, \; i = 1, \ldots, j, \end{aligned} \tag{6}$$

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.,

$$\widetilde{H}_k = H_{\text{prior}} + (B - H_{\text{prior}} R)(R^\top R)^{-1} R^\top, \tag{7}$$

where $R = (r_1, \ldots, r_j)$, $B = (b_1, \ldots, b_j) \in \mathbb{R}^{\ell \times j}$. 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 [11].

In what follows, we use a similar idea and 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 [19, 56, 48] 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.

3.2. QPI Algorithm

For $k \in \{0, 1, 2, \ldots\}$, 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 (6), we propose the generic QPI update rule

$$v_{k+1} = v_k - (I - \gamma \widetilde{P}_k)^{-1}(v_k - T_k), \tag{8}$$

where

$$\begin{aligned} \widetilde{P}_k = \underset{P \in \mathbb{R}^{n \times n}}{\operatorname{argmin}} \quad & \|P - P_{\text{prior}}\|_F^2 \\ \text{s.t.} \quad & P r_i = b_i, \ i = 1, \ldots, j. \end{aligned} \tag{9}$$

Observe that instead of approximating the complete Hessian $H_k := (I - \gamma P_k)^{-1}$ 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., we have the *global* structural constraint

$$P_k \mathbf{1} = \mathbf{1}, \tag{10}$$

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, we have the *local* structural constraint

$$T_k = c_k + \gamma P_k v_k \ \Rightarrow \ P_k v_k = \gamma^{-1}(T_k - c_k), \tag{11}$$

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 (10) and (11) hold *exactly*. Incorporating these constraints, we propose the approximation

$$\begin{aligned} \widetilde{P}_k = \underset{P \in \mathbb{R}^{n \times n}}{\operatorname{argmin}} \quad & \|P - P_{\text{prior}}\|_F^2 \\ \text{s.t.} \quad & P\mathbf{1} = \mathbf{1}, \ P v_k = \gamma^{-1}(T_k - c_k). \end{aligned} \tag{12}$$

The update rule (8) using the approximation (12) is, however, not necessarily a contraction. The same problem also arises in similar algorithms such as Anderson accelerated VI [56] and Nesterov accelerated VI [22]. 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

$$\begin{aligned} &\text{(QPI)} && \text{compute } v_{k+1} \text{ according to (8), (12);} \\ &\text{(Safeguard)} && \textbf{if } \theta_{k+1} > \gamma^{k+1}\theta_0, \ \textbf{then } v_{k+1} = T_k. \end{aligned} \tag{13}$$

The following theorem summarizes the discussion above by providing the QPI update rule explicitly (see Section 4.1 for the proof).

Theorem 3.1 (QPI)**.** *Consider the update rule* (8) *using the approximation* (12) *where* $P_{\text{prior}}\mathbf{1} = \mathbf{1}$ *and let* $G_{\text{prior}} = (I - \gamma P_{\text{prior}})^{-1}$*. We have*

$$v_{k+1} = v_k - \widetilde{G}_k(v_k - T_k), \tag{14a}$$

where

$$\begin{aligned}
&w_k = T_k - c_k - \gamma P_{\text{prior}} v_k, \ \ \breve{w}_k = G_{\text{prior}} w_k \in \mathbb{R}^n, \\
&u_k = v_k - \tfrac{\mathbf{1}^\top v_k}{n}\mathbf{1}, \ \ \breve{u}_k = G_{\text{prior}}^\top u_k \in \mathbb{R}^n, \\
&\tau_k = \begin{cases} 0 & \textit{if } u_k^\top v_k = 0, \\ (u_k^\top v_k)^{-1} & \textit{otherwise} \end{cases} \in \mathbb{R}, \\
&\eta_k = \begin{cases} 0 & \textit{if } u_k^\top v_k = 0, \\ \big(u_k^\top (v_k - \breve{w}_k)\big)^{-1} & \textit{otherwise} \end{cases} \in \mathbb{R}, \\
&\widetilde{P}_k = P_{\text{prior}} + \gamma^{-1}\tau_k w_k u_k^\top, \\
&\widetilde{G}_k = G_{\text{prior}} + \eta_k \breve{w}_k \breve{u}_k^\top.
\end{aligned} \tag{14b}$$

Moreover, the iterates v_k *of the QPI update rule* (14) *with the safeguarding* (13) *converge to* $v^\star$ *linearly with rate* γ *and a per-iteration 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 QNMs. We also note that there is also a one-time computational cost of $O(n^3)$ in the QPI update rule (14) for computing G_{prior} (assuming direct inversion and if G_{prior} is not available in closed form).

3.2.1. *The prior.* Next to be addressed is the choice of P_{prior}. First, note that for a fixed prior in all iterations, computation of τ_k and $\widetilde{P}_k$ in (14b) is not needed since the update (14a) 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 4.2 for the proof):

Corollary 3.2 (Uniform prior)**.** *The QPI update rule* (14) *with the uniform prior* $P_{\text{prior}} = \frac{1}{n}E$*, equivalently reads as*

$$v_{k+1} = (1 - \delta_k)T_k + \delta_k c_k + \lambda_k \mathbf{1}, \tag{15a}$$

where the scalar coefficients are given by

$$\begin{aligned}
&z_k = c_k - \tfrac{\mathbf{1}^\top c_k}{n}\mathbf{1} \in \mathbb{R}^n,\\
&g_k = v_k - T_k \in \mathbb{R}^n, \quad y_k = g_k - \tfrac{\mathbf{1}^\top g_k}{n}\mathbf{1} \in \mathbb{R}^n,\\
&\delta_k = \begin{cases} 0 & \text{if } v_k^\top(y_k+z_k)=0,\\ \frac{v_k^\top y_k}{v_k^\top(y_k+z_k)} & \text{otherwise} \end{cases} \in \mathbb{R},\\
&\lambda_k = \tfrac{\gamma}{n(1-\gamma)}\mathbf{1}^\top\big((\delta_k-1)g_k+\delta_k c_k\big) \in \mathbb{R}.
\end{aligned} \tag{15b}$$

Observe that the QPI update rule (15a) 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 (12) 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 [11]. This can be achieved by choosing an initialization $\widetilde{P}_{-1}$ such that $\widetilde{P}_{-1}\mathbf{1} = \mathbf{1}$ and defining $\widetilde{G}_{-1} := (1-\gamma\widetilde{P}_{-1})^{-1}$ (e.g., $\widetilde{P}_{-1} = \frac{1}{n}E$ and $\widetilde{G}_{-1} = I + \frac{\gamma}{n(1-\gamma)}E$). We note that for this recursive scheme, one might also need to introduce some form of projection to avoid unbounded growth of $\widetilde{P}_k$ and $\widetilde{G}_k$. In particular, using the fact that $\widetilde{P}_k$ is in principle an approximation of a row stochastic matrix, one can use a simple normalization to ensure $\|\widetilde{P}_k\|_\infty \le 1$ and $\|\widetilde{G}_k\|_\infty \le (1-\gamma)^{-1}$ for all k.

3.2.2. *Implementation via backtracking.* The QPI update rule (14a) can be alternatively implemented without safeguarding. The proposed alternative again exploits the fact that the VI update leads to contraction in the Bellman error and combines that with the *backtracking* approach from optimization. To be precise, let us rewrite the QPI update rule (14a) as

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

and introduce a step-size α_k as follows

$$v_{k+1} = T_k - \alpha_k(\widetilde{G}_k - I)(v_k - T_k). \tag{16}$$

Then, at each iteration k, we introduce an inner iteration that finds the step-size α_k via backtracking such that $\theta_{k+1} \le \gamma'\theta_k$ for a chosen $\gamma' \in (\gamma, 1)$. For instance, at each iteration k, we can set

$$\begin{array}{ll}
(0) & \alpha_k = 1;\\
(1) & \text{compute } v_{k+1} \text{ according to (16), (14b)};\\
(2) & \textbf{if } \theta_{k+1} > \gamma'\theta_k, \textbf{ then } \alpha_k \leftarrow \tfrac{1}{2}\alpha_k, \text{ go to (1).}
\end{array} \tag{17}$$

We note that the preceding backtracking scheme is guaranteed to terminate if $v_k \neq v^\star$ (i.e, $\theta_k \neq 0$). Indeed, we have (see Section 4.3 for the proof):

Lemma 3.3 (Backtracking)**.** *Let $\theta_v := \|v - T(v)\|_\infty$ for $v \in \mathbb{R}^n$. Fix $v \in \mathbb{R}^n$ such that $\theta_v \neq 0$. Then, for all $w \in \mathbb{R}^n$ with $\|w - T(v)\|_\infty \le \frac{\gamma'-\gamma}{1+\gamma}\theta_v$, we have $\theta_w \le \gamma'\theta_v$.*

In particular, the preceding lemma implies that the backtracking scheme (17) terminates in at most $\log_2(\frac{1+\gamma}{\gamma'-\gamma}\|\widetilde{G}_k - I\|_\infty)$ steps at iteration k. We also note that iterates of the proposed QPI update rule with backtracking converge to the optimal value function linearly with rate γ'.

3.2.3. *Modified implementation with superlinear convergence.* For the proposed QPI scheme to achieve the classic local *superlinear* convergence of the class of QNMs, we need to modify the approximation (12) to also include secant-type conditions. To this end, for $k \geq 1$, we consider a modified approximations as follows

if $\pi_{v_k} = \pi_{v_{k-1}}$

$$\begin{aligned} \widetilde{P}_k = \underset{P \in \mathbb{R}^{n \times n}}{\operatorname{argmin}} \quad & \|P - \widetilde{P}_{k-1}\|_F^2 \\ \text{s.t.} \quad & P\mathbf{1} = \mathbf{1},\ Pv_k = \gamma^{-1}(T_k - c_k),\ \gamma P(v_k - v_{k-1}) = T_k - T_{k-1}; \end{aligned} \tag{18a}$$

else

$$\begin{aligned} \widetilde{P}_k = \underset{P \in \mathbb{R}^{n \times n}}{\operatorname{argmin}} \quad & \|P - \widetilde{P}_{k-1}\|_F^2 \\ \text{s.t.} \quad & P\mathbf{1} = \mathbf{1},\ \gamma P(v_k - v_{k-1}) = T_k - T_{k-1}; \end{aligned} \tag{18b}$$

with the corresponding gain matrix

$$\widetilde{G}_k = \big(I - \gamma \widetilde{P}_k\big)^{-1}. \tag{18c}$$

Above, the only difference between (18a) and (18b) is the omission of the *local structural* constraint in (18b). We note that the condition $\pi_{v_k} = \pi_{v_{k-1}}$ implies that v_k and v_{k-1} have the same greedy policy and hence lie on the same affine region of the map T (recall that T is piece-wise affine). Therefore, the selection rule is to avoid "constraint collision" between the *local structural* and the *secant* constraints when $\pi_{v_k} \neq \pi_{v_{k-1}}$. Also, observe that in both approximations the prior is the previous approximation $\widetilde{P}_{k-1}$. Most importantly, note that the solution $\widetilde{P}_k$ is a low-rank (rank-three, at most) update of $\widetilde{P}_{k-1}$; c.f. the optimization (6) and its solution (7). This means that $\widetilde{G}_k$ can also be computed with a low cost using the Woodbury formula and $\widetilde{G}_{k-1} = (I - \gamma \widetilde{P}_{k-1})^{-1}$. Our next result concerns the convergence of the proposed modified implementation of the QPI algorithm (see Section 4.4 for the proof):

Theorem 3.4 (Superlinear convergence)**.** *Consider the update rule* (16) *with the gain matrix* (18) *and the backtracking scheme* (17)*. Then, the iterates v_k converge to $v^\star$ with a global linear rate and a local superlinear rate.*

3.2.4. *Other constraints.* We finish this section with the following remark. One can also add extra constraints to the minimization problem (12) 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 existing knowledge on the structure of the MDP. However, incorporating such information may lead to the problem (12) not having a closed-form and/or low-rank solution, and hence undermining the computational efficiency of the proposed algorithm. In this regard, we note that the problem (12) has a closed-form solution which is a rank-one update of the prior; see $\widetilde{P}_k$ in (14b).

3.3. 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 (15) with a uniform prior. However, we note that the extension can be similarly applied to the QPI algorithm (14) 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 (12), 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. However, the *local* structural constraint is in this case formed based on the *sampled* Bellman operator $\widehat{T}(\cdot)$, 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

$$\widehat{T}_k := \widehat{T}(q_k),$$

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

$$\begin{aligned} \widetilde{P}_k = \quad & \underset{P \in \mathbb{R}^{nm \times nm}}{\operatorname{argmin}} \; \|P - P_{\text{prior}}\|_F^2 \\ & \text{s.t.} \;\; P\mathbf{1} = \mathbf{1}, \; Pq_k = \gamma^{-1}\big(\widehat{T}_k - c\big). \end{aligned} \tag{19}$$

The minimization problem above also has a closed-form solution as a rank-one update of the prior, which allows us to compute $\widetilde{G}_k = (I - \gamma \widetilde{P}_k)^{-1}$ efficiently using Woodbury formula. Having $\widetilde{G}_k$ at hand, we can consider the update rule

$$q_{k+1} = q_k - \alpha_k \widetilde{G}_k (q_k - \widehat{T}_k), \tag{20}$$

where α_k is the diminishing learning rate of the algorithm, e.g., $\alpha_k = 1/(k+1)$. However, similar to the model-based case, the update rule (20) is not convergent. To address this issue, we consider a convex combination of the update rule (20) with the "standard" synchronous Q-learning (QL) update rule[3] [53, 29]

$$q_{k+1} = q_k - \alpha_k (q_k - \widehat{T}_k), \tag{21}$$

with a diminishing effect. That is, we consider the update rule

$$\begin{aligned} q_{k+1} &= q_k - \alpha_k \Big((1 - \beta_k)(q_k - \widehat{T}_k) + \beta_k \widetilde{G}_k (q_k - \widehat{T}_k) \Big) \\ &= q_k + \alpha_k (\widehat{T}_k - q_k) + \alpha_k \beta_k (\widetilde{G}_k - I)(\widehat{T}_k - q_k), \end{aligned}$$

where β_k is a diminishing coefficient, e.g., $\beta_k = 1/(k+1)$. Moreover, to assure convergence, we bound the effect of the extra term $p_k = (\widetilde{G}_k - I)(\widehat{T}_k - q_k)$ by passing it through the operator Π_M :

[3] 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].

$\mathbb{R}^{nm} \to \mathbb{R}^{nm}$, with $M = \frac{2\gamma}{(1-\gamma)^2} \|c\|_\infty$, given by

$$\Pi_M(p) := \frac{\min\{M, \|p\|_\infty\}}{\|p\|_\infty} p.$$

We note that the bound M is chosen using the fact that for any row stochastic matrix P and $G = (1 - \gamma P)^{-1}$, we have

$$\|G - I\|_\infty = \left\|\textstyle\sum_{i=1}^\infty (\gamma P)^i\right\|_\infty \le \textstyle\sum_{i=1}^\infty \gamma^i \left\|P^i\right\|_\infty \le \frac{\gamma}{1-\gamma},$$

and the generic bound $\|q^\pi\|_\infty \le \|c\|_\infty / (1 - \gamma)$ for the value q^π of any policy π. The update rule of the model-free QPL algorithm in its generic form is hence

$$\begin{aligned} p_k &= (\widetilde{G}_k - I)(\widehat{T}_k - q_k), \\ q_{k+1} &= q_k + \alpha_k(\widehat{T}_k - q_k) + \alpha_k \beta_k \Pi_M(p_k). \end{aligned} \tag{22}$$

In particular, by using the uniform prior $P_{\text{prior}} = \frac{1}{nm}E$, the QPL update rule reduces to

$$\begin{aligned} p_k &= \delta_k(c - \widehat{T}_k) + \lambda_k \mathbf{1}, \\ q_{k+1} &= q_k + \alpha_k(\widehat{T}_k - q_k) + \alpha_k \beta_k \Pi_M(p_k), \end{aligned} \tag{23a}$$

where

$$\begin{aligned} & z = c - \tfrac{\mathbf{1}^\top c}{nm}\mathbf{1} \in \mathbb{R}^{nm}, \\ & \widehat{g}_k = q_k - \widehat{T}_k \in \mathbb{R}^{nm}, \quad y_k = \widehat{g}_k - \tfrac{\mathbf{1}^\top \widehat{g}_k}{nm}\mathbf{1} \in \mathbb{R}^{nm}, \\ & \delta_k = \begin{cases} 0 & \text{if } q_k^\top(y_k + z) = 0, \\ \frac{q_k^\top y_k}{q_k^\top(y_k+z)} & \text{otherwise} \end{cases} \in \mathbb{R}, \\ & \lambda_k = \tfrac{\gamma}{nm(1-\gamma)} \mathbf{1}^\top \big((\delta_k - 1)\widehat{g}_k + \delta_k c\big) \in \mathbb{R}. \end{aligned} \tag{23b}$$

Observe that in this case QPL uses the two additional vectors $c - \widehat{T}_k$ and the all-one vector $\mathbf{1}$ with adaptive coefficients in its update rule. The following theorem summarizes properties of the proposed QPL algorithm (see Section 4.5 for the proof).

Theorem 3.5 (QPL)**.** *Assume the learning rates α_k and β_k are such that $\sum_k \alpha_k = \infty$, $\sum_k \alpha_k^2 < \infty$, and $\beta_k \to 0$. Then, the iterates q_k of QPL algorithm (23) converge to $q^\star$ almost surely. Moreover, the algorithm has a per-iteration complexity of $\mathcal{O}(nm^2)$.*

Regarding the preceding result, we note that the per-iteration time complexity of QPL is the same as that of the (synchronous) QL algorithm. We finish with the following remark on the *asynchronous* implementation of QPL.

Remark 3.6 (Asynchronous QPL)**.** *The proposed QPL update rule (23) 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*

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

Cf. approximation (19)*. In particular, by using the uniform prior* $P_{\text{prior}} = \frac{1}{nm}E$*, the corresponding update rule reads as*

$$\begin{aligned} p_k &= \tau_k\big(\delta_k e_{(s_k,a_k)} + \tfrac{\gamma(1+\delta_k)}{nm(1-\gamma)}\mathbf{1}\big), \\ q_{k+1} &= q_k + \alpha_k\tau_k e_{(s_k,a_k)} + \alpha_k\beta_k\Pi_M(p_k), \end{aligned}$$

where the scalar coefficients are given by

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

Note that the preceding update rule leads to an update in all entries of the Q-function q_k *in each iteration.*

4. Technical Proofs

4.1. Proof of Theorem 3.1

First, let us show that the two equality constraints in the minimization problem (12) 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 (12) are linearly dependent if and only if $u_k^\top v_k = 0$.

We first consider the update rule (14) for the case $u_k^\top v_k \neq 0$. Define $R := (\mathbf{1}, v_k)$, $B := \big(\mathbf{1}, \gamma^{-1}(T_k - c_k)\big) \in \mathbb{R}^{n\times 2}$ so that the minimization problem (12) can be written as

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

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 (12) become linearly dependent, and, in particular, the second constraint can be discarded. The solution to the problem (12) in this case is then $\widetilde{P}_k = P_{\text{prior}}$. Hence, the approximation (12) 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

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

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

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

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 nonsingular with their eigenvalues having strictly positive real parts (because the eigenvalues of P_k all reside within the unit disc), we have

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

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. We will use induction to show that

$$\theta_k \le \gamma^k \theta_0, \quad \forall k \ge 0. \tag{25}$$

The base case $k = 0$ holds trivially. Let us now assume the inequality (25) holds for some $k \ge 0$ and recall that T is a γ-contraction in ∞-norm. Then, the safeguarding against standard VI as in (13) implies that

$$\theta_{k+1} \le \max\{\gamma^{k+1}\theta_0,\ \gamma\theta_k\} \le \gamma^{k+1}\theta_0,$$

where we used the induction hypothesis for the last inequality. This completes the proof.

Finally, the per-iteration time complexity of each iteration of the safeguarded QPI update rule: The update rule (14a) 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 (14b) involves vector/matrix additions and matrix-vector multiplications (all of size n) and hence requires $\mathcal{O}(n^2)$ operations. For the safeguarding (13), 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 $\mathcal{O}(n^2 m)$.

4.2. **Proof of Corollary 3.2**

The result follows from Theorem 3.1 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 (15b) is equivalent to the condition $u_k^\top v_k = 0$ in (14b). 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 3.1 in Section 4.1. 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

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

This completes the proof.

4.3. **Proof of Lemma 3.3**

Recall that T is a γ-contraction observe that

$$\theta_{T(v)} = \|T(v) - T(T(v))\|_\infty \leq \gamma \|v - T(v)\|_\infty = \gamma \theta_v.$$

Also, note that the map $\theta_{(\cdot)}$ is $(1+\gamma)$-Lipschitz continuous: For any $w_1, w_2 \in \mathbb{R}^n$, we have

$$\begin{aligned}|\theta_{w_1} - \theta_{w_2}| &= |\, \|w_1 - T(w_1)\|_\infty - \|w_2 - T(w_2)\|_\infty | \leq \|w_1 - T(w_1) - w_2 + T(w_2)\|_\infty \\ &\leq \|w_1 - w_2\|_\infty + \|T(w_1) - T(w_2)\|_\infty \leq (1+\gamma) \|w_1 - w_2\|_\infty .\end{aligned}$$

Setting $w_1 = w$ with $\|w - T(v)\|_\infty \leq \frac{\gamma' - \gamma}{1+\gamma} \theta_v$ and $w_2 = T(v)$, we then have

$$|\theta_w - \theta_{T(v)}| \leq (1+\gamma) \|w - T(v)\|_\infty \leq (\gamma' - \gamma)\theta_v.$$

Hence,

$$\theta_w \leq \theta_{T(v)} + (\gamma' - \gamma)\theta_v \leq \gamma' \theta_v.$$

This completes the proof.

4.4. **Proof of Theorem 3.4**

In what follows, we show that the proposed algorithm is an instance of semismooth QNM with a structured least-change secant update and a standard backtracking scheme. To this end, observe that we are looking for the unique root $v^\star$ of the map $g(v) = v - T(v)$. First, note that g is piece-wise affine, Lipschitz-continuous, and, in particular, *strongly semismooth* [17, Prop. 7.4.7]. Moreover, the any generalized Jacobian $H_k \in \partial g(v) \subset \{I - \gamma P \;:\; P \geq 0,\; P\mathbf{1} = \mathbf{1}\}$ is *non-singular* at any point v (note that for a row stochastic matrix P and $\gamma \in (0,1)$, we have $(I - \gamma P)^{-1} = \sum_{t=0}^{\infty} \gamma^t P^t$).

Denote $g_k := g(v_k)$ and $\widetilde{H}_k := \widetilde{G}_k^{-1}$ for each $k \geq 0$. The update rule (16) then can be equivalently written as

$$v_{k+1} = v_k - \alpha_k \widetilde{H}_k^{-1} g_k - (1 - \alpha_k) g_k,$$

where α_k is determined using the backtracking scheme (17) ensuring

$$\|g_{k+1}\|_\infty \leq \gamma' \|g_k\|_\infty ,$$

for some $\gamma' \in (\gamma, 1)$. This means that, as long as the approximations $\widetilde{H}_k$ of the Jacobian of g are uniformly bounded, we have the global linear convergence of the iterates v_k to $v^\star$ with rate γ'; see Lemma 3.3.

Finally, let us look at the approximations $\widetilde{H}_k$ of the Jacobian of g. Using the change of variable $\widetilde{P}_k = \gamma^{-1}(I - \widetilde{H}_k)$, we can show that

$$\begin{aligned}
&\textbf{if } \pi_{v_k} = \pi_{v_{k-1}} \\
&\qquad \widetilde{H}_k = \underset{H \in \mathbb{R}^{n\times n}}{\operatorname{argmin}} \quad \|H - \widetilde{H}_{k-1}\|_F^2 \\
&\qquad\qquad\qquad \text{s.t.} \quad H\mathbf{1} = (1-\gamma)\mathbf{1},\ Hv_k = g_k + c_k,\ H(v_k - v_{k-1}) = g_k - g_{k-1}; \\
&\textbf{else} \\
&\qquad \widetilde{H}_k = \underset{H \in \mathbb{R}^{n\times n}}{\operatorname{argmin}} \quad \|H - \widetilde{H}_{k-1}\|_F^2 \\
&\qquad\qquad\qquad \text{s.t.} \quad H\mathbf{1} = (1-\gamma)\mathbf{1},\ H(v_k - v_{k-1}) = g_k - g_{k-1}.
\end{aligned}$$

One can now clearly observe that the approximation $\widetilde{H}_k$ satisfies the *secant* condition, i.e.,

$$\widetilde{H}_k(v_k - v_{k-1}) = g_k - g_{k-1},$$

besides the structural linear constraints that the true (generalized) Jacobian must satisfy. This implies that the proposed algorithm indeed involves a least-change in $\|\widetilde{H}_k - \widetilde{H}_{k-1}\|_F$ subject to secant and structural constraints. Now, since g is strongly semismooth, there exists $H_k \in \partial g(v_k)$ such that [17, Def. 7.4.2]

$$\|g_k - g_{k-1} - H_k(v_k - v_{k-1})\| = o(\|v_k - v_{k-1}\|).$$

Therefore,

$$\|(\widetilde{H}_k - H_k)(v_k - v_{k-1})\| = o(\|v_k - v_{k-1}\|).$$

That is, Dennis-Moré condition is satisfied and, by [47, Thm. 4.2], the convergence is locally superlinear.

4.5. Proof of Theorem 3.5

The per-iteration time complexity of each iteration of the QPL update rule is as follows: The update rule (23) requires $\mathcal{O}(nm^2)$ operations for computing the vectors $\widehat{T}_k = \widehat{T}(q_k)$, and $\mathcal{O}(nm)$ operations for computing the step-sizes δ_k and λ_k and the vector additions. Summing up the aforementioned complexities, the total time complexity is $\mathcal{O}(nm^2)$.

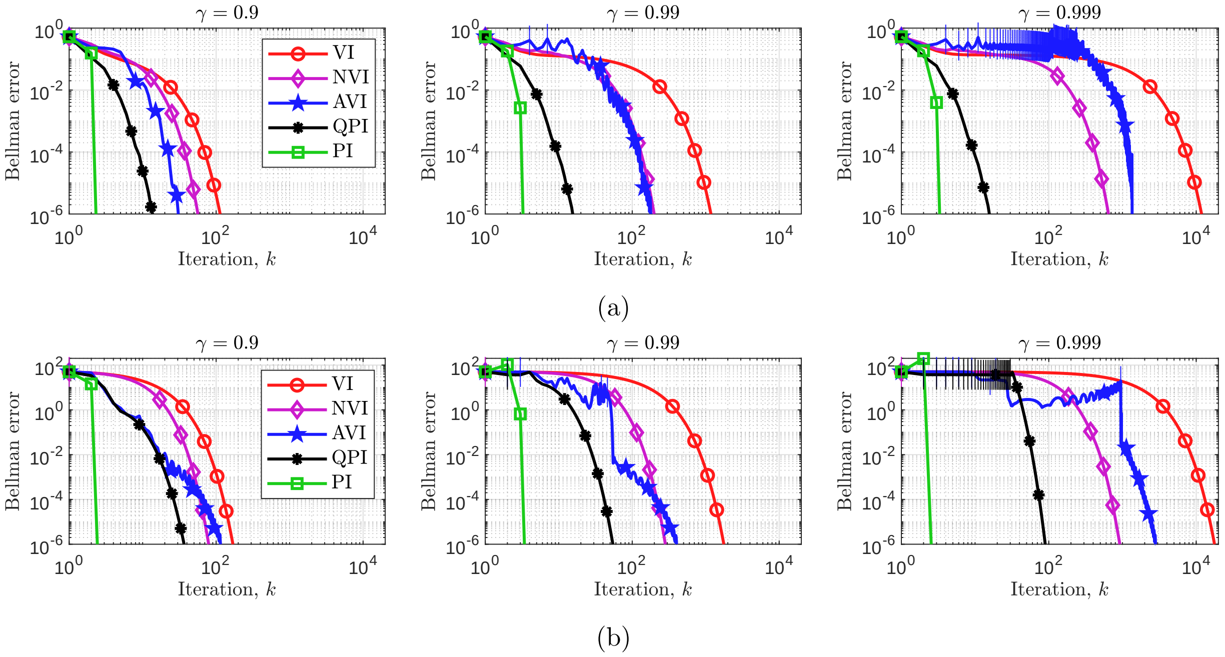
Regarding the convergence, define

$$F(q) := \bar{T}(q) - q,\ V_k := \widehat{T}_k - T(q),\ d_k := \beta_k \Pi_M(p_k),$$

where $\bar{T}$ is the corresponding true Bellman operator for Q-functions, i.e., $\bar{T}(q) = \mathbb{E}[\widehat{T}(q)]$ for each q. Then, the convergence of q_k to the unique equilibrium $q^\star$ of the map F follows from [39, Cor. 3.2]. In particular, observe that $d_k \to 0$ since $\beta_k \to 0$ and $\|\Pi_M(p_k)\|_\infty \leq M$.

5. 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



(a)

(b)

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 (15) and (23) in Sections 5.1 and 5.2, respectively. The results of numerical experiments with alternative priors are then reported in Section 5.3.

5.1. Model-based algorithms

For model-based algorithms we consider two MDPs: a randomly generated Garnet MDP [3] and the Healthcare MDP [22] with an *absorbing* state. The proposed QPI algorithm (15) is compared with the following algorithms: VI (value iteration); NVI (VI with Nesterov acceleration) [22]; AVI (VI with Anderson acceleration) [19]; 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 (13) used for QPI). All the algorithms are initialized by $v_0 = \mathbf{0}$ with termination condition $\|v_k - T(v_k)\|_\infty \leq 10^{-6}$. 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 PI

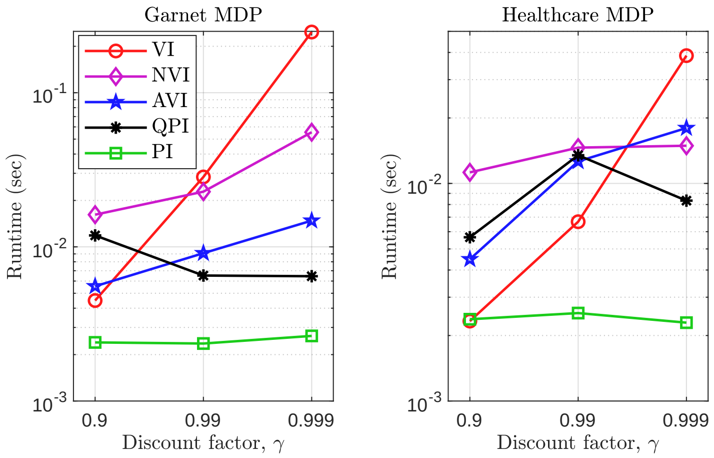


FIGURE 2. The running time of the model-based algorithms for three values of γ corresponding to Figure 1.

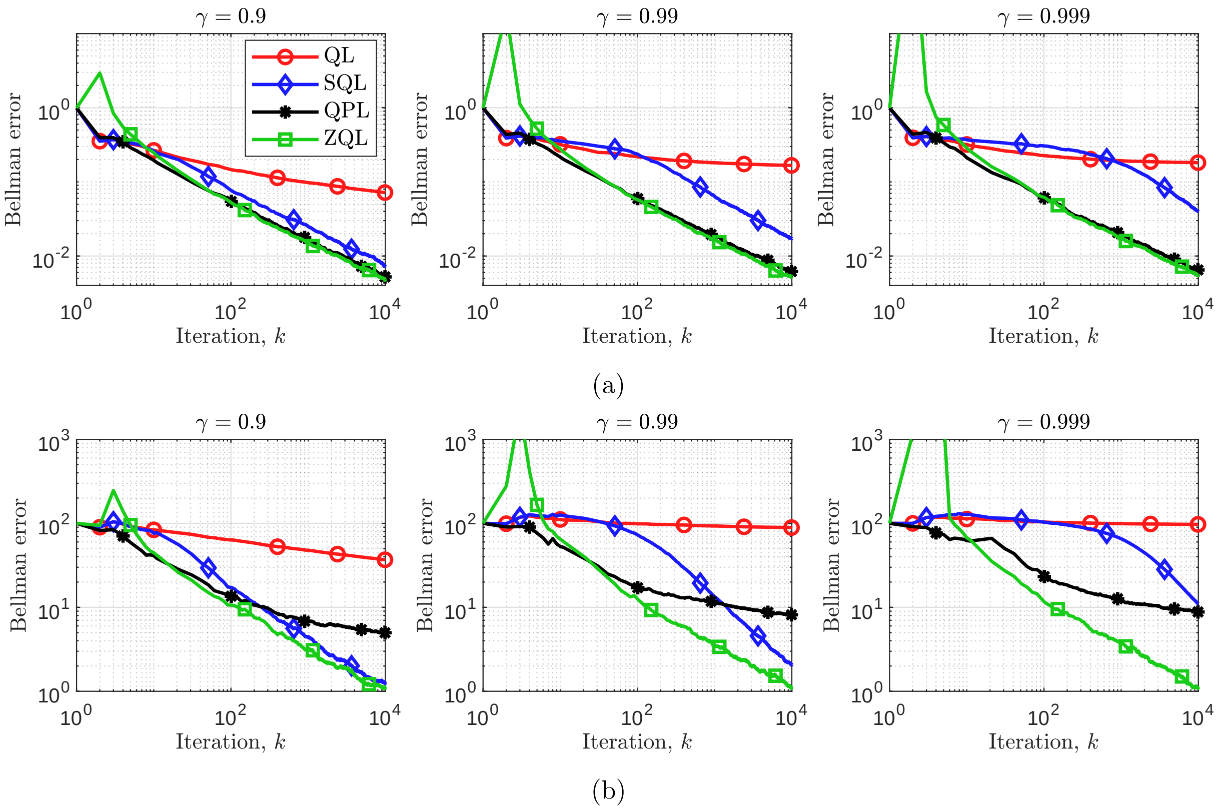
and terminating in approximately the same number of iterations, independent of γ, in both MDPs. Moreover, comparing the performance of QPI with AVI (its counterpart in the class of QNMs), we also see the importance of newly introduced linear constraints and prior in the approximation of the transition 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.

5.2. **Model-free algorithms**

For model-free algorithms we also consider two MDPs: again a randomly generated Garnet MDP [3] and the Graph MDP [15]. The proposed QPL algorithm (23) with $\alpha_k = \frac{1}{(k+1)}$ and $\beta_k = \frac{1}{(k+1)^{0.1}}$ is compared with the following algorithms: QL (Q-learning) as in (21) with $\alpha_k = \frac{1}{(k+1)}$; SQL (speedy QL) [20]; and, ZQL (zap QL) [15]. See Appendix B for the exact update rules of SQL and ZQL. 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 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 1.

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



(a)

(b)

FIGURE 3. Performance of model-free algorithms (averaged over 20 runs) for three values of γ: (a) Garnet MDP; (b) Graph MDP.

TABLE 1. 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	1.7	3.1	3.3	16	71
Graph	0.16	0.27	0.31	0.59	6.5

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 deviates from ZQL as k increases for Graph MDP (Figures 3b). This means that for structured MDPs, QPL may not lead to a better performance compared to SQL or ZQL. 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 (19). 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 1 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 1, which is indeed the dominating factor in the actual runtime of the model-free algorithms.

5.3. 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 $P_{\text{prior}} \propto \mathbf{1}\mathbf{1}^\top$, (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 actions uniformly at random so that the prior has the same sparsity pattern as the true transition probability matrix.

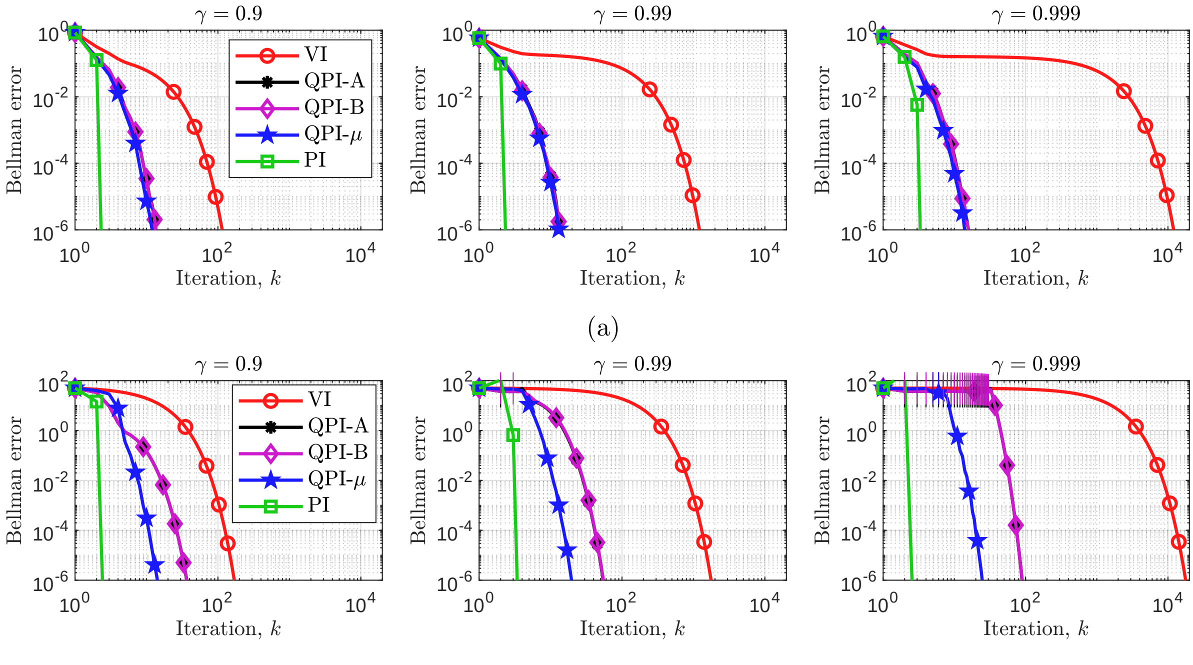
As depicted in Figures 4a and 5a, the experiments with alternative priors shows no improvement in the performance of the QPI and QPL algorithms in comparison with the uniform prior for random Garnet MDPs. For structured MDPs, however, we observe contradictory results as shown in Figures 4b and 5b: Using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

6. Limitations and Future Research

In this paper, we proposed 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.

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 5 showed examples of MDPs in which the safeguard is activated. Our modified implementation of Section 3.2.2 guarantees the linear convergence while removing the need for safeguarding and using backtracking instead. Another possible solution is the use of the operator splitting method introduced in [44] 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 (12) instead of the true transition matrix P_k 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 $\|P_k - \widetilde{P}_k\|_\infty \leq 1 - \gamma$ [44, Thm. 1], which is difficult to achieve for a low-rank approximation $\widetilde{P}_k$ of P_k.

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. The main difficulty to be addressed is the fact that the linear constraints in (12) are not the standard



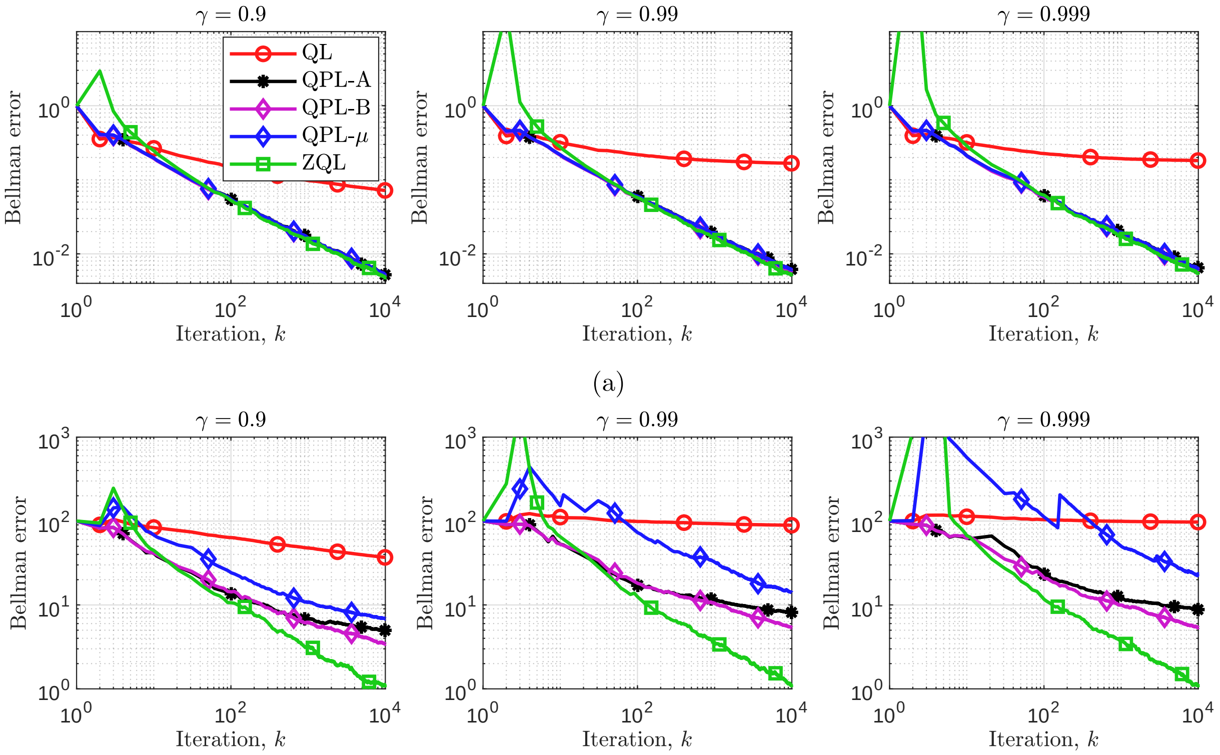
(a)

(b)

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.

secant conditions used in QNMs. The modified implementation of Section 3.2.3 partially addresses this issue by including the secant conditions in its approximation. An interesting result, however, is the establishment of a local superlinear convergence rate for QPI without imposing the secant condition. Another possibility is to use the results for Anderson acceleration in [16] to establish an improved linear rate for convergence. To that end, similar to what is done in [48], 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.

The proposed algorithms in this study heavily rely on the approximation (12) 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. However, our numerical simulations using these alternative priors did not demonstrate a clear improvement in the performance of the proposed algorithms, highlighting the need for further investigation on other MDPs. In this regard, we also note that the main drawback of the approximation (12) 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.



(a)

(b)

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.

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 [22]. The MDP 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 [22, 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 [15]. The MDP has 18 state-action pairs in total and corresponds to a simple path-finding problem. We refer the reader to [15, Sec. 3] for the description of the MDP.

Appendix B. Accelerated VI and QL algorithms

The update rules are as follows:

(1) Nesterov accelerated VI [22] – with $v_{-1} = v_0$:

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

(2) Anderson accelerated VI [19] – with $v_{-1} = v_0$:

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

(3) Speedy QL [20] – with $q_{-1} = q_0$, $\alpha_k = \frac{1}{k+1}$:

$$\begin{cases} d_k = \widehat{T}(q_k) - \widehat{T}(q_{k-1}), \\ q_{k+1} = q_k - \alpha_k\big(q_k - \widehat{T}(q_{k-1})\big) + (1-\alpha_k)d_k. \end{cases}$$

(4) Zap QL [15] – with $D_{-1} = \mathbf{0}$, $\alpha_k = \beta_k = \frac{1}{k+1}$:

$$\begin{cases} D_k = (1-\beta_k)D_{k-1} + \beta_k\big(I - \gamma\widehat{P}(q_k)\big), \\ q_{k+1} = q_k - \alpha_k D_k^{-1}\big(q_k - \widehat{T}(q_k)\big), \end{cases}$$

where $\widehat{P}(q)$ is the synchronously sampled state-action transition matrix of the Markov chain under the greedy policy π_q w.r.t. q with elements $[\widehat{P}(q)]\big((s,a),(s',a')\big) = 1$ if $s' = \hat{s}^+$, $a' = \pi_q(\hat{s}^+)$ and $= 0$ otherwise, 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).

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. (2022a). *Abstract dynamic programming.* Athena Scientific.

[8] Bertsekas, D. (2022b). *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control.* Athena Scientific.

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

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

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

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

[13] 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.

[14] 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.

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

[16] 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.

[17] Facchinei, F. and Pang, J. (2003). *Finite-dimensional variational inequalities and complementarity problems*, volume 2. Springer.

[18] 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.

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

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

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

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

[23] 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.*

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

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

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

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

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

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

[35] Liu, Y. and Kolarijani, M. A. S. (2024). Fitted Q-iteration via max-plus-linear approximation. *IEEE Control Systems Letters*, 8:3201–3206.

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

[37] 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.

[38] 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.

[39] Perkins, S. and Leslie, D. S. (2013). Asynchronous stochastic approximation with differential inclusions. *Stochastic Systems*, 2(2):409–446.

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

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

[42] Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons.

[43] 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.

[44] 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.

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

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

[47] Sun, D. and Han, J. (1997). Newton and quasi-newton methods for a class of nonsmooth equations and related problems. *SIAM Journal on Optimization*, 7(2):463–480.

[48] 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.

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

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

[51] 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.

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

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

[54] 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.

[55] 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*.

[56] 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.