Rank-One Modified Value Iteration

Arman S. Kolarijani [1] **Tolga Ok** [1] **Peyman Mohajerin Esfahani** [1 2] **Mohamad Amin Sharifi Kolarijani** [1]

Abstract

In this paper, we provide a novel algorithm for solving planning and learning problems of Markov decision processes. The proposed algorithm follows a policy iteration-type update by using a rank-one approximation of the transition probability matrix in the policy evaluation step. This rank-one approximation is closely related to the stationary distribution of the corresponding transition probability matrix, which is approximated using the power method. We provide theoretical guarantees for the convergence of the proposed algorithm to optimal (action-)value function with the same rate and computational complexity as the value iteration algorithm in the planning problem and as the Q-learning algorithm in the learning problem. Through our extensive numerical simulations, however, we show that the proposed algorithm consistently outperforms first-order algorithms and their accelerated versions for both planning and learning problems.

1. Introduction

Value iteration (*VI*) and *policy iteration* (*PI*) algorithms lie at the heart of most if not all algorithms for optimal control of *Markov decision processes* (*MDPs*) in both cases of the *planning problem* (i.e., with access to the true model of the MDP) and the *reinforcement learning problem* (i.e., with access to samples of the MDP) (Sutton & Barto, 2018; Bertsekas, 2023). Their widespread application stems from their simple implementation and straightforward combination with function approximation schemes such as neural networks. Both VI and PI are iterative algorithms that ultimately find the fixed-point of the *Bellman* (*optimality*) *operator* $\mathbf{T}$: For γ-*discounted*, *finite state-action* MDPs, the *value function* $\boldsymbol{v}_k$ at iteration k is given by

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + \boldsymbol{G}_k\big(\mathbf{T}(\boldsymbol{v}_k) - \boldsymbol{v}_k\big), \quad k = 0, 1, \dots, \tag{1}$$

where $\boldsymbol{G}_k = \boldsymbol{I}$ in the VI algorithm and $\boldsymbol{G}_k = (\boldsymbol{I} - \gamma \boldsymbol{P}_k)^{-1}$ in the PI algorithm, $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{P}_k$ is the *state transition probability matrix* of the MDP under the *greedy policy* with respect to $\boldsymbol{v}_k$. Both algorithms are guaranteed to converge to the optimal value function and control policy; see, e.g., (Puterman, 2014, Thms. 6.3.3 and 6.4.2). When it comes to computational complexity, one observes a trade-off between the two algorithms: VI has a lower per-iteration complexity compared to PI, while PI converges in a smaller number of iterations compared to VI. The faster convergence of PI is partially explained by its second-order nature which leads to a local quadratic convergence rate (Puterman & Brumelle, 1979; Bertsekas, 2022; Gargiani et al., 2022), compared to the linear convergence rate of VI (Puterman, 2014, Thms. 6.3.3).

This trade-off between the two algorithms has motivated much research to improve the convergence rate of VI and/or the per-iteration complexity of PI. One of the first improvements is the *Relaxed VI* algorithm (Kushner & Kleinman, 1971; Porteus & Totten, 1978) which allows for a greater step size compared to the standard VI algorithm. A large body of research in this area originates from the correspondence between VI and gradient descent method and between PI and Newton method (Grand-Clément, 2021; Kolarijani & Mohajerin Esfahani, 2023). Here, approaches such as accelerated gradient descent and quasi-Newton methods from the optimization literature are adapted to develop modified versions of VI and PI. For instance, the combination of the VI algorithm with *Nesterov acceleration* (Nesterov, 1983) and *Anderson acceleration* (Anderson, 1965) have been explored in (Goyal & Grand-Clément, 2022) and (Zhang et al., 2020), respectively, for solving the planning problem. More recently, *Halpern's anchoring acceleration* scheme (Halpern, 1967) has been used to introduce the *Anchored VI* algorithm (Lee & Ryu, 2024) which in particular exhibits a $\mathcal{O}(1/k)$-rate for large values of discount factor and even for $\gamma = 1$. In the case of the learning problem, *Speedy Q-Learning* (Ghavamzadeh et al., 2011), *Momentum Q-Learning* (Weng et al., 2021), and *Nesterov Stochastic Approximation* (Devraj et al., 2019) are among the algorithms that use the idea of momentum to achieve a better rate of convergence compared to standard Q-learning (QL).

[1]Delft University of Technology, The Netherlands [2]University of Toronto, Canada. Correspondence to: Arman S. Kolarijani <a.sharifikolarijani@tudelft.nl>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

The *Zap Q-Learning* algorithm (Devraj & Meyn, 2017) can be thought of as a second-order learning algorithm which was inspired by the stochastic Newton-Raphson (SNR) algorithm (Ruppert, 1985). The *Quasi-Policy Iteration/Learning* algorithms (Kolarijani & Mohajerin Esfahani, 2023) are, on the other hand, an example of using the idea of quasi-Newton methods for developing algorithms for optimal control of MDPs. Another class of modified VI algorithms is the *Generalized Second-Order VI* algorithm (Kamanchi et al., 2022) which applies the Newton method on a *smoothed* version of the Bellman operator. Tools and techniques from linear algebra have also been exploited to modify the VI algorithm, particularly for policy evaluation. The *Operator Splitting VI* algorithm (Rakhsha et al., 2022) is an example that exploits the matrix splitting method for solving the linear equation corresponding to policy evaluation for a given "cheap-to-access" model of the underlying MDP. Recently, in (Lee et al., 2024), the authors combined the matrix splitting method with the matrix deflation techniques for removing the dominant eigenstructure of the transition probability matrix to speed up the policy evaluation.

Contribution. In this paper, we propose a novel algorithm that modifies the VI algorithm by incorporating a computationally efficient PI-type update rule. To this end, we consider the update rule (1) with a matrix gain of the form $\boldsymbol{G}_k = (\boldsymbol{I} - \widetilde{\boldsymbol{P}}_k)^{-1}$, where $\widetilde{\boldsymbol{P}}_k$ is a *rank-one approximation* of $\boldsymbol{P}_k$. To be precise, we consider the approximation $\widetilde{\boldsymbol{P}}_k = \mathbf{1}\boldsymbol{d}_k^\top$, where $\boldsymbol{d}_k = \boldsymbol{P}_k^\top \boldsymbol{d}_k$ is the *stationary* distribution of $\boldsymbol{P}_k$ (assuming it exists) and $\mathbf{1}$ is the all-one vector. The proposed algorithm then uses the *power method* for approximating $\boldsymbol{d}_k$ iteratively using the true matrix $\boldsymbol{P}_k$ in the planning problem and its sampled version in the learning problem. In particular,

(1) we propose the *Rank-one VI* (*R1-VI*) Algorithm 1 as the modified VI algorithm for solving the planning problem and prove its convergence to the optimal value function (Theorem 3.3);

(2) we propose the *Rank-one QL* (*R1-QL*) Algorithm 2 as the modified QL algorithm for solving the learning problem and prove its convergence to the optimal Q-function (Theorem 4.2);

(3) we compare the proposed R1-VI and R1-QL algorithms with the state-of-the-art algorithms for solving planning and learning problems of MDPs and show the empirically faster convergence of the proposed algorithms compared the ones with the same per-iteration computational complexity (i.e., the first-order algorithms and their accelerated versions).

Paper organization. In Section 2, we provide the necessary background and the problem definition along with the standard VI and QL algorithms for solving the planning and learning problems of MDPs. The proposed R1-VI algorithm for solving the planning problem and its analysis are discussed in Section 3. Section 4 presents the R1-QL algorithm for solving the learning problem and its analysis. In Section 5, we provide the results of our extensive numerical simulations and compare the proposed algorithms with a range of existing algorithms for solving the optimal control problem of MDPs. Finally, some limitations of the proposed algorithms and future research directions are discussed in Section 6. All the technical proofs are provided in Appendix A.

Notations. The set of real numbers is denoted by $\mathbb{R}$. For a vector $\boldsymbol{v} \in \mathbb{R}^n$, we use $\boldsymbol{v}(i)$ and $[\boldsymbol{v}](i)$ to denote its i-th element. Similarly, $\boldsymbol{M}(i,j)$ and $[\boldsymbol{M}](i,j)$ denote the element in i-th row and j-th column of the matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$. We use $\langle \boldsymbol{v}, \boldsymbol{u}\rangle = \sum_{i=1}^n \boldsymbol{v}(i)\cdot\boldsymbol{u}(i)$ to denote the the inner product of the two vectors $\boldsymbol{v}, \boldsymbol{u} \in \mathbb{R}^n$. $\|\boldsymbol{v}\|_1 = \sum_{i=1}^n |\boldsymbol{v}(i)|$, $\|\boldsymbol{v}\|_2 = \sqrt{\langle \boldsymbol{v}, \boldsymbol{v}\rangle}$, and $\|\boldsymbol{v}\|_\infty = \max_{i=1}^n |\boldsymbol{v}(i)|$ denote the 1-norm, 2-norm, and ∞-norm of the vector $\boldsymbol{v} \in \mathbb{R}^n$, respectively. We use $\rho(\boldsymbol{M})$ to denote the spectral radius (i.e., the largest eigenvalue in absolute value) of a square matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$. Given a set $\mathcal{X}$, $\Delta(\mathcal{X})$ denotes the set of probability distributions on $\mathcal{X}$. Let $x \sim P$ be a random variable with distribution $P \in \Delta(\mathcal{X})$. We use $\widehat{x} \sim P$ to denote *a sample of the random variable* x drawn from the sample space $\mathcal{X}$ of x according to the distribution P. We use $\mathbf{1}$, $\mathbf{0}$, and $\boldsymbol{I}$ to denote the all-one vector, the all-zero vector, and the identity matrix, respectively, with their dimension being clear from the context.

2. Optimal control of MDPs

Consider a finite MDP $(\mathcal{S}, \mathcal{A}, P, c, \gamma)$. Here, $\mathcal{S} := \{1, 2, \dots, n\}$ and $\mathcal{A} := \{1, 2, \dots, m\}$ are the *state* and *action spaces*, respectively. The *transition kernel* $P : \mathcal{S}\times\mathcal{A} \to \Delta(\mathcal{S})$ is the conditional probability $P(s^+|s,a)$ of the transition to state s^+ given the current state-action pair (s,a). The function $\boldsymbol{c} \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|} = \mathbb{R}^{nm}$, bounded from below, represents the *stage cost* $\boldsymbol{c}(s,a)$ of taking the control action a while the system is in state s. And, $\gamma \in (0,1)$ is the *discount factor* which can be seen as a trade-off parameter between short- and long-term costs. A *control policy* $\pi : \mathcal{S} \to \mathcal{A}$ is a mapping from states to actions. Fix policy π. For the corresponding Markov chain under the policy π we define: (i) the *state transition probability matrix* $\boldsymbol{P}^\pi \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|} = \mathbb{R}^{n\times n}$ where $\boldsymbol{P}^\pi(s, s^+) = P\big(s^+|s,\pi(s)\big)$ for $s, s^+ \in \mathcal{S}$, (ii) the *state-action transition probability matrix* $\overline{\boldsymbol{P}}^\pi \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|\times|\mathcal{S}\times\mathcal{A}|} = \mathbb{R}^{(nm)\times(nm)}$ where $\overline{\boldsymbol{P}}^\pi\big((s,a),(s^+,a^+)\big) = P(s^+|s,a)$ if $a^+ = \pi(s^+)$ and $= 0$ otherwise for $(s,a),(s^+,a^+) \in \mathcal{S}\times\mathcal{A}$, and (iii) the stage cost $\boldsymbol{c}^\pi \in \mathbb{R}^{|\mathcal{S}|} = \mathbb{R}^n$ where $\boldsymbol{c}^\pi(s) = \boldsymbol{c}\big(s,\pi(s)\big)$ for $s \in \mathcal{S}$.

Under policy π, the *value function* $\boldsymbol{v}^\pi \in \mathbb{R}^{|\mathcal{S}|} = \mathbb{R}^n$ is the *expected discounted cost* endured by following policy π over an infinite-horizon trajectory, that is, for each $s \in \mathcal{S}$,

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

The *action-value function* $\boldsymbol{q}^\pi \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|} = \mathbb{R}^{nm}$ (or the so-called *Q-function*) under policy π is defined as follows: for each $(s,a) \in \mathcal{S}\times\mathcal{A}$

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

These functions can be shown to satisfy the fixed-point equations (Puterman, 2014, Thm. 6.1.1)

$$\boldsymbol{v}^\pi = \boldsymbol{c}^\pi + \gamma \boldsymbol{P}^\pi \boldsymbol{v}^\pi, \quad \boldsymbol{q}^\pi = \boldsymbol{c} + \gamma \overline{\boldsymbol{P}}^\pi \boldsymbol{q}^\pi. \tag{2}$$

The problem of interest is to *control* the MDP in a manner that the expected, discounted, infinite-horizon cost is minimized. To do so, one aims to find the *optimal policy* π^* such that for any policy π,

$$\boldsymbol{v}^\star(s) \coloneqq \boldsymbol{v}^{\pi^\star}(s) \le \boldsymbol{v}^\pi(s), \quad \forall s \in \mathcal{S},$$

or, equivalently,

$$\boldsymbol{q}^\star(s,a) \coloneqq \boldsymbol{q}^{\pi^\star}(s,a) \le \boldsymbol{q}^\pi(s,a), \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A}.$$

The optimal value function can be characterized as the solution to the fixed-point equation $\boldsymbol{v}^\star = \mathbf{T}(\boldsymbol{v}^\star)$, where $\mathbf{T} : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ is the so-called *Bellman* (*optimality*) *operator* defined as follows: for each $s \in \mathcal{S}$

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

The operator $\mathbf{T}$ is a γ-contraction in the ∞-norm (i.e., $\|\mathbf{T}(\boldsymbol{v}) - \mathbf{T}(\boldsymbol{w})\|_\infty \le \gamma \|\boldsymbol{v} - \boldsymbol{w}\|_\infty$ for all $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^n$) (Puterman, 2014, Prop. 6.2.4). This contraction property is essentially the basis for the VI algorithm, introduced in (4).

$$\begin{array}{l}\texttt{intialize } \boldsymbol{v}_0 \in \mathbb{R}^{|\mathcal{S}|} = \mathbb{R}^n\\ \texttt{for } k = 0, 1, \ldots\\ \quad \boldsymbol{v}_{k+1}(s) = [\mathbf{T}(\boldsymbol{v}_k)](s), \quad \forall s \in \mathcal{S}\\ \texttt{endfor}\end{array} \tag{4}$$

From the Banach fixed-point theorem (see, e.g., (Puterman, 2014, Thm. 6.2.3)), the VI algorithm converges to $\boldsymbol{v}^\star$ with a linear rate γ. Correspondingly, one can derive the fixed-point characterization $\boldsymbol{q}^\star = \overline{\mathbf{T}}(\boldsymbol{q}^\star)$ of the optimal Q-function, where

$$[\overline{\mathbf{T}}(\boldsymbol{q}^\star)](s,a) \coloneqq \boldsymbol{c}(s,a) + \gamma\, \mathbb{E}_{s^+\sim P(\cdot|s,a)}\left[\min_{a^+\in\mathcal{A}} \boldsymbol{q}^\star(s^+,a^+)\right],$$

for each $(s,a) \in \mathcal{S}\times\mathcal{A}$ (Szepesvári, 2009, Fact 3). This characterization is particularly useful when one only has access to *samples* $\widehat{s}^{\,+} \sim P(\cdot|s,a)$ of the next state s^+ for each state-action pair $(s,a) \in \mathcal{S}\times\mathcal{A}$ (and not to the true transition probability kernel of the MDP). In particular, let us define the *empirical Bellman operator* $\widehat{\mathbf{T}} : \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|} \times \mathcal{S} \to \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$ as follows: for each $(s,a,s^+) \in \mathcal{S}\times\mathcal{A}\times\mathcal{S}$

$$[\widehat{\mathbf{T}}(\boldsymbol{q}, s^+)](s,a) \coloneqq \boldsymbol{c}(s,a) + \gamma \min_{a^+\in\mathcal{A}} \boldsymbol{q}(s^+,a^+).$$

A classic algorithm for the learning problem is then the (*synchronous*) QL algorithm (Watkins & Dayan, 1992; Kearns & Singh, 1998), given in (5).

$$\begin{array}{l}\texttt{intialize } \boldsymbol{q}_0 \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|} = \mathbb{R}^{nm}\\ \texttt{for } k = 0, 1, \ldots\\ \quad \texttt{for } (s,a) \in \mathcal{S}\times\mathcal{A}\\ \quad\quad \widehat{s}_k^{\,+} \sim P(\cdot|s,a)\\ \quad\quad \delta_k = \boldsymbol{q}_k(s,a) - [\widehat{\mathbf{T}}(\boldsymbol{q}_k, \widehat{s}_k^{\,+})](s,a)\\ \quad\quad \boldsymbol{q}_{k+1}(s,a) = \boldsymbol{q}_k(s,a) - \lambda_k \delta_k\\ \quad \texttt{endfor}\\ \texttt{endfor}\end{array} \tag{5}$$

Above, $\lambda_k \ge 0$ are the step-sizes, The QL algorithm is also guaranteed to converge to $\boldsymbol{q}^\star$ almost surely given that the step-sizes satisfy the *Robbins–Monro conditions* (i.e., $\sum_{k=0}^\infty \lambda_k = \infty$ and $\sum_{k=0}^\infty \lambda_k^2 < \infty$) (Tsitsiklis, 1994; Jaakkola et al., 1993). In particular, with a polynomial step-size $\lambda_k = 1/(1+k)^\omega$ with $\omega \in (1/2, 1)$, QL outputs an ϵ-accurate Q-function with high probability after $\tilde{\mathcal{O}}(\tau^{-4/\omega}\cdot\epsilon^{-2/\omega} + \tau^{1/(1-\omega)})$ iterations of synchronous sampling with $\tau = 1-\gamma$ (Even-Dar & Mansour, 2003), while with a re-scaled linear step-size $\lambda_k = 1/(1+\tau k)$, QL has been shown to require $\tilde{\mathcal{O}}(\tau^{-5}\cdot\epsilon^{-2})$ iterations of synchronous sampling for the same performance (Wainwright, 2019).

3. Rank-one value iteration (R1-VI)

Another well-known algorithm for solving the planning problem in MDPs is the PI algorithm. In order to provide this algorithm in a compact form, let us first introduce the notion of *greedy policy*. Given a value function $\boldsymbol{v} \in \mathbb{R}^n$, the greedy policy with respect to $\boldsymbol{v}$, denoted by $\pi^{\boldsymbol{v}} : \mathcal{S} \to \mathcal{A}$, is

$$\pi^{\boldsymbol{v}}(s) \in \operatorname*{argmin}_{a\in\mathcal{A}} \mathbb{E}_{s^+\sim P(\cdot|s,a)}\left[\boldsymbol{c}(s,a) + \gamma \boldsymbol{v}(s^+)\right], \forall s \in \mathcal{S}. \tag{6}$$

The PI algorithm is then summarized in (7).

$$\begin{array}{l}\texttt{intialize } \pi_0 : \mathcal{S} \to \mathcal{A}\\ \texttt{for } k = 0, 1, \ldots\\ \quad \boldsymbol{v}_k = \boldsymbol{v}^{\pi_k} \quad \text{[policy evaluation – eq. (2)]}\\ \quad \pi_{k+1} = \pi^{\boldsymbol{v}_k} \quad \text{[policy improvement – eq. (6)]}\\ \texttt{endfor}\end{array} \tag{7}$$

The algorithm to be proposed in this section is based on an alternative representation of the iterations of the PI algorithm; see also (Puterman, 2014, Prop. 6.5.1).

Lemma 3.1 (Policy iteration)**.** *Each iteration of the PI algorithm (7) equivalently reads as*

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + (\boldsymbol{I} - \gamma \boldsymbol{P}_k)^{-1}\big(\mathbf{T}(\boldsymbol{v}_k) - \boldsymbol{v}_k\big), \tag{8}$$

where $\boldsymbol{P}_k \coloneqq \boldsymbol{P}^{\pi^{\boldsymbol{v}_k}}$ is the state transition probability matrix of the MDP under the greed policy $\pi^{\boldsymbol{v}_k}$.

The PI algorithm outputs the optimal policy in a finite number of iterations (Puterman, 2014, 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 (Bertsekas, 2022; Gargiani et al., 2022). The faster convergence of PI compared to VI comes however 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 (8). To address this issue, we propose to use a low-rank approximation of $\boldsymbol{P}_k$ instead. Such approach allows us to approximate $(\boldsymbol{I} - \gamma \boldsymbol{P}_k)^{-1}$ with a reduced computational cost by using the Woodbury formula (Hager, 1989). To be precise, we propose the *rank-one VI (R1-VI) algorithm*

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + (\boldsymbol{I} - \gamma \widetilde{\boldsymbol{P}}_k)^{-1}\big(\mathbf{T}(\boldsymbol{v}_k) - \boldsymbol{v}_k\big), \tag{9}$$

where

$$\widetilde{\boldsymbol{P}}_k = \mathbf{1}\boldsymbol{d}_k^\top, \tag{10}$$

is a *rank-one* approximation of the true transition probability matrix $\boldsymbol{P}_k$ at iteration k, with $\boldsymbol{d}_k \coloneqq \boldsymbol{d}_{\pi_k} \in \Delta(\mathcal{S})$ being a stationary distribution of the greedy policy $\pi_k = \pi_{\boldsymbol{v}_k}$, i.e., a solution of $\boldsymbol{d}_k^\top \boldsymbol{P}_k = \boldsymbol{d}_k^\top$. Under certain conditions, (10) is indeed "the best" rank-1 approximation of $\boldsymbol{P}_k$:

Lemma 3.2 (Rank-1 approximation)**.** *Assume that the transition probability matrix $\boldsymbol{P}_k$ is ergodic (i.e., irreducible and aperiodic). Then,*

$$\begin{aligned} \mathbf{1}\boldsymbol{d}_k^\top = \operatorname*{argmin}_{\boldsymbol{P} \in \mathbb{R}^{n\times n}} \ & \rho(\boldsymbol{P} - \boldsymbol{P}_k) \\ \text{s.t. } & \boldsymbol{P} \geq 0,\ \boldsymbol{P}\mathbf{1} = \mathbf{1},\ \operatorname{rank}(\boldsymbol{P}) = 1. \end{aligned} \tag{11}$$

where $\boldsymbol{d}_k$ is the unique stationary distribution of $\boldsymbol{P}_k$. That is, $\widetilde{\boldsymbol{P}}_k = \mathbf{1}\boldsymbol{d}_k^\top$ is the best rank-1 approximation of $\boldsymbol{P}_k$ in terms of the spectral radius.

Using the Woodbury formula, we then have

$$(\boldsymbol{I} - \gamma \widetilde{\boldsymbol{P}}_k)^{-1} = (\boldsymbol{I} - \gamma \mathbf{1}\boldsymbol{d}_k^\top)^{-1} = I + \tfrac{\gamma}{1-\gamma}\mathbf{1}\boldsymbol{d}_k^\top,$$

and hence the R1-VI update (9) reads as

$$\boldsymbol{v}_{k+1} = \mathbf{T}(\boldsymbol{v}_k) + \tfrac{\gamma}{1-\gamma}\big\langle \boldsymbol{d}_k, \mathbf{T}(\boldsymbol{v}_k) - \boldsymbol{v}_k \big\rangle \mathbf{1}. \tag{12}$$

Algorithm 1 Rank-One Value Iteration (R1-VI)

Input: transition kernel $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$; cost function $\boldsymbol{c} \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$; discount factor $\gamma \in (0,1)$;
Output: optimal value function $v^\star$

initialize: $\boldsymbol{v}_0 \in \mathbb{R}^{|\mathcal{S}|}$, $\boldsymbol{d}_{-1} \in \Delta(\mathcal{S})$;
for $k = 0, 1, 2, \ldots$ **do**
  $\boldsymbol{P}_k = \mathbf{0} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$;
  for $s \in \mathcal{S}$ **do**

$$\begin{cases} a_k \in \operatorname*{argmin}_{a\in\mathcal{A}} \big\{\boldsymbol{c}(s,a) + \gamma\, \mathbb{E}_{s^+}\big[\boldsymbol{v}_k(s^+)\big]\big\}, \\ [\mathbf{T}(\boldsymbol{v}_k)](s) = \min_{a\in\mathcal{A}} \big\{\boldsymbol{c}(s,a) + \gamma\, \mathbb{E}_{s^+}\big[\boldsymbol{v}_k(s^+)\big]\big\}; \end{cases}$$

    $\boldsymbol{P}_k(s, s^+) = P(s^+|s, a_k),\ \forall s^+ \in \mathcal{S}$;
  end for
  $\boldsymbol{f} = \boldsymbol{P}_k^\top \boldsymbol{d}_{k-1}$; $\boldsymbol{d}_k = \boldsymbol{f}/\|\boldsymbol{f}\|_1$;
  $\boldsymbol{v}_{k+1} = \mathbf{T}(\boldsymbol{v}_k) + \frac{\gamma}{1-\gamma}\big\langle \boldsymbol{d}_k, \mathbf{T}(\boldsymbol{v}_k) - \boldsymbol{v}_k \big\rangle \mathbf{1}$;
end for

Next to be addressed is the computation of the vector $\boldsymbol{d}_k$. Considering the fact that $\boldsymbol{d}_k$ is a left eigenvector of $\boldsymbol{P}_k$ corresponding to the eigenvalue 1, we can use the power method (Golub & Van Loan, 2013, Sec. 7.3) to compute it as follows

$$\boldsymbol{f} = \boldsymbol{P}_k^\top \boldsymbol{d}_k^{(i)},\ \boldsymbol{d}_k^{(i+1)} = \frac{\boldsymbol{f}}{\|\boldsymbol{f}\|_1},\ i = 0, 1, \ldots, I-1, \tag{13}$$

with some initialization $\boldsymbol{d}_k^{(0)} \in \Delta(\mathcal{S})$ and $I \in \{1, 2, \ldots\}$. We then use $\boldsymbol{d}_k = \boldsymbol{d}_k^{(I)}$ in the update rule (12). We note that the normalization is only to avoid the accumulation of numerical errors; to see this note that for the row stochastic matrix $\boldsymbol{P}_k$, we have $\boldsymbol{P}_k^\top \boldsymbol{d} \in \Delta(\mathcal{S})$ for any $\boldsymbol{d} \in \Delta(\mathcal{S})$. Under the assumptions of Lemma 3.2, the preceding iteration converges linearly to the unique stationary distribution with a rate equal to the second largest eigenvalue modulus of $\boldsymbol{P}_k$ (Gallager, 2011, Thm. 3.4.1).

The complete description of the proposed R1-VI algorithm is provided in Algorithm 1. We note that Algorithm 1 includes a single iteration (i.e., $I = 1$) of the power method in (13) initialized by $\boldsymbol{d}_k^{(0)} = \boldsymbol{d}_{k-1}$. The reason for this choice is that the greedy policy $\pi^{\boldsymbol{v}_k}$ and hence the corresponding transition matrix $\boldsymbol{P}_k$ usually stays the same over multiple iterations k of the algorithm in the value space. This means that the algorithm effectively performs multiple iterations of the power method. Despite using this *approximation* $\boldsymbol{d}_k$ of the stationary distribution of $\boldsymbol{P}_k$ with a single iteration of the power method, the proposed algorithm can be shown to converge.

Theorem 3.3 (Convergence of R1-VI)**.** *The iterates $\boldsymbol{v}_k$ of the R1-VI Algorithm 1 converge to the optimal value function $\boldsymbol{v}^\star = \mathbf{T}(\boldsymbol{v}^\star)$ with at least the same rate as VI, i.e., with linear rate γ.*

Let us also note that the per-iteration complexity of the

proposed R1-VI Algorithm 1 is $\mathcal{O}(n^2m)$, i.e., the same as that of VI. We finish this section with the following remark.

Remark 3.4 (Generalization to modified policy iteration)**.** *Recall the generic value update rule*

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + \boldsymbol{G}_k\big(\mathbf{T}(\boldsymbol{v}_k) - \boldsymbol{v}_k\big), \quad k = 0, 1, \ldots, \tag{14}$$

with the gain matrix $\boldsymbol{G}_k = \boldsymbol{I}$ in the VI algorithm and $\boldsymbol{G}_k = (\boldsymbol{I} - \gamma \boldsymbol{P}_k)^{-1}$ in the PI algorithm. Also, note that $(\boldsymbol{I} - \gamma \boldsymbol{P}_k)^{-1} = \sum_{\ell=0}^{\infty} \gamma^\ell \boldsymbol{P}_k^\ell$ since $\rho(\boldsymbol{I} - \gamma \boldsymbol{P}_k) < 1$ (Puterman, 2014, Cor. C.4). Inserting the truncated sum

$$\boldsymbol{G}_k = \sum_{\ell=0}^{L} \gamma^\ell \boldsymbol{P}_k^\ell,$$

with $L \in \{0, 1, \ldots\}$ in the update rule (14), we derive the Modified PI *(*MPI*) algorithm which converges linearly with rate γ for any choice of L (Puterman, 2014, Thm. 6.5.5). (Observe that $L = 0$ and $L = \infty$ correspond to the standard VI and PI algorithms, respectively). The proposed rank-one modification can be in general combined with the MPI algorithm. Indeed, we have*

$$\begin{aligned}(\boldsymbol{I} - \gamma \boldsymbol{P}_k)^{-1} &= \sum_{\ell=0}^{\infty} \gamma^\ell \boldsymbol{P}_k^\ell = \sum_{\ell=0}^{L-1} \gamma^\ell \boldsymbol{P}_k^\ell + \gamma^L \boldsymbol{P}_k^L \sum_{\ell=0}^{\infty} \gamma^\ell \boldsymbol{P}_k^\ell \\ &= \sum_{\ell=0}^{L-1} \gamma^\ell \boldsymbol{P}_k^\ell + \gamma^L \boldsymbol{P}_k^L (\boldsymbol{I} - \gamma \boldsymbol{P}_k)^{-1}.\end{aligned}$$

Then, by using the approximation $\widetilde{\boldsymbol{P}}_k = \mathbf{1}\boldsymbol{d}_k^\top$ in the matrix inversion on the right-hand side of the equation above, we derive the gain matrix of the rank-one MPI (R1-MPI) *algorithm to be*

$$\begin{aligned}\boldsymbol{G}_k &= \sum_{\ell=0}^{L-1} \gamma^\ell \boldsymbol{P}_k^\ell + \gamma^L \boldsymbol{P}_k^L (\boldsymbol{I} - \gamma \widetilde{\boldsymbol{P}}_k)^{-1} \\ &= \sum_{\ell=0}^{L} \gamma^\ell \boldsymbol{P}_k^\ell + \frac{\gamma^{L+1}}{1-\gamma} \mathbf{1}\boldsymbol{d}_k^\top.\end{aligned}$$

Observe that R1-VI is now a special case of R1-MPI with $L = 0$.

4. Rank-one Q-learning (R1-QL)

In this section, we focus on the learning problem in which we have access to a generative model that provides us with samples of the MDP (as opposed to access to the true transition probability kernel of the MDP in the planning problem). To start, let us provide the PI update rule for the Q-function. The proof is similar to the proof of Lemma 3.1 and omitted.

Lemma 4.1 (Policy iteration for Q-function)**.** *Each iteration of the PI algorithm for the Q-function is given by*

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + (\boldsymbol{I} - \gamma \overline{\boldsymbol{P}}_k)^{-1}\big(\overline{\mathbf{T}}(\boldsymbol{q}_k) - \boldsymbol{q}_k\big), \tag{15}$$

where $\overline{\boldsymbol{P}}_k := \overline{\boldsymbol{P}}^{\pi^{\boldsymbol{q}_k}}$ is the state-action transition probability matrix of the MDP under the greed policy $\pi^{\boldsymbol{q}_k}$.

The idea is again to use the rank-one approximation $\widetilde{\boldsymbol{P}}_k = \mathbf{1}\boldsymbol{d}_k^\top$ of the matrix $\overline{\boldsymbol{P}}_k$ in the update rule (15), where $\boldsymbol{d}_k$ is now the stationary distribution of $\overline{\boldsymbol{P}}_k$. This leads to the update rule

$$\boldsymbol{q}_{k+1} = \overline{\mathbf{T}}(\boldsymbol{q}_k) + \frac{\gamma}{1-\gamma}\big\langle \boldsymbol{d}_k, \overline{\mathbf{T}}(\boldsymbol{q}_k) - \boldsymbol{q}_k \big\rangle \mathbf{1},$$

at each iteration of the planning problem. Then, for the learning problem, considering the *synchronous* update of all state-action pairs $(s, a) \in \mathcal{S} \times \mathcal{A}$ at each iteration k, we arrive at the *rank-one Q-learning (R1-QL)* update rule

$$\begin{aligned}\alpha_k &= \frac{\gamma \lambda_k}{1-\gamma}\big\langle \widehat{\boldsymbol{d}}_k, \widehat{\mathbf{T}}_k(\boldsymbol{q}_k) - \boldsymbol{q}_k \big\rangle, \\ \boldsymbol{q}_{k+1} &= (1 - \lambda_k)\boldsymbol{q}_k + \lambda_k \widehat{\mathbf{T}}_k(\boldsymbol{q}_k) + \alpha_k \mathbf{1},\end{aligned} \tag{16}$$

where $\lambda_k \geq 0$ are properly chosen step-sizes satisfying the Robbins–Monro conditions (e.g., $\lambda_k = 1/(k+1)$), and $\widehat{\mathbf{T}}_k$ is the empirical Bellman operator evaluated at iteration k, i.e., $[\widehat{\mathbf{T}}_k(\boldsymbol{q}_k)](s, a) := [\widehat{\mathbf{T}}(\boldsymbol{q}_k, \widehat{s}_k^+)](s, a)$ with $\widehat{s}_k^+ \sim P(\cdot|s, a)$ for each $(s, a) \in \mathcal{S} \times \mathcal{A}$ – the subscript k in $\widehat{\mathbf{T}}_k$ denotes the dependence on the next-state sample $\widehat{s}_k^+$ generated at iteration k.

What remains to be addressed is computing the estimation $\widehat{\boldsymbol{d}}_k$ of the stationary distribution in (16) using the samples. At each iteration k, define the sparse matrix $\boldsymbol{F}_k \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|\times|\mathcal{S}\times\mathcal{A}|} = \mathbb{R}^{(nm)\times(nm)}$ with exactly one nonzero entry equal to 1 in each row $(s, a) \in \mathcal{S} \times \mathcal{A}$ corresponding to the column (s^+, a^+), where

$$\begin{aligned}s^+ &= \widehat{s}_k^+ \sim P(\cdot|s, a), \\ a^+ &= \widehat{a}_k^+ = \operatorname*{argmin}_{a^+ \in \mathcal{A}} \boldsymbol{q}_k(\widehat{s}_k^+, a^+).\end{aligned} \tag{17}$$

Observe that the matrix $\boldsymbol{F}_k$ is a sampled version of the state-action transition probability matrix $\overline{\boldsymbol{P}}_k$. Using this sample, we can form the stochastic approximation

$$\widehat{\boldsymbol{P}}_k = (1 - \lambda_k)\widehat{\boldsymbol{P}}_{k-1} + \lambda_k \boldsymbol{F}_k.$$

for the state-action transition probability matrix. We note that the same approximation is used in the Zap Q-learning algorithm (Devraj & Meyn, 2017). With this approximation in hand, we can again use the power method for finding the stationary distribution. In particular, with a single iteration of the power method initialized by the previous stationary distribution $\widehat{\boldsymbol{d}}_{k-1}$, we have

$$\widehat{\boldsymbol{d}}_k = \widehat{\boldsymbol{P}}_k^\top \widehat{\boldsymbol{d}}_{k-1} = (1 - \lambda_k)\widehat{\boldsymbol{P}}_{k-1}^\top \widehat{\boldsymbol{d}}_{k-1} + \lambda_k \boldsymbol{F}_k^\top \widehat{\boldsymbol{d}}_{k-1}.$$

Now, using the approximation $\widehat{\boldsymbol{d}}_{k-1} \approx \widehat{\boldsymbol{P}}_{k-1}^\top \widehat{\boldsymbol{d}}_{k-1}$ (i.e., assuming $\widehat{\boldsymbol{d}}_{k-1}$ is the stationary distribution of $\widehat{\boldsymbol{P}}_{k-1}$ which

Algorithm 2 Rank-One Q-Learning (R1-QL)

Input: samples from transition kernel $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$; cost function $\boldsymbol{c} \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$; discount factor $\gamma \in (0,1)$;
Output: optimal Q-function $q^\star$

initialize: $\boldsymbol{q}_0 \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$, $\widehat{\boldsymbol{d}}_{-1} \in \Delta(\mathcal{S}\times\mathcal{A})$;
for $k = 0, 1, 2, \ldots$ **do**
  $\lambda_k = 1/(k+1)$;
  $\boldsymbol{f} = \mathbf{0} \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$;
  for $(s,a) \in \mathcal{S}\times\mathcal{A}$ **do**
    $\widehat{s}_k^{\,+} \sim P(\cdot|s,a)$;
    $$\begin{cases} \widehat{a}_k^{\,+} \in \operatorname*{argmin}_{a^+\in\mathcal{A}} \boldsymbol{q}_k(\widehat{s}_k^{\,+}, a^+), \\ \big[\widehat{\mathbf{T}}_k(\boldsymbol{q}_k)\big](s,a) = \boldsymbol{c}(s,a) + \gamma \min_{a^+\in\mathcal{A}} \boldsymbol{q}_k(\widehat{s}_k^{\,+}, a^+); \end{cases}$$
    $\boldsymbol{f}(s,a) = (1-\lambda_k)\widehat{\boldsymbol{d}}_{k-1}(s,a) + \lambda_k \widehat{\boldsymbol{d}}_{k-1}(s^+, a^+)$;
  end for
  $\widehat{\boldsymbol{d}}_k = \boldsymbol{f}/\|\boldsymbol{f}\|_1$;
  $\alpha_k = \frac{\gamma\lambda_k}{1-\gamma}\langle \widehat{\boldsymbol{d}}_k, \widehat{\mathbf{T}}_k(\boldsymbol{q}_k) - \boldsymbol{q}_k\rangle$;
  $\boldsymbol{q}_{k+1} = (1-\lambda_k)\boldsymbol{q}_k + \lambda_k \widehat{\mathbf{T}}_k(\boldsymbol{q}_k) + \alpha_k \mathbf{1}$;
end for

does not hold exactly since $\widehat{\boldsymbol{d}}_{k-1}$ is only an approximation of the stationary distribution of $\widehat{\boldsymbol{P}}_{k-1}$), we derive,

$$\widehat{\boldsymbol{d}}_k = (1-\lambda_k)\widehat{\boldsymbol{d}}_{k-1} + \lambda_k \boldsymbol{F}_k^\top \widehat{\boldsymbol{d}}_{k-1}$$

or, equivalently, for each $(s,a) \in \mathcal{S}\times\mathcal{A}$

$$\widehat{\boldsymbol{d}}_k(s,a) = (1-\lambda_k)\widehat{\boldsymbol{d}}_{k-1}(s,a) + \lambda_k \widehat{\boldsymbol{d}}_{k-1}(s^+, a^+), \quad (18)$$

where s^+ and a^+ are given in (17). Observe that the approximation $\widehat{\boldsymbol{d}}_{k-1} \approx \widehat{\boldsymbol{P}}_{k-1}^\top \widehat{\boldsymbol{d}}_{k-1}$ significantly reduces the memory and time complexity of the algorithm since we do *not* need to keep track of the estimates $\widehat{\boldsymbol{P}}_k$ of the state-action transition probability matrix and perform full matrix-vector multiplications for updating the estimates $\widehat{\boldsymbol{d}}_k$ of the stationary distribution.

The complete description of the proposed R1-QL algorithm is provided in Algorithm 2. We again note that the vector $\boldsymbol{f}$ and its normalization are only introduced to avoid the accumulation of numerical errors. The following result discusses the convergence of the proposed algorithm.

Theorem 4.2 (Convergence of R1-QL)**.** *The iterates $\boldsymbol{q}_k$ of the R1-QL Algorithm 2 converge to the optimal Q-function $\boldsymbol{q}^\star = \overline{\mathbf{T}}(\boldsymbol{q}^\star)$ almost surely with at least the same rate as QL.*

Finally, we note that the per-iteration time complexity of the R1-QL Algorithm 2 is the same as that of the synchronous QL algorithm, i.e., $\mathcal{O}(nm^2)$.

5. Numerical simulations

In this section, we compare the performance of several planning and learning algorithms with our proposed methods. The experiments are conducted on Garnet (Archibald et al., 1995) and Graph MDPs (Devraj & Meyn, 2017), focusing on the Bellman errors $\|\mathbf{T}(\boldsymbol{v}_k) - \boldsymbol{v}_k\|_\infty$ and $\big\|\overline{\mathbf{T}}(\boldsymbol{q}_k) - \boldsymbol{q}_k\big\|_\infty$ and the value errors $\|\boldsymbol{v}_k - \boldsymbol{v}^\star\|_\infty$ and $\|\boldsymbol{q}_k - \boldsymbol{q}^\star\|_\infty$. In all of our experiments, we run policy iteration (PI) until it converges to calculate the optimal values $\boldsymbol{v}^\star$ and $\boldsymbol{q}^\star$ for reference. Garnet and Graph MDPs are particularly compelling for our empirical analysis as we can run PI to compute the optimal values, enabling us to measure value errors throughout the iterations. The Garnet MDPs have a state size of $n = 200$, an action size of $m = 5$, and randomly generated transition probabilities and costs with a branching factor of 10. For our numerical experiments, we consider 25 randomly generated instances of Garnet MDPs and report the three quantiles of the errors. For the Graph MDPs, we use the same configuration as described in (Devraj & Meyn, 2017), providing a complementary benchmark to validate the effectiveness of our proposed method. We also note that the supplementary material includes the Python package implementing the algorithms and the experiments discussed in this section. In what follows, we report the result of our simulations for the planning and learning problems.

Planning Algorithms. We compare several value iteration (VI) algorithms with the same per-iteration time complexity as our proposed R1-VI Algorithm 1. We also include PI for reference. We mainly focus on comparing R1-VI with accelerated VI methods, namely, Nesterov-VI (Goyal & Grand-Clément, 2022) and Anderson-VI (Geist & Scherrer, 2018). In order to keep the time complexity the same, we use Anderson-VI with the memory parameter equal to one leading to a rank-one approximation of the Hessian matrix. The update rule of the accelerated VI algorithms is provided in Appendix B.1.

In Figure 1, we report the median number of iterations required to reach a certain error threshold for each algorithm across both MDPs for four different values of the discount factors γ. Our results indicate that the convergence performance of R1-VI is comparable with PI while maintaining the same per-iteration time complexity as VI. Additionally, R1-VI significantly outperforms the VI algorithm and its accelerated versions, particularly when the discount factor is close to 1. This observation can be partially explained by the fact that the proposed R1-VI algorithm forms an approximation of the inverse of the "Hessian", i.e., $(\boldsymbol{I} - \gamma \boldsymbol{P}_k)^{-1}$, by incorporating its largest eigenvalue $\frac{1}{1-\gamma}$. A more detailed comparison of the algorithms is provided in Appendix B.2.

Learning Algorithms. In the learning experiments, we report the Bellman and value errors of the algorithms trained in a synchronous fashion, following the methodology outlined in (Ghavamzadeh et al., 2011), ensuring a consistent evaluation. In synchronous learning, in each iteration k, a sample $\widehat{s}^{\,+} \sim P(\cdot|s,a)$ of the next state is generated for each state-

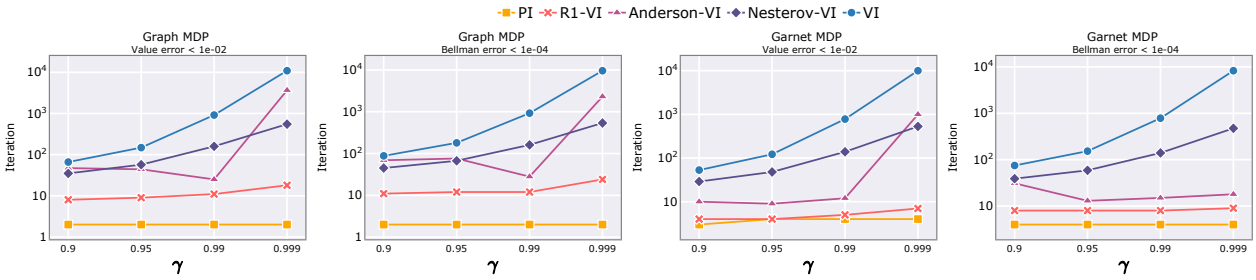

Figure 1: Planning algorithms – the median number of iterations required for each algorithm to reach a fixed error threshold across four discount factors γ for the two MDPs.

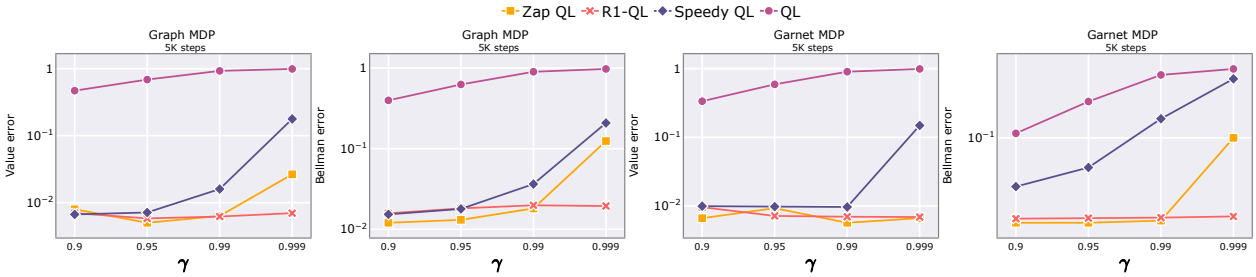

Figure 2: Learning algorithms – the median error values achieved by each learning algorithm over the course of 5000 iterations across four discount factors γ for the two MDPs.

action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ and the action-value function $\boldsymbol{q}_k$ is updated in all state-action pairs; see the update rule in QL algorithm (5) and R1-VI Algorithm 2. All the learning algorithms use the same samples generated through the training. Besides the proposed R1-QL Algorithm 2, we report the performance of Speedy QL (Ghavamzadeh et al., 2011), Zap QL (Devraj & Meyn, 2017), and the standard QL (5) in Garnet and Graph MDPs for several discount factors; see Appendix B.1 for the update rule of Speedy QL and Zap QL. We run each algorithm using the same step-size schedule $(\lambda_k)_{k=0}^{\infty}$, namely, linearly decaying $\lambda_k = 1/(1+k)$, to ensure a fair comparison.

Figure 2 shows the final error values after running each algorithm for 5000 iterations. R1-QL achieves comparable or lower error values across both MDPs. In contrast to the other algorithms, R1-QL consistently maintains a similar level of error values across various discount factors, particularly at higher discount factors – a characteristic attributed to Policy Iteration (PI) algorithms. It is worth mentioning that since both Zap QL and R1-QL estimate $(\boldsymbol{I} - \gamma \overline{\boldsymbol{P}}_k)^{-1}$ using the samples, they behave more robustly against the increase in the discount factor γ; see Figure 2. Regarding per iteration complexity, R1-QL has the same time and memory complexity as QL and Speedy QL. In contrast, Zap QL, due to the inherent full-rank matrix inversion, incurs higher time and memory complexity. Albeit Zap QL can be efficiently implemented with lower complexity, it typically exhibits a higher computational cost in practice. A comprehensive analysis of the error trajectories observed during training is provided in Appendix B.2.

6. Limitations and future research directions

We finish the paper by discussing some limitations of the proposed rank-one modification of the VI and QL algorithms along with some future research directions.

Let us start by noting that the provided theoretical results in Theorems 3.3 and 4.2 guarantee the convergence of the proposed algorithms with the same rate as the standard VI and QL algorithms. However, our numerical experiments with Garnet and Graph MDPs in Section 5 show that the proposed algorithms have a faster convergence rate compared to standard VI and QL and their accelerated versions. This gap can be explained by the fact that our proof technique does not exploit that the vectors $\boldsymbol{d}_k$ and $\widehat{\boldsymbol{d}}_k$ used in the update rule are specifically constructed to approximate the stationary distribution of the Markov chain induced by the greedy policy (see Appendices A.3 and A.4 for details). That is, the convergence of R1-VI and R1-QL algorithms is guaranteed for any choice of $\boldsymbol{d}_k \in \Delta(\mathcal{S})$ and $\widehat{\boldsymbol{d}}_k \in \Delta(\mathcal{S} \times \mathcal{A})$. In fact, when the stationary distribution concentrates on a single state, as happens when there is an absorbing state with zero reward, the second term in the R1VI update (Equation (12)) vanishes. Appendix B.3 provides an empirical analysis in

Gridworld (Sutton & Barto, 2018), which includes an absorbing state and thus violates the assumption of Lemma 3.2. Moreover, the provided proof of convergence shows that the greedy policies generated with respect to the iterates of R1-VI and R1-QL are the same as those for the standard VI and QL algorithms, respectively (see Lemmas A.2 and A.4). In other words, the proposed algorithms do not affect the speed of convergence to the optimal policy compared to VI and QL. Nevertheless, at least in the case of R1-VI, the faster convergence in the value spaces leads to a faster termination of the algorithm for a given performance bound for the greedy policy. In this regard, let us also note that the mismatch between convergence in value space and policy space also arises in other "accelerated" VI/QL algorithms; see Appendix B.4.

Second, the proposed algorithms heavily depend on the structure of the transition probability matrices $\boldsymbol{P}_k$ and $\overline{\boldsymbol{P}}_k$ and their rank-one approximation using the corresponding stationary distributions. This dependence particularly hinders the application of the proposed algorithms to generic function approximation setups in solving the optimal control problem of MDPs with continuous state-action spaces. We note that a similar issue for the Zap Q-leaning algorithm (Devraj & Meyn, 2017) has been successfully addressed in (Chen et al., 2020).

Third, we note that the proposed R1-QL algorithm 2 is a *synchronous* algorithm that updates *all* state-action pairs $(s,a) \in \mathcal{S} \times \mathcal{A}$ of the Q-function $\boldsymbol{q}_k$ at each iteration k. This algorithm can be modified in a standard fashion for the *asynchronous* case. However, the provided convergence analysis can not be extended for the corresponding asynchronous algorithm in a straightforward manner. Moreover, the straightforward asynchronous implementation of R1-QL leads to an $\mathcal{O}(nm)$ per-iteration complexity for updating a *single* entry $(s,a) \in \mathcal{S} \times \mathcal{A}$ at each iteration k, which is higher than the $\mathcal{O}(m)$ per-iteration complexity of the standard asynchronous QL algorithm. We note that the Zap Q-leaning algorithm (Devraj & Meyn, 2017) also suffers from this issue. Addressing these issues requires a more involved analysis and modification of the proposed algorithm in the asynchronous case, which we leave for future research.

Finally, the basic idea of the proposed algorithms can also be used for developing the rank-one modified version of the existing algorithms for the *average* cost setting. For example, consider the PI algorithm that uses the *relative VI* algorithm in the policy evaluation step for unichains (Puterman, 2014, Sec. 8.6.1). This algorithm can be characterized via the following update rule in the value space: For a fixed $s \in \mathcal{S}$,

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + \big((I - \boldsymbol{P}_k)(\boldsymbol{I} - e_s e_s^\top) + \mathbf{1} e_s^\top\big)^{-1} (\mathbf{T}(\boldsymbol{v}_k) - \boldsymbol{v}_k),$$
$$\boldsymbol{v}_{k+1}(s) = 0,$$

where e_s is the s-th unit vector and $\mathbf{T}$ is now the *undiscounted* Bellman operator. Now, observe that

$$\begin{aligned}\boldsymbol{G}_k &= \big((\boldsymbol{I} - \boldsymbol{P}_k)(\boldsymbol{I} - e_s e_s^\top) + \mathbf{1} e_s^\top\big)^{-1} \\ &= \big(\boldsymbol{I} - \boldsymbol{P}_k + (\boldsymbol{p}_k - e_s + \mathbf{1}) e_s^\top\big)^{-1} \\ &= \big(\boldsymbol{I} - \mathbf{1}\boldsymbol{d}_k^\top + (\boldsymbol{p}_k - e_s + \mathbf{1}) e_s^\top\big)^{-1},\end{aligned}$$

where $\boldsymbol{p}_k = \boldsymbol{P}_k(\cdot, s)$ is the s-th column of $\boldsymbol{P}_k$ and we used the approximation $\boldsymbol{P}_k \approx \mathbf{1}\boldsymbol{d}_k^\top$ in the last equality. The matrix inversion can then be handled efficiently using the Woodbury formula. However, the convergence of this algorithm and any possible improvement in the convergence rate when $\boldsymbol{d}_k$ is approximated via the power method requires further investigation.

Acknowledgements

The authors would like to thank the reviewers for their useful comments. This work was supported by the Horizon Europe Pathfinder Open project RELIEVE-101099481 and by the European Research Council (ERC) project TRUST-949796.

References

Anderson, D. G. Iterative Procedures for Nonlinear Integral Equations. *Journal of the ACM (JACM)*, 12(4):547–560, 1965.

Archibald, T., McKinnon, K., and Thomas, L. On the Generation of Markov Decision Processes. *Journal of the Operational Research Society*, 46(3):354–361, 1995.

Bertsekas, D. *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control*. Athena Scientific, 2022.

Bertsekas, D. *A Course in Reinforcement Learning*. Athena Scientific, 2023.

Chen, S., Devraj, A. M., Lu, F., Bušić, A., and Meyn, S. Zap Q-Learning with Nonlinear Function Approximation. In *Advances in Neural Information Processing Systems*, volume 33, pp. 16879–16890, 2020.

Devraj, A. M. and Meyn, S. Zap Q-Learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Devraj, A. M., Bušić, A., and Meyn, S. On Matrix Momentum Stochastic Approximation and applications to Q-Learning. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 749–756, 2019.

Even-Dar, E. and Mansour, Y. Learning Rates for Q-Learning. *Journal of Machine Learning Research*, 5 (1):1–25, 2003.

Gallager, R. G. *Discrete Stochastic Processes*. 2011.

Gargiani, M., Zanelli, A., Liao-McPherson, D., Summers, T., and Lygeros, J. Dynamic Programming Through the Lens of Semismooth Newton-Type Methods. *IEEE Control Systems Letters*, 6:2996–3001, 2022.

Geist, M. and Scherrer, B. Anderson Acceleration for Reinforcement Learning. *preprint arXiv:1809.09501*, 2018.

Ghavamzadeh, M., Kappen, H., Azar, M., and Munos, R. Speedy Q-Learning. In *Advances in Neural Information Processing Systems*, volume 24, 2011.

Golub, G. H. and Van Loan, C. F. *Matrix Computations*. JHU Press, 2013.

Goyal, V. and Grand-Clément, J. A First-Order Approach to Accelerated Value Iteration. *Operations Research*, 71 (2):517–535, 2022.

Grand-Clément, J. From Convex Optimization to MDPs: A Review of First-Order, Second-Order and Quasi-Newton Methods for MDPs. *preprint arXiv:2104.10677*, 2021.

Hager, W. W. Updating the Inverse of a Matrix. *SIAM Review*, 31(2):221–239, 1989.

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

Jaakkola, T., Jordan, M., and Singh, S. Convergence of Stochastic Iterative Dynamic Programming Algorithms. In *Advances in neural information processing systems*, volume 6, 1993.

Kamanchi, C., Diddigi, R. B., and Bhatnagar, S. Generalized Second-Order Value Iteration in Markov Decision Processes. *IEEE Transactions on Automatic Control*, 67 (8):4241–4247, 2022.

Kearns, M. and Singh, S. Finite-Sample Convergence Rates for Q-Learning and Indirect Algorithms. In *Advances in neural information processing systems*, volume 11, 1998.

Kolarijani, M. A. S. and Mohajerin Esfahani, P. From Optimization to Control: Quasi Policy Iteration. *arXiv preprint arXiv:2311.11166*, 2023.

Kushner, H. and Kleinman, A. Accelerated Procedures for the Solution of Discrete Markov Control Problems. *IEEE Transactions on Automatic Control*, 16(2):147–152, 1971.

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

Lee, J., Rakhsha, A., Ryu, E. K., and Farahmand, A.-m. Deflated dynamics value iteration. *arXiv preprint arXiv:2407.10454*, 2024.

Nesterov, Y. E. A Method for Solving the Convex Programming Problem with Convergence Rate $O(1/k^2)$. *Doklady Akademii Nauk SSSR*, 269(3):543–547, 1983.

Porteus, E. L. and Totten, J. C. Accelerated Computation of the Expected Discounted Return in a Markov Chain. *Operations Research*, 26(2):350–358, 1978.

Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

Puterman, M. L. and Brumelle, S. L. On the Convergence of Policy Iteration in Stationary Dynamic Programming. *Mathematics of Operations Research*, 4(1):60–69, 1979.

Rakhsha, A., Wang, A., Ghavamzadeh, M., and Farahmand, A.-m. Operator splitting value iteration. *Advances in Neural Information Processing Systems*, 35:38373–38385, 2022.

Ruppert, D. A Newton-Raphson Version of the multivariate Robbins-Monro Procedure. *The Annals of Statistics*, 13 (1):236–245, 1985.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT press, 2018.

Szepesvári, C. *Algorithms for Reinforcement Learning*. 2009.

Tsitsiklis, J. N. Asynchronous Stochastic Approximation and Q-Learning. *Machine learning*, 16:185–202, 1994.

Wainwright, M. J. Stochastic Approximation with Cone-Contractive Operators: Sharp ℓ_∞-Bounds for Q-Learning. *arXiv preprint arXiv:1905.06265*, 2019.

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

Weng, B., Xiong, H., Zhao, L., Liang, Y., and Zhang, W. Finite-Time Theory for Momentum Q-Learning. In *Uncertainty in Artificial Intelligence (UAI)*, pp. 665–674, 2021.

Zhang, J., O'Donoghue, B., and Boyd, S. Globally Convergent Type-I Anderson Acceleration for Nonsmooth Fixed-Point Iterations. *SIAM Journal on Optimization*, 30(4):3170–3197, 2020.

A. Technical Proofs

A.1. Proof of Lemma 3.1

We begin by providing two basic results on MDPs. First, recall that given a policy π, the value $\boldsymbol{v}^\pi$ of π solves the (*Bellman consistency*) equation (Puterman, 2014, Thm. 6.1.1)

$$\boldsymbol{v}^\pi(s) = \boldsymbol{c}^\pi(s) + \gamma\, \mathbb{E}_{s^+\sim \boldsymbol{P}^\pi(s,\cdot)}\left[\boldsymbol{v}^\pi(s^+)\right], \quad \forall s \in \mathcal{S}.$$

Hence, we have

$$\boldsymbol{v}^\pi = (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-1}\boldsymbol{c}^\pi. \tag{19}$$

Moreover, the definitions of the Bellman operator in (3) and of the greedy policy in (6) imlpy that

$$\mathbf{T}(\boldsymbol{v}) = \boldsymbol{c}^{\pi^{\boldsymbol{v}}} + \gamma \boldsymbol{P}^{\pi^{\boldsymbol{v}}}\boldsymbol{v}, \quad \forall \boldsymbol{v} \in \mathbb{R}^n. \tag{20}$$

Recall $\boldsymbol{P}_k \coloneqq \boldsymbol{P}^{\pi^{\boldsymbol{v}_k}}$. Using these two results, we have at each iteration of the PI algorithm (7),

$$\begin{aligned}
\boldsymbol{v}_{k+1} = \boldsymbol{v}^{\pi_{k+1}} = \boldsymbol{v}^{\pi^{\boldsymbol{v}_k}} &\overset{(19)}{=} (\boldsymbol{I} - \gamma \boldsymbol{P}_k)^{-1}\boldsymbol{c}^{\pi^{\boldsymbol{v}_k}} \overset{(20)}{=} (\boldsymbol{I} - \gamma \boldsymbol{P}_k)^{-1}\big(\mathbf{T}(\boldsymbol{v}_k) - \gamma \boldsymbol{P}_k \boldsymbol{v}_k\big) \\
&= (\boldsymbol{I} - \gamma \boldsymbol{P}_k)^{-1}\big(\mathbf{T}(\boldsymbol{v}_k) - \boldsymbol{v}_k + (\boldsymbol{I} - \gamma \boldsymbol{P}_k)\boldsymbol{v}_k\big) \\
&= \boldsymbol{v}_k + (\boldsymbol{I} - \gamma \boldsymbol{P}_k)^{-1}\big(\mathbf{T}(\boldsymbol{v}_k) - \boldsymbol{v}_k\big).
\end{aligned}$$

This concludes the proof.

A.2. Proof of Lemma 3.2

Let $(\rho_i)_{i=1}^n$ be the eigenvalues of $\boldsymbol{P}_k$ such that $|\rho_1| \geq |\rho_2| \geq \cdots \geq |\rho_n|$. Since $\boldsymbol{P}_k$ is a row stochastic matrix, we have $\rho_1 = 1$ (Gallager, 2011, Thm. 3.4.1). Moreover, the assumption that $\boldsymbol{P}_k$ is irreducible and aperiodic implies that $|\rho_i| < 1$ for all $i \neq 1$ (Gallager, 2011, Thm. 3.4.1), that is, $\rho_1 = 1$ is the unique eigenvalue of $\boldsymbol{P}_k$ on the unit circle in the complex plane and all other eigenvalues lie inside the unit disc. The unique (up to scaling) right and left eigenvectors corresponding to $\rho_1 = 1$ are the all-one vector $\mathbf{1}$ and the stationary distribution $\boldsymbol{d}_k$. From these results it follows that $\widetilde{\boldsymbol{P}}_k = \mathbf{1}\boldsymbol{d}_k^\top$ is the unique solution of (11).

A.3. Proof of Theorem 3.3

For each iteration $k \geq 0$ of the R1-VI Algorithm 1, we have

$$\begin{cases}
\boldsymbol{v}_{k+1} = \mathbf{T}(\boldsymbol{v}_k) + \alpha_k \mathbf{1} \\
\alpha_k = \dfrac{\gamma}{1-\gamma}\big\langle \boldsymbol{d}_k, \mathbf{T}(\boldsymbol{v}_k) - \boldsymbol{v}_k \big\rangle,
\end{cases} \tag{21}$$

where $\boldsymbol{d}_k \in \Delta(\mathcal{S})$ is an approximation of the stationary distribution of the MDP under the greedy policy with respect to $\boldsymbol{v}_k$. We start with analyzing the effect of a constant shift in the argument of the Bellman operator.

Lemma A.1. *For all $\alpha \in \mathbb{R}$ and $\boldsymbol{v} \in \mathbb{R}^n$, we have*

$$\mathbf{T}(\boldsymbol{v} + \alpha \mathbf{1}) = \mathbf{T}(\boldsymbol{v}) + \gamma\alpha \mathbf{1}.$$

Proof. For each $s \in \mathcal{S}$, we have

$$\begin{aligned}
[\mathbf{T}(\boldsymbol{v} + \alpha\mathbf{1})](s) &= \min_{a\in\mathcal{A}}\left\{\boldsymbol{c}(s,a) + \gamma \sum_{s^+\in\mathcal{S}} P(s^+|s,a)\left[\boldsymbol{v} + \alpha\mathbf{1}\right](s^+)\right\} \\
&= \gamma\alpha + \min_{a\in\mathcal{A}}\left\{\boldsymbol{c}(s,a) + \gamma \sum_{s^+\in\mathcal{S}} P(s^+|s,a)\boldsymbol{v}(s^+)\right\} \\
&= \gamma\alpha + [\mathbf{T}(\boldsymbol{v})](s).
\end{aligned}$$

□

We next use the preceding result to provide an alternative characterization of the iterates of R1-VI in (21).

Lemma A.2. *For each $k \geq 0$, the iterates in (21) are equivalently given by*

$$\begin{cases} \boldsymbol{v}_{k+1} = \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) + \beta_{k+1}\mathbf{1} \\ \beta_{k+1} = \gamma\beta_k + \alpha_k, \quad \text{with } \beta_0 = 0. \end{cases}$$

Proof. (Proof by induction.) Consider $k = 0$ and observe that

$$\boldsymbol{v}_1 = \mathbf{T}(\boldsymbol{v}_0) + \beta_1\mathbf{1},$$

with $\beta_1 = \gamma\beta_0 + \alpha_0 = \alpha_0$. Next, for some $k \geq 1$ and $\beta_k \in \mathbb{R}$, assume $\boldsymbol{v}_k = \mathbf{T}^{(k)}(\boldsymbol{v}_0) + \beta_k\mathbf{1}$. Then,

$$\boldsymbol{v}_{k+1} = \mathbf{T}(\boldsymbol{v}_k) + \alpha_k\mathbf{1} = \mathbf{T}\big(\mathbf{T}^{(k)}(\boldsymbol{v}_0) + \beta_k\mathbf{1}\big) + \alpha_k\mathbf{1} = \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) + \gamma\beta_k\mathbf{1} + \alpha_k\mathbf{1},$$

where the last equality follows from Lemma A.1. Therefore,

$$\boldsymbol{v}_{k+1} = \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) + (\gamma\beta_k + \alpha_k)\mathbf{1} = \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) + \beta_{k+1}\mathbf{1},$$

which concludes the proof. □

We now employ the preceding lemmas to provide an alternative characterization of the constant shifts α_k in the R1-VI updates in (21).

Lemma A.3. *For each $k \geq 0$, one has*

$$\alpha_k = \frac{\gamma}{1-\gamma}\left\langle \boldsymbol{d}_k, \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) - \mathbf{T}^{(k)}(\boldsymbol{v}_0)\right\rangle - \gamma\beta_k.$$

Proof. From Lemma A.2, we have

$$\boldsymbol{v}_k = \mathbf{T}^{(k)}(\boldsymbol{v}_0) + \beta_k\mathbf{1}, \quad k = 0, 1, \ldots.$$

(Notice that for $k = 0$, the preceding equation simply implies $\boldsymbol{v}_0 = \boldsymbol{v}_0$ since $\mathbf{T}^{(0)}$ is the identity operator and $\beta_0 = 0$.) Then, from Lemma A.1, it follows that

$$\mathbf{T}(\boldsymbol{v}_k) = \mathbf{T}\big(\mathbf{T}^{(k)}(\boldsymbol{v}_0) + \beta_k\mathbf{1}\big) = \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) + \gamma\beta_k\mathbf{1}.$$

Thus,

$$\mathbf{T}(\boldsymbol{v}_k) - \boldsymbol{v}_k = \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) - \mathbf{T}^{(k)}(\boldsymbol{v}_0) - (1-\gamma)\beta_k\mathbf{1}$$

and

$$\begin{aligned} \big\langle \boldsymbol{d}_k, \mathbf{T}(\boldsymbol{v}_k) - \boldsymbol{v}_k\big\rangle &= \big\langle \boldsymbol{d}_k, \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) - \mathbf{T}^{(k)}(\boldsymbol{v}_0)\big\rangle - (1-\gamma)\beta_k\langle \boldsymbol{d}_k, \mathbf{1}\rangle \\ &= \big\langle \boldsymbol{d}_k, \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) - \mathbf{T}^{(k)}(\boldsymbol{v}_0)\big\rangle - (1-\gamma)\beta_k, \end{aligned}$$

where we used $\boldsymbol{d}_k \in \Delta(\mathcal{S})$ (i.e., $\langle \boldsymbol{d}_k, \mathbf{1}\rangle = 1$) in the second equality above. Therefore, for α_k in (21), we have

$$\alpha_k = \frac{\gamma}{1-\gamma}\left\langle \boldsymbol{d}_k, \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) - \mathbf{T}^{(k)}(\boldsymbol{v}_0)\right\rangle - \gamma\beta_k.$$

This completes the proof. □

Now, observe that plugging in α_k from Lemma A.3 in the update rule of Lemma A.2 leads to

$$\boldsymbol{v}_{k+1} = \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) + \frac{\gamma}{1-\gamma}\big\langle \boldsymbol{d}_k, \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) - \mathbf{T}^{(k)}(\boldsymbol{v}_0)\big\rangle\mathbf{1}, \quad k = 0, 1, \ldots. \tag{22}$$

Then, using the fact that $\boldsymbol{v}^\star = \mathbf{T}(\boldsymbol{v}^\star)$ and the Bellman operator is a γ-contraction in the ∞-norm, we have

$$\begin{aligned}
\|\boldsymbol{v}_{k+1} - \boldsymbol{v}^\star\|_\infty &= \left\| \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) - \mathbf{T}(\boldsymbol{v}^\star) + \frac{\gamma}{1-\gamma} \langle \boldsymbol{d}_k, \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) - \mathbf{T}^{(k)}(\boldsymbol{v}_0) \rangle \mathbf{1} \right\|_\infty \\
&\leq \left\| \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) - \mathbf{T}(\boldsymbol{v}^\star) \right\|_\infty + \frac{\gamma}{1-\gamma} \left| \langle \boldsymbol{d}_k, \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) - \mathbf{T}^{(k)}(\boldsymbol{v}_0) \rangle \right| \\
&\leq \gamma^{k+1} \|\boldsymbol{v}_0 - \boldsymbol{v}^\star\|_\infty + \frac{\gamma}{1-\gamma} \left\| \mathbf{T}^{(k+1)}(\boldsymbol{v}_0) - \mathbf{T}^{(k)}(\boldsymbol{v}_0) \right\|_\infty \\
&\leq \gamma^{k+1} \|\boldsymbol{v}_0 - \boldsymbol{v}^\star\|_\infty + \frac{\gamma^{k+1}}{1-\gamma} \|\mathbf{T}(\boldsymbol{v}_0) - \boldsymbol{v}_0\|_\infty \\
&\leq \gamma^{k+1} \big(\|\boldsymbol{v}_0 - \boldsymbol{v}^\star\|_\infty + \frac{1}{1-\gamma} \|\mathbf{T}(\boldsymbol{v}_0) - \boldsymbol{v}_0\|_\infty \big).
\end{aligned}$$

That is, $\boldsymbol{v}_k \to \boldsymbol{v}^\star$ as $k \to \infty$ linearly with rate γ.

A.4. Proof of Theorem 4.2

Let us begin with recalling the definition of the empirical Bellman operator

$$\big[\widehat{\mathbf{T}}_k(\boldsymbol{q})\big](s,a) := \boldsymbol{c}(s,a) + \gamma \min_{a^+ \in \mathcal{A}} \boldsymbol{q}(\widehat{s}_k^+, a^+), \tag{23}$$

where $\widehat{s}_k^+ \sim P(\cdot|s,a)$, that is to say $\widehat{s}^+$ is sampled according to the law $P(\cdot|s,a)$ at iteration k. From this definition, it immediately follows that

$$\widehat{\mathbf{T}}_k(\boldsymbol{q} + \alpha \mathbf{1}) = \widehat{\mathbf{T}}_k(\boldsymbol{q}) + \gamma\alpha \mathbf{1}, \quad \forall\, \alpha \in \mathbb{R},\ \boldsymbol{q} \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}. \tag{24}$$

Also, recall that each iteration $k \geq 0$ of the R1-QL Algorithm 2 reads as

$$\begin{cases}
\boldsymbol{q}_{k+1} = (1-\lambda_k)\boldsymbol{q}_k + \lambda_k \widehat{\mathbf{T}}_k(\boldsymbol{q}_k) + \alpha_k \mathbf{1} \\
\alpha_k = \lambda_k \Big(\dfrac{\gamma}{1-\gamma} \Big) \langle \widehat{\boldsymbol{d}}_k, \widehat{\mathbf{T}}_k(\boldsymbol{q}_k) - \boldsymbol{q}_k \rangle,
\end{cases} \tag{25}$$

where $\widehat{\boldsymbol{d}}_k \in \Delta(\mathcal{S}\times\mathcal{A})$ is an estimation of the stationary distribution of the state-action transition probability matrix of the MDP under the greedy policy with respect to $\boldsymbol{q}_k$. Let us also consider the standard QL iterates

$$\boldsymbol{q}_{k+1}^{\mathrm{QL}} = (1-\lambda_k)\boldsymbol{q}_k^{\mathrm{QL}} + \lambda_k \widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}), \quad k = 0, 1, \ldots,$$

with the same initialization $\boldsymbol{q}_0^{\mathrm{QL}} = \boldsymbol{q}_0$ and empirical Bellman operator $\widehat{\mathbf{T}}_k(\cdot)$ for all k as the R1-QL algorithm (25).

The first result concerns an alternative characterization of the iterates in (25).

Lemma A.4. *For each $k \geq 0$, the iterates of R1-QL algorithm (25) equivalently read as*

$$\begin{cases}
\boldsymbol{q}_{k+1} = \boldsymbol{q}_{k+1}^{\mathrm{QL}} + \beta_{k+1}\mathbf{1} \\
\beta_{k+1} = (1-\lambda_k)\beta_k + \gamma\lambda_k\beta_k + \alpha_k, \quad \text{with } \beta_0 = 0.
\end{cases} \tag{26}$$

Proof. (Proof by induction) For $k = 0$, since $\boldsymbol{q}_0^{\mathrm{QL}} = \boldsymbol{q}_0$, we can write

$$\begin{aligned}
\boldsymbol{q}_1 &= (1-\lambda_0)\boldsymbol{q}_0 + \lambda_0 \widehat{\mathbf{T}}_k(\boldsymbol{q}_0) + \alpha_0 \mathbf{1} \\
&= (1-\lambda_0)\boldsymbol{q}_0^{\mathrm{QL}} + \lambda_0 \widehat{\mathbf{T}}_k(\boldsymbol{q}_0^{\mathrm{QL}}) + \alpha_0 \mathbf{1} \\
&= \boldsymbol{q}_1^{\mathrm{QL}} + \beta_1 \mathbf{1},
\end{aligned}$$

where $\beta_1 = \alpha_0$. Assume next $\boldsymbol{q}_k = \boldsymbol{q}_k^{\mathrm{QL}} + \beta_k \mathbf{1}$ for some $k \geq 0$. Then, it follows that

$$\begin{aligned}
\boldsymbol{q}_{k+1} &= (1-\lambda_k)\boldsymbol{q}_k + \lambda_k \widehat{\mathbf{T}}_k(\boldsymbol{q}_k) + \alpha_k \mathbf{1} \\
&= (1-\lambda_k)(\boldsymbol{q}_k^{\mathrm{QL}} + \beta_k \mathbf{1}) + \lambda_k \widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}} + \beta_k \mathbf{1}) + \alpha_k \mathbf{1} \\
&\overset{(24)}{=} (1-\lambda_k)(\boldsymbol{q}_k^{\mathrm{QL}} + \beta_k \mathbf{1}) + \lambda_k \big(\widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) + \gamma \beta_k \mathbf{1}\big) + \alpha_k \mathbf{1} \\
&= (1-\lambda_k)\boldsymbol{q}_k^{\mathrm{QL}} + \lambda_k \widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) + \big((1-\lambda_k)\beta_k + \gamma \lambda_k \beta_k + \alpha_k\big)\mathbf{1} \\
&= \boldsymbol{q}_{k+1}^{\mathrm{QL}} + \beta_{k+1}\mathbf{1}.
\end{aligned}$$

This concludes the proof. □

We next provide a useful characterization of the constant shifts α_k in the R1-QL update rule (25).

Lemma A.5. *For each $k \geq 0$, one has*

$$\alpha_k = \lambda_k \Big(\frac{\gamma}{1-\gamma}\Big) \big\langle \widehat{\boldsymbol{d}}_k, \widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) - \boldsymbol{q}_k^{\mathrm{QL}} \big\rangle - \gamma \lambda_k \beta_k.$$

Proof. From Lemma A.4 and since $\boldsymbol{q}_0 = \boldsymbol{q}_0^{\mathrm{QL}}$ and $\beta_0 = 0$, we have

$$\boldsymbol{q}_k = \boldsymbol{q}_k^{\mathrm{QL}} + \beta_k \mathbf{1}, \quad k = 0, 1, \ldots.$$

Hence, we can use (24) to write

$$\widehat{\mathbf{T}}_k(\boldsymbol{q}_k) = \widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}} + \beta_k \mathbf{1}) = \widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) + \gamma \beta_k \mathbf{1}.$$

As a result,

$$\begin{aligned}
\widehat{\mathbf{T}}_k(\boldsymbol{q}_k) - \boldsymbol{q}_k &= \widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) + \gamma \beta_k \mathbf{1} - \boldsymbol{q}_k^{\mathrm{QL}} - \beta_k \mathbf{1} \\
&= \widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) - \boldsymbol{q}_k^{\mathrm{QL}} - (1-\gamma)\beta_k \mathbf{1},
\end{aligned}$$

and

$$\begin{aligned}
\big\langle \widehat{\boldsymbol{d}}_k, \widehat{\mathbf{T}}_k(\boldsymbol{q}_k) - \boldsymbol{q}_k \big\rangle &= \big\langle \widehat{\boldsymbol{d}}_k, \widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) - \boldsymbol{q}_k^{\mathrm{QL}} \big\rangle - (1-\gamma)\beta_k \langle \widehat{\boldsymbol{d}}_k, \mathbf{1} \rangle \\
&= \big\langle \widehat{\boldsymbol{d}}_k, \widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) - \boldsymbol{q}_k^{\mathrm{QL}} \big\rangle - (1-\gamma)\beta_k,
\end{aligned}$$

where we use the identity $\langle \widehat{\boldsymbol{d}}_k, \mathbf{1} \rangle = 1$ in the second line since $\widehat{\boldsymbol{d}}_k \in \Delta(\mathcal{S} \times \mathcal{A})$. Recalling the definition of α_k in (25), one thus have

$$\alpha_k = \lambda_k \Big(\frac{\gamma}{1-\gamma}\Big) \big\langle \widehat{\boldsymbol{d}}_k, \widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) - \boldsymbol{q}_k^{\mathrm{QL}} \big\rangle - \gamma \lambda_k \beta_k.$$

□

Plugging the expression for α_k from Lemma A.5 into the update rule of Lemma A.4, we derive the iteration

$$\beta_{k+1} = (1-\lambda_k)\beta_k + \lambda_k \Big(\frac{\gamma}{1-\gamma}\Big) \big\langle \widehat{\boldsymbol{d}}_k, \widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) - \boldsymbol{q}_k^{\mathrm{QL}} \big\rangle, \quad k = 0, 1, \ldots, \tag{27}$$

initialized by $\beta_0 = 0$. Next, using the convergence of QL, we can show the convergence of β_k:

Lemma A.6. *The iterates β_k in (27) converge to zero almost surely.*

Proof. Define $\beta_{1,0} = \beta_{2,0} = 0$ and consider the iterations

$$\beta_{1,k+1} = (1-\lambda_k)\beta_{1,k} + \lambda_k \Big(\frac{\gamma}{1-\gamma}\Big) \big\langle \widehat{\boldsymbol{d}}_k, \widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) - \overline{\mathbf{T}}(\boldsymbol{q}_k^{\mathrm{QL}}) \big\rangle, \tag{28}$$

$$\beta_{2,k+1} = (1-\lambda_k)\beta_{2,k} + \lambda_k \Big(\frac{\gamma}{1-\gamma}\Big) \big\langle \widehat{\boldsymbol{d}}_k, \overline{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) - \boldsymbol{q}_k^{\mathrm{QL}} \big\rangle, \tag{29}$$

for $k = 0, 1, \ldots$, so that $\beta_k = \beta_{1,k} + \beta_{2,k}$ for all $k \geq 0$. In what follows, we use the fact that the QL iterates $\boldsymbol{q}_k^{\mathrm{QL}}$ are bounded and converge to $\boldsymbol{q}^\star = \overline{\mathbf{T}}(\boldsymbol{q}^\star)$ almost surely (Tsitsiklis, 1994, Thm. 4). First, observe that the iteration (28) converges to zero almost surely using (Tsitsiklis, 1994, Lem. 1) and the fact that (see also (Tsitsiklis, 1994, Sec. 7))

$$\mathbb{E}_{\widehat{s}_k^+}\left[\left\langle \widehat{\boldsymbol{d}}_k, \widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) - \overline{\mathbf{T}}(\boldsymbol{q}_k^{\mathrm{QL}})\right\rangle\right] = \left\langle \widehat{\boldsymbol{d}}_k, \mathbb{E}_{\widehat{s}_k^+}\left[\widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) - \overline{\mathbf{T}}(\boldsymbol{q}_k^{\mathrm{QL}})\right]\right\rangle = 0,$$
$$\mathbb{E}_{\widehat{s}_k^+}\left[\left\langle \widehat{\boldsymbol{d}}_k, \widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) - \overline{\mathbf{T}}(\boldsymbol{q}_k^{\mathrm{QL}})\right\rangle^2\right] \leq \mathbb{E}_{\widehat{s}_k^+}\left[\left\|\widehat{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) - \overline{\mathbf{T}}(\boldsymbol{q}_k^{\mathrm{QL}})\right\|_\infty^2\right] \leq \left\|\boldsymbol{q}_k^{\mathrm{QL}}\right\|_\infty^2.$$

The iteration (29) also converges to zero almost surely since

$$\beta_{2,k+1} = \left(\frac{\gamma}{1-\gamma}\right) \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \widehat{\boldsymbol{d}}_k(s,a) \left\{\frac{1}{k+1}\sum_{\ell=0}^{k}\left([\overline{\mathbf{T}}_\ell(\boldsymbol{q}_\ell^{\mathrm{QL}}) - \boldsymbol{q}_\ell^{\mathrm{QL}}](s,a)\right)\right\},$$

is a scaled, weighted average of Cesaro means of the sequences $[\overline{\mathbf{T}}_k(\boldsymbol{q}_k^{\mathrm{QL}}) - \boldsymbol{q}_k^{\mathrm{QL}}](s,a)$ that converge to zero almost surely for each $(s,a) \in \mathcal{S}\times\mathcal{A}$ (recall that $\lambda_k = 1/(k+1)$ and $\boldsymbol{q}_k^{\mathrm{QL}} \to \boldsymbol{q}^\star = \overline{\mathbf{T}}(\boldsymbol{q}^\star)$ almost surely). □

Finally, recall the characterization $\boldsymbol{q}_k = \boldsymbol{q}_k^{\mathrm{QL}} + \beta_k \mathbf{1}$ in Lemma A.4 and observe that $\boldsymbol{q}_k^{\mathrm{QL}} \to \boldsymbol{q}^\star$ and $\beta_k \to 0$ almost surely. Therefore, $\boldsymbol{q}_k \to \boldsymbol{q}^\star$ almost surely.

B. On numerical experiments

B.1. Algorithms

Below, we provide the update rules of the algorithms we employed in our numerical experiments. We adapt the update rules provided by (Kolarijani & Mohajerin Esfahani, 2023) in our implementations for the planning algorithms.

- **Nesterov VI algorithm (Goyal & Grand-Clément, 2022)**:

$$\boldsymbol{z}_k = \boldsymbol{v}_k + \frac{1-\sqrt{1-\gamma^2}}{\gamma}(\boldsymbol{v}_k - \boldsymbol{v}_{k-1}),$$
$$\boldsymbol{v}_{k+1} = \boldsymbol{z}_k + \frac{1}{1+\gamma}(\mathbf{T}(\boldsymbol{z}_k) - \boldsymbol{z}_k).$$

- **Anderson VI algorithm (Geist & Scherrer, 2018)**: (The following update rule is for Anderson acceleration with memory equal to 1 which corresponds to a rank-one approximation of the Hessian.)

$$\boldsymbol{z}_k = \boldsymbol{v}_k - \boldsymbol{v}_{k-1},$$
$$\boldsymbol{z}'_k = \mathbf{T}(\boldsymbol{v}_k) - \mathbf{T}(\boldsymbol{v}_{k-1}),$$
$$\delta_k = \begin{cases} 0, & \boldsymbol{z}_k^\top(\boldsymbol{z}_k - \boldsymbol{z}'_k) = 0, \\ \frac{\boldsymbol{z}_k^\top\left(\boldsymbol{v}_k - \mathbf{T}(\boldsymbol{v}_k)\right)}{\boldsymbol{z}_k^\top(\boldsymbol{z}_k - \boldsymbol{z}'_k)}, & \text{otherwise}, \end{cases}$$
$$\boldsymbol{v}_{k+1} = (1-\delta_k)\mathbf{T}(\boldsymbol{v}_k) + \delta_k \mathbf{T}(\boldsymbol{v}_{k-1}).$$

- **Speedy QL algorithm (Ghavamzadeh et al., 2011)**: (The following update rule is the synchronous implementation of Speedy QL.)

$$
\begin{aligned}
&\texttt{for } (s,a) \in \mathcal{S} \times \mathcal{A} \\
&\quad \widehat{s}^{\,+} \sim P(\cdot|s,a), \\
&\quad \boldsymbol{z}_k(s,a) = \boldsymbol{c}(s,a) + \gamma \min_{a^+ \in \mathcal{A}} \boldsymbol{q}_k(\widehat{s}^{\,+}, a^+), \\
&\quad \boldsymbol{z}'_k(s,a) = \boldsymbol{c}(s,a) + \gamma \min_{a^+ \in \mathcal{A}} \boldsymbol{q}_{k-1}(\widehat{s}^{\,+}, a^+), \\
&\texttt{endfor} \\
&\quad \boldsymbol{q}_{k+1} = \boldsymbol{q}_k + \frac{1}{1+k}(\boldsymbol{z}'_k - \boldsymbol{q}_k) + \frac{k}{1+k}(\boldsymbol{z}_k - \boldsymbol{z}'_k).
\end{aligned}
$$

- **Zap QL algorithm (Devraj & Meyn, 2017)**: (The following update rule is also the synchronous implementation of Zap QL without eligibility trace. The matrix $\boldsymbol{F}_k \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|\times|\mathcal{S}\times\mathcal{A}|}$ below denotes the sampled transition matrix at iteration k.)

$$
\begin{aligned}
&\boldsymbol{F}_k = \mathbf{0} \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|\times|\mathcal{S}\times\mathcal{A}|}, \\
&\texttt{for } (s,a) \in \mathcal{S} \times \mathcal{A} \\
&\quad \widehat{s}^{\,+} \sim P(\cdot|s,a), \\
&\quad \widehat{a}^{\,+} = \operatorname*{argmin}_{a^+ \in \mathcal{A}} \boldsymbol{q}_k(\widehat{s}^{\,+}, a^+), \\
&\quad \big[\widehat{\mathbf{T}}_k(\boldsymbol{q}_k)\big](s,a) = \boldsymbol{c}(s,a) + \gamma \min_{a^+ \in \mathcal{A}} \boldsymbol{q}_k(\widehat{s}^{\,+}, a^+) - \boldsymbol{q}_k(s,a), \\
&\quad \boldsymbol{F}_k\big((s,a),(\widehat{s}^{\,+},\widehat{a}^{\,+})\big) = 1, \\
&\texttt{endfor} \\
&\widehat{\boldsymbol{P}}_k = \widehat{\boldsymbol{P}}_{k-1} + \frac{1}{2+k}(\boldsymbol{F}_k - \widehat{\boldsymbol{P}}_{k-1}), \\
&\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + \frac{1}{1+k}(\boldsymbol{I} - \gamma\widehat{\boldsymbol{P}}_k)^{-1}\big(\widehat{\mathbf{T}}_k(\boldsymbol{q}_k) - \boldsymbol{q}_k\big).
\end{aligned}
$$

B.2. Extended numerical analysis

In this appendix, we provide the Bellman and value errors observed throughout the iterations of planning and learning algorithms. We run each planning algorithm until the error thresholds in Table 1 are achieved.

Table 1: Error thresholds for planning algorithms.

MDP	Error	$\gamma = 0.9$	$\gamma = 0.9$	$\gamma = 0.9$	$\gamma = 0.9$
Garnet	value	10^{-5}	10^{-4}	10^{-4}	10^{-2}
	Bellman	10^{-5}	10^{-5}	10^{-5}	10^{-4}
Graph	value	10^{-5}	10^{-4}	10^{-3}	10^{-2}
	Bellman	10^{-5}	10^{-5}	10^{-5}	10^{-4}

We consider four different values for discount factor γ across 25 realizations of the Garnet MDP. Note that the planning algorithms are deterministic in nature, hence, the only source of variation in the errors is due to the random realization of the Garnet MDPs. Figure 3 shows the range of error values observed within the span of the iterations of the planning algorithms. The solid curve represents the median error values, while the shaded region around the curve indicates the errors between the first and the third quantiles. We follow the same style of presentation in the other figures in this section.

We observe in Figure 3 that Anderson VI has the highest error variance. Furthermore, the error curves for both Anderson VI and Nesterov VI are not monotonically decreasing, which is particularly visible for Anderson VI. This is because the iterations in these algorithms are not necessarily a contraction with a guaranteed reduction in the Bellman/value error.

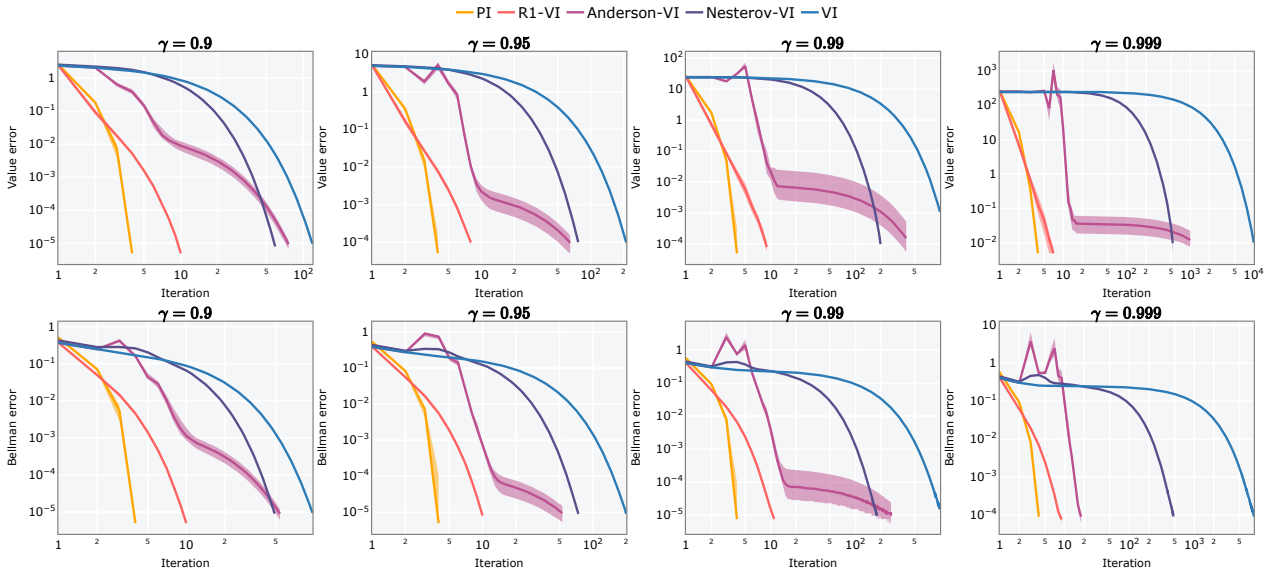

Figure 3: Comparison of the planning algorithms in Garnet MDP with various γ values.

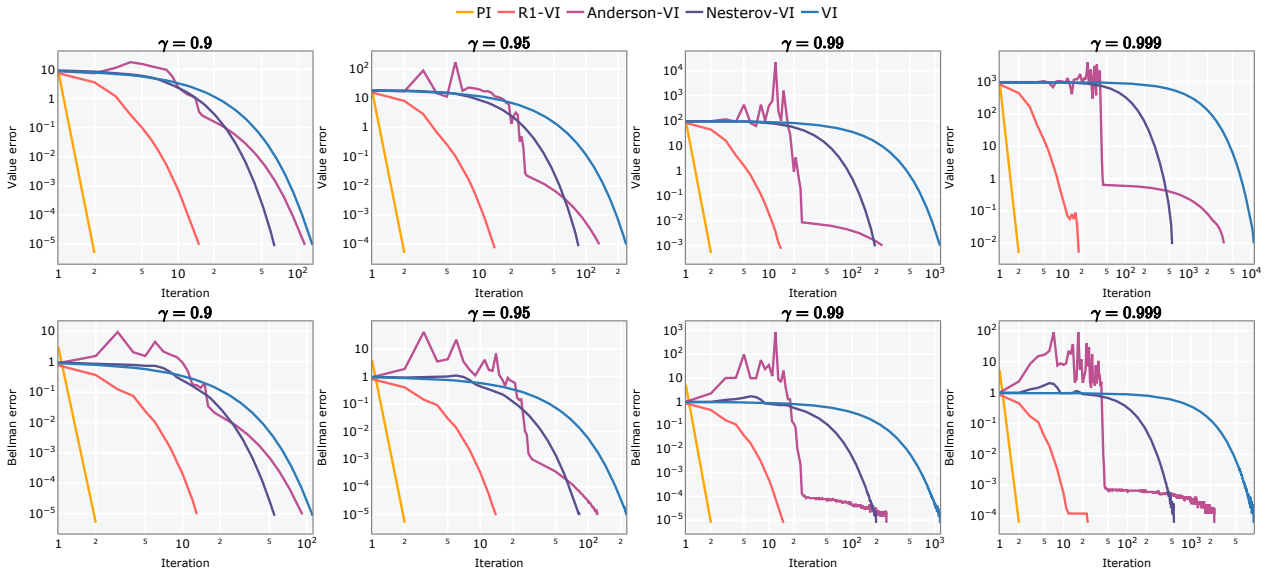

Figure 4: Comparison of the planning algorithms in Graph MDP with various γ values.

Nevertheless, in our numerical experiments, both algorithms seem to be convergent. Figure 4 shows the error curves of planning algorithms for the Graph MDP. Here, the non-monotonic behavior of Anderson VI is more apparent as the error values initially increase with an oscillating behavior. Similar to Garnet MDPs, R1-V1 consistently provides lower errors throughout the iterations.

Figure 5 and 6 show the error curves for the learning algorithms in Garnet and Graph MDPs, respectively. In these experiments, we run each learning algorithm with 5 different seeds to marginalize the randomness in the sampling process. We observe that the difference between Zap QL, Speedy QL, and R1-QL is not noticeable at lower values of the discount factors (i.e., $\gamma \leq 0.95$). However, as the discount factor increases, particularly at $\gamma = 0.999$, the gap between the error values increases. At higher values of the discount factors, R1-QL consistently yields lower error values, while QL struggles to minimize the errors due to the linearly decaying step-size λ_k. Furthermore, we observe that Zap QL displays higher error

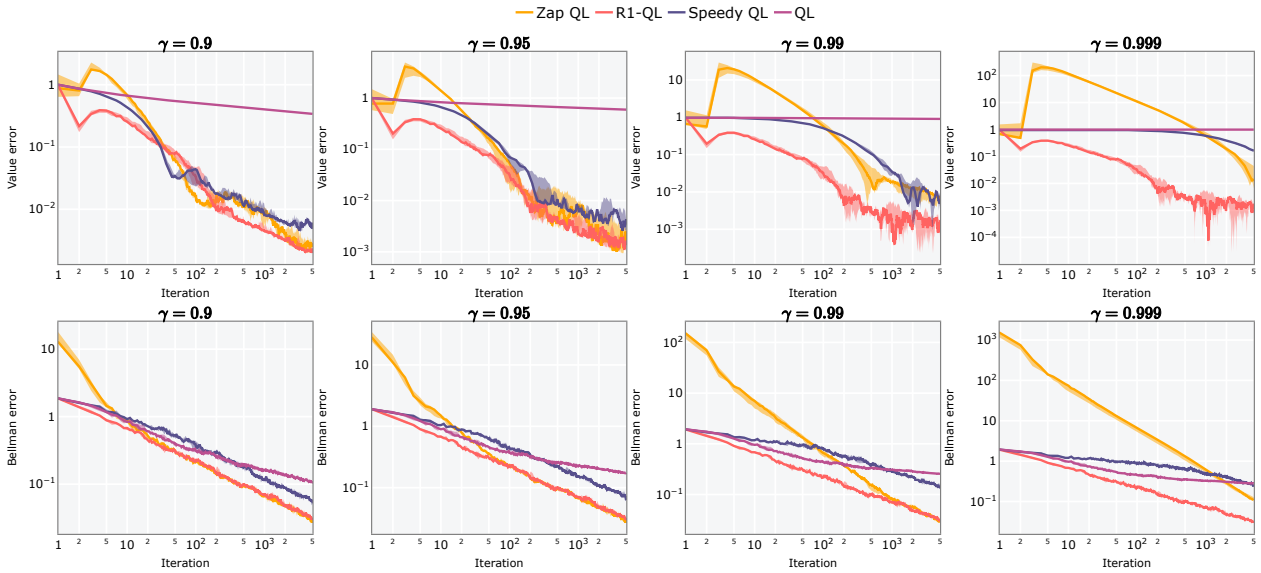

Figure 5: Comparison of the learning algorithms in Garnet MDP with various γ values.

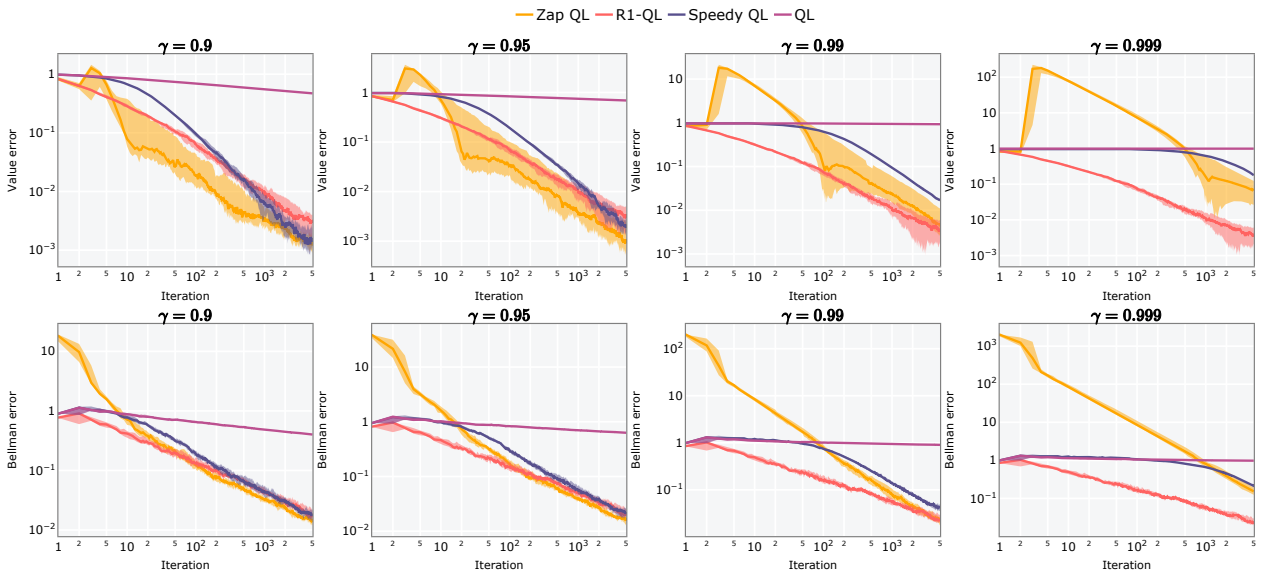

Figure 6: Comparison of the learning algorithms in Graph MDP with various γ values.

variance, which may be due to the inversion of the estimated "Hessain" $(\boldsymbol{I} - \gamma \widehat{\boldsymbol{P}})$. In contrast, R1-QL exhibits considerably lower variance, despite implicitly performing a similar inversion. We argue that the lower error variance observed with R1-QL is due to the low-rank approximation of the transition probability matrix via estimation of the corresponding stationary distribution.

B.3. Reducible MDPs

To illustrate R1-VI's behavior in a reducible MDP, we replicate the comparison from Section 5 within the Gridworld environment of (Sutton & Barto, 2018), which inherently contains an absorbing state. We consider two Gridworld variants:

1. **Absorbing Gridworld**, in which the absorbing state yields positive reward.

2. **Terminal Gridworld**, in which the absorbing state grants a zero reward.

Here, "reducibility" refers to the Markov chain induced by the optimal policy. Note, however, that even in the absence of an absorbing state, where all actions from a state lead back to itself, a non-optimal policy may still produce a reducible chain in Gridworld.

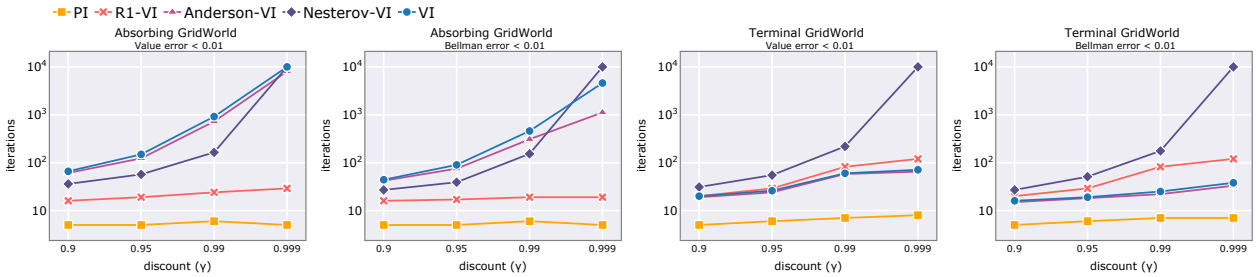

Figure 7: Comparison of planning algorithms on two reducible Gridworld MDP instances, one with a zero-reward absorbing state and one with a positive-reward absorbing state, across various γ values.

In Absorbing-Gridworld (left side of Figure 7), R1-VI yields the lowest value and Bellman error, apart from PI, across all γ values. However, in Terminal Gridworld (right side of Figure 7), R1-VI performs slightly worse than the other accelerated algorithms. This is explained by the fact that under the optimal policy, the stationary distribution concentrates on the absorbing state, which provides zero reward and hence zero Bellman error when the values are initialized to zero. Consequently, the second term in the R1-VI update (Equation (9)) vanishes. In contrast, in Absorbing Gridworld, the Bellman error at the absorbing state is not immediately zero, hence the second term in R1-VI contributes to improve convergence.

B.4. Policy performance

Throughout Section 5, we compared both the planning and learning algorithms, including our proposed R1VI and R1QL methods, using the value and Bellman error metrics. However, rapid convergence in value does not necessarily translate into equally rapid convergence in policy space, which is the ultimate criterion of policy optimization. In this section, we present a comparative analysis based on the policy evaluation metric in the Graph and Garnet MDPs.

Figure 8 shows that the planning algorithms yield exactly the same policy evaluation, except for PI and Anderson-VI. Anderson-VI initially struggles to find the optimal policy due to instabilities in the value space (shown in Figures 3 and 4), but eventually converges to the optimal policy. In both MDPs, policy convergence occurs in fewer than five steps, except for Anderson-VI, whereas convergence in the value space requires several orders of magnitude more steps.

The convergence of policies among the learning algorithms is more varied than that of the planning algorithms. Figure 9 shows that, for the Graph MDP, all algorithms except Zap-QL achieve almost identical performance, converging within five steps for every value of γ. In the Garnet MDP, convergence requires more steps, and improvements over iterations are slower.

In both Figures 9 and 8, the proposed R1VI and R1QL algorithms match the policy performance of VI and QL, respectively, across all iterations. Moreover, VI in the planning setting and QL in the learning setting produce among the highest policy performance observed across both MDPs, despite exhibiting the slowest convergence in the value space.

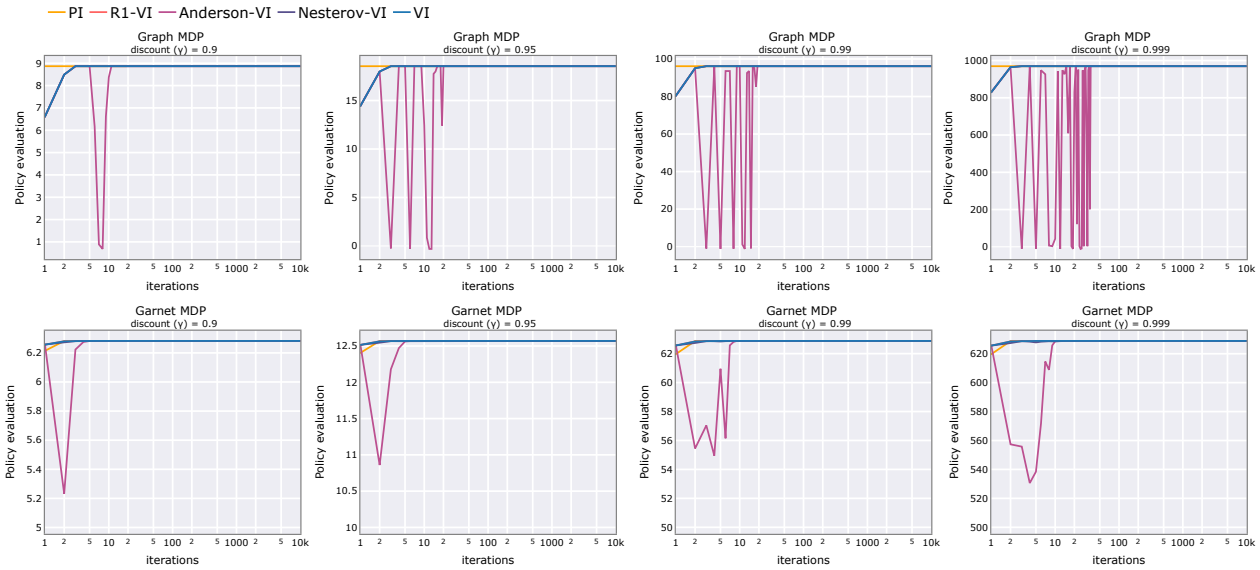

Figure 8: Comparison of the value of the greedy policy produced by various planning algorithms (R1-VI, VI, PI, Anderson-VI, and Nesterov-VI) on Garnet and Graph MDPs over a range of discount factors γ. Note that R1-VI, VI, and Nesterov-VI yield essentially overlapping results.

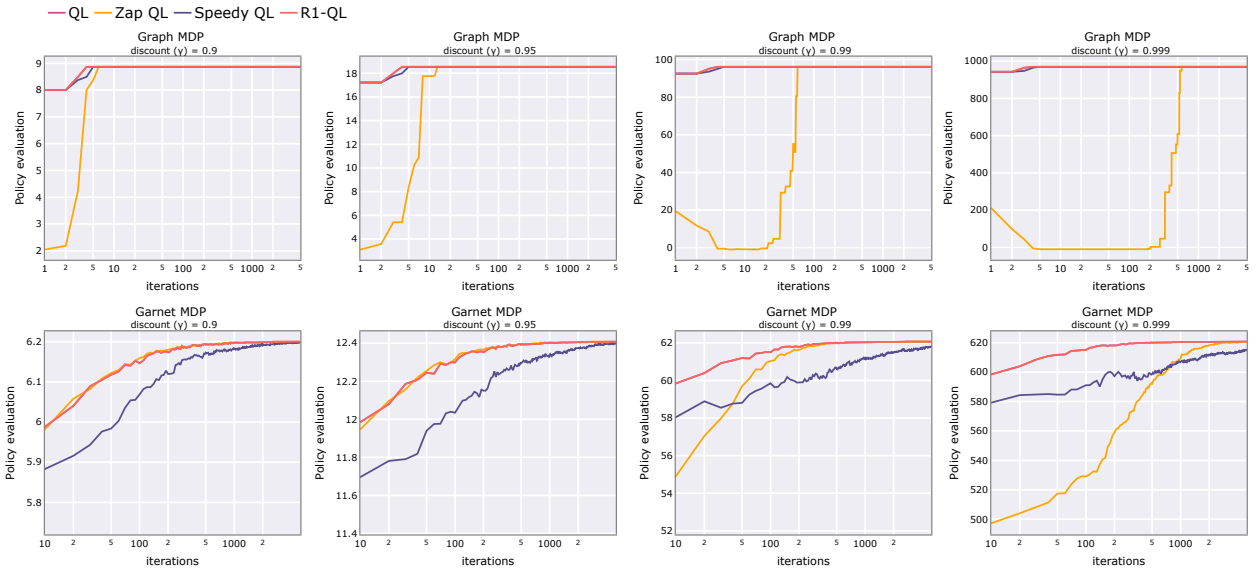

Figure 9: Comparison of the value of the greedy policy produced by the R1-QL and QL learning algorithms on Garnet and Graph MDPs as a function of the discount factor γ. Results for R1-QL and QL coincide. The y-axis shows the median policy-evaluation score of the corresponding greedy policies across 5 different seeds.