A Stochastic Approximation Method with Max-Norm Projections and its Applications to the Q-Learning Algorithm

Sumit Kunnumkal
Indian School of Business, Gachibowli, Hyderabad, 500032, India
Email: sumit_kunnumkal@isb.edu
Tel: 91-040-23187168

Huseyin Topaloglu
School of Operations Research and Information Engineering,
Cornell University, Ithaca, New York 14853, USA
Email: topaloglu@orie.cornell.edu
Tel: 1-607-255-0698

January 20, 2009

Abstract

In this paper, we develop a stochastic approximation method to solve a monotone estimation problem and use this method to enhance the empirical performance of the Q-learning algorithm when applied to Markov decision problems with monotone value functions. We begin by considering a monotone estimation problem where we want to estimate the expectation of a random vector η. We assume that the components of $\mathbb{E}\{\eta\}$ are known to be in increasing order. The stochastic approximation method that we propose is designed to exploit this information by projecting its iterates onto the set of vectors with increasing components. The novel aspect of the method is that it uses projections with respect to the max norm. We show the almost sure convergence of the stochastic approximation method. After this result, we consider the Q-learning algorithm when applied to Markov decision problems with monotone value functions. We study a variant of the Q-learning algorithm that uses projections to ensure that the value function approximation that is obtained at each iteration is also monotone. Computational results indicate that the performance of the Q-learning algorithm can be improved significantly by exploiting the monotonicity property of the value functions.

Stochastic approximation methods are often used for solving online learning, simulation optimization and stochastic control problems. The practical appeal of these methods usually comes from the fact that they are incremental and iterative in nature. In particular, stochastic approximation methods do not require the knowledge of the probability distributions of the underlying random variables. Instead, they iteratively use samples of the random variables that are generated either through computer simulation or by observing the real system. For this reason, stochastic approximation methods are often referred to as model free methods.

In this paper, we develop a stochastic approximation method to solve a monotone estimation problem and use this method to enhance the empirical performance of the Q-learning algorithm when applied to Markov decision problems with monotone value functions. We begin by considering a monotone estimation problem, where the goal is to estimate the expectation of a random vector η. We assume that the components of the vector $\mathbb{E}\{\eta\}$ are known to be in increasing order and our stochastic approximation method is designed to exploit this information. In particular, our stochastic approximation method iteratively obtains samples of η to generate a sequence of approximations to $\mathbb{E}\{\eta\}$. We use projections to ensure that each approximation in the sequence also has increasing components. We show the almost sure convergence of the sequence of approximations to $\mathbb{E}\{\eta\}$. After this result, we consider the Q-learning algorithm when applied to Markov decision problems with monotone value functions. The Q-learning algorithm works very much like a stochastic approximation method and generates a sequence of value function approximations by using samples of the current state of the system, the decision chosen by the user and the cost incurred in the state transition. The links between the Q-learning algorithm and stochastic approximation methods are recognized in the literature, and, similarly to many stochastic approximation methods, the Q-learning algorithm may suffer from slow empirical convergence. In this paper, we focus on Markov decision problems with monotone value functions. Such Markov decision problems arise in a variety of settings including queue admission, batch service, marketing, and aging and replacement problems. We develop a variant of the Q-learning algorithm that ensures that the value function approximation that is obtained at each iteration is also monotone. Our goal is to enhance the empirical performance of the Q-learning algorithm by imposing the monotonicity property of the value function on the value function approximations.

Our work draws on two papers in particular. The first paper is Powell, Ruszczynski, and Topaloglu [2004], where the authors propose a stochastic approximation method to construct approximations to the convex recourse functions that arise in stochastic programs. The primary idea behind their stochastic approximation method is to project the recourse function approximations onto the set of convex functions to ensure that the recourse function approximations are convex. Our use of projections in the Q-learning algorithm to impose the monotonicity of the value function on the value function approximations is inspired by their work. However, a crucial distinguishing feature of our work is that we use projections with respect to the max norm, whereas Powell et al. [2004] use projections with respect to the Euclidean norm. There are several reasons for our use of projections with respect to the max norm. First, our proof of convergence for the new variant of the Q-learning algorithm relies on an order preserving property of the projection, which can roughly be stated as follows. We let x_0 and y_0 be two vectors in $\Re^n$ with increasing components and assume that x_1 and y_1 are respectively obtained from x_0 and y_0 by perturbing only the jth components of x_0 and y_0. Letting π_{x_1} and π_{y_1} respectively be the projections of x_1 and y_1 onto the set of vectors with increasing components, the order preserving property states that if we have $x_1 \leq y_1$, then we also have $\pi_{x_1} \leq \pi_{y_1}$. Kunnumkal and Topaloglu [2007] show that the order preserving property is quite tedious to establish when the projections are with respect to the Euclidean norm, but it follows almost by definition when the projections are with respect to the max norm. Second, our computational experience indicates that projections with respect to the max norm may provide better empirical convergence rate than projections with respect to the Euclidean norm. Third, projections with respect to the max norm can be computed quite efficiently. In particular, we show that there is a closed form expression for the projections that we use in this paper and this closed form expression can be computed in $O(n)$ time when dealing with a vector in $\Re^n$. Furthermore, projections with respect to the max norm can, in general, be computed by solving a linear program.

The second paper that is closely related to our work is Tsitsiklis [1994]. Tsitsiklis [1994] provides an alternative convergence proof for the Q-learning algorithm by posing the Q-learning algorithm as a stochastic approximation method. In this paper, we extend his convergence proof in two directions to be able to embed projections into the Q-learning algorithm. First, we establish the almost sure convergence of a stochastic approximation method that uses projections with respect to the max norm. Second, we show that our projections satisfy the aforementioned order preserving property. It is important to note that projections with respect to the max norm do not have the nonexpansiveness property in the sense of Proposition 2.2.1.c in Bertsekas, Nedic, and Ozdaglar [2003] and the convergence proof that we use for our stochastic approximation method relies on a somewhat nonstandard argument. In particular, our proof technique considers a version of our stochastic approximation method that does not use projections at all. This version is trivially known to be convergent. Our approach relies on showing that the distance between the iterates of the projected and unprojected versions diminishes as the iterations progress. It turns out that Vazquez-Abad, Cassandras, and Julka [1998] and van Ryzin and McGill [2000] also use similar auxiliary unprojected stochastic approximation methods. However, they need to keep track of the iterates of their auxiliary stochastic approximation methods when they apply their approaches in practice, whereas we use the unprojected auxiliary stochastic approximation method as an aid in establishing convergence and the practical application of our approach does not require keeping track of the iterates of the auxiliary stochastic approximation method. We also note that the stochastic approximation method that we develop in this paper may be of independent interest, since we are not aware of other stochastic approximation methods that work under projections with respect to the max norm.

There is a large body of literature revolving around stochastic approximation methods and the books by Kushner and Clark [1978], Benveniste, Metivier, and Priouret [1991], Bertsekas and Tsitsiklis [1996] and Kushner and Yin [2003] provide comprehensive coverage of the theory. The use of projections in stochastic approximation methods is common, but the role of projections has traditionally been to ensure the boundedness or feasibility of the iterates. Examples of using projections to ensure the boundedness of the iterates can be found in Ljung [1977] and Andradottir [1995], whereas Vazquez-Abad et al. [1998], Vazquez-Abad [1999] and Bertsekas et al. [2003] use projections to ensure the feasibility of the iterates. On the other hand, we use projections to impose structural properties on the iterates and the use of projections in learning algorithms to impose the structure of the value function on the value function approximations appears to be rare. We have not seen other papers that use projections in learning algorithms to impose structural properties, with only one exception being Powell et al. [2004].

The Q-learning algorithm dates back to Watkins [1989] and Watkins and Dayan [1992] and this algorithm is now considered as a standard tool for solving Markov decision problems. The basic idea behind the algorithm is to obtain information about the system by using sampled trajectories. The algorithm starts with an arbitrary value function approximation and constructs a sequence of value function approximations by iteratively updating the previous approximation. One of the appealing features of the Q-learning algorithm is that it updates the previous value function approximation only by using a sample of the current state of the system, the decision chosen by the user, the next state following the state transition and the cost incurred along the way. In particular, the full set of transition probabilities and the costs are not required. As a result, the Q-learning algorithm is often referred to as a model free method to solve Markov decision problems. Barto, Bradtke, and Singh [1995], Sutton and Barto [1998] and Si, Barto, Powell, and Wunsch II [2004] give a nice overview of the different ways in which the Q-learning algorithm may be applied in practice.

The literature on approximate dynamic programming is also related to our paper. There is recent work in this area indicating that the performance of approximate dynamic programming algorithms can be improved by exploiting structural properties. For example, Godfrey and Powell [2001], Topaloglu and Powell [2003], Papadaki and Powell [2003], Powell et al. [2004] and Topaloglu [2005] consider dynamic programs where the value functions are known to be convex. These papers propose approximate dynamic programming algorithms that ensure that the estimates of the value functions obtained during the intermediate iterations are also convex. Similarly, a generic approach to approximate dynamic programming involves using value function approximations of the form $\sum_{p=1}^{P} r_p \, B_p(\cdot)$, where $(r_1, \ldots, r_P)$ are adjustable parameters and $(B_1(\cdot), \ldots, B_P(\cdot))$ are fixed basis functions. The challenge is to choose the basis functions and adjust the parameters so that $\sum_{p=1}^{P} r_p \, B_p(\cdot)$ represents a good value function approximation; see Bertsekas and Tsitsiklis [1996], Tsitsiklis and Van Roy [1997], Tsitsiklis and Van Roy [2001], de Farias and Van Roy [2003] and Powell [2007]. The basis function terminology here comes from the fact that the set of possible value function approximations of the form $\sum_{p=1}^{P} r_p \, B_p(\cdot)$ that can be generated by adjusting the parameters $(r_1, \ldots, r_P)$ are limited by the span of the fixed functions $(B_1(\cdot), \ldots, B_P(\cdot))$. The emerging theme from this literature is that the basis functions should be chosen to reflect the known structural properties of the value function as closely as possible.

We make the following research contributions in this paper. 1) We show the almost sure convergence of a stochastic approximation method that can be used to estimate the expectation of a random vector η. We assume that the components of $\mathbb{E}\{\eta\}$ are known be in increasing order and our stochastic approximation method exploits this information by projecting its iterates onto the set of vectors with increasing components. 2) We consider Markov decision problems with monotone value functions and develop a variant of the Q-learning algorithm that uses our stochastic approximation method to ensure that the value function approximation that is obtained at each iteration is also monotone. We establish the almost sure convergence of the new variant of the Q-learning algorithm. 3) We present computational results on a batch service problem that compare the new variant of the Q-learning algorithm with the standard version and report significant gains in empirical performance.

The rest of the paper is organized as follows. In Section 1, we describe our stochastic approximation method and establish its convergence. In Section 2, we work with Markov decision problems with monotone value functions and show how to use our stochastic approximation method in the context of the Q-learning algorithm to ensure that the value function approximation that is obtained at each iteration is also monotone. In Section 3, we present our computational results.

1 The Stochastic Approximation Method

We begin by defining some notation that we use throughout the paper. We use $v(j)$ to denote the jth component of $v \in \Re^n$. For fixed scalars L and U, we define $\mathcal{V}(L,U) \subset \Re^n$ as $\mathcal{V}(L,U) = \{v \in \Re^n : L \leq v(1) \leq \ldots \leq v(n) \leq U\}$ so that $\mathcal{V}(L,U)$ is the set of vectors in $\Re^n$ whose components are increasing and bounded between L and U. We let $\mathbf{1}(\cdot)$ be the indicator function and $\|\cdot\|$ be the max norm on $\Re^n$ defined as $\|v\| = \max_{j \in \{1,\ldots,n\}} |v(j)|$. We write w.p.1 for with probability 1.

In this section, we analyze the stochastic approximation method given in Algorithm 1 below. The goal of this stochastic approximation method is to generate a sequence $\{v_k\}$ of approximations to the expectation $\mathbb{E}\{\eta\}$ of a random vector η by using a sequence $\{\eta_k\}$ of "samples" of η. We assume that the components of $\mathbb{E}\{\eta\}$ are known to be in increasing order.

Algorithm 1

(1) Choose $v_1 \in \mathcal{V}(L,U)$. Set $k = 1$.

(2) Letting η_k be a random variable taking values in $\Re^n$, J_k be a random variable taking values in $\{1,\ldots,n\}$ and $\alpha_k \geq 0$ be a step size parameter, set

$$z_k(j) = v_k(j) + \alpha_k \, \mathbf{1}(j = J_k) \, [\eta_k(j) - v_k(j)] \tag{1}$$

for all $j \in \{1,\ldots,n\}$.

(3) Set $v_{k+1} \in \operatorname{argmin}_{w \in \mathcal{V}(L,U)} \|z_k - w\|$.

(4) Increase k by 1 and go to Step 2.

We would like to show that $\lim_{k\to\infty} v_k = \mathbb{E}\{\eta\}$ w.p.1. In Step 2, we observe the realizations of the random variables η_k and J_k. One interpretation of the random variable η_k is that we obtain a sample of η through the random variable η_k. For example, η may correspond to a random output from a simulation model, in which case η_k would correspond to the output from the kth replication of the simulation model. To observe a realization of η_k, we simply need to run the simulation model once and observing a realization of η_k does not necessarily require knowing the distribution of η. In Step 3, we project z_k onto $\mathcal{V}(L,U)$ so that we have $v_{k+1} \in \mathcal{V}(L,U)$. This step ensures that the sequence of approximations $\{v_k\}$ have increasing components and its purpose is to exploit the information that $\mathbb{E}\{\eta\}$ has increasing components. We note that the result of the projection operator in Step 3 may not be unique. We shortly address this issue.

We use $\mathcal{F}_k$ to denote the history of Algorithm 1 up to the beginning of iteration k, which is captured by the random variables $\{\eta_1, \ldots, \eta_{k-1}, J_1, \ldots, J_{k-1}\}$. In this case, if the step size parameter α_k is a deterministic function of $\mathcal{F}_k$, then the whole evolution of Algorithm 1 up to the beginning of iteration k is deterministically specified by the knowledge of $\mathcal{F}_k$. We assume that the initial iterates of all stochastic approximation methods that we consider are deterministic so that they do not need to be included in the history. We assume that the following statements hold for Algorithm 1.

(A1) Using $\bar{\eta}$ to denote $\mathbb{E}\{\eta\}$, there exist finite scalars L, U and A such that we have $\bar{\eta} \in \mathcal{V}(L,U)$, $\mathbb{E}\{\eta_k(J_k) \,|\, \mathcal{F}_k, J_k\} = \bar{\eta}(J_k)$ and $\mathbb{E}\{[\eta_k(j)]^2 \,|\, \mathcal{F}_k, J_k\} \leq A$ w.p.1 for all $j \in \{1, \ldots, n\}$, $k = 1, 2, \ldots$.

(A2) The step size parameter α_k is positive and is a deterministic function of $\mathcal{F}_k$ for all $k = 1, 2, \ldots$.

(A3) The random variable J_k and the step size parameter α_k satisfy $\sum_{k=1}^{\infty}[\mathbf{1}(j = J_k)\,\alpha_k] = \infty$ w.p.1 and $\sum_{k=1}^{\infty}\mathbb{E}\{\mathbf{1}(j = J_k)\,\alpha_k^2\} < \infty$ for all $j \in \{1, \ldots, n\}$.

By (A1), the components of $\mathbb{E}\{\eta\}$ are increasing and bounded between the finite scalars L and U. To ensure that $\mathbb{E}\{\eta_k(J_k) \,|\, \mathcal{F}_k, J_k\} = \bar{\eta}(J_k)$, we can sample η_k from the probability distribution of η and independent of $\mathcal{F}_k$ and J_k. If we sample η_k in this fashion, then we trivially have $\mathbb{E}\{[\eta_k(j)]^2 \,|\, \mathcal{F}_k, J_k\} \leq A < \infty$ as long as η has finite variance. Nevertheless, η_k can be dependent on the history of the algorithm and our results continue to hold as long as (A1)-(A3) are satisfied. (A2) and (A3) are standard assumptions on step size parameters.

The next proposition shows that an element of the set $\operatorname{argmin}_{w \in \mathcal{V}(L,U)} \|z_k - w\|$ in Step 3 of Algorithm 1 can be computed by mere inspection. In this proposition, we omit the subscripts for the iteration number and write Step 2 of Algorithm 1 as

$$z(j) = v(j) + \alpha\,\mathbf{1}(j = J)\,[\eta(j) - v(j)], \tag{2}$$

where $v \in \mathcal{V}(L,U)$. Letting $z(0) = L$ and $z(n+1) = U$ for notational uniformity, since we have $v \in \mathcal{V}(L,U)$ and z differs from v only in the Jth component, we have either $z \in \mathcal{V}(L,U)$ or $z(J) > z(J+1)$ or $z(J-1) > z(J)$. We are now ready to show the result.

Proposition 1 *Let $v \in \mathcal{V}(L,U)$, z be as in (2) and*

$$M = \begin{cases} z(J) & \text{if } z \in \mathcal{V}(L,U) \\ \min\left\{\dfrac{z(J) + z(J+1)}{2}, U\right\} & \text{if } z(J) > z(J+1) \\ \max\left\{\dfrac{z(J-1) + z(J)}{2}, L\right\} & \text{if } z(J-1) > z(J) \end{cases} \tag{3}$$

$$\Pi_z^{L,U}(j) = \begin{cases} \min\{z(j), M\} & \text{if } j \in \{1, \ldots, J-1\} \\ M & \text{if } j = J \\ \max\{z(j), M\} & \text{if } j \in \{J+1, \ldots, n\} \end{cases} \tag{4}$$

for all $j \in \{1, \ldots, n\}$. Then, we have $\Pi_z^{L,U} \in \operatorname{argmin}_{w \in \mathcal{V}(L,U)} \|z - w\|$.

Proof The proof proceeds in three parts. First, we show that $\Pi_z^{L,U} \in \mathcal{V}(L,U)$. After this, we show that $\|z - \Pi_z^{L,U}\| = |z(J) - M|$. Finally, we show that $\|z - w\| \geq |z(J) - M|$ for all $w \in \mathcal{V}(L,U)$. These three parts collectively imply that $\Pi_z^{L,U} \in \operatorname{argmin}_{w \in \mathcal{V}(L,U)} \|z - w\|$. We provide the complete proof in the appendix. $\square$

Since we have $\Pi_{z_k}^{L,U} \in \operatorname{argmin}_{w \in \mathcal{V}(L,U)} \|z_k - w\|$ and $\Pi_{z_k}^{L,U}$ can be computed by mere inspection, we assume throughout the paper that we let $v_{k+1} = \Pi_{z_k}^{L,U}$ in Step 3 of Algorithm 1. We note that the computation in (3) can be carried out in constant time, whereas the computation in (4) can be carried out in $O(n)$ time. Therefore, we can compute $\Pi_z^{L,U}$ in $O(n)$ time.

The next proposition gives a convergence result for Algorithm 1. Its proof is divided between the next two subsections. The first subsection establishes some preliminary results and the second subsection finishes the proof. These two subsections can be skipped without loss of continuity if the reader is only interested in the practical implications of Proposition 2.

Proposition 2 *Let the sequence $\{v_k\}$ be generated by Algorithm 1. Assume that (A1)-(A3) hold and v_{k+1} in Step 3 of Algorithm 1 is chosen as $\Pi_{z_k}^{L,U}$. Then, we have $\lim_{k\to\infty} v_k = \mathbb{E}\{\eta\}$ w.p.1.*

1.1 Preliminary Results

In this subsection, we consider the following stochastic approximation method.

Algorithm 2

(1) Choose $w_1 \in \mathcal{V}(L,U)$. Set $k = 1$.

(2) Letting η_k, J_k and α_k be as in Step 2 of Algorithm 1, set

$$w_{k+1}(j) = w_k(j) + \alpha_k \, \mathbf{1}(j = J_k) \, [\eta_k(j) - w_k(j)]$$

for all $j \in \{1, \ldots, n\}$.

(3) Increase k by 1 and go to Step 2.

Our understanding is that the random variables $\{\eta_k\}$ and $\{J_k\}$ in Algorithm 1 are the same as the random variables $\{\eta_k\}$ and $\{J_k\}$ in Algorithm 2. In other words, Algorithms 1 and 2 are subjected to the same sequence of random variables when they are compared trajectory by trajectory. Algorithm 2 is a standard stochastic approximation method and it is widely known that if $\{w_k\}$ is generated by Algorithm 2 and (A1)-(A3) hold, then we have $\lim_{k\to\infty} w_k = \mathbb{E}\{\eta\}$ w.p.1; see Proposition 4.1 in Bertsekas and Tsitsiklis [1996]. The proof of Proposition 2 shows that the distance between the iterates of Algorithms 1 and 2 gets arbitrarily small as the iterations progress, in which case we obtain $\lim_{k\to\infty} v_k = \mathbb{E}\{\eta\}$ w.p.1. We begin with the next two preliminary lemmas.

Lemma 3 *Let $\{v_k\}$ be generated by Algorithm 1, $\{w_k\}$ be generated by Algorithm 2 and $\epsilon > 0$. Assume that (A1)-(A3) hold and v_{k+1} in Step 3 of Algorithm 1 is chosen as $\Pi_{z_k}^{L,U}$. Then, there exists a random* *iteration number K such that, w.p.1, K is finite and*

$$\begin{aligned}
&\min\Big\{[1-\alpha_k\,\mathbf{1}(j=J_k)]\,[v_k(j)-w_k(j)],\ \min_{i\in\{j+1,\dots,n\}}\Big\{[1-\alpha_k\,\mathbf{1}(i=J_k)]\,[v_k(i)-w_k(i)]\Big\}-\epsilon,-\epsilon\Big\}\\
&\quad\le v_{k+1}(j)-w_{k+1}(j)\\
&\quad\le \max\Big\{[1-\alpha_k\,\mathbf{1}(j=J_k)]\,[v_k(j)-w_k(j)],\ \max_{i\in\{1,\dots,j-1\}}\Big\{[1-\alpha_k\,\mathbf{1}(i=J_k)]\,[v_k(i)-w_k(i)]\Big\}+\epsilon,\epsilon\Big\}
\end{aligned}$$

for all $j\in\{1,\dots,n\}$, $k\ge K$.

Proof The proof uses the fact that $\lim_{k\to\infty} w_k=\mathbb{E}\{\eta\}$ w.p.1 and follows from a lengthy but simple accounting argument that keeps track of how $v_{k+1}(j)$ and $w_{k+1}(j)$ are computed in Algorithms 1 and 2. We provide the complete proof in the appendix. □

Lemma 4 *Let $\{v_k\}$ be generated by Algorithm 1, $\{w_k\}$ be generated by Algorithm 2, $j\in\{1,\dots,n\}$ and $\epsilon>0$. Assume that (A1)-(A3) hold and v_{k+1} in Step 3 of Algorithm 1 is chosen as $\Pi^{L,U}_{z_k}$. Then, we have the following results.*

(1) *If there exists a finite iteration number K w.p.1 such that $v_{k+1}(j)-w_{k+1}(j)\ge\min\{[1-\alpha_k\,\mathbf{1}(j=J_k)]\,[v_k(j)-w_k(j)],-\epsilon\}$ for all $k\ge K$, then there exists a finite iteration number $\tilde K$ w.p.1 such that $v_{k+1}(j)-w_{k+1}(j)\ge-\epsilon$ for all $k\ge\tilde K$.*

(2) *If there exists a finite iteration number K w.p.1 such that $v_{k+1}(j)-w_{k+1}(j)\le\max\{[1-\alpha_k\,\mathbf{1}(j=J_k)]\,[v_k(j)-w_k(j)],\epsilon\}$ for all $k\ge K$, then there exists a finite iteration number $\tilde K$ w.p.1 such that $v_{k+1}(j)-w_{k+1}(j)\le\epsilon$ for all $k\ge\tilde K$.*

Proof We only show Part 1 and Part 2 follows similarly. All statements in the proof are in w.p.1 sense. In other words, we use a deterministic analysis to compare the trajectories of Algorithms 1 and 2 over one sample path, but we omit the dependence of $\{v_k\}$ and $\{w_k\}$ on the sample path for notational brevity. Noting the assumption in Part 1, we let K be a finite iteration number such that $v_{k+1}(j)-w_{k+1}(j)\ge\min\{[1-\alpha_k\,\mathbf{1}(j=J_k)]\,[v_k(j)-w_k(j)],-\epsilon\}$ for all $k\ge K$. By (A3), there exists a finite iteration number N such that $N\ge K$ and $\alpha_k\in[0,1]$ for all $k\ge N$. We first show that

$$v_{k+1}(j)-w_{k+1}(j)\ge\min\Big\{\prod_{\kappa=N}^{k}[1-\alpha_\kappa\,\mathbf{1}(j=J_\kappa)]\,[v_N(j)-w_N(j)],-\epsilon\Big\}\tag{5}$$

holds for all $k\ge N$ by using induction on k. By our choice of K and the fact that $N\ge K$, one can see that (5) holds for $k=N$. Assuming that (5) holds for $k\ge N$, our choice of K implies that

$$\begin{aligned}
&v_{k+2}(j)-w_{k+2}(j)\\
&\qquad\ge\min\Big\{[1-\alpha_{k+1}\,\mathbf{1}(j=J_{k+1})]\,[v_{k+1}(j)-w_{k+1}(j)],-\epsilon\Big\}\\
&\qquad\ge\min\Big\{[1-\alpha_{k+1}\,\mathbf{1}(j=J_{k+1})]\,\min\Big\{\prod_{\kappa=N}^{k}[1-\alpha_\kappa\,\mathbf{1}(j=J_\kappa)]\,[v_N(j)-w_N(j)],-\epsilon\Big\},-\epsilon\Big\}\\
&\qquad=\min\Big\{\prod_{\kappa=N}^{k+1}[1-\alpha_\kappa\,\mathbf{1}(j=J_\kappa)]\,[v_N(j)-w_N(j)],-\epsilon\Big\},
\end{aligned}$$

where the second inequality uses the induction assumption and the fact that $\alpha_{k+1} \in [0,1]$. The chain of inequalities above complete the induction argument and establish that (5) holds.

Lemma 3.3. in Bertsekas and Tsitsiklis [1996] shows that $\lim_{k\to\infty} \prod_{\kappa=N}^{k} [1 - \alpha_\kappa \mathbf{1}(j = J_\kappa)] = 0$ as long as (A3) holds. In this case, (5) implies that there exists a finite iteration number $\tilde{K}$ such that $\tilde{K} \geq N$ and $v_{k+1}(j) - w_{k+1}(j) \geq -\epsilon$ for all $k \geq \tilde{K}$. □

1.2 Convergence Proof for Algorithm 1

In this subsection, we prove Proposition 2 by using Lemmas 3 and 4. We let $\{w_k\}$ be generated by Algorithm 2 and choose an arbitrary $\epsilon > 0$. As we mention above, by Proposition 4.1 in Bertsekas and Tsitsiklis [1996], we have $\lim_{k\to\infty} w_k = \mathbb{E}\{\eta\}$ w.p.1. We now show that there exists a finite iteration number K w.p.1 such that $\|v_k - w_k\| \leq n\epsilon$ for all $k \geq K$. Since $\lim_{k\to\infty} w_k = \mathbb{E}\{\eta\}$ w.p.1 and ϵ is arbitrary, we obtain $\lim_{k\to\infty} v_k = \mathbb{E}\{\eta\}$ w.p.1. All statements in the proof are in w.p.1 sense.

We first use induction on j to show that there exists a finite iteration number $\tilde{K}$ such that $v_k(j) - w_k(j) \geq -n\epsilon$ for all $j \in \{1, \ldots, n\}$, $k \geq \tilde{K}$. By Lemma 3, there exists a finite iteration number $K(n)$ such that

$$v_{k+1}(n) - w_{k+1}(n) \geq \min\Big\{[1 - \alpha_k \mathbf{1}(n = J_k)]\,[v_k(n) - w_k(n)], -\epsilon\Big\}$$

for all $k \geq K(n)$, in which case, Lemma 4 implies that there exists a finite iteration number $\tilde{K}(n)$ such that $v_{k+1}(n) - w_{k+1}(n) \geq -\epsilon$ for all $k \geq \tilde{K}(n)$. Assuming that there exists a finite iteration number $\tilde{K}(j)$ such that $v_{k+1}(i) - w_{k+1}(i) \geq -(n-j+1)\epsilon$ for all $i \in \{j, \ldots, n\}$, $k \geq \tilde{K}(j)$, we now show that there exists a finite iteration number $\tilde{K}(j-1)$ such that $v_{k+1}(i) - w_{k+1}(i) \geq -(n-j+2)\epsilon$ for all $i \in \{j-1, \ldots, n\}$, $k \geq \tilde{K}(j-1)$. By Lemma 3, there exists a finite iteration number $K(j)$ such that $K(j) > \tilde{K}(j)$ and

$$\begin{aligned}
&v_{k+1}(j-1) - w_{k+1}(j-1) \\
&\qquad \geq \min\Big\{[1 - \alpha_k \mathbf{1}(j-1 = J_k)]\,[v_k(j-1) - w_k(j-1)], \\
&\qquad\qquad\qquad \min_{i\in\{j,\ldots,n\}} \Big\{[1 - \alpha_k \mathbf{1}(i = J_k)]\,[v_k(i) - w_k(i)]\Big\} - \epsilon, -\epsilon\Big\} \\
&\qquad \geq \min\Big\{[1 - \alpha_k \mathbf{1}(j-1 = J_k)]\,[v_k(j-1) - w_k(j-1)], -(n-j+1)\epsilon - \epsilon, -\epsilon\Big\}
\end{aligned}$$

for all $k \geq K(j)$, where the second inequality uses the fact that $v_k(i) - w_k(i) \geq -(n-j+1)\epsilon$ for all $i \in \{j, \ldots, n\}$, $k > \tilde{K}(j)$. In this inequality, we also assume that $K(j)$ is large enough so that $\alpha_k \in [0,1]$ for all $k \geq K(j)$. Therefore, we obtain

$$v_{k+1}(j-1) - w_{k+1}(j-1) \geq \min\Big\{[1 - \alpha_k \mathbf{1}(j-1 = J_k)]\,[v_k(j-1) - w_k(j-1)], -(n-j+2)\epsilon\Big\}$$

for all $k \geq K(j)$ so that Lemma 4 implies that there exists a finite iteration number $\tilde{K}(j-1)$ such that $\tilde{K}(j-1) \geq K(j)$ and $v_{k+1}(j-1) - w_{k+1}(j-1) \geq -(n-j+2)\epsilon$ for all $k \geq \tilde{K}(j-1)$. Therefore, we have $v_{k+1}(i) - w_{k+1}(i) \geq -(n-j+2)\epsilon$ for all $i \in \{j-1, \ldots, n\}$, $k \geq \tilde{K}(j-1)$. This completes the induction argument, and letting $\tilde{K} = \tilde{K}(1)$, we have $v_{k+1}(j) - w_{k+1}(j) \geq -n\epsilon$ for all $j \in \{1, \ldots, n\}$, $k \geq \tilde{K}$.

Using a similar argument, we can also show that there exists a finite iteration number $\tilde{K}'$ such that $v_{k+1}(j) - w_{k+1}(j) \leq n\epsilon$ for all $j \in \{1, \ldots, n\}$, $k \geq \tilde{K}'$. Letting $K = \max\{\tilde{K}, \tilde{K}'\} + 1$, we have $\|v_k - w_k\| \leq n\epsilon$ for all $k \geq K$ and this establishes Proposition 2.

2 Applications to the Q-Learning Algorithm

We consider discounted cost Markov decision problems with finite sets of states and actions. We denote the sets of states and actions respectively by $\{1, \ldots, n\}$ and $\mathcal{A}$. If the system is in state j and we take action a, then the system transitions to state s with probability $p_{js}(a)$ and we incur a finite cost of $g(j, a, s)$. The costs in the future time periods are discounted by a factor $\lambda \in [0, 1)$. For notational brevity, we assume that every action is admissible in every state. We are interested in finding a Markovian policy that minimizes the discounted total expected cost. In particular, a Markovian policy μ is a mapping from the set of states to the set of actions, prescribing which action to take in each possible state. Therefore, the states visited by the system under policy μ evolve according to the Markov chain that transitions from state j to state s with probability $p_{js}(\mu(j))$. If we let $\{X_t^\mu : t = 0, 1, \ldots\}$ be the states visited by this Markov chain, then the discounted total expected cost incurred by starting from state j and using policy μ can be written as

$$V^\mu(j) = \lim_{T \to \infty} \mathbb{E}\left\{ \sum_{t=0}^{T} \lambda^t \, g(X_t^\mu, \mu(X_t^\mu), X_{t+1}^\mu) \,|\, X_0^\mu = j \right\}.$$

Letting Θ be the set of all possible Markovian policies, the optimal policy μ^* satisfies $V^{\mu^*}(j) = \min_{\mu \in \Theta} V^\mu(j)$ for all $j \in \{1, \ldots, n\}$.

Bertsekas and Tsitsiklis [1996] show that the optimal policy can be found by computing the Q-factors $\{\tilde{Q}^a(j) : j = 1, \ldots, n, \ a \in \mathcal{A}\}$ through the optimality equation

$$\tilde{Q}^a(j) = \sum_{s=1}^{n} p_{js}(a) \Big\{ g(j, a, s) + \lambda \min\nolimits_{b \in \mathcal{A}} \tilde{Q}^b(s) \Big\}. \tag{6}$$

In this case, the optimal policy takes an action in the set $\operatorname{argmin}_{a \in \mathcal{A}} \tilde{Q}^a(j)$ whenever the state of the system is j. The Q-factor $\tilde{Q}^a(j)$ corresponding to state-action pair (j, a) can be interpreted as the discounted total expected cost that we incur when the initial state of the system is j and we immediately take action a and follow the optimal policy afterwards. Throughout the paper, we view the Q-factors as matrices taking values in $\Re^{n \times |\mathcal{A}|}$. For $\tilde{Q} \in \Re^{n \times |\mathcal{A}|}$, we use $\tilde{Q}^a \in \Re^n$ to denote the column of $\tilde{Q}$ corresponding to action a and $\tilde{Q}^a(j) \in \Re$ to denote the jth component of the vector $\tilde{Q}^a$. In other words, $\tilde{Q} \in \Re^{n \times |\mathcal{A}|}$ captures the Q-factors for all state-action pairs, whereas $\tilde{Q}^a \in \Re^n$ is the vector of Q-factors corresponding to action a.

The Q-learning algorithm is intended to solve the optimality equation in (6) through stochastic approximation. The algorithm starts with arbitrary Q-factor approximations $\{Q_1^a(j) : j = 1, \ldots, n, \ a \in \mathcal{A}\}$. At the kth iteration of the algorithm, we observe a state-action pair (J_k, A_k). Given that the system transitions to state S_k after taking action A_k in state J_k, we update the Q-factor approximation for the state-action pair (J_k, A_k) as

$$Q_{k+1}^{A_k}(J_k) = Q_k^{A_k}(J_k) + \alpha_k \, [g(J_k, A_k, S_k) + \lambda \, \min_{b \in \mathcal{A}} Q_k^b(S_k) - Q_k^{A_k}(J_k)]. \tag{7}$$

The Q-factor approximations for the other state-action pairs remain unchanged. If certain assumptions on the state-action pair (J_k, A_k) and the step size parameter α_k are satisfied, then it can be shown that we have $\lim_{k \to \infty} Q_k^a(j) = \tilde{Q}^a(j)$ w.p.1 for all $j \in \{1, \ldots, n\}$, $a \in \mathcal{A}$, where $\{\tilde{Q}^a(j) : j = 1, \ldots, n, \; a \in \mathcal{A}\}$ is the solution to the optimality equation in (6); see Section 5.6 in Bertsekas and Tsitsiklis [1996].

2.1 New Variant of the Q-Learning Algorithm

The Q-learning algorithm, in general, does not exploit the structural properties of the underlying Markov decision problem and may take a large number of iterations to provide a good policy. In this section, we investigate a variant of the Q-learning algorithm that is applicable to Markov decision problems where the Q-factors are known to satisfy the monotonicity property

$$\tilde{Q}^a(j) \le \tilde{Q}^a(j+1) \tag{8}$$

for all $j \in \{1, \ldots, n-1\}$, $a \in \mathcal{A}$. There is a variety of Markov decision problems in the queue admission, batch service, marketing, and aging and replacement settings, where the Q-factors satisfy the property above. In Section 3, we present two problems in the batch service setting. Furthermore, the next corollary to Theorem 6.11.6 in Puterman [1994] gives sufficient conditions under which the Q-factors satisfy the monotonicity property in (8).

Lemma 5 *Let $\{\tilde{Q}^a(j) : j = 1, \ldots, n, \; a \in \mathcal{A}\}$ be the solution to the optimality equation in* (6)*. Assume that $\sum_{s=1}^n p_{js}(a) \, g(j, a, s)$ is increasing in j for all $a \in \mathcal{A}$ and $\sum_{s=s'}^n p_{js}(a)$ is increasing in j for all $s' \in \{1, \ldots, n\}$, $a \in \mathcal{A}$. Then, we have $\tilde{Q}^a(j) \le \tilde{Q}^a(j+1)$ for all $j \in \{1, \ldots, n-1\}$, $a \in \mathcal{A}$.*

Proof Letting $V(j) = \min_{a \in \mathcal{A}} \tilde{Q}^a(j)$, the proof first shows that $V(j)$ is increasing in j. After this, the result follows by a simple substitution in (6). We provide the details in the appendix. □

We note that one can often check the assumptions of Lemma 5 by using the problem structure without having access to the actual values of the transition probabilities and costs. For example, for a queue admission application where the state is the number of entities in the system and the action is the service rate, the assumption that $\sum_{s=1}^n p_{js}(a) \, g(j, a, s)$ is increasing in j implies that the expected one period cost is increasing in the number of entities in the system. This is certainly the case when the cost components of interest are the waiting cost of keeping the entities in the system and the service cost of running the server. On the other hand, the assumption that $\sum_{s=s'}^n p_{js}(a)$ is increasing in j implies that the probability that the number of entities in the system at the next time period exceeds s' increases as the number of entities in the system at the current time period increases. This assumption is satisfied by essentially any plausible arrival process.

The basic idea behind our variant of the Q-learning algorithm is to use projections to impose the monotonicity property in (8) on the Q-factor approximations. Our goal is to improve the empirical performance of the Q-learning algorithm by imposing the structural properties of the Q-factors on the Q-factor approximations. In particular, we study the following variant of the Q-learning algorithm.

Algorithm 3

(1) Choose $Q^a_1 \in \mathcal{V}(-C, C)$ for all $a \in \mathcal{A}$, where C is a sufficiently large scalar. Set $k = 1$.

(2) Observe a state-action pair (J_k, A_k) and a subsequent state S_k.

(3) Letting $\alpha_k \geq 0$ be a step size parameter, set

$$R^a_k(j) = Q^a_k(j) + \alpha_k \, \mathbf{1}(j = J_k, a = A_k) \, [g(j, a, S_k) + \lambda \, \min_{b \in \mathcal{A}} Q^b_k(S_k) - Q^a_k(j)] \tag{9}$$

for all $j \in \{1, \ldots, n\}$, $a \in \mathcal{A}$.

(4) Set $Q^a_{k+1} = \Pi^{-C,C}_{R^a_k}$ for all $a \in \mathcal{A}$, where $\Pi^{-C,C}_{R^a_k}$ is as in (4).

(5) Increase k by 1 and go to Step 2.

The updating procedure in Step 3 of Algorithm 3 is the same as the one in (7). In Step 4, we project R^a_k onto $\mathcal{V}(-C, C)$ so that Q^a_{k+1} satisfies the monotonicity property in (8). Our goal is to show that $\lim_{k \to \infty} Q^a_k(j) = \tilde{Q}^a(j)$ w.p.1 for all $j \in \{1, \ldots, n\}$, $a \in \mathcal{A}$. Letting $\mathcal{G}_k$ be the history of Algorithm 3 captured by the random variables $\{J_1, \ldots, J_{k-1}, A_1, \ldots, A_{k-1}, S_1, \ldots, S_{k-1}\}$, we assume that the following statements hold.

(B1) We have $\mathbb{P}\{S_k = s \,|\, \mathcal{G}_k, J_k, A_k\} = p_{J_k s}(A_k)$ w.p.1 for all $k = 1, 2, \ldots$.

(B2) Letting $\{\tilde{Q}^a(j) : j = 1, \ldots, n, \; a \in \mathcal{A}\}$ be the solution to the optimality equation in (6), we have $\tilde{Q}^a \in \mathcal{V}(-C, C)$ for all $a \in \mathcal{A}$, where C is as in Steps 1 and 4 of Algorithm 3.

(B3) The step size parameter α_k is positive and is a deterministic function of $\mathcal{G}_k$ for all $k = 1, 2, \ldots$.

(B4) The random variables J_k and A_k, and the step size parameter α_k satisfy $\sum_{k=1}^{\infty} [\mathbf{1}(j = J_k, a = A_k) \, \alpha_k] = \infty$ w.p.1 and $\sum_{k=1}^{\infty} \mathbb{E}\{\mathbf{1}(j = J_k, a = A_k) \, \alpha_k^2\} < \infty$ for all $j \in \{1, \ldots, n\}$, $a \in \mathcal{A}$.

By (B1), the subsequent state S_k is sampled according to the transition probabilities of the Markov decision problem. (B2) assumes that the Q-factors satisfy the monotonicity property in (8) and they are uniformly bounded by C. Since the uniform bound C is also used in Algorithm 3, one needs to know or estimate this bound to be able to apply Algorithm 3. If we know the problem data, then letting $C = \max_{j, s \in \{1, \ldots, n\}, a \in \mathcal{A}} |g(j, a, s)|/(1 - \lambda)$ suffices. Otherwise, we need to estimate C through some insight into the problem at hand, but we can always choose C quite large as our computational experience indicates that choosing C too large does not affect the performance of Algorithm 3 in an undesirable fashion. One can construct simple sequences $\{J_k\}$, $\{A_k\}$ and $\{\alpha_k\}$ that satisfy (B3) and (B4). For example, sampling (J_k, A_k) uniformly over $\{1, \ldots, n\} \times \mathcal{A}$ and letting $\alpha_k = 1/k$ suffice.

It is interesting to note that Proposition 2 does not immediately provide a convergence result for Algorithm 3. In particular, if we compare (1) in Algorithm 1 with (9) in Algorithm 3, then we can see that the index j in Algorithm 1 corresponds to the state-action pair (j, a) in Algorithm 3. Furthermore, the random variables J_k and $\eta_k(J_k)$ in Algorithm 1 respectively correspond to the state action pair (J_k, A_k) and the random variable $g(J_k, A_k, S_k) + \lambda \, \min_{b \in \mathcal{A}} Q^b_k(S_k)$ in Algorithm 3. (A1), which is needed for Proposition 2 to hold, requires that the expectation of $\eta_k(J_k)$ conditional on $\mathcal{F}_k$ and J_k is equal to the J_kth component of a fixed vector $\bar{\eta}$. On the other hand, noting that the Q-factor approximations $\{Q^a_k(j) : j = 1, \ldots, n,\ a \in \mathcal{A}\}$ keep changing over the iterations, it is unrealistic to expect that the expectation of $g(J_k, A_k, S_k) + \lambda \min_{b \in \mathcal{A}} Q^b_k(S_k)$ conditional on $\mathcal{G}_k$ and (J_k, A_k) would be equal to the (J_k, A_k)th component of a fixed matrix.

The next proposition gives a convergence result for Algorithm 3. Its proof is divided between the next three subsections. Section 2.2 shows an order preserving property of the projection operator in Step 4 of Algorithm 3. Section 2.3 follows a standard argument to write Algorithm 3 as an alternative stochastic approximation method. Finally, Section 2.4 shows the convergence of Algorithm 3 by establishing the convergence of the alternative stochastic approximation method. The order preserving property that we show in the Section 2.2 becomes useful when establishing the convergence of the alternative stochastic approximation method.

Proposition 6 *Let $\{Q^a_k(j) : j = 1, \ldots, n,\ a \in \mathcal{A}\}$ be generated by Algorithm* 3 *and assume that* (B1)-(B4) *hold. Then, we have $\lim_{k \to \infty} Q^a_k(j) = \tilde{Q}^a(j)$ w.p.1 for all $j \in \{1, \ldots, n\}$, $a \in \mathcal{A}$, where $\{\tilde{Q}^a(j) : j = 1, \ldots, n,\ a \in \mathcal{A}\}$ is the solution to the optimality equation in* (6).

2.2 Order Preserving Property

In this section, we show an order preserving property of the projection operator that we use in Algorithm 3. An important implication of this property will be the following. Assume that we run Algorithm 3 twice, once starting with the initial Q-factor approximations $\{Q^a_1(j) : j = 1, \ldots, n,\ a \in \mathcal{A}\}$ and once starting with the initial Q-factor approximations $\{\hat{Q}^a_1(j) : j = 1, \ldots, n,\ a \in \mathcal{A}\}$. Other than the difference in the initial conditions, the two runs of Algorithm 3 are subjected to the same sequence of random variables. Letting $\{Q^a_k(j) : j = 1, \ldots, n,\ a \in \mathcal{A}\}$ and $\{\hat{Q}^a_k(j) : j = 1, \ldots, n,\ a \in \mathcal{A}\}$ be the Q-factor approximations obtained at the kth iteration of the two runs, the order preserving property ensures that if we have $Q^a_1 \leq \hat{Q}^a_1$ for all $a \in \mathcal{A}$ in the initial conditions, then we also have $Q^a_k \leq \hat{Q}^a_k$ w.p.1 for all $a \in \mathcal{A}$, $k = 1, 2, \ldots$.

Lemma 7 *Let $v \in \mathcal{V}(L, U)$, $\hat{v} \in \mathcal{V}(\hat{L}, \hat{U})$ and $J \in \{1, \ldots, n\}$. Assume that z and $\hat{z}$ are obtained respectively from v and $\hat{v}$ by perturbing only the Jth component. If $v \leq \hat{v}$, $z \leq \hat{z}$, $L \leq \hat{L}$ and $U \leq \hat{U}$, then we have $\Pi^{L,U}_z \leq \Pi^{\hat{L},\hat{U}}_{\hat{z}}$.*

Proof The proof follows by keeping track of how $\Pi^{L,U}_z$ and $\Pi^{\hat{L},\hat{U}}_{\hat{z}}$ are computed through (3) and (4). We provide the details in the appendix. $\Box$

If the two sets of initial Q-factor approximations $\{Q^a_1(j) : j = 1, \ldots, n,\ a \in \mathcal{A}\}$ and $\{\hat{Q}^a_1(j) : j = 1, \ldots, n,\ a \in \mathcal{A}\}$ satisfy $Q^a_1 \leq \hat{Q}^a_1$ for all $a \in \mathcal{A}$ and $\alpha_k \in [0, 1]$ for all $k = 1, 2, \ldots$, then (9) in Step 3 of Algorithm 3 implies that

$$\begin{aligned} R^{A_1}_1(J_1) &= [1 - \alpha_1]\, Q^{A_1}_1(J_1) + \alpha_1 \left[g(J_1, A_1, S_1) + \lambda \min\nolimits_{b \in \mathcal{A}} Q^b_1(S_1)\right] \\ &\leq [1 - \alpha_1]\, \hat{Q}^{A_1}_1(J_1) + \alpha_1 \left[g(J_1, A_1, S_1) + \lambda \min\nolimits_{b \in \mathcal{A}} \hat{Q}^b_1(S_1)\right] = \hat{R}^{A_1}_1(J_1) \end{aligned}$$

and $R_1^{A_1}(j) = Q_1^{A_1}(j) \leq \hat{Q}_1^{A_1}(j) = \hat{R}_1^{A_1}(j)$ w.p.1 for all $j \in \{1, \ldots, n\} \setminus \{J_1\}$. Therefore, we have $R_1^{A_1} \leq \hat{R}_1^{A_1}$. Similarly, we also have $R_1^a = Q_1^a \leq \hat{Q}_1^a = \hat{R}_1^a$ w.p.1 for all $a \in \mathcal{A} \setminus \{A_1\}$. In this case, Lemma 7 implies that $Q_2^a = \Pi_{R_1^a}^{-C,C} \leq \Pi_{\hat{R}_1^a}^{-C,C} = \hat{Q}_2^a$ w.p.1 for all $a \in \mathcal{A}$. Continuing in the same fashion for the subsequent iterations, we obtain $Q_k^a \leq \hat{Q}_k^a$ w.p.1 for all $a \in \mathcal{A}$, $k = 1, 2, \ldots$.

2.3 The Q-Learning Algorithm as a Stochastic Approximation Method

It is possible to write Algorithm 3 in an alternative fashion by following a standard argument that can be found in Section 5.6 in Bertsekas and Tsitsiklis [1996]. For this purpose, we define the operator Γ on $\Re^{n \times |\mathcal{A}|}$ and the random variable $\omega_k \in \Re^{n \times |\mathcal{A}|}$ as

$$[\Gamma Q]^a(j) = \sum_{s=1}^{n} p_{js}(a) \Big\{ g(j, a, s) + \lambda \min\nolimits_{b \in \mathcal{A}} Q^b(s) \Big\} \tag{10}$$

$$\omega_k^a(j) = g(j, a, S_k) + \lambda \min\nolimits_{b \in \mathcal{A}} Q_k^b(S_k) - \sum_{s=1}^{n} p_{js}(a) \Big\{ g(j, a, s) + \lambda \min\nolimits_{b \in \mathcal{A}} Q_k^b(s) \Big\}, \tag{11}$$

where $Q = \{Q^a(j) : j = 1, \ldots, n, \ a \in \mathcal{A}\}$ is a matrix taking values in $\Re^{n \times |\mathcal{A}|}$ and we use $[\Gamma Q]^a(j)$ to denote the (j, a)th component of ΓQ. With this definition of Γ and ω_k, we have $[\Gamma Q_k]^a(j) + \omega_k^a(j) = g(j, a, S_k) + \lambda \min_{b \in \mathcal{A}} Q_k^b(S_k)$ and Step 3 of Algorithm 3 can be written as

$$R_k^a(j) = Q_k^a(j) + \alpha_k \, \mathbf{1}(j = J_k, a = A_k) \, [[\Gamma Q_k]^a(j) + \omega_k^a(j) - Q_k^a(j)] \tag{12}$$

for all $j \in \{1, \ldots, n\}$, $a \in \mathcal{A}$.

The operator Γ and the random variable ω_k have important properties that become useful when establishing the convergence of Algorithm 3. To begin with, (6) and (10) imply that $\tilde{Q} = \Gamma \tilde{Q}$ so that the solution $\tilde{Q} = \{\tilde{Q}^a(j) : j = 1, \ldots, n, \ a \in \mathcal{A}\}$ to the optimality equation in (6) is a fixed point of the operator Γ. Furthermore, the operator Γ corresponds to the so called dynamic programming operator and the discussion that follows (5.63) in Bertsekas and Tsitsiklis [1996] indicates that this operator is a contraction mapping with respect to the max norm with contraction factor λ. In other words, if we define the max norm $\| \cdot \|$ on $\Re^{n \times |\mathcal{A}|}$ as $\|Q\| = \max_{a \in \mathcal{A}} \|Q^a\|$, then we have

$$\|\Gamma Q - \tilde{Q}\| \leq \lambda \, \|Q - \tilde{Q}\| \tag{13}$$

for all $Q = \{Q^a(j) : j = 1, \ldots, n, \ a \in \mathcal{A}\} \in \Re^{n \times |\mathcal{A}|}$. Finally, it is possible to interpret the random variable $\omega_k^{A_k}(J_k)$ as an error term with zero expectation. To see this, we use (B1) and (11) to obtain

$$\mathbb{E}\{\omega_k^{A_k}(J_k) \,|\, \mathcal{G}_k, J_k, A_k\} = \sum_{s=1}^{n} \mathbb{P}\{S_k = s \,|\, \mathcal{G}_k, J_k, A_k\} \Big\{ g(J_k, A_k, s) + \lambda \min\nolimits_{b \in \mathcal{A}} Q_k^b(s) \Big\} \\ - \sum_{s=1}^{n} p_{J_k s}(A_k) \Big\{ g(J_k, A_k, s) + \lambda \min\nolimits_{b \in \mathcal{A}} Q_k^b(s) \Big\} = 0. \tag{14}$$

2.4 Convergence Proof for Algorithm 3

We are now ready to prove Proposition 6. The proof is an extension of the proof of Proposition 4.4 in Bertsekas and Tsitsiklis [1996] and it is designed to deal with the effects of the projection operator in

Step 4 of Algorithm 3. All statements in the proof should be understood in w.p.1 sense. By (B4), there exists a finite iteration number N such that $\alpha_k \in [0,1]$ for all $k \geq N$. Therefore, we assume throughout the proof that $\alpha_k \in [0,1]$ for all $k = 1, 2, \ldots$ without loss of generality.

By Step 4 of Algorithm 3, we have $Q^a_k \in \mathcal{V}(-C, C)$ for all $a \in \mathcal{A}$, $k = 1, 2, \ldots$. Since $\tilde{Q}^a \in \mathcal{V}(-C, C)$ for all $a \in \mathcal{A}$ by (B2), we have $\|Q_k - \tilde{Q}\| \leq \|Q_k\| + \|\tilde{Q}\| \leq 2\,C$ for all $k = 1, 2, \ldots$. Noting that we have $\lambda \in [0, 1)$, we choose $\epsilon > 0$ with $\lambda + \epsilon < 1$. Letting $D_0 = 2\,C$, we define the sequence $\{D_t\}$ through $D_{t+1} = [\lambda + \epsilon]\, D_t$. We have $\|Q_k - \tilde{Q}\| \leq D_0$ for all $k = 1, 2, \ldots$. To show the result by induction, we assume that there exists a finite iteration number $k(t)$ such that $\|Q_k - \tilde{Q}\| \leq D_t$ for all $k \geq k(t)$. We show that this assumption implies that there exists a finite iteration number $k(t+1)$ such that $\|Q_k - \tilde{Q}\| \leq D_{t+1}$ for all $k \geq k(t+1)$. In this case, since we have $\lim_{t\to\infty} D_t = 0$, we obtain $\lim_{k\to\infty} \|Q_k - \tilde{Q}\| = 0$.

We fix $a \in \mathcal{A}$ and let $e \in \Re^n$ be the vector whose components are all ones. For $k \geq k(t)$, we start with $v^a_{k(t)} = \tilde{Q}^a - D_t\, e$ and define the sequence of vectors $\{v^a_k\}$ as

$$z^a_k(j) = v^a_k(j) + \mathbf{1}(a = A_k)\, \alpha_k\, \mathbf{1}(j = J_k)\, [\tilde{Q}^a(j) - \lambda\, D_t + \omega^a_k(j) - v^a_k(j)] \tag{15}$$

for all $j \in \{1, \ldots, n\}$ and $v^a_{k+1} = \Pi^{-3C,C}_{z^a_k}$. Identifying $\mathbf{1}(a = A_k)\, \alpha_k$ and $\tilde{Q}^a(j) - \lambda\, D_t + \omega^a_k(j)$ in (15) respectively with α_k and $\eta_k(j)$ in Step 2 of Algorithm 1, it is easy to see that the sequence of vectors $\{v^a_k\}$ are essentially generated by Algorithm 1. We now verify the assumptions in (A1)-(A3) so that we can use Proposition 2 to give a convergence result for the sequence of vectors $\{v^a_k\}$.

Since $\tilde{Q}^a \in \mathcal{V}(-C, C)$ by (B2) and $D_t \leq D_0 = 2\,C$, we have $-3\,C \leq -C - \lambda\, D_t \leq \tilde{Q}^a(j) - \lambda\, D_t \leq \tilde{Q}^a(j) \leq C$ for all $j \in \{1, \ldots, n\}$, which implies that $\tilde{Q}^a - \lambda\, D_t\, e \in \mathcal{V}(-3\,C, C)$. By (14), we have

$$\mathbb{E}\{\tilde{Q}^{A^k}(J_k) - \lambda\, D_t + \omega^{A_k}_k(J_k) \,|\, \mathcal{G}_k, J_k, A_k\} = \tilde{Q}^{A^k}(J_k) - \lambda\, D_t.$$

Letting G be such that $|g(j, a, s)| \leq G < \infty$ for all $j, s \in \{1, \ldots, n\}$, $a \in \mathcal{A}$, we have $|\tilde{Q}^a(j) - \lambda\, D_t + \omega^a_k(j)| \leq C + 2\,C + 2\,[G + C]$ by (11). Therefore, if we identify $-3C$, C and $\tilde{Q}^a - \lambda\, D_t\, e$ respectively with L, U and $\mathbb{E}\{\eta\}$ in (A1), then (A1) is satisfied by the updating procedure in (15). By (B3) and (B4), the other assumptions in (A2) and (A3) that are needed for Proposition 2 to hold are satisfied. Therefore, we have $\lim_{k\to\infty} \|v^a_k - \tilde{Q}^a + \lambda\, D_t\, e\| = 0$ by Proposition 2.

We next use induction on k to show that $v^a_k \leq Q^a_k$ for all $k \geq k(t)$. Since we have $v^a_{k(t)} = \tilde{Q}^a - D_t\, e$ and $\|Q_{k(t)} - \tilde{Q}\| \leq D_t$, the result holds for $k = k(t)$. Assuming that $v^a_k \leq Q^a_k$, we have

$$\begin{aligned} R^a_k(j) &= [1 - \alpha_k\, \mathbf{1}(j = J_k, a = A_k)]\, Q^a_k(j) + \alpha_k\, \mathbf{1}(j = J_k, a = A_k)\, [[\Gamma Q_k]^a(j) + \omega^a_k(j)] \\ &\geq [1 - \alpha_k\, \mathbf{1}(j = J_k, a = A_k)]\, v^a_k(j) + \alpha_k\, \mathbf{1}(j = J_k, a = A_k)\, [\tilde{Q}^a(j) - \lambda\, \|Q_k - \tilde{Q}\| + \omega^a_k(j)] \\ &\geq [1 - \alpha_k\, \mathbf{1}(j = J_k, a = A_k)]\, v^a_k(j) + \alpha_k\, \mathbf{1}(j = J_k, a = A_k)\, [\tilde{Q}^a(j) - \lambda\, D_t + \omega^a_k(j)] = z^a_k(j) \end{aligned}$$

for all $j \in \{1, \ldots, n\}$, where the first equality follows from (12), the first inequality follows from (13) and the second inequality follows from the assumption that $\|Q_k - \tilde{Q}\| \leq D_t$ for all $k \geq k(t)$. Thus, we have $z^a_k \leq R^a_k$. We have $v^a_k \in \mathcal{V}(-3\,C, C)$ by definition and z^a_k is obtained from v^a_k by perturbing only the J_kth component. Similarly, we have $Q^a_k \in \mathcal{V}(-C, C)$ by definition and R^a_k is obtained from Q^a_k

by perturbing only the J_kth component. In this case, Lemma 7 and the fact that $z_k^a \leq R_k^a$ imply that $v_{k+1}^a = \Pi_{z_k^a}^{-3C,C} \leq \Pi_{R_k^a}^{-C,C} = Q_{k+1}^a$. Therefore, we have $v_k^a \leq Q_k^a$ for all $k \geq k(t)$.

For $k \geq k(t)$, we can also start with $\hat{v}_{k(t)}^a = \tilde{Q}^a + D_t\, e$ and define the sequence of vectors $\{\hat{v}_k^a\}$ as

$$\hat{z}_k^a(j) = \hat{v}_k^a(j) + \mathbf{1}(a = A_k)\, \alpha_k\, \mathbf{1}(j = J_k)\, [\tilde{Q}^a(j) + \lambda\, D_t + \omega_k^a(j) - \hat{v}_k^a(j)]$$

for all $j \in \{1, \ldots, n\}$ and $\hat{v}_{k+1}^a = \Pi_{\hat{z}_k^a}^{-C,3C}$. Following the same line of reasoning in the last three paragraphs, it is possible to show that $\lim_{k\to\infty} \|\hat{v}_k^a - \tilde{Q}^a - \lambda\, D_t\, e\| = 0$ and $\hat{v}_k^a \geq Q_k^a$ for all $k \geq k(t)$.

Since $\lim_{k\to\infty} \|v_k^a - \tilde{Q}^a + \lambda\, D_t\, e\| = 0$ and $\lim_{k\to\infty} \|\hat{v}_k^a - \tilde{Q}^a - \lambda\, D_t\, e\| = 0$, there exists a finite iteration number $k(t,a)$ such that $k(t,a) \geq k(t)$, and $v_k^a - \tilde{Q}^a + \lambda\, D_t\, e \geq -\epsilon\, D_t\, e$ and $\hat{v}_k^a - \tilde{Q}^a - \lambda\, D_t\, e \leq \epsilon\, D_t\, e$ for all $k \geq k(t,a)$. In this case, since $v_k^a \leq Q_k^a \leq \hat{v}_k^a$ for all $k \geq k(t)$, we have $-D_{t+1}\, e = -[\lambda + \epsilon]\, D_t\, e \leq v_k^a - \tilde{Q}^a \leq Q_k^a - \tilde{Q}^a \leq \hat{v}_k^a - \tilde{Q}^a \leq [\lambda + \epsilon]\, D_t\, e = D_{t+1}\, e$ for all $k \geq k(t,a)$. Letting $k(t+1) = \max_{a \in \mathcal{A}} k(t,a)$, we have $\|Q_k - \tilde{Q}\| \leq D_{t+1}$ for all $k \geq k(t+1)$. This completes the proof of Proposition 6.

3 Computational Experiments

This section compares the performances of several versions of the Q-learning algorithm on two batch service problems. Our goal is to demonstrate how much the empirical performance of the Q-learning algorithm can be improved by exploiting the monotonicity property of the Q-factors.

3.1 Batch Service Problem with a Single Product

We have a service station with capacity κ to serve the products that arrive randomly over time. The arriving products queue for service and we incur a cost of h per waiting product per time period. At any time period, we can decide to fill the service station up to its capacity and serve the products, in which case we incur a fixed cost of ϕ and the served products leave the system. The product arrivals at each time period have a geometric distribution with parameter ρ. We assume that we cannot have more than n waiting products in the system and the products that arrive into a full system are lost. We use the number of waiting products as the state variable so that the state space is $\{0, \ldots, n\}$. The set of actions is $\{0, 1\}$ with the interpretation that 0 and 1 respectively correspond to not running and running the service station. This problem has been studied extensively and its Q-factors satisfy the monotonicity property in (8); see, for example, Ignall and Kolesar [1972], Kosten [1973], Deb and Serfozo [1973], Ignall and Kolesar [1974] and Papadaki and Powell [2002].

We compare three versions of the Q-learning algorithm. The first version exactly corresponds to Algorithm 3. The second version is the standard version of the Q-learning algorithm. This version does not use projections at all and simply sets $Q_{k+1}^a = R_k^a$ for all $a \in \mathcal{A}$ in Step 4 of Algorithm 3. In other words, the standard version of the Q-learning algorithm follows Steps 1, 2, 3 and 5 as they are stated in Algorithm 3 and replaces Step 4 of Algorithm 3 with $Q_{k+1}^a = R_k^a$ for all $a \in \mathcal{A}$. The third version is an alternative version of Algorithm 3 that uses projections with respect to the Euclidean norm. This alternative version follows Steps 1, 2, 3 and 5 as they are stated in Algorithm 3 and replaces Step 4 of Algorithm 3 with $Q_{k+1}^a = \operatorname{argmin}_{w \in \mathcal{V}(L,U)} \|R_k^a - w\|_2$ for all $a \in \mathcal{A}$, where $\|\cdot\|_2$ is the Euclidean norm on $\Re^n$. We refer to Algorithm 3, the standard version of the Q-learning algorithm and the alternative version of Algorithm 3 respectively as A3, SQ and EQ.

We run A3, SQ and EQ for 10,000 iterations. We sample J_k, A_k and S_k by following the greedy policy obtained at iteration k. The greedy policy obtained at iteration k takes an action in the set $\text{argmin}_{a \in \mathcal{A}} Q_k^a(j)$ when the system is in state j. Starting with an arbitrary initial state, given that the system is in state J_k at iteration k, we set $A_k = \text{argmin}_{a \in \mathcal{A}} Q_k^a(J_k)$, breaking ties arbitrarily. We sample S_k such that $\mathbb{P}\{S_k = s \,|\, J_k, A_k\} = p_{J_k s}(A_k)$ and S_k is independent of the states and actions $\{J_1, \ldots, J_{k-1}, A_1, \ldots, A_{k-1}, S_1, \ldots, S_{k-1}\}$ that are observed up to iteration k. We set $J_{k+1} = S_k$ at iteration $k+1$ and continue in a similar fashion for the subsequent iterations. At each iteration, we stop following the greedy policy with probability 0.1 and sample (J_k, A_k) uniformly over $\{0, \ldots, n\} \times \{0, 1\}$. This ensures that (B4) is satisfied. We make 100 runs to eliminate the effect of sampling noise and report the average of the results. We use common random numbers when running A3, SQ and EQ.

We vary the values of κ, n, ϕ, ρ and λ to obtain different test problems. The value of the holding cost is fixed at 1 throughout. Figures 1 and 2 respectively show the performances of the greedy policies obtained after 4,000 and 10,000 iterations for different test problems. We label the test problems by $(\kappa, n, \phi, \rho, \lambda)$ on the horizontal axis. Letting $C_k(j)$ be the discounted total expected cost that is incurred by the greedy policy obtained after k iterations when the initial state of the system is j and $V^*(j)$ be the discounted total expected cost that is incurred by the optimal policy when the initial state of the system is j, the performance measure that we use in Figure 1 is

$$\max_{j \in \{0, \ldots, n\}} 100 \, \frac{C_{4,000}(j) - V^*(j)}{V^*(j)}.$$

This performance measure gives a feel for the worst percent penalty that is incurred by using the greedy policy obtained after 4,000 iterations instead of the optimal policy. We compute $\{V^*(j) : j = 0, \ldots, n\}$ by solving the optimality equation for the problem. We use the same performance measure in Figure 2, but we focus on the greedy policy obtained after 10,000 iterations.

The results indicate that both A3 and EQ significantly improve the performance of SQ and the performance gap is more noticeable when n is relatively large. Therefore, exploiting the monotonicity property appears to be particularly helpful when we have to estimate a relatively large number of Q-factors. The greedy policies obtained by A3 after 4,000 iterations perform noticeably better than the greedy policies obtained by EQ, but EQ catches up with A3 after 10,000 iterations for many test problems. Nevertheless, for all of the test problems with $\lambda = 0.99$, A3 performs better than EQ. It is widely known that discounted cost Markov decision problems become computationally more difficult as the discount factor gets close to one and it is encouraging A3 performs better than EQ when the discount factor is large. Figure 3 plots $\max_{j \in \{0, \ldots, n\}} 100 \, [C_k(j) - V^*(j)]/V^*(j)$ for test problem $(200, 300, 200, 0.1, 0.90)$ as a function of the iteration counter k. The performance gap between A3 and EQ is significant during the early iterations, but EQ catches up later.

The results are encouraging as they indicate that A3 may be a viable alternative to SQ and EQ especially for online learning settings where fast response is crucial. The gap between the performances of A3 and SQ can be quite significant for some test problems, indicating that explicitly exploiting

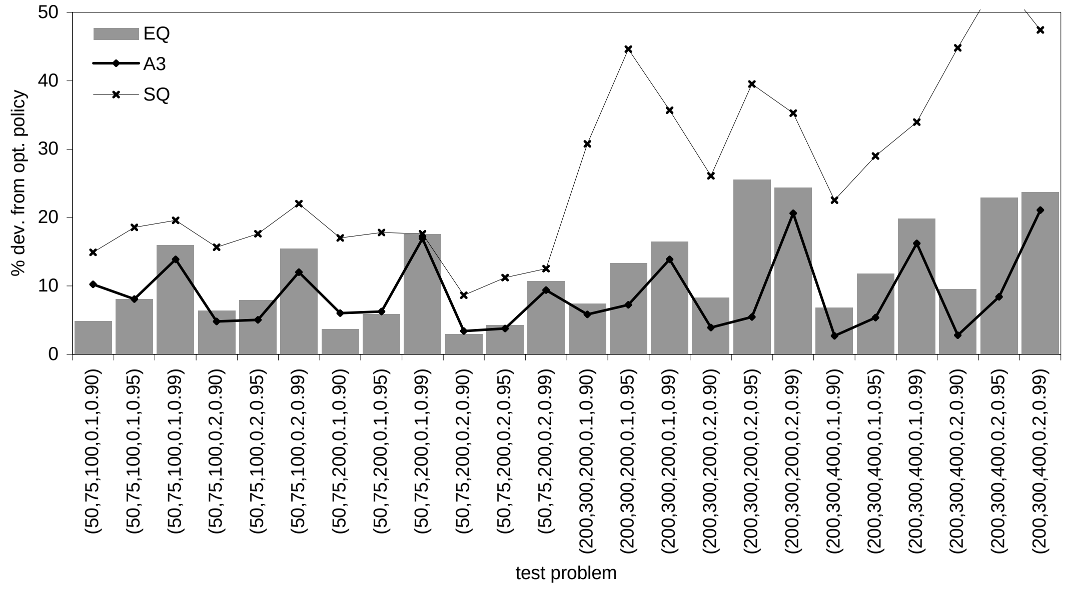

Figure 1: Performances of the greedy policies obtained by A3, SQ and EQ after 4,000 iterations.

the known monotonicity property of the Q-factors can significantly improve the performance of the Q-learning algorithm. This type of behavior holds over a wide range of problem parameters for our problem class, but we still caution the reader that these are empirical results and one should consider carrying out numerical experiments before generalizing them to other problem classes.

3.2 Batch Service Problem with Two Products

We have a service station that can serve two types of products. We refer to the two types of products as type 1 and type 2. The capacity of the service station is κ and we incur a fixed cost of ϕ every time we run the service station. The holding costs for the products of type 1 and type 2 are respectively h_1 and h_2. We assume that $h_1 \geq h_2$ so that it is desirable to serve the products of type 1 first when the number of products in the system exceeds the capacity of the service station. The product arrivals of type 1 and type 2 at each time period have geometric distributions respectively with parameters ρ_1 and ρ_2. The arrival processes for the two products are independent. We assume that we cannot have more than n_1 products of type 1 and n_2 products of type 2 in the system. We use $(j, i) \in \{0, \ldots, n_1\} \times \{0, \ldots, n_2\}$ as the state space. The set of actions is $\{0, 1\}$ with the same interpretation as in the previous subsection. Since $h_1 \geq h_2$, if we decide to run the service station, then we fill it up to its capacity with products of type 1 first. If there is still available capacity, then we fill it with products of type 2.

This problem is studied in Papadaki and Powell [2003] in the finite horizon setting. Following the approach in Papadaki and Powell [2003], it is possible to show that the Q-factors for this problem satisfy

$$Q^a(j,i) \leq Q^a(j+1,i) \qquad \text{for all } j \in \{0,\ldots,n_1-1\},\ i \in \{0,\ldots,n_2\},\ a \in \{0,1\}$$
$$Q^a(j,i) \leq Q^a(j,i+1) \qquad \text{for all } j \in \{0,\ldots,n_1\},\ i \in \{0,\ldots,n_2-1\},\ a \in \{0,1\}.$$

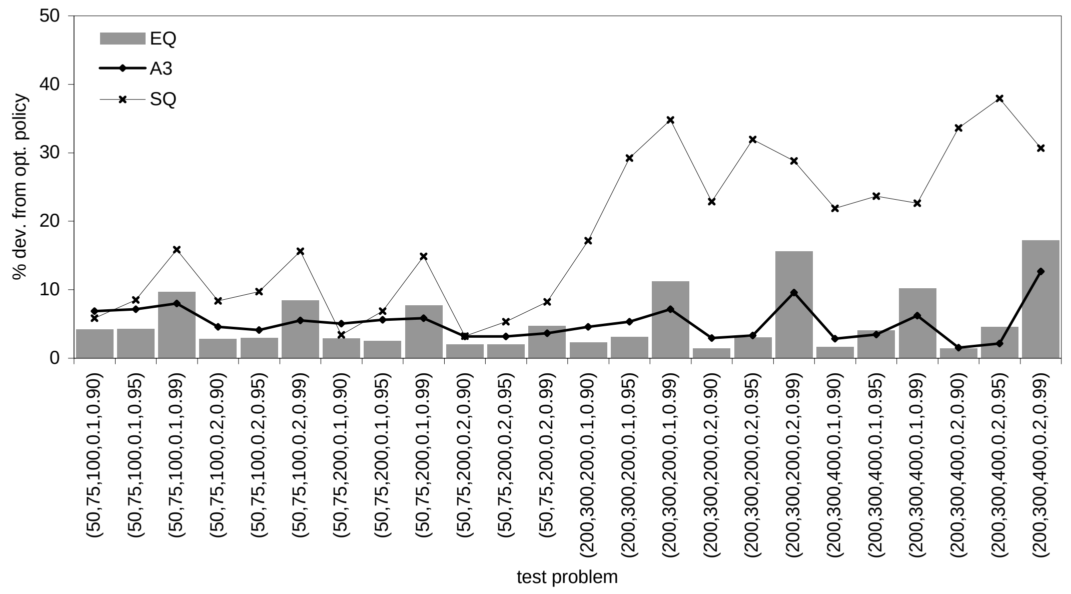

Figure 2: Performances of the greedy policies obtained by A3, SQ and EQ after 10,000 iterations.

A3 and EQ cannot impose both of these monotonicity properties on the Q-factor approximations and we arbitrarily choose to impose the first property. In particular, letting $\{Q_k^a(j,i) : j = 0,\ldots,n_1,\ i = 0,\ldots,n_2,\ a = 0,1\}$ be the Q-factor approximations at iteration k, $Q_k^a(i)$ be the vector $(Q_k^a(0,i),\ldots,$ $Q_k^a(n_1,i))$ and $\mathcal{V}(-C,C)$ be the set $\{v \in \Re^{n_1+1} : -C \leq v(0) \leq \ldots \leq v(n_1) \leq C\}$ for an appropriate scalar C, we project $Q_k^a(i)$ onto $\mathcal{V}(-C,C)$ for all $i \in \{0,\ldots,n_2\}$, $a \in \{0,1\}$.

Figure 4 shows the performances of the greedy policies obtained after 5,000,000 iterations. The performance measure that we use in this figure is the same as the ones that we use in Figures 1 and 2. The values of the holding costs for the two products are fixed at 1 and 0.5 throughout. The system capacities and the arrival probability distributions for the two products are the same. We label the test problems by $(\kappa, n, \phi, \rho, \lambda)$ on the horizontal axis, where n is the common value for n_1 and n_2, and ρ is the common value for ρ_1 and ρ_2.

The results indicate that both A3 and EQ improve the performance of SQ significantly. The gap in the performance becomes especially large when the discount factor is close to one. As mentioned above, discounted cost Markov decision problems become computationally more difficult as the discount factor gets close to one and it is encouraging that A3 and EQ significantly improve on SQ when the discount factor is large. It is also interesting to note that A3 appears to have a small but noticeable advantage over EQ for most of the test problems.

4 Conclusions

We analyzed a stochastic approximation method that is useful for estimating the expectation of a random vector when the expectation is known to have increasing components. We used this stochastic approximation method to establish the convergence of a variant of the Q-learning algorithm that is

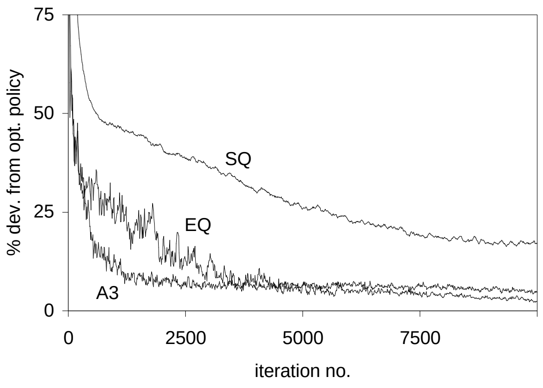

Figure 3: Performances of the greedy policies obtained by A3, SQ and EQ as a function of the iteration counter for test problem $(200, 300, 200, 0.1, 0.90)$.

applicable to Markov decision problems with monotone value functions. Computational experiments on a batch service problem indicated that it is possible to improve the empirical performance of the Q-learning algorithm by exploiting the monotonicity property of the Q-factors. The state spaces in our test problems were relatively small and our test problems could be solved to optimality by using standard dynamic programming tools as long as the transition probabilities and costs are known. Even for such test problems, the performance of the Q-learning algorithm could be unsatisfactory and the new variant that we proposed in this paper provided improvements.

It is worthwhile to emphasize that the asymptotic convergence behavior of the new variant of the Q-learning algorithm cannot be different from the asymptotic convergence behavior of the standard version. To see this, we note that both versions of the Q-learning algorithm converge to the optimal Q-factors. Therefore, considering the optimal Q-factors $\{\tilde{Q}^a(j) : j = 1, \ldots, n,\ a \in \mathcal{A}\}$, if $\tilde{Q}^a$ lies in the interior of $\mathcal{V}(-C, C)$ for all $a \in \mathcal{A}$, then the Q-factor approximations generated by the standard version of the Q-learning algorithm always stays in $\mathcal{V}(-C, C)$ after a finite number of iterations. In this case, using a projection onto $\mathcal{V}(-C, C)$ would not change the convergence behavior of the standard version. Nevertheless, the number of iterations required for the Q-factor approximations to start staying in $\mathcal{V}(-C, C)$ may be quite large and the new variant of the Q-learning algorithm may improve on the standard version when we look at the empirical performance over a finite number of iterations.

A natural direction for further research is to exploit properties besides monotonicity. Convexity, in particular, often appears in inventory control and revenue management settings. Robust variants of the Q-learning algorithm are likely to be useful for developing model free methods in these settings.

Acknowledgements

The authors thank the area editor, the associate editor and two anonymous referees for their useful comments that especially strengthened the exposition. This work was supported in part by National Science Foundation grants DMI-0422133 and CMMI-0758441.

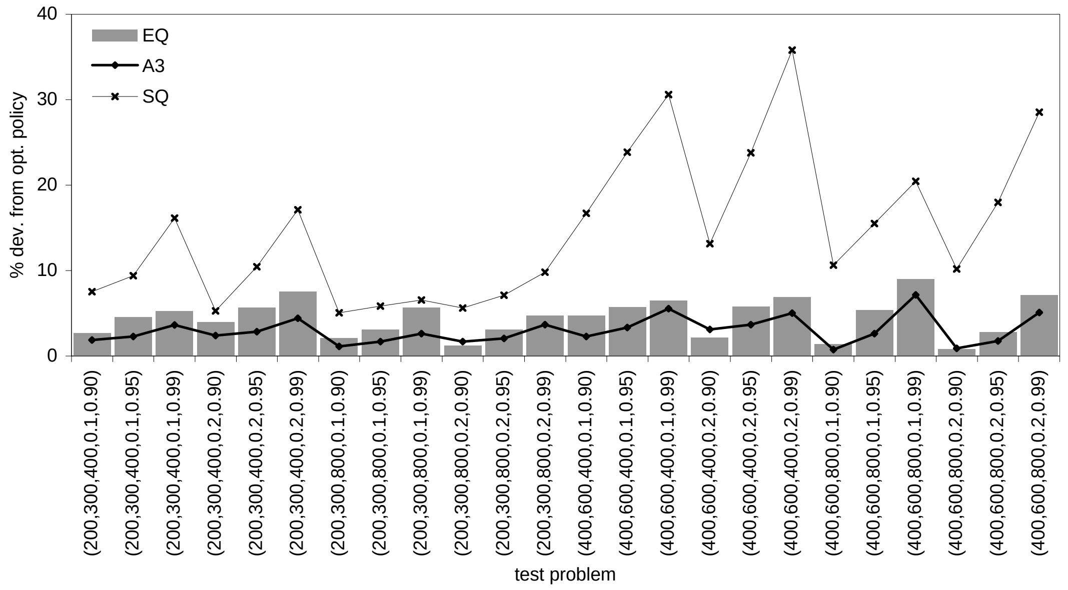

Figure 4: Performances of the greedy policies obtained by A3, SQ and EQ after 5,000,000 iterations.

References

Andradottir, S. 1995. A method for discrete stochastic optimization. *Management Science 41,* 12, 1946–1961.

Barto, A. G., Bradtke, S. J., and Singh, S. P. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence 72*, 81–138.

Benveniste, A., Metivier, M., and Priouret, P. 1991. *Adaptive Algorithms and Stochastic Approximations.* Springer.

Bertsekas, D. P., Nedic, A., and Ozdaglar, A. E. 2003. *Convex Analysis and Optimization.* Athena Scientific, Belmont, Massachusetts.

Bertsekas, D. P. and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming.* Athena Scientific, Belmont, MA.

de Farias, D. P. and Van Roy, B. 2003. The linear programming approach to approximate dynamic programming. *Operations Research 51,* 6, 850–865.

Deb, R. and Serfozo, R. 1973. Optimal control of batch service queues. *Advances in Applied Probability 5*, 340–361.

Godfrey, G. A. and Powell, W. B. 2001. An adaptive, distribution-free approximation for the newsvendor problem with censored demands, with applications to inventory and distribution problems. *Management Science 47,* 8, 1101–1112.

Ignall, E. and Kolesar, P. 1972. Operating characteristics of a simple shuttle under local dispatching rules. *Operations Research 20*, 1077–1088.

Ignall, E. and Kolesar, P. 1974. Operating characteristics of an infinite capacity shuttle: Control at a single terminal. *Operations Research 22*, 1008–1024.

Kosten, L. 1973. *Stochastic Theory of Service Systems.* Pergamon Press, New York.

Kunnumkal, S. and Topaloglu, H. 2007. Exploiting the structural properties of the underlying Markov decision problem in the Q-learning algorithm. *INFORMS Journal on Computing* to appear.

Kushner, H. J. and Clark, D. S. 1978. *Stochastic Approximation Methods for Constrained and Unconstrained Systems.* Springer-Verlang, Berlin.

Kushner, H. J. and Yin, G. G. 2003. *Stochastic Approximation and Recursive Algorithms and Applications.* Springer, New York.

Ljung, L. 1977. Analysis of recursive stochastic algorithms. *IEEE Transactions on Automatic Control 22*, 551–575.

Papadaki, K. and Powell, W. B. 2002. Exploiting structure in adaptive dynamic programming algorithms for a stochastic batch service problem. *European Journal of Operational Research 142,* 1, 108–127.

Papadaki, K. and Powell, W. B. 2003. An adaptive dynamic programming algorithm for a stochastic multiproduct batch dispatch problem. *Naval Research Logistics 50,* 7, 742–769.

Powell, W. B. 2007. *Approximate Dynamic Programming: Solving the Curses of Dimensionality.* John Wiley & Sons, Hoboken, NJ.

Powell, W. B., Ruszczynski, A., and Topaloglu, H. 2004. Learning algorithms for separable approximations of stochastic optimization problems. *Mathematics of Operations Research 29,* 4, 814–836.

Puterman, M. L. 1994. *Markov Decision Processes.* John Wiley and Sons, Inc., New York.

Si, J., Barto, A. G., Powell, W. B., and Wunsch II, D., Eds. 2004. *Handbook of Learning and Approximate Dynamic Programming.* Wiley-Interscience, Piscataway, NJ.

Sutton, R. S. and Barto, A. G. 1998. *Reinforcement Learning.* The MIT Press, Cambridge, MA.

Topaloglu, H. 2005. An approximate dynamic programming approach for a product distribution problem. *IIE Transactions 37*, 711–724.

Topaloglu, H. and Powell, W. B. 2003. An algorithm for approximating piecewise linear functions from sample gradients. *Operations Research Letters 31*, 66–76.

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

Tsitsiklis, J. and Van Roy, B. 2001. Regression methods for pricing complex American-style options. *IEEE Transactions on Neural Networks 12,* 4, 694–703.

Tsitsiklis, J. N. 1994. Asynchronous stochastic approximation and Q-learning. *Machine Learning 16*, 185–202.

van Ryzin, G. and McGill, J. 2000. Revenue management without forecasting or optimization: An adaptive algorithm for determining airline seat protection levels. *Management Science 46,* 6, 760–775.

Vazquez-Abad, F. J. 1999. Strong points of weak convergence: A study using RPA gradient estimation for automatic learning. *Automatica 35,* 7, 1255–1274.

Vazquez-Abad, F. J., Cassandras, C. G., and Julka, V. 1998. Centralized and decentralized asynchronous optimization of stochastic discrete event systems. *IEEE Transactions on Automatic Control 43,* 5, 631–655.

Watkins, C. J. C. H. 1989. Learning from delayed rewards. Ph.D. thesis, Cambridge University, Cambridge, England.

Watkins, C. J. C. H. and Dayan, P. 1992. Q-learning. *Machine Learning 8*, 279–292.

A Appendix: Proof of Proposition 1

To avoid clutter, the proof uses $p(j)$ to denote $\Pi_z^{L,U}(j)$. We consider three cases.

Case 1 Assume that $z(J) > z(J+1)$. First, we show that $p \in \mathcal{V}(L,U)$. Since $v \in \mathcal{V}(L,U)$ and z differs from v only in the Jth component, we have

$$L \leq z(1) \leq z(2) \leq \ldots \leq z(J-1) \leq z(J+1) \leq \ldots \leq z(n) \leq U. \tag{16}$$

By (16), we have $z(J) > z(J+1) \geq L$, which implies that $[z(J)+z(J+1)]/2 > L$ and we obtain $L \leq M = \min\{[z(J)+z(J+1)]/2, U\} \leq U$ by (3). Therefore, (4) and (16) imply that $L \leq p(j) \leq U$ for all $j \in \{1,\ldots,n\}$. By (4), we have $p(J-1) \leq p(J) \leq p(J+1)$, and by (4) and (16), we have $p(1) \leq p(2) \leq \ldots \leq p(J-1)$ and $p(J+1) \leq p(J+2) \leq \ldots \leq p(n)$. Therefore, we obtain $p \in \mathcal{V}(L,U)$.

Second, we show that $\|p-z\| = z(J) - M$. We let $\mathcal{J}' = \{j \in \{J+1,\ldots,n\} : z(j) \leq M\}$ and $\mathcal{J}'' = \{j \in \{J+1,\ldots,n\} : z(j) > M\}$. By (4), we have $p(j) = M \geq z(j)$ for all $j \in \mathcal{J}'$, which noting (16), implies that $|p(j)-z(j)| = M - z(j) \leq M - z(J+1) \leq [z(J)-z(J+1)]/2$ for all $j \in \mathcal{J}'$, where the second inequality follows from (3). We have $|p(j)-z(j)| = 0$ for all $j \in \mathcal{J}''$ by (4). We have $p(J) = M \leq [z(J)+z(J+1)]/2 < z(J)$, which implies that $|p(J)-z(J)| = z(J)-M \geq [z(J)-z(J+1)]/2$. Finally, (16) implies that $[z(J)+z(J+1)]/2 > z(J+1) \geq z(J-1) \geq z(J-2) \geq \ldots \geq z(1)$. Since we also have $U \geq z(J-1) \geq z(J-2) \geq \ldots \geq z(1)$ by (16), we have $M \geq z(j)$ for all $j \in \{1,\ldots,J-1\}$, which, noting (4), implies that $|p(j)-z(j)| = 0$ for all $j \in \{1,\ldots,J-1\}$. Therefore, we obtain $\|p-z\| = z(J) - M$.

Finally, we show that $\|z-w\| \geq z(J) - M$ for all $w \in \mathcal{V}(L,U)$. We consider two subcases.

Case 1.a Assume that $w(J) \leq M$. Since we have $z(J) > [z(J)+z(J+1)]/2$, (3) implies that $M < z(J)$ and we obtain $\|z-w\| \geq |z(J)-w(J)| = z(J)-w(J) \geq z(J)-M$.

Case 1.b Assume that $w(J) > M$. Letting $w(n+1) = U$, since $w \in \mathcal{V}(L,U)$, we have $U \geq w(J+1) \geq w(J) > M = \min\{[z(J)+z(J+1)]/2, U\}$, which implies that $M = [z(J)+z(J+1)]/2 > z(J+1)$ and we obtain $\|z-w\| \geq |z(J+1)-w(J+1)| = w(J+1)-z(J+1) > M - z(J+1) = z(J)-M$.

Therefore, we obtain $\|z-w\| \geq z(J)-M = \|p-z\|$ for all $w \in \mathcal{V}(L,U)$ so that $p \in \operatorname{argmin}_{w\in\mathcal{V}(L,U)} \|z-w\|$. The cases $z \in \mathcal{V}(L,U)$ and $z(J-1) > z(J)$ can be handled by using similar arguments. □

B Appendix: Proof of Lemma 3

The following lemma is useful when showing Lemma 3.

Lemma 8 *Let $\{w_k\}$ be generated by Algorithm 2 and $\epsilon > 0$, and for notational uniformity, $w_k(0) = L$ and $w_k(n+1) = U$ for all $k = 1,2,\ldots$. Assume that (A1)-(A3) hold. Then, there exists a finite iteration number K w.p.1 such that $w_k(j) \leq w_k(i) + \epsilon$ for all $j \in \{0,\ldots,n+1\}$, $i \in \{j,\ldots,n+1\}$, $k \geq K$.*

Proof All statements in the proof are in w.p.1 sense. We let $\mathbb{E}\{\eta(0)\} = L$ and $\mathbb{E}\{\eta(n+1)\} = U$ for notational uniformity. Since we have $\lim_{k\to\infty} w_k = \mathbb{E}\{\eta\}$, there exists a finite iteration number K such that $\|w_k - \mathbb{E}\{\eta\}\| \leq \epsilon/2$ for all $k \geq K$. Noting that $\mathbb{E}\{\eta\} \in \mathcal{V}(L,U)$, we obtain $w_k(j) \leq \mathbb{E}\{\eta(j)\} + \epsilon/2 \leq \mathbb{E}\{\eta(i)\} + \epsilon/2 \leq \omega_k(i) + \epsilon/2 + \epsilon/2$ for all $j \in \{0,\dots,n+1\}$, $i \in \{j,\dots,n+1\}$, $k \geq K$. □

We are now ready to prove Lemma 3. We only show the first inequality. The proof of the second inequality is similar. All statements in the proof are in w.p.1 sense. We let K be as in Lemma 8, $k \geq K$, $z_k(n+1) = v_k(n+1) = w_k(n+1) = U$ and M be computed as in (3) but by using (z_k, J_k) instead of (z, J). We consider three cases.

Case 1 Assume that $j \in \{J_k{+}1,\dots,n\}$. Since $v_{k+1} = \Pi^{L,U}_{z_k}$, (4) implies that $v_{k+1}(j) = \max\{z_k(j), M\} \geq z_k(j)$. We have $v_{k+1}(j) - w_{k+1}(j) \geq z_k(j) - w_{k+1}(j) = [1 - \alpha_k \mathbf{1}(j = J_k)]\,[v_k(j) - w_k(j)]$.

Case 2 Assume that $j \in \{1,\dots,J_k - 1\}$. We consider two subcases.

Case 2.a Assume that $z_k(j) \leq M$. We have $v_{k+1}(j) = \min\{z_k(j), M\} = z_k(j)$ by (4), which implies that $v_{k+1}(j) - w_{k+1}(j) = z_k(j) - w_{k+1}(j) = [1 - \alpha_k \mathbf{1}(j = J_k)]\,[v_k(j) - w_k(j)]$.

Case 2.b Assume that $z_k(j) > M$. We first show that $z_k(J_k - 1) > z_k(J_k)$. To see this, we note that we have either $z_k(J_k) > z_k(J_k + 1)$ or $z_k(J_k - 1) > z_k(J_k)$ because, otherwise, we have $z_k \in \mathcal{V}(L,U)$ and $z_k(j) > M = z_k(J_k)$ by (3), which contradict the fact that $z_k \in \mathcal{V}(L,U)$ and $j \in \{1,\dots,J_k - 1\}$. On the other hand, if $z_k(J_k) > z_k(J_k + 1)$, then we have $M = \min\{[z_k(J_k) + z_k(J_k + 1)]/2, U\} \geq \min\{z_k(J_k + 1), U\}$. Since $v_k \in \mathcal{V}(L,U)$ and z_k differs from v_k only in the J_kth component, we have $M \geq \min\{z_k(J_k + 1), U\} = \min\{v_k(J_k + 1), U\} = v_k(J_k + 1) \geq v_k(J_k - 1) = z_k(J_k - 1) \geq v_k(J_k - 2) = z_k(J_k - 2) \geq \dots \geq v_k(1) = z_k(1)$, which contradicts the fact that $j \in \{1,\dots,J_k - 1\}$ and $z_k(j) > M$. Therefore, we must have $z_k(J_k - 1) > z_k(J_k)$ and $M = \max\{[z_k(J_k - 1) + z_k(J_k)]/2, L\}$, which imply that $M = \max\{[z_k(J_k - 1) + z_k(J_k)]/2, L\} \geq [z_k(J_k - 1) + z_k(J_k)]/2 > z_k(J_k)$. We have $v_{k+1}(j) = \min\{z_k(j), M\} = M > z_k(J_k)$ by (4), which implies that

$$
\begin{aligned}
&v_{k+1}(j) - w_{k+1}(j) > z_k(J_k) - w_{k+1}(j) \geq z_k(J_k) - w_{k+1}(J_k) - \epsilon \\
&\qquad = [1 - \alpha_k \mathbf{1}(J_k = J_k)]\,[v_k(J_k) - w_k(J_k)] - \epsilon \geq \min_{i \in \{j+1,\dots,n\}} \Big\{[1 - \alpha_k \mathbf{1}(i = J_k)]\,[v_k(i) - w_k(i)]\Big\} - \epsilon,
\end{aligned}
$$

where the second inequality follows from Lemma 8 and the third inequality follows from the fact that $J_k \in \{j + 1,\dots,n\}$.

Case 3 Assume that $j = J_k$. We have $v_{k+1}(J_k) = M$ by (4). We consider two subcases.

Case 3.a Assume that $z_k(J_k) \leq M$. We obtain $v_{k+1}(J_k) - w_{k+1}(J_k) = M - w_{k+1}(J_k) \geq z_k(J_k) - w_{k+1}(J_k) = [1 - \alpha_k \mathbf{1}(J_k = J_k)]\,[v_k(J_k) - w_k(J_k)]$.

Case 3.b Assume that $z_k(J_k) > M$. In this case, we must have $z_k(J_k) > z_k(J_k{+}1)$. To see this, by (3), we have either $z_k(J_k) > z_k(J_k{+}1)$ or $z_k(J_k{-}1) > z_k(J_k)$. However, if $z_k(J_k{-}1) > z_k(J_k)$, then we have $M = \max\{[z_k(J_k{-}1){+}z_k(J_k)]/2, L\} \geq \max\{z_k(J_k), L\} \geq z_k(J_k)$, which contradicts the assumption that $z_k(J_k) > M$. Therefore, we must have $z_k(J_k) > z_k(J_k + 1)$ and $M = \min\{[z_k(J_k) + z_k(J_k + 1)]/2, U\}$. Since $v_k \in \mathcal{V}(L,U)$ and $z_k(J_k + 1) = v_k(J_k + 1)$ by the definition of Algorithm 1, we have

$$
\begin{aligned}
M = \min\{[z_k(J_k) + z_k(J_k + 1)]/2, U\} &\geq \min\{z_k(J_k + 1), U\} \\
&= \min\{v_k(J_k + 1), U\} = v_k(J_k + 1) = z_k(J_k + 1).
\end{aligned}
$$

Therefore, since $v_{k+1}(J_k) = M$ by (4), we obtain

$$\begin{aligned} v_{k+1}(J_k) - w_{k+1}(J_k) &= M - w_{k+1}(J_k) \geq z_k(J_k+1) - w_{k+1}(J_k) \\ &\geq z_k(J_k+1) - w_{k+1}(J_k+1) - \epsilon = [1 - \alpha_k \mathbf{1}(J_k+1 = J_k)]\,[v_k(J_k+1) - w_k(J_k+1)] - \epsilon, \end{aligned}$$

where the second inequality follows from Lemma 8. We prefer not replacing $\mathbf{1}(J_k = J_k)$ with 1 or $\mathbf{1}(J_k+1 = J_k)$ with 0 for notational uniformity. Since $v_k(J_k+1) - w_k(J_k+1) = 0$ when $J_k = n$ in the chain of inequalities above, the result follows by merging the results of the Cases 1, 2.a, 2.b and 3.

C Appendix: Proof of Lemma 5

We let $V(j) = \min_{a \in \mathcal{A}} \tilde{Q}^a(j)$ for all $j \in \{1, \ldots, n\}$, in which case the optimality equation in (6) can be written as

$$V(j) = \min_{a \in \mathcal{A}} \left\{ \sum_{s=1}^{n} p_{js}(a) \Big\{ g(j,a,s) + \lambda V(s) \Big\} \right\}.$$

Under the assumptions of Lemma 5, Proposition 4.7.3 and Theorem 6.11.6 in Puterman [1994] use the optimality equation above to show that $V(j)$ is increasing in j. Noting that $\sum_{s=s'}^{n} p_{js}(a) \leq \sum_{s=s'}^{n} p_{j+1,s}(a)$ and using the fact that $V(j)$ in increasing in j, Lemma 4.7.2 in Puterman [1994] shows that we have $\sum_{s=1}^{n} p_{js}(a)\, V(s) \leq \sum_{s=1}^{n} p_{j+1,s}(a)\, V(s)$. Since we also have $\sum_{s=1}^{n} p_{js}(a)\, g(j,a,s) \leq \sum_{s=1}^{n} p_{j+1,s}(a)\, g(j+1,a,s)$ by the assumption in the lemma, (6) implies that

$$\tilde{Q}^a(j) = \sum_{s=1}^{n} p_{js}(a) \Big\{ g(j,a,s) + \lambda V(s) \Big\} \leq \sum_{s=1}^{n} p_{j+1,s}(a) \Big\{ g(j+1,a,s) + \lambda V(s) \Big\} = \tilde{Q}^a(j+1).$$

D Appendix: Proof of Lemma 7

We let M and $\hat{M}$ be as in (3), but respectively computed by using (L, U, z) and $(\hat{L}, \hat{U}, \hat{z})$. Since we have $z(j) \leq \hat{z}(j)$ for all $j \in \{1, \ldots, n\}$, if we can show that $M \leq \hat{M}$, then the result follows from (4). We let $v(0) = z(0) = L$, $v(n+1) = z(n+1) = U$, $\hat{v}(0) = \hat{z}(0) = \hat{L}$ and $\hat{v}(n+1) = \hat{z}(n+1) = \hat{U}$ for notational uniformity and consider three cases.

Case 1 Assume that $z \in \mathcal{V}(L, U)$. By (3), we have $M = z(J)$ and $z(J-1) \leq z(J) \leq z(J+1)$. Noting that z may differ from v only in the Jth component, we obtain $v(J-1) \leq M \leq v(J+1)$.

Case 2 Assume that $z(J) > z(J+1)$. By (3), we have $M = \min\{[z(J) + z(J+1)]/2, U\} \geq \min\{z(J+1), U\} = \min\{v(J+1), U\} = v(J+1) \geq v(J-1)$, where the second equality follows from the fact that z may differ from v only in the Jth component, and the last equality and the last inequality follow from the fact that $v \in \mathcal{V}(L, U)$.

Case 3 Assume that $z(J-1) > z(J)$. By (3), we have $M = \max\{[z(J-1) + z(J)]/2, L\} \leq \max\{z(J-1), L\} = \max\{v(J-1), L\} = v(J-1) \leq v(J+1)$.

Combining the three cases above, it is easy to see that the largest possible value of M is $\min\{[z(J) + z(J+1)]/2, U\}$ and this occurs when $z(J) > z(J+1)$. Similarly, one can show that the smallest possible value of $\hat{M}$ is $\max\{[\hat{z}(J-1)+\hat{z}(J)]/2, \hat{L}\}$ and this occurs when $\hat{z}(J-1) > \hat{z}(J)$. Therefore, even if M takes its largest possible value and $\hat{M}$ takes its smallest possible value, we still have

$$\begin{aligned} M = \min\{[z(J)+z(J+1)]/2, U\} &\leq [z(J)+z(J+1)]/2 \\ &\leq [\hat{z}(J-1)+\hat{z}(J)]/2 \leq \max\{[\hat{z}(J-1)+\hat{z}(J)]/2, \hat{L}\} = \hat{M}, \end{aligned}$$

where the second inequality follows from the fact that $\hat{z}(J-1) > \hat{z}(J) \geq z(J) > z(J+1)$.