Exploiting the Structural Properties of the Underlying Markov Decision Problem in the Q-Learning Algorithm

Sumit Kunnumkal, Huseyin Topaloglu
School of Operations Research and Information Engineering,
Cornell University, Ithaca, NY 14853, USA
{sumit,topaloglu}@orie.cornell.edu

May 25, 2007

Abstract

This paper shows how to exploit the structural properties of the underlying Markov decision problem to improve the convergence behavior of the Q-learning algorithm. In particular, we consider infinite-horizon discounted-cost Markov decision problems where there is a natural ordering between the states of the system and the value function is known to be monotone in the state. We propose a new variant of the Q-learning algorithm that ensures that the value function approximations obtained during the intermediate iterations are also monotone in the state. We establish the convergence of the proposed algorithm and experimentally show that it significantly improves the convergence behavior of the standard version of the Q-learning algorithm.
Keywords: Markov decision processes; Q-learning; stochastic approximation methods.

1 Introduction

The Q-learning algorithm was proposed by Watkins (1989) and Watkins and Dayan (1992) as a method for solving Markov decision problems. This algorithm becomes especially useful when one does not have access to the complete set of transition probabilities and costs that characterize how the state of the system evolves in response to the decisions chosen by the decision-maker and what costs are incurred along the way. The fundamental idea is to gather information about the transition probabilities and the costs through sampled trajectories of the system. The algorithm starts with an arbitrary approximation to the value function, and iteratively updates this approximation by using samples of the current state of the system, the decision chosen by the decision-maker and the subsequent state of the system following the decision, which can all be generated either by simulation or by experimenting with the real system. The appealing aspect of the algorithm is that it does not require the complete set of transition probabilities or costs to update the value function approximation. For this reason, the Q-learning algorithm is generally 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 comprehensive overview of the research revolving around the Q-learning algorithm.

Tsitsiklis (1994) and Bertsekas and Tsitsiklis (1996) show that the Q-learning algorithm fits within the general framework of stochastic approximation methods. However, similar to many other

stochastic approximation methods, it can exhibit slow convergence behavior. This paper proposes a new variant of the Q-learning algorithm that exploits the structural properties of the underlying Markov decision problem to improve the convergence behavior. In particular, we consider infinite-horizon discounted-cost Markov decision problems where there is a natural ordering between the states of the system and the value function is known to be monotone in the state. We propose a new variant of the Q-learning algorithm that updates the value function approximations in such a way that the approximations obtained during the intermediate iterations are also monotone in the state. We prove the convergence of the new variant. We experimentally show that we can significantly improve the convergence behavior of the Q-learning algorithm by imposing the monotonicity property of the value function on the approximations that are obtained during the intermediate iterations.

Our work is based on previous research. Tsitsiklis (1994) and Bertsekas and Tsitsiklis (1996) give a new proof of convergence for the Q-learning algorithm by posing it as a stochastic approximation method and extending the standard stochastic approximation theory. On the other hand, Powell, Ruszczynski and Topaloglu (2004) propose an iterative method to construct approximations to the recourse functions arising from two-stage stochastic optimization problems. Their method projects the recourse function approximations onto the set of convex functions to ensure that the approximations obtained during the intermediate iterations are convex. The variant of the Q-learning algorithm we propose also uses a projection operator to project the value function approximations onto the set of monotone functions, and thereby, to ensure that the value function approximations obtained during the intermediate iterations are monotone. To prove the convergence of our variant of the Q-learning algorithm, we extend the convergence results in Tsitsiklis (1994) and Bertsekas and Tsitsiklis (1996) to deal with the effects of the projection operator. Our extension requires the projection operator to satisfy a certain order-preserving property, which roughly states that if the functions $p(\cdot)$ and $q(\cdot)$ satisfy $p(x) \leq q(x)$ for all x in their common domain, and $\tilde{p}(\cdot)$ and $\tilde{q}(\cdot)$ are respectively the projections of $p(\cdot)$ and $q(\cdot)$ onto the set of monotone functions, then we have $\tilde{p}(x) \leq \tilde{q}(x)$ for all x. Ultimately, we establish the convergence of our variant of the Q-learning algorithm by showing that the projection operator that we use indeed satisfies this property.

The model-free nature of the Q-learning algorithm can be quite useful in practical settings. In particular, the Q-learning algorithm enables us to work directly with the sampled state and action trajectories of the system. In contrast, a conventional stochastic optimization model usually requires carrying out three steps. First, we collect data on the evolution of the system over time. Next, we use the data to estimate the probability distributions that govern the evolution of the system. Finally, we compute the optimal policy by using the estimated probability distributions. The novel aspect of the Q-learning algorithm is that it bypasses the estimation step, and thus, enables us to

move directly from the data to optimization. Therefore, we can use the Q-learning algorithm for real-time optimization and search for the optimal policy in an iterative and incremental fashion rather than waiting for a large amount of data to be available to estimate the probability distributions. Since the Q-learning algorithm bypasses the estimation step, it also avoids the fit errors that occur when one employs parametric forms for the probability distributions and uses the data to estimate the parameters. For example, often times, a conventional stochastic optimization model does not provide satisfactory results simply because one fits a geometric distribution to the data when a negative binomial distribution is a better candidate. Finally, the Q-learning algorithm provides a remedy to the censored data problem that arises in many practical settings. The censored data problem refers to the discrepancy between what is naturally recorded as the system evolves over time and what is needed to estimate the probability distributions. For example, in inventory control settings, it is natural to record the amount of inventory sold rather than the amount of demand, and these two figures are different when the demand exceeds the available inventory. Similarly, in queue admission settings, it is natural to record the number of entities that are admitted into the system rather than the number of entities that arrive, and these two figures are different when the number of entities in the system reaches the system capacity. When dealing with the censored data, we may not have access to the samples of the random variables that we are interested in and this makes estimating the probability distributions very difficult. In practice, the data are usually "uncensored" by using heuristic methods. On the other hand, the Q-learning algorithm naturally avoids the censored data problem since it directly works with the sampled state and action trajectories of the system without estimating the probability distributions. For example, in inventory control settings, the Q-learning algorithm works with the trajectories of the available inventory and the amount of inventory sold. In queue admission settings, the Q-learning algorithm works with the trajectories of the admissions into and departures from the system. Ding (2002) provides a nice overview of the censored data problem from an operations management perspective.

There has been work on exploiting the structural properties of the underlying Markov decision problem to improve the performance of different solution methods. Puterman (1994) gives a variant of the value iteration algorithm that can be used when the value function is known to be monotone in the state. There is substantial literature on approximating the value function by using linear architectures of the form $\sum_{k \in \mathcal{K}} r_k \, \phi_k(x)$, where $\{r_k : k \in \mathcal{K}\}$ are adjustable parameters, $\{\phi_k(\cdot) : k \in \mathcal{K}\}$ are fixed basis functions and x is the state of the system (see, for example, Schweitzer and Seidmann (1985), Bertsekas and Tsitsiklis (1996), Tsitsiklis and Van Roy (1997) and de Farias and Van Roy (2003)). The consensus in this literature is that the basis functions $\{\phi_k(\cdot) : k \in \mathcal{K}\}$ should be chosen to capture the known structural properties of the value function as accurately as possible. Topaloglu and Powell (2006) formulate a dynamic programming-based fleet management model, and motivated by the fact

that the value function for the problem is concave, they use concave value function approximations. Papadaki and Powell (2002) and Papadaki and Powell (2003) show that the value function arising from a batch service problem is monotone in the number of products waiting to be served and exploit this property to enhance the performance of their approximate dynamic programming algorithm.

Stochastic approximation methods commonly use projections to ensure the boundedness of the iterates by projecting them onto bounded sets (see, for example, Ljung (1977), Kushner and Clark (1978) and Andradottir (1995)). However, the use of projections in learning algorithms to impose the structural properties of the value function on the value function approximations is novel.

This paper makes the following research contributions. 1) We consider infinite-horizon discounted-cost Markov decision problems where the value function is known to be monotone in the state of the system. We propose a new variant of the Q-learning algorithm that uses a projection operator to ensure that the value function approximations obtained during the intermediate iterations are monotone in the state. We establish the convergence of this algorithm. 2) To establish the convergence, we extend the available stochastic approximation theory and prove the aforementioned order-preserving property of the projection operator. 3) We present computational experiments on two batch service problems that show that we can significantly improve the convergence behavior of the Q-learning algorithm by exploiting the structural properties of the underlying Markov decision problem.

The rest of the paper is organized as follows. Section 2 describes the new variant of the Q-learning algorithm that we propose. We prove the convergence of this algorithm under certain assumptions in Section 3. In Section 4, we review the assumptions that we make in Section 3 and show that the new variant of the Q-learning algorithm that we propose indeed satisfies these assumptions. Section 5 presents our computational experiments.

2 Description of the Algorithm

We are interested in infinite-horizon discounted-cost Markov decision problems with finite sets of states and actions, which we respectively denote by $\mathcal{S}$ and $\mathcal{U}$. If the system is in state i and we use action u, then the system moves to state j with probability $p_{ij}(u)$ and we incur a cost of $g(i,u,j)$, where $|g(i,u,j)| < \infty$. The costs in the future time periods are discounted by a factor $\lambda \in [0,1)$ per time period. For notational brevity, we assume that $\mathcal{S} = \{1,\ldots,n\}$ and each action in $\mathcal{U}$ is admissible for every state.

A stationary Markovian deterministic policy d is a mapping from $\mathcal{S}$ to $\mathcal{U}$ that describes which action to take for each possible state. Therefore, the states visited by the system under policy d evolve

as a Markov chain with the transition probability matrix $\{p_{ij}(d(i)) : i, j \in \mathcal{S}\}$. Letting $\{X_0^d, X_1^d, \ldots\}$ be the Markov chain that evolves according to the transition probability matrix $\{p_{ij}(d(i)) : i, j \in \mathcal{S}\}$, the infinite-horizon discounted cost incurred by starting from state i and using policy d is

$$J^d(i) = \lim_{T\to\infty} \mathbb{E}\left\{ \sum_{t=0}^{T} \lambda^t g(X_t^d, d(X_t^d), X_{t+1}^d) \mid X_0 = i \right\}.$$

Letting $\mathcal{D}$ be the set of stationary Markovian deterministic policies, the optimal policy d^* satisfies $J^{d^*}(i) = \min_{d\in\mathcal{D}} J^d(i)$ for all $i \in \mathcal{S}$. This policy can be obtained by solving the so-called optimality equation

$$J(i) = \min_{u\in\mathcal{U}} \sum_{j=1}^{n} p_{ij}(u)\big[\, g(i,u,j) + \lambda\, J(j)\big] \qquad \text{for all } i \in \mathcal{S} \tag{1}$$

and letting

$$d^*(i) = \operatorname*{argmin}_{u\in\mathcal{U}} \sum_{j=1}^{n} p_{ij}(u)\big[\, g(i,u,j) + \lambda\, J(j)\big] \qquad \text{for all } i \in \mathcal{S}. \tag{2}$$

The Q-learning algorithm is based on an alternative interpretation of the optimality equation. In particular, we let

$$Q^u(i) = \sum_{j=1}^{n} p_{ij}(u)\big[\, g(i,u,j) + \lambda\, J(j)\big] \qquad \text{for all } i \in \mathcal{S},\ u \in \mathcal{U}, \tag{3}$$

in which case (1) implies that $J(i) = \min_{u\in\mathcal{U}} Q^u(i)$ for all $i \in \mathcal{S}$. Using this in (3), we have

$$Q^u(i) = \sum_{j=1}^{n} p_{ij}(u)\Big[g(i,u,j) + \lambda \min_{v\in\mathcal{U}} Q^v(j)\Big] \qquad \text{for all } i \in \mathcal{S},\ u \in \mathcal{U}. \tag{4}$$

Then, noting (2) and (3), the optimal policy can be obtained by solving (4) and letting $d^*(i) = \operatorname{argmin}_{u\in\mathcal{U}} Q^u(i)$ for all $i \in \mathcal{S}$. We refer to $Q^u(i)$ as the Q-factor for the state-action pair (i, u).

The Q-learning algorithm solves (4) through stochastic approximation. It starts with arbitrary Q-factor approximations $\{Q_0^u(i) : i \in \mathcal{S},\ u \in \mathcal{U}\}$. At the t-th iteration of the algorithm, a state-action pair (i, u) and a successor state s are sampled such that $\mathbb{P}\{s = j \mid \text{history of the algorithm up to iteration } t\} = p_{ij}(u)$. (We shortly make clear what we mean by the history of the algorithm up to iteration t.) The Q-factor approximation for the state-action pair (i, u) is updated as

$$Q_{t+1}^u(i) = \big[\, 1 - \alpha_t^u(i)\big]\, Q_t^u(i) + \alpha_t^u(i)\Big[g(i,u,s) + \lambda \min_{v\in\mathcal{U}} Q_t^v(s)\Big], \tag{5}$$

where $\alpha_t^u(i)$ is a step-size parameter. The Q-factor approximations for the other state-action pairs remain unchanged. Under certain assumptions, it can be shown that $\lim_{t\to\infty} Q_t^u(i) = Q^u(i)$ with probability 1 (w.p.1) for all $i \in \mathcal{S},\ u \in \mathcal{U}$, where $\{Q^u(i) : i \in \mathcal{S},\ u \in \mathcal{U}\}$ is the solution to (4).

The Q-learning algorithm, as described above, does not exploit the structural properties of the underlying Markov decision problem and may require a large number of iterations to provide a good

1) Initialize the Q-factor approximations $Q_0 \in \mathbb{R}^{n\times|\mathcal{U}|}$ such that $Q_0^u \in \mathcal{P}^{-C,C}$ for all $u \in \mathcal{U}$. Set $t = 0$.

2) Sample a state-action pair (i_t, u_t) and a successor state s_t. Set

$$R_t^u(i) = Q_t^u(i) + \delta_t^u(i)\,\alpha_t^u(i)\Big[g(i,u,s_t) + \lambda \min_{v\in\mathcal{U}} Q_t^v(s_t) - Q_t^u(i)\Big] \qquad \text{for all } i \in \mathcal{S},\, u \in \mathcal{U}, \quad (7)$$

where $\alpha_t^u(i)$ is a step-size parameter and

$$\delta_t^u(i) = \begin{cases} 1 & \text{if } (i,u) = (i_t, u_t) \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

3) Set $Q_{t+1}^u = \Pi^{-C,C} R_t^u$ for all $u \in \mathcal{U}$.

4) Increase t by 1 and go to Step 2.

Figure 1: The monotone Q-learning algorithm.

policy. In this paper, we propose a new variant of the Q-learning algorithm applicable to problems where the Q-factors are known to satisfy

$$Q^u(i) \leq Q^u(i+1) \qquad \text{for all } i \in \{1,\ldots,n-1\},\, u \in \mathcal{U}. \quad (6)$$

The idea behind our variant of the Q-learning algorithm is to use a projection operator to impose the property $Q_t^u(i) \leq Q_t^u(i+1)$ for all $i \in \{1,\ldots,n-1\}$, $u \in \mathcal{U}$ on the Q-factor approximations that are obtained during the intermediate iterations. Our objective is to facilitate faster convergence by imposing the known structural properties of the Q-factors on the Q-factor approximations.

Throughout the paper, we denote the i-th component of $y \in \mathbb{R}^n$ by $y(i)$, the (i,u)-th component of $Q \in \mathbb{R}^{n\times|\mathcal{U}|}$ by $Q^u(i)$ and the vector $\{Q^u(i) : i \in \mathcal{S}\}$ by Q^u. For scalars L and U with $L \leq U$, we define the convex set $\mathcal{P}^{L,U}$ and the projection operator $\Pi^{L,U}$ onto this set as

$$\mathcal{P}^{L,U} = \{y \in \mathbb{R}^n : L \leq y(1) \leq y(2) \leq \ldots \leq y(n) \leq U\}$$
$$\Pi^{L,U} y = \operatorname*{argmin}_{z\in\mathcal{P}^{L,U}} \|z - y\|_2,$$

where $\|\cdot\|_2$ is the Euclidean norm on $\mathbb{R}^n$. Since the Q-factors are known to satisfy (6), if we let C be large enough so that $|Q^u(i)| \leq C < \infty$ for all $i \in \mathcal{S}$, $u \in \mathcal{U}$, then we have $Q^u \in \mathcal{P}^{-C,C}$ for all $u \in \mathcal{U}$. Letting $\{Q_t^u(i) : i \in \mathcal{S},\ u \in \mathcal{U}\}$ be the Q-factor approximations obtained by the Q-learning algorithm at iteration t, the idea behind our variant of the Q-learning algorithm is to project Q_t^u onto $\mathcal{P}^{-C,C}$ when $Q_t^u \notin \mathcal{P}^{-C,C}$ for some action u.

The variant of the Q-learning algorithm we propose is described in Figure 1. We refer to this algorithm as the monotone Q-learning algorithm. Letting $\mathcal{F}_t$ be the σ-subalgebra generated by the random variables $\{Q_0, i_0, \ldots, i_{t-1}, u_0, \ldots, u_{t-1}, s_0, \ldots, s_{t-1}\}$ in this algorithm, the successor state s_t

is sampled such that $\mathbb{P}\{s_t = j \mid \mathcal{F}_t, i_t, u_t\} = p_{i_t j}(u_t)$. The updating procedure in (7) is the same as the one in (5). The random variable $\delta_t^u(i)$ ensures that $R_t^u(i) = Q_t^u(i)$ when $(i,u) \neq (i_t, u_t)$. In Step 3, we project the vector R_t^u onto $\mathcal{P}^{-C,C}$. This step ensures that we have $Q_{t+1}^u(i) \leq Q_{t+1}^u(i+1)$ for all $i \in \{1,\ldots,n-1\}$, $u \in \mathcal{U}$. In the next section, we show that the iterates of this algorithm converge to the solution to (4) w.p.1 under certain assumptions.

Closing this section, we note that Q-factors satisfying (6) appear in numerous queue admission, equipment replacement, batch service and pricing applications (see, for example, Ignall and Kolesar (1974), Papadaki and Powell (2002) and Puterman (1994)). In particular, the following corollary to Lemma 4.7.2 and Theorem 6.11.6 in Puterman (1994) gives sufficient conditions under which the Q-factors satisfy (6).

Lemma 1 *Let $\{Q^u(i) : i \in \mathcal{S},\ u \in \mathcal{U}\}$ be the solution to (4). Assume that $\sum_{j=1}^n p_{ij}(u)\, g(i,u,j)$ is increasing in i for all $u \in \mathcal{U}$ and $\sum_{j=k}^n p_{ij}(u)$ is increasing in i for all $k \in \mathcal{S}$, $u \in \mathcal{U}$. Then, we have $Q^u(i) \leq Q^u(i+1)$ for all $i \in \{1,\ldots,n-1\}$, $u \in \mathcal{U}$.*

Proof We let $\{J(i) : i \in \mathcal{S}\}$ be the solution to (1). Under the conditions stated above, the proof of Theorem 6.11.6 in Puterman (1994) shows that $J(j) \leq J(j+1)$ for all $j \in \{1,\ldots,n-1\}$. We fix $i \in \{1,\ldots,n-1\}$ and $u \in \mathcal{U}$. Since we have $\sum_{j=k}^n p_{ij}(u) \leq \sum_{j=k}^n p_{i+1,j}(u)$ for all $k \in \mathcal{S}$, Lemma 4.7.2 in Puterman (1994) implies that $\sum_{j=1}^n p_{ij}(u) J(j) \leq \sum_{j=1}^n p_{i+1,j}(u) J(j)$. Since we also have $\sum_{j=1}^n p_{ij}(u)\, g(i,u,j) \leq \sum_{j=1}^n p_{i+1,j}(u)\, g(i+1,u,j)$, we obtain

$$Q^u(i) = \sum_{j=1}^n p_{ij}(u)\big[\, g(i,u,j) + \lambda\, J(j)\big] \leq \sum_{j=1}^n p_{i+1,j}(u)\big[\, g(i+1,u,j) + \lambda\, J(j)\big] = Q^u(i+1). \qquad \square$$

It is important to note that we can often check the assumptions of Lemma 1 by using the problem structure without having access to the actual values of $\{p_{ij}(u) : i,j \in \mathcal{S},\ u \in \mathcal{U}\}$ and $\{g(i,u,j) : i,j \in \mathcal{S},\ u \in \mathcal{U}\}$. 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 $\sum_{j=1}^n p_{ij}(u)\, g(i,u,j)$ is increasing in i 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. The assumption that $\sum_{j=k}^n p_{ij}(u)$ is increasing in i implies that the probability that the number of entities in the system at the next time period exceeds k increases as the number of entities in the system at the current time period increases. This assumption is satisfied by essentially any sensible arrival process. For an equipment replacement application where the state is the age of the equipment, the action is the replacement decision and the cost components of interest are the maintenance and replacement costs, the assumption that $\sum_{j=1}^n p_{ij}(u)\, g(i,u,j)$ is increasing in i is

1) Initialize $y_0 \in \mathbb{R}^n$ such that $y_0 \in \mathcal{P}^{L,U}$. Set $t = 0$.

2) Sample $i_t \in \mathcal{S}$ and $\theta_t \in \mathbb{R}^n$. Set

$$\hat{y}_t(i) = y_t(i) + \rho_t(i)\,\gamma_t(i)\big[\,y^*(i) + \theta_t(i) - y_t(i)\big] \qquad \text{for all } i \in \mathcal{S}, \tag{9}$$

where $\gamma_t(i)$ is a step-size parameter, $y^* \in \mathbb{R}^n$ is a fixed vector, $\theta_t \in \mathbb{R}^n$ is a random error term and

$$\rho_t(i) = \begin{cases} 1 & \text{if } i = i_t \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

3) Set $y_{t+1} = \Pi^{L,U}\hat{y}_t$.

4) Increase t by 1 and go to Step 2.

Figure 2: A stochastic approximation method that uses the projection operator $\Pi^{L,U}$.

satisfied when the maintenance and replacement costs increase with age. Similarly, the assumption that $\sum_{j=k}^{n} p_{ij}(u)$ is increasing in i is satisfied by essentially any sensible aging process. We refer the reader to Papadaki and Powell (2002) and Puterman (1994) for similar observations for batch service and pricing applications.

3 Convergence Analysis

In this section, we show that the iterates of the monotone Q-learning algorithm converge to the solution to (4) w.p.1. For this purpose, we consider two stochastic approximation methods that make use of the projection operator $\Pi^{L,U}$. The first one of these methods is useful to prove the convergence of the second one. Ultimately, we prove the convergence of the monotone Q-learning algorithm by showing that it is a special case of the second stochastic approximation method.

The first stochastic approximation method that we consider is described in Figure 2. We let $\mathcal{G}_t$ be the σ-subalgebra generated by the random variables $\{y_0, i_0, \ldots, i_{t-1}, \theta_0, \ldots, \theta_{t-1}\}$ in this algorithm, and assume that $y^* \in \mathcal{P}^{L,U}$, $\gamma_t(i)$ is positive and $\mathcal{G}_t$-measurable for all $i \in \mathcal{S}$, $t = 0, 1, \ldots$ w.p.1, and

$$\liminf_{t\to\infty} \mathbb{P}\{i_t = i \mid \mathcal{G}_t\} > 0 \qquad \text{for all } i \in \mathcal{S} \text{ w.p.1} \tag{11}$$

$$\sum_{t=0}^{\infty} \min_{i\in\mathcal{S}} \gamma_t(i) = \infty \qquad \text{w.p.1} \tag{12}$$

$$\sum_{t=0}^{\infty} \mathbb{E}\big\{\rho_t(i)\,[\gamma_t(i)]^2\big\} < \infty \qquad \text{for all } i \in \mathcal{S} \tag{13}$$

$$\mathbb{E}\big\{\theta_t(i_t)\,|\,\mathcal{G}_t, i_t\big\} = 0 \qquad \text{for all } t = 0, 1, \ldots \text{ w.p.1} \tag{14}$$

$$\mathbb{E}\big\{[\theta_t(i_t)]^2\,|\,\mathcal{G}_t, i_t\big\} \le A < \infty \qquad \text{for all } t = 0, 1, \ldots \text{ w.p.1} \tag{15}$$

1) Initialize $Q_0 \in \mathbb{R}^{n\times|\mathcal{U}|}$ such that $Q_0^u \in \mathcal{P}^{-C,C}$ for all $u \in \mathcal{U}$. Set $t = 0$.

2) Sample (i_t, u_t) in $\mathcal{S} \times \mathcal{U}$ and $\omega_t \in \mathbb{R}^{n\times|\mathcal{U}|}$. Set

$$R_t^u(i) = Q_t^u(i) + \delta_t^u(i)\,\alpha_t^u(i)\Big\{[\Gamma Q_t]^u(i) + \omega_t^u(i) - Q_t^u(i)\Big\} \qquad \text{for all } i \in \mathcal{S},\ u \in \mathcal{U}, \tag{16}$$

where $\alpha_t^u(i)$ is a step-size parameter, $\delta_t^u(i)$ is as defined in (8), Γ is an operator on $\mathbb{R}^{n\times|\mathcal{U}|}$ and $\omega_t \in \mathbb{R}^{n\times|\mathcal{U}|}$ is a random error term.

3) Set $Q_{t+1}^u = \Pi^{-C,C} R_t^u$ for all $u \in \mathcal{U}$.

4) Increase t by 1 and go to Step 2.

Figure 3: A stochastic approximation method that is useful to show the convergence of the monotone Q-learning algorithm.

for some scalar A. We have the following convergence result for the algorithm in Figure 2.

Lemma 2 *Assume that $y^* \in \mathcal{P}^{L,U}$, $\gamma_t(i)$ is positive and $\mathcal{G}_t$-measurable for all $i \in \mathcal{S}$, $t = 0, 1, \ldots$ w.p.1, and* (11)-(15) *hold. Let the sequence $\{y_t\}$ be generated by the algorithm in Figure 2. Then, we have $\lim_{t\to\infty} \|y_t - y^*\|_2 = 0$ w.p.1.*

Proof See the online supplement. □

Powell et al. (2004) analyze a similar algorithm under the assumption that the step-size parameter $\gamma_t(i)$ does not depend on i. However, this assumption is frequently violated in practical applications. Our proof also makes use of the supermartingale convergence theorem directly and is more compact.

The second stochastic approximation method that we consider is described in Figure 3. We establish the convergence of this algorithm by using Lemma 2. To this end, for $y \in \mathbb{R}^n$ and $Q \in \mathbb{R}^{n\times|\mathcal{U}|}$, we define the norm $\|\cdot\|$ as $\|y\| = \max_{i\in\mathcal{S}} |y(i)|$ and $\|Q\| = \max_{i\in\mathcal{S},u\in\mathcal{U}} |Q^u(i)|$. We also define the partial ordering $\leq$ on $\mathbb{R}^n$, whereby we write $y \leq z$ when $y(i) \leq z(i)$ for all $i \in \mathcal{S}$.

We let $\mathcal{F}_t$ be the σ-subalgebra generated by the random variables $\{Q_0, i_0, \ldots, i_{t-1}, u_0, \ldots, u_{t-1}, \omega_0, \ldots, \omega_{t-1}\}$ in the algorithm in Figure 3 and assume that the following statements hold.

(A.1) We have $\liminf_{t\to\infty} \mathbb{P}\{(i_t, u_t) = (i, u) \,|\, \mathcal{F}_t\} > 0$ for all $i \in \mathcal{S}$, $u \in \mathcal{U}$ w.p.1.

(A.2) The step-size parameters are positive, $\mathcal{F}_t$-measurable and satisfy

$$\sum_{t=0}^{\infty} \min_{i\in\mathcal{S}} \alpha_t^u(i) = \infty \qquad \text{for all } u \in \mathcal{U} \text{ w.p.1}$$

$$\sum_{t=0}^{\infty} \mathbb{E}\big\{\delta_t^u(i)\,[\alpha_t^u(i)]^2\big\} < \infty \qquad \text{for all } i \in \mathcal{S},\ u \in \mathcal{U}.$$

(A.3) For some scalar A, the random error terms satisfy

$$\mathbb{E}\{\omega_t^{u_t}(i_t) \,|\, \mathcal{F}_t, i_t, u_t\} = 0 \qquad \text{for all } t = 0, 1, \ldots \text{ w.p.1}$$
$$\mathbb{E}\{[\omega_t^{u_t}(i_t)]^2 \,|\, \mathcal{F}_t, i_t, u_t\} \leq A < \infty \qquad \text{for all } t = 0, 1, \ldots \text{ w.p.1.}$$

(A.4) There exist a scalar C, $\tilde{Q} \in \mathbb{R}^{n \times |\mathcal{U}|}$ and $\lambda \in [0, 1)$ that satisfy $\tilde{Q}^u \in \mathcal{P}^{-C,C}$ for all $u \in \mathcal{U}$ and

$$\|\Gamma Q_t - \tilde{Q}\| \leq \lambda \|Q_t - \tilde{Q}\| \qquad \text{for all } t = 0, 1, \ldots \text{ w.p.1.}$$

(A.5) For $L_1 \leq L_2$ and $U_1 \leq U_2$, let $y \in \mathcal{P}^{L_1,U_1}$ and $z \in \mathcal{P}^{L_2,U_2}$ be such that $y \leq z$. Let the scalars A and B satisfy $A \leq B$. Fix $i^* \in \mathcal{S}$, and let $\hat{y}$ and $\hat{z} \in \mathbb{R}^n$ be obtained by

$$\hat{y}(i) = \begin{cases} y(i) + \alpha\,[A - y(i)] & \text{if } i = i^* \\ y(i) & \text{otherwise} \end{cases}$$
$$\hat{z}(i) = \begin{cases} z(i) + \alpha\,[B - z(i)] & \text{if } i = i^* \\ z(i) & \text{otherwise,} \end{cases}$$

where $\alpha \in [0, 1]$. We have $\Pi^{L_1,U_1}\hat{y} \leq \Pi^{L_2,U_2}\hat{z}$.

(A.1) states that the probability that an element of $\mathcal{S} \times \mathcal{U}$ is sampled at iteration t stays away from 0 as t gets large. (A.2) seems to be a nonstandard assumption on the step-size parameters, but Section 4 shows that it is satisfied by commonly used step-size parameters. (A.3) is a standard assumption on the random error terms. (A.4) holds when the operator Γ is a contraction mapping with respect to the norm $\|\cdot\|$ and with contraction factor λ, and $\tilde{Q}$ is the fixed point of this operator. Finally, (A.5) imposes an order-preserving property on the projection operator. Since $y \leq z$, $A \leq B$ and $\alpha \in [0, 1]$, we have $\hat{y} \leq \hat{z}$. Therefore, (A.5) roughly states that the projection operator maintains the ordering between $\hat{y}$ and $\hat{z}$. We verify all of these assumptions in Section 4.

The following convergence result for the algorithm in Figure 3 is an extension of Proposition 4.4 in Bertsekas and Tsitsiklis (1996) designed to deal with the effects of the projection operator.

Proposition 3 *Assume that* (A.1)-(A.5) *hold. Let the sequence* $\{Q_t\}$ *be generated by the algorithm in Figure 3 and* $\tilde{Q}$ *be as in* (A.4)*. Then, we have* $\lim_{t\to\infty} \|Q_t - \tilde{Q}\| = 0$ *w.p.1.*

Proof Appealing to the monotone convergence theorem, we have

$$\sum_{t=0}^{\infty} \mathbb{E}\{\delta_t^u(i)\,[\alpha_t^u(i)]^2\} = \mathbb{E}\left\{\sum_{t=0}^{\infty} \mathbb{E}\{\delta_t^u(i)\,[\alpha_t^u(i)]^2 \,|\, \mathcal{F}_t\}\right\} = \mathbb{E}\left\{\sum_{t=0}^{\infty} \mathbb{P}\{(i_t, u_t) = (i, u) \,|\, \mathcal{F}_t\}\,[\alpha_t^u(i)]^2\right\},$$

which, noting (A.2), implies that $\sum_{t=0}^{\infty} \mathbb{P}\{(i_t, u_t) = (i, u) \,|\, \mathcal{F}_t\}\,[\alpha_t^u(i)]^2 < \infty$ w.p.1. Then, by (A.1), there exists a finite iteration number $\bar{t}$ w.p.1 such that $\alpha_t^u(i) \in [0, 1]$ for all $t \geq \bar{t}$. Therefore, without loss of generality, we assume that $\alpha_t^u(i) \in [0, 1]$ for all $i \in \mathcal{S}$, $u \in \mathcal{U}$, $t = 0, 1, \ldots$ w.p.1.

All statements in the rest of the proof should be understood in w.p.1 sense. Since Q_t^u is the projection of R_{t-1}^u onto $\mathcal{P}^{-C,C}$, we have $Q_t^u \in \mathcal{P}^{-C,C}$ for all $u \in \mathcal{U}$, $t = 0, 1, \ldots$. Then, since $\tilde{Q}^u \in \mathcal{P}^{-C,C}$ for all $u \in \mathcal{U}$ by (A.4), we have $\|Q_t - \tilde{Q}\| \leq \|Q_t\| + \|\tilde{Q}\| \leq 2C$ for all $t = 0, 1, \ldots$. We choose $\epsilon > 0$ with $\lambda + \epsilon < 1$. Letting $D_0 = 2C$, we define the sequence $\{D_k\}$ through $D_{k+1} = (\lambda + \epsilon) D_k$. We have $\|Q_t - \tilde{Q}\| \leq D_0$ for all $t = 0, 1, \ldots$. To show the result by induction, we assume that there exists a finite iteration number t_k such that $\|Q_t - \tilde{Q}\| \leq D_k$ for all $t \geq t_k$. We show that this assumption implies that there exists a finite iteration number t_{k+1} such that $\|Q_t - \tilde{Q}\| \leq D_{k+1}$ for all $t \geq t_{k+1}$. Then, since $\lim_{k\to\infty} D_k = 0$, we obtain $\lim_{t\to\infty} \|Q_t - \tilde{Q}\| = 0$.

We fix $u \in \mathcal{U}$ and let $e \in \mathbb{R}^n$ be the vector whose components are all 1's. For $t \geq t_k$, starting with $y_{t_k}^u = \tilde{Q}^u - D_k e$ and $z_{t_k}^u = \tilde{Q}^u + D_k e$, we define the sequences of vectors $\{y_t^u\}$ and $\{z_t^u\}$ through

$$\hat{y}_t^u(i) = y_t^u(i) + \delta_t^u(i)\, \alpha_t^u(i) \big[\tilde{Q}^u(i) - \lambda D_k + \omega_t^u(i) - y_t^u(i) \big] \qquad \text{for all } i \in \mathcal{S} \tag{17}$$

$$\hat{z}_t^u(i) = z_t^u(i) + \delta_t^u(i)\, \alpha_t^u(i) \big[\tilde{Q}^u(i) + \lambda D_k + \omega_t^u(i) - z_t^u(i) \big] \qquad \text{for all } i \in \mathcal{S}, \tag{18}$$

$y_{t+1}^u = \Pi^{-3C,C} \hat{y}_t^u$ and $z_{t+1}^u = \Pi^{-C,3C} \hat{z}_t^u$. Therefore, the sequences $\{y_t^u\}$ and $\{z_t^u\}$ are generated by the stochastic approximation method in Figure 2.

Since $\tilde{Q}^u \in \mathcal{P}^{-C,C}$ by (A.4) and $D_k \leq D_0 = 2C$, we have

$$-3C \leq -C - \lambda D_k \leq \tilde{Q}^u(i) - \lambda D_k \leq \tilde{Q}^u(i) \leq C$$

$$-C \leq \tilde{Q}^u(i) \leq \tilde{Q}^u(i) + \lambda D_k \leq C + \lambda D_k \leq 3C$$

for all $i \in \mathcal{S}$, which imply that $\tilde{Q}^u - \lambda D_k e \in \mathcal{P}^{-3C,C}$ and $\tilde{Q}^u + \lambda D_k e \in \mathcal{P}^{-C,3C}$. By (A.1)-(A.3), the other assumptions in (11)-(15) that are needed for Lemma 2 to hold are satisfied. Therefore, we have $\lim_{t\to\infty} \|y_t^u - \tilde{Q}^u + \lambda D_k e\|_2 = 0$ and $\lim_{t\to\infty} \|z_t^u - \tilde{Q}^u - \lambda D_k e\|_2 = 0$ by Lemma 2.

We now show by induction that $y_t^u \leq Q_t^u \leq z_t^u$ for all $t \geq t_k$. The result holds for $t = t_k$ since we have $y_{t_k}^u = \tilde{Q}^u - D_k e$, $z_{t_k}^u = \tilde{Q}^u + D_k e$ and $\|Q_{t_k} - \tilde{Q}\| \leq D_k$. Assume that $y_t^u \leq Q_t^u \leq z_t^u$ holds for some $t \geq t_k$. Then, we have

$$\begin{aligned}
R_t^u(i) &= \big[1 - \delta_t^u(i)\, \alpha_t^u(i) \big] Q_t^u(i) + \delta_t^u(i)\, \alpha_t^u(i) \Big\{ [\Gamma Q_t]^u(i) + \omega_t^u(i) \Big\} \\
&\leq \big[1 - \delta_t^u(i)\ \alpha_t^u(i) \big] z_t^u(i) + \delta_t^u(i)\, \alpha_t^u(i) \Big\{ \tilde{Q}^u(i) + \lambda \|Q_t - \tilde{Q}\| + \omega_t^u(i) \Big\} \\
&\leq \big[1 - \delta_t^u(i)\ \alpha_t^u(i) \big] z_t^u(i) + \delta_t^u(i)\, \alpha_t^u(i) \Big\{ \tilde{Q}^u(i) + \lambda D_k + \omega_t^u(i) \Big\} = \hat{z}_t^u(i)
\end{aligned}$$

for all $i \in \mathcal{S}$, where the first inequality uses the induction hypothesis and (A.4), and the second inequality uses the assumption that $\|Q_t - \tilde{Q}\| \leq D_k$ for all $t \geq t_k$. Using a similar argument, we can also show that $\hat{y}_t^u(i) \leq R_t^u(i)$ for all $i \in \mathcal{S}$. Therefore, we have $\hat{y}_t^u \leq R_t^u \leq \hat{z}_t^u$. Since y_t^u and z_t^u are respectively the projections of $\hat{y}_{t-1}^u$ and $\hat{z}_{t-1}^u$ onto $\mathcal{P}^{-3C,C}$ and $\mathcal{P}^{-C,3C}$, we have $y_t^u \in \mathcal{P}^{-3C,C}$ and

$z_t^u \in \mathcal{P}^{-C,3C}$. Then, (A.5) implies that $y_{t+1}^u = \Pi^{-3C,C}\hat{y}_t^u \leq \Pi^{-C,C}R_t^u = Q_{t+1}^u \leq \Pi^{-C,3C}\hat{z}_t^u = z_{t+1}^u$. Therefore, we have $y_t^u \leq Q_t^u \leq z_t^u$ for all $t \geq t_k$.

Since we have $\lim_{t\to\infty} \|y_t^u - \tilde{Q}^u + \lambda D_k e\|_2 = 0$ and $\lim_{t\to\infty} \|z_t^u - \tilde{Q}^u - \lambda D_k e\|_2 = 0$, there exists a finite iteration number $\bar{t}_{k+1}^u$ such that $\bar{t}_{k+1}^u \geq t_k$, and $y_t^u - \tilde{Q}^u + \lambda D_k e \geq -\epsilon D_k e$ and $z_t^u - \tilde{Q}^u - \lambda D_k e \leq \epsilon D_k e$ for all $t \geq \bar{t}_{k+1}^u$. Then, using the fact that $y_t^u \leq Q_t^u \leq z_t^u$ for all $t \geq t_k$, we have $-D_{k+1}e = -(\lambda+\epsilon)D_k e \leq y_t^u - \tilde{Q}^u \leq Q_t^u - \tilde{Q}^u \leq z_t^u - \tilde{Q}^u \leq (\lambda+\epsilon)D_k e \leq D_{k+1}e$ for all $t \geq \bar{t}_{k+1}^u$. Letting $t_{k+1} = \max_{u\in\mathcal{U}} \bar{t}_{k+1}^u$, we have $\|Q_t - \tilde{Q}\| \leq D_{k+1}$ for all $t \geq t_{k+1}$. □

Using a standard argument, the monotone Q-learning algorithm can be posed in the form of the algorithm in Figure 3 by letting the operator Γ and the random error term $\omega_t^u(i)$ be defined as

$$[\Gamma Q_t]^u(i) = \sum_{j=1}^{n} p_{ij}(u)\Big[g(i,u,j) + \lambda \min_{v\in\mathcal{U}} Q_t^v(j)\Big] \tag{19}$$

$$\omega_t^u(i) = g(i,u,s_t) + \lambda \min_{v\in\mathcal{U}} Q_t^v(s_t) - \sum_{j=1}^{n} p_{ij}(u)\Big[g(i,u,j) + \lambda \min_{v\in\mathcal{U}} Q_t^v(j)\Big]. \tag{20}$$

With this definition of Γ and $\omega_t^u(i)$, (7) in Step 2 of the monotone Q-learning algorithm takes the form of (16) in Step 2 of the algorithm in Figure 3. Therefore, if we can show that (A.1)-(A.5) are satisfied by the monotone Q-learning algorithm, then the convergence of the monotone Q-learning algorithm follows from Proposition 3. We do this in the next section.

4 Verifying the Assumptions

In this section, we show that (A.1)-(A.5) are satisfied by the monotone Q-learning algorithm, as long as we choose the state-action pair and the step-size parameters at each iteration in a sensible way.

To ensure that (A.1) is satisfied, with probability p_t, we sample the state-action pair at iteration t from the uniform distribution over $\mathcal{S}\times\mathcal{U}$. If we have $\liminf_{t\to\infty} p_t > 0$, then (A.1) is satisfied. This approach is common in the learning literature and p_t is referred to as the exploration probability.

For any deterministic sequence $\{a_t\}$ that satisfies $\sum_{t=0}^{\infty} a_t = \infty$ and $\sum_{t=0}^{\infty} a_t^2 < \infty$, letting $\alpha_t^u(i) = a_t$ for all $i \in \mathcal{S}$, $u \in \mathcal{U}$ is sufficient to ensure that (A.2) is satisfied. Another common step-size parameter choice is to let

$$\alpha_t^u(i) = \frac{A}{B + C_t^u(i)} \qquad \text{for all } i \in \mathcal{S},\ u \in \mathcal{U}, \tag{21}$$

where A and B are positive scalars, and $C_t^u(i)$ is the number of times that state-action pair (i,u) has been sampled up to iteration t. The following lemma shows that (A.2) is also satisfied for this step-size parameter choice.

Lemma 4 *Let $\alpha_t^u(i)$ be as in* (21) *for all $i \in \mathcal{S}$, $u \in \mathcal{U}$. Then,* (A.2) *holds.*

Proof We fix $u \in \mathcal{U}$. We show by induction that $\min_{i\in\mathcal{S}} \alpha_t^u(i) \geq A/(B+t)$ for all $t = 0, 1, \ldots$ w.p.1. Since $C_0^u(i) = 0$ for all $i \in \mathcal{S}$, the result holds for $t = 0$. Assume that $\min_{i\in\mathcal{S}} \alpha_t^u(i) \geq A/(B+t)$, which implies that $C_t^u(i) \leq t$ for all $i \in \mathcal{S}$. Since $C_{t+1}^u(i) - C_t^u(i) \leq 1$, we have $C_{t+1}^u(i) \leq t+1$ for all $i \in \mathcal{S}$, which implies that $\min_{i\in\mathcal{S}} \alpha_{t+1}^u(i) \geq A/(B+t+1)$. This completes the induction argument and we obtain $\sum_{t=0}^{\infty} \min_{i\in\mathcal{S}} \alpha_t^u(i) \geq \sum_{t=0}^{\infty} A/(B+t) = \infty$ w.p.1. We now also fix $i \in \mathcal{S}$ and let $\mathcal{T}$ be the set of iterations at which the state-action pair (i, u) is sampled. We have

$$\sum_{t=0}^{\infty} \delta_t^u(i) \, [\alpha_t^u(i)]^2 = \sum_{t\in\mathcal{T}} \left[\frac{A}{B + C_t^u(i)}\right]^2 \leq \sum_{t=0}^{\infty} \left[\frac{A}{B+t}\right]^2 \leq \frac{A^2}{B^2} + \frac{(A\pi)^2}{6} \qquad \text{w.p.1,}$$

which implies that $\sum_{t=0}^{\infty} \mathbb{E}\{\delta_t^u(i) \, [\alpha_t^u(i)]^2\} < \infty$. □

We now consider (A.3). Since the successor state s_t is sampled such that $\mathbb{P}\{s_t = j \,|\, \mathcal{F}_t, i_t, u_t\} = p_{i_t j}(u_t)$, (20) implies that

$$\begin{aligned}
&\mathbb{E}\{\omega_t^{u_t}(i_t) \,|\, \mathcal{F}_t, i_t, u_t\} \\
&\quad = \sum_{j=1}^{n} P\{s_t = j \,|\, \mathcal{F}_t, i_t, u_t\} \Big[g(i_t, u_t, j) + \lambda \min_{v\in\mathcal{U}} Q_t^v(j)\Big] - \sum_{j=1}^{n} p_{i_t j}(u_t) \Big[g(i_t, u_t, j) + \lambda \min_{v\in\mathcal{U}} Q_t^v(j)\Big] = 0.
\end{aligned}$$

We let M be large enough so that $|g(i, u, j)| \leq M < \infty$ for all $i, j \in \mathcal{S}$, $u \in \mathcal{U}$. Since Q_t^u is the projection of R_{t-1}^u onto $\mathcal{P}^{-C,C}$, we have $Q_t^u \in \mathcal{P}^{-C,C}$ for all $u \in \mathcal{U}$. Then, (20) also implies that $[\omega_t^{u_t}(i_t)]^2 \leq (2M + 2\lambda C)^2$ w.p.1. Therefore, (A.3) is satisfied.

We now turn to (A.4). We let $Q \in \mathbb{R}^{n\times|\mathcal{U}|}$ be the solution to (4). If $|g(i, u, j)| \leq M < \infty$ for all $i, j \in \mathcal{S}$, $u \in \mathcal{U}$, then the infinite-horizon discounted cost of any policy is bounded by $M/(1-\lambda)$ and we have $|J(i)| \leq M/(1-\lambda)$ for all $i \in \mathcal{S}$. In this case, (3) implies that $|Q^u(i)| \leq M + \lambda[M/(1-\lambda)] = M/(1-\lambda)$ for all $i \in \mathcal{S}$, $u \in \mathcal{U}$. We assume that we are dealing with a problem where the Q-factors satisfy (6). Therefore, letting $C = M/(1-\lambda)$, we have $Q^u \in \mathcal{P}^{-C,C}$ for all $u \in \mathcal{U}$. For the particular choice of the operator Γ in (19), (4) implies that $Q = \Gamma Q$. It is also well-known that this operator is a contraction mapping with respect to the norm $\|\cdot\|$ and with contraction factor λ, which implies that $\|\Gamma Q_t - Q\| = \|\Gamma Q_t - \Gamma Q\| \leq \lambda \|Q_t - Q\|$ for all $Q_t \in \mathbb{R}^{n\times|\mathcal{U}|}$ (see, for example, Bertsekas and Tsitsiklis (1996)). Therefore, (A.4) is satisfied when we let $\tilde{Q} = Q$. We also emphasize that, with this choice of $\tilde{Q}$, Proposition 3 shows that $\{Q_t\}$ converges to the solution to (4) w.p.1, as desired.

We now show that (A.5) is satisfied. The following lemma is useful when we prove the final result in Lemma 6.

Lemma 5 *Assume that $y \in \mathcal{P}^{L,U}$. Fix $i^* \in \mathcal{S}$ and let $\hat{y} \in \mathbb{R}^n$ be obtained by*

$$\hat{y}(i) = \begin{cases} y(i) + \alpha\left[A - y(i)\right] & \text{if } i = i^* \\ y(i) & \text{otherwise,} \end{cases} \tag{22}$$

where A is a scalar and $\alpha \in [0,1]$. Let $v = \Pi^{L,U}\hat{y}$. Then, there exists $\ell \in \mathcal{S}$ satisfying the following.

1) *We have*

$$v(i) = \begin{cases} \dfrac{\hat{y}(i^*) + \ldots + \hat{y}(\ell)}{\ell - i^* + 1} \wedge U & \text{if } \ell > i^* \text{ and } i \in \{i^*, \ldots, \ell\} \\ \dfrac{\hat{y}(\ell) + \ldots + \hat{y}(i^*)}{i^* - \ell + 1} \vee L & \text{if } \ell < i^* \text{ and } i \in \{\ell, \ldots, i^*\} \\ [\hat{y}(i^*) \wedge U] \vee L & \text{if } \ell = i^* = i \\ \hat{y}(i) & \text{otherwise,} \end{cases} \tag{23}$$

where we use $a \wedge b = \min\{a, b\}$ and $a \vee b = \max\{a, b\}$.

2) *If $\ell > i^*$, then we have*

$$\hat{y}(i^*+1) \le \hat{y}(i^*+2) \le \ldots \le \hat{y}(\ell) \le \frac{\hat{y}(i^*) + \ldots + \hat{y}(\ell)}{\ell - i^* + 1} \le \hat{y}(i^*). \tag{24}$$

If $\ell < i^*$, then we have

$$\hat{y}(i^*) \le \frac{\hat{y}(\ell) + \ldots + \hat{y}(i^*)}{i^* - \ell + 1} \le \hat{y}(\ell) \le \hat{y}(\ell+1) \le \ldots \le \hat{y}(i^*-1). \tag{25}$$

Proof See the online supplement. □

Therefore, (23) implies that if $\hat{y}$ is obtained by (22), then the result of the projection $\Pi^{L,U}\hat{y}$ can be characterized simply by specifying ℓ and i^*. The following lemma shows that (A.5) is satisfied.

Lemma 6 (A.5) *holds.*

Proof We let $L_1, L_2, U_1, U_2, y, z, A, B, i^*, \hat{y}$ and $\hat{z}$ be as defined in (A.5). We let $v = \Pi^{L_1,U_1}\hat{y}$ and $w = \Pi^{L_2,U_2}\hat{z}$. We want to show that $v \le w$. By Lemma 5, there exist ℓ and $\kappa \in \mathcal{S}$ such that

$$v(i) = \begin{cases} \dfrac{\hat{y}(i^*) + \ldots + \hat{y}(\ell)}{\ell - i^* + 1} \wedge U_1 & \text{if } \ell > i^* \text{ and } i \in \{i^*, \ldots, \ell\} \\ \dfrac{\hat{y}(\ell) + \ldots + \hat{y}(i^*)}{i^* - \ell + 1} \vee L_1 & \text{if } \ell < i^* \text{ and } i \in \{\ell, \ldots, i^*\} \\ [\hat{y}(i^*) \wedge U_1] \vee L_1 & \text{if } \ell = i^* = i \\ \hat{y}(i) & \text{otherwise} \end{cases} \tag{26}$$

$$w(i) = \begin{cases} \dfrac{\hat{z}(i^*) + \ldots + \hat{z}(\kappa)}{\kappa - i^* + 1} \wedge U_2 & \text{if } \kappa > i^* \text{ and } i \in \{i^*, \ldots, \kappa\} \\ \dfrac{\hat{z}(\kappa) + \ldots + \hat{z}(i^*)}{i^* - \kappa + 1} \vee L_2 & \text{if } \kappa < i^* \text{ and } i \in \{\kappa, \ldots, i^*\} \\ [\hat{z}(i^*) \wedge U_2] \vee L_2 & \text{if } \kappa = i^* = i \\ \hat{z}(i) & \text{otherwise.} \end{cases} \tag{27}$$

1	2	3	4	5	6	7	8	9
$\ell = i^*$	$\ell = i^*$	$\ell = i^*$	$\ell > i^*$	$\ell > i^*$	$\ell > i^*$	$\ell < i^*$	$\ell < i^*$	$\ell < i^*$
$\kappa = i^*$	$\kappa > i^*$	$\kappa < i^*$	$\kappa = i^*$	$\kappa > i^*$	$\kappa < i^*$	$\kappa = i^*$	$\kappa > i^*$	$\kappa < i^*$

Table 1: List of cases that need to be considered in the proof of Lemma 6.

Our proof requires investigating the nine cases listed in Table 1. For brevity, we investigate only Case 5 here. The other cases can be handled using similar arguments.

Case 5 – Assume that $\ell > i^*$ and $\kappa > i^*$. Letting $\bar{v} = [\hat{y}(i^*) + \ldots + \hat{y}(\ell)]/[\ell - i^* + 1]$ and $\bar{w} = [\hat{z}(i^*) + \ldots + \hat{z}(\kappa)]/[\kappa - i^* + 1]$, (24) implies that

$$\hat{y}(i^*+1) \leq \hat{y}(i^*+2) \leq \ldots \leq \hat{y}(\ell) \leq \bar{v} \leq \hat{y}(i^*) \tag{28}$$

$$\hat{z}(i^*+1) \leq \hat{z}(i^*+2) \leq \ldots \leq \hat{z}(\kappa) \leq \bar{w} \leq \hat{z}(i^*). \tag{29}$$

Since $y \leq z$, $A \leq B$ and $\alpha \in [0,1]$, we have $\hat{y} \leq \hat{z}$. Since v is the projection of $\hat{y}$ onto $\mathcal{P}^{L_1,U_1}$, we have $v \in \mathcal{P}^{L_1,U_1}$. Similarly, we have $w \in \mathcal{P}^{L_2,U_2}$. We consider the following three subcases.

Case 5.a – Assume that $\ell < \kappa$. Since $\ell > i^*$, (26) implies that $v(\ell) = \bar{v} \wedge U_1$ and $v(\ell+1) = \hat{y}(\ell+1)$. Since $v \in \mathcal{P}^{L_1,U_1}$, we have $v(\ell) \leq v(\ell+1)$. Since $i^* < \ell < \kappa$, (29) implies that $\hat{z}(\ell+1) \leq \hat{z}(\ell+2) \leq \ldots \leq \hat{z}(\kappa)$. Therefore, using the fact that $\hat{y} \leq \hat{z}$, we have $\bar{v} \wedge U_1 = v(\ell) \leq v(\ell+1) = \hat{y}(\ell+1) \leq \hat{z}(\ell+1) \leq \hat{z}(\ell+2) \leq \ldots \leq \hat{z}(\kappa)$. Then, we obtain

$$\begin{aligned}
\bar{w} &= \frac{\ell - i^* + 1}{\kappa - i^* + 1} \, \frac{\hat{z}(i^*) + \ldots + \hat{z}(\ell)}{\ell - i^* + 1} + \frac{\kappa - \ell}{\kappa - i^* + 1} \, \frac{\hat{z}(\ell+1) + \ldots + \hat{z}(\kappa)}{\kappa - \ell} \\
&\geq \frac{\ell - i^* + 1}{\kappa - i^* + 1} \, \frac{\hat{y}(i^*) + \ldots + \hat{y}(\ell)}{\ell - i^* + 1} + \frac{\kappa - \ell}{\kappa - i^* + 1} \, \frac{\hat{z}(\ell+1) + \ldots + \hat{z}(\kappa)}{\kappa - \ell} \\
&\geq \frac{\ell - i^* + 1}{\kappa - i^* + 1} \, \bar{v} + \frac{\kappa - \ell}{\kappa - i^* + 1} \, [\bar{v} \wedge U_1] \geq \bar{v} \wedge U_1,
\end{aligned}$$

where the first inequality uses the fact that $\hat{z} \geq \hat{y}$. Therefore, since $U_2 \geq U_1$, we have $\bar{w} \wedge U_2 \geq \bar{v} \wedge U_1$.

Since $\ell > i^*$, $z \in \mathcal{P}^{L_2,U_2}$ and $\hat{z}$ differs from z only in the i^*-th component, we have $\hat{z}(\ell+1) \leq \hat{z}(\ell+2) \leq \ldots \leq \hat{z}(\kappa) \leq U_2$. Since $\ell > i^*$, (29) implies that $\hat{z}(\ell+1) \leq \hat{z}(\ell+2) \ldots \leq \hat{z}(\kappa) \leq \bar{w}$. Therefore, we have $\hat{z}(\ell+1) \leq \hat{z}(\ell+2) \leq \ldots \leq \hat{z}(\kappa) \leq \bar{w} \wedge U_2$. Then, (26) and (27) imply that

$$v(i) = \begin{cases} \bar{v} \wedge U_1 \leq \bar{w} \wedge U_2 = w(i) & \text{if } i \in \{i^*, \ldots, \ell\} \\ \hat{y}(i) \leq \hat{z}(i) \leq \bar{w} \wedge U_2 = w(i) & \text{if } i \in \{\ell+1, \ldots, \kappa\} \\ \hat{y}(i) \leq \hat{z}(i) = w(i) & \text{otherwise,} \end{cases}$$

which shows that $v \leq w$.

Case 5.b – Assume that $\ell = \kappa$. Since $\hat{y} \leq \hat{z}$ and $U_1 \leq U_2$, we have $\bar{v} \wedge U_1 \leq \bar{w} \wedge U_2$ and the result immediately follows from (26) and (27).

Case 5.c – Assume that $\kappa < \ell$. Since $i^* < \kappa < \ell$, (28) implies that $\hat{y}(\kappa+1) \leq \hat{y}(\kappa+2) \leq \ldots \leq \hat{y}(\ell) \leq \bar{v}$. Then, using the fact that $\hat{y} \leq \hat{z}$, we obtain

$$\begin{aligned}\bar{v} - \bar{v} &= \frac{\kappa - i^* + 1}{\ell - i^* + 1} \frac{\hat{y}(i^*) + \ldots + \hat{y}(\kappa)}{\kappa - i^* + 1} + \frac{\ell - \kappa}{\ell - i^* + 1} \frac{\hat{y}(\kappa+1) + \ldots + \hat{y}(\ell)}{\ell - \kappa} - \bar{v} \\ &\leq \frac{\kappa - i^* + 1}{\ell - i^* + 1} \frac{\hat{z}(i^*) + \ldots + \hat{z}(\kappa)}{\kappa - i^* + 1} + \frac{\ell - \kappa}{\ell - i^* + 1} \bar{v} - \bar{v} = \frac{\kappa - i^* + 1}{\ell - i^* + 1} \bar{w} - \frac{\kappa - i^* + 1}{\ell - i^* + 1} \bar{v},\end{aligned}$$

which implies that $\bar{w} \geq \bar{v}$. Since $U_2 \geq U_1$, we have $\bar{w} \wedge U_2 \geq \bar{v} \wedge U_1$.

Since $\kappa > i^*$, (27) implies that $w(\kappa) = \bar{w} \wedge U_2$ and $w(\kappa+1) = \hat{z}(\kappa+1)$. Since $w \in \mathcal{P}^{L_2,U_2}$, we have $w(\kappa) \leq w(\kappa+1)$. Since $\kappa > i^*$, $z \in \mathcal{P}^{L_2,U_2}$ and $\hat{z}$ differs from z only in the i^*-th component, we have $\hat{z}(\kappa+1) \leq \hat{z}(\kappa+2) \leq \ldots \leq \hat{z}(\ell)$. Therefore, we have $\bar{w} \wedge U_2 = w(\kappa) \leq w(\kappa+1) = \hat{z}(\kappa+1) \leq \hat{z}(\kappa+2) \leq \ldots \leq \hat{z}(\ell)$. Then, (26) and (27) imply that

$$v(i) = \begin{cases} \bar{v} \wedge U_1 \leq \bar{w} \wedge U_2 = w(i) & \text{if } i \in \{i^*, \ldots, \kappa\} \\ \bar{v} \wedge U_1 \leq \bar{w} \wedge U_2 \leq \hat{z}(i) = w(i) & \text{if } i \in \{\kappa+1, \ldots, \ell\} \\ \hat{y}(i) \leq \hat{z}(i) = w(i) & \text{otherwise,} \end{cases}$$

which shows that $v \leq w$. □

Therefore, Proposition 3 and the discussion in this section show that the iterates of the monotone Q-learning algorithm converge to the solution to (4) w.p.1, as long as we choose the state-action pair and the step-size parameters at each iteration in such a way that (A.1) and (A.2) are satisfied.

5 Computational Experiments

In this section, we test the performance of the monotone Q-learning algorithm on two batch service problems. Our objective is to show how much the convergence behavior of the Q-learning algorithm can be improved by exploiting the structural properties of the underlying Markov decision problem.

5.1 A single-product batch service problem

This problem arises in situations where a service station with finite capacity is used to serve the products that arrive randomly over time. At each time period, we decide whether we run the service station. If we run the service station, then the waiting products receive service and leave the system. If we do not run the service station or the number of waiting products is greater than the capacity of the service station, then the products that are not served are held until the next time period. The capacity of the service station is K. The cost of running the service station is R. The holding cost is h per time period per product. We assume that $(Kh)/(1-\lambda) > R$ so that the cost of holding K products for infinite number of time periods is greater than the cost of running the service station. This ensures that it is optimal to run the service station when there are K or more waiting products.

We use the number of waiting products as the state. We assume that it is not possible to hold more than $2K$ products in the system and we simply lose the products that arrive into a full system. This ensures that we can bound the number of waiting products by $2K$ and use $\{0,\ldots,2K\}$ as the state space. The set of actions is $\{0,1\}$, where actions 0 and 1 respectively correspond to not running and running the service station. Letting A be the random variable representing the number of product arrivals at each time period, the transition probabilities are given by

$$p_{ij}(u) = \begin{cases} \mathbb{P}\{A = j-i\} & \text{if } u = 0 \text{ and } i \leq j < 2K \\ \mathbb{P}\{A \geq 2K - i\} & \text{if } u = 0 \text{ and } i \leq j = 2K \\ \mathbb{P}\{A = j - [i-K]^+\} & \text{if } u = 1 \text{ and } [i-K]^+ \leq j < 2K \\ \mathbb{P}\{A \geq 2K - [i-K]^+\} & \text{if } u = 1 \text{ and } [i-K]^+ \leq j = 2K \\ 0 & \text{otherwise,} \end{cases} \tag{30}$$

where we let $[a-b]^+ = \max\{a-b,0\}$ and use the fact that the arriving products that find $2K$ products in the system are lost. It is interesting that due to the finite capacity of the system, we may not observe all of the arrivals during the time periods at which the number of arrivals exceeds the available capacity. Therefore, if the system operates under a policy that tends to hold a large number of products in the system before serving them, then it is difficult to obtain unbiased samples of the arrival random variables. This is an example of the censored data problem described in the introduction. It is certainly intriguing to investigate the quality of the solutions that are obtained by ignoring the censored data problem, but we leave this issue aside since our focus is on showing how much the convergence behavior of the Q-learning algorithm can be improved by exploiting the structural properties of the underlying Markov decision problem. The costs are given by

$$g(i,u,j) = \begin{cases} h\,i & \text{if } u = 0 \\ R + h\,[i-K]^+ & \text{if } u = 1. \end{cases}$$

This problem has been studied extensively and it is known that its Q-factors satisfy (6) (see Ignall and Kolesar (1972), Kosten (1973), Deb and Serfozo (1973), Ignall and Kolesar (1974) and Papadaki and Powell (2002)).

We compare the performance of the monotone Q-learning algorithm (MQL) with that of the standard version of the Q-learning algorithm (SQL). We note that we obtain SQL by letting $Q^u_{t+1} = R^u_t$ for all $u \in \mathcal{U}$ in Step 3 of the algorithm in Figure 1. We apply MQL and SQL for 10,000 iterations. We sample the state-action pair at iteration t by following the greedy policy d_t defined as

$$d_t(i) = \begin{cases} \text{argmin}_{u\in\{0,1\}}\ Q^u_t(i) & \text{if } i \in \{0,\ldots,K-1\} \\ 1 & \text{otherwise.} \end{cases} \tag{31}$$

Since we know that it is optimal to run the service station when there are K or more waiting products, we let $d_t(i) = 1$ for all $i \in \{K,\ldots,2K\}$. Therefore, letting i_0 be an arbitrary initial state, the state-action pair sampled at iteration 0 is $(i_0, d_0(i_0))$. The successor state s_0 is sampled such

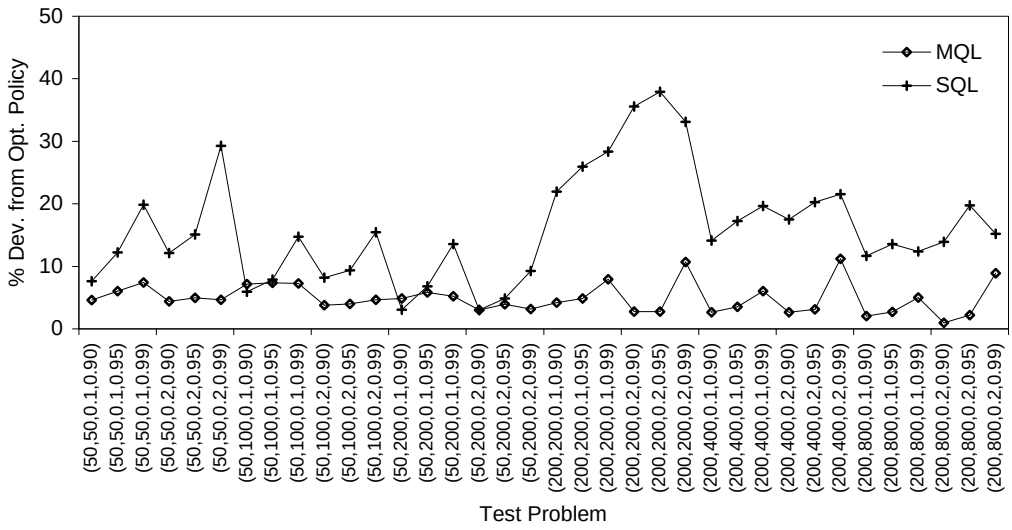

Figure 4: Performances of the greedy policies obtained by MQL and SQL after 10,000 iterations for different test problems.

that $\mathbb{P}\{s_0 = j \,|\, \mathcal{F}_0, i_0, d_0(i_0)\} = p_{i_0 j}(d_0(i_0))$. Then, we let $i_1 = s_0$ and the state-action pair sampled at iteration 1 is $(i_1, d_1(i_1))$. The state-action pairs at the subsequent iterations are sampled in a similar way. To ensure that (A.1) is satisfied, at any iteration, we stop following the greedy policy with probability 0.1 and sample the state-action pair at that iteration from the uniform distribution over $\{0, \ldots, 2K\} \times \{0, 1\}$.

We let $\{Q^u(i) : i \in \{0, \ldots, 2K\},\ u \in \{0, 1\}\}$ be the solution to (4) and d^* be the optimal policy. Since we have $d^*(i) = \text{argmin}_{u \in \{0,1\}} Q^u(i)$, (31) implies that if $Q^u_t(i)$ is "close to" $Q^u(i)$ for all $u \in \{0, 1\}$, then we have $d_t(i) = d^*(i)$. Therefore, we compare the convergence behavior of MQL and SQL by comparing the performances of the greedy policies obtained by the two algorithms. To evaluate the performance of the greedy policy d_t, we let $J^{d_t}(i)$ be the infinite-horizon discounted cost incurred by starting from state i and using policy d_t, and compute this quantity by solving

$$J^{d_t}(i) = \sum_{j=0}^{2K} p_{ij}(d_t(i)) \big[g(i, d_t(i), j) + \lambda\, J^{d_t}(j) \big] \qquad \text{for all } i \in \{0, \ldots, 2K\}.$$

We assume that the number of product arrivals at each time period has a geometric distribution with parameter ρ. We let $h = 1$ throughout, and vary K, R, ρ and λ to obtain different test problems. Figure 4 shows the performances of the greedy policies obtained by MQL and SQL after 10,000 iterations for different test problems. We label each test problem by (K, R, ρ, λ) on the horizontal axis. Letting $J(i)$ be the optimal infinite-horizon discounted cost incurred by starting from state i, the performance measure plotted in Figure 4 is $\max_{i \in \{0,\ldots,2K\}} \big[J^{d_{10,000}}(i) - J(i) \big] / J(i)$, where $d_{10,000}$ is the greedy policy obtained after 10,000 iterations. We try to eliminate the effect of noise by making 100 runs for MQL and SQL under common random numbers and present the

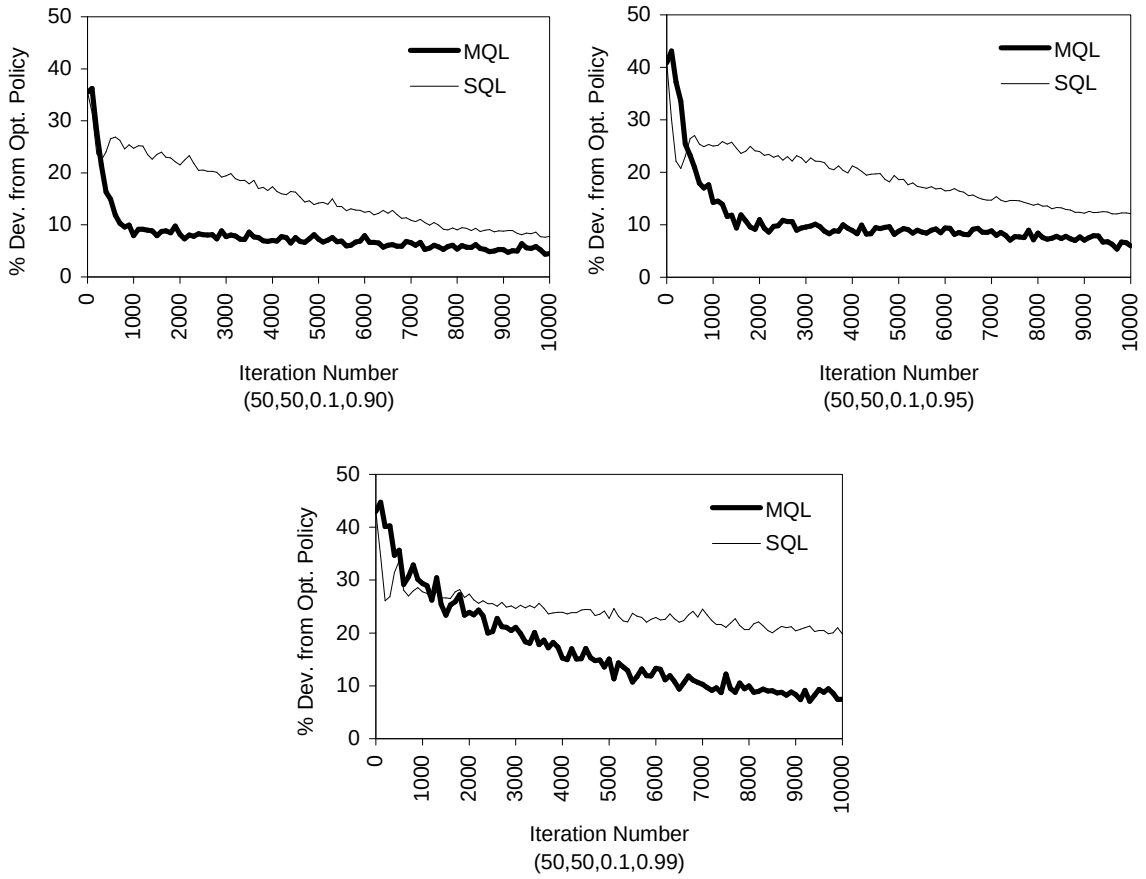

Figure 5: Performances of the greedy policies obtained by MQL and SQL as a function of the iteration number for test problems $(50, 50, 0.1, 0.90)$, $(50, 50, 0.1, 0.95)$ and $(50, 50, 0.1, 0.99)$.

averages of the results over 100 runs. For all of our test problems, it takes less than a second to compute $\{J(i) : i \in \{0, \ldots, 2K\}\}$ by using the value iteration algorithm, but this is naturally under the assumption that all problem parameters are known.

For a majority of the test problems, the greedy policy obtained by MQL performs significantly better than the greedy policy obtained by SQL. The performance gap between MQL and SQL becomes more noticeable when K is large. Exploiting the known structural properties of the problem appears to be particularly helpful when we need to approximate a larger number of Q-factors.

Figure 5 shows $\max_{i \in \{0,\ldots,2K\}} \left[J^{d_t}(i) - J(i)\right]/J(i)$ as a function of the iteration number t. Each chart in this figure corresponds to a different value for the discount factor. The performance gap between MQL and SQL is significant especially in the early iterations. As the discount factor gets large, the performance gap between MQL and SQL becomes more pronounced. It is well-known that infinite-horizon discounted-cost Markov decision problems get more difficult as the discount factor approaches 1, and it is encouraging that MQL significantly improves the performance of SQL when the discount factor is large.

5.2 A two-product batch service problem

In this problem, two types of products have to be served by a single service station. We refer to the two product types as type 1 and type 2. The holding costs for the products of type 1 and type 2 are respectively h_1 and h_2. We assume that $h_1 > h_2$ so that if the total number of waiting products is greater than the capacity of the service station, then it is optimal to serve the products of type 1 first. We also assume that $(Kh_2)/(1-\lambda) > R$ so that it is optimal to run the service station when there are K or more waiting products. This problem is the infinite-horizon version of the problem studied by Papadaki and Powell (2003).

We assume that it is not possible to hold more than $2K$ products of each type in the system so that we can use the pair $(i,\kappa) \in \{0,\ldots,2K\} \times \{0,\ldots,2K\}$ as the state. Similar to (30), letting A_1 and A_2 respectively be the random variables representing the numbers of product arrivals of type 1 and type 2 at each time period, the transition probabilities are given by

$$
\begin{aligned}
&p_{(i,\kappa),(j,\ell)}(u) \\
&\quad = \begin{cases}
\mathbb{P}\{A_1 = j-i,\ A_2 = \ell-\kappa\} & \text{if } u=0,\ i \le j < 2K \text{ and } \kappa \le \ell < 2K \\
\mathbb{P}\{A_1 \ge 2K-i,\ A_2 = \ell-\kappa\} & \text{if } u=0,\ i \le j = 2K \text{ and } \kappa \le \ell < 2K \\
\mathbb{P}\{A_1 = j-i,\ A_2 \ge 2K-\kappa\} & \text{if } u=0,\ i \le j < 2K \text{ and } \kappa \le \ell = 2K \\
\mathbb{P}\{A_1 \ge 2K-i,\ A_2 \ge 2K-\kappa\} & \text{if } u=0,\ i \le j = 2K \text{ and } \kappa \le \ell = 2K \\
\mathbb{P}\{A_1 = j-[i-K]^+,\ A_2 = \ell-[\kappa-[K-i]^+]^+\} & \text{if } u=1,\ [i-K]^+ \le j < 2K \\
& \qquad \text{and } [\kappa-[K-i]^+]^+ \le \ell < 2K \\
\mathbb{P}\{A_1 \ge 2K-[i-K]^+,\ A_2 = \ell-[\kappa-[K-i]^+]^+\} & \text{if } u=1,\ [i-K]^+ \le j = 2K \\
& \qquad \text{and } [\kappa-[K-i]^+]^+ \le \ell < 2K \\
\mathbb{P}\{A_1 = j-[i-K]^+,\ A_2 \ge 2K-[\kappa-[K-i]^+]^+\} & \text{if } u=1,\ [i-K]^+ \le j < 2K \\
& \qquad \text{and } [\kappa-[K-i]^+]^+ \le \ell = 2K \\
\mathbb{P}\{A_1 \ge 2K-[i-K]^+,\ A_2 \ge 2K-[\kappa-[K-i]^+]^+\} & \text{if } u=1,\ [i-K]^+ \le j = 2K \\
& \qquad \text{and } [\kappa-[K-i]^+]^+ \le \ell = 2K \\
0 & \text{otherwise,}
\end{cases}
\end{aligned}
$$

where we use the fact if the total number of waiting products is greater than the capacity of the service station, then it is optimal to serve the products of type 1 first. The costs are given by

$$
g((i,\kappa),u,(j,\ell)) = \begin{cases} h_1\, i + h_2\, \kappa & \text{if } u=0 \\ R + h_1\,[i-K]^+ + h_2\left[\kappa-[K-i]^+\right]^+ & \text{if } u=1. \end{cases}
$$

Following the approach in Papadaki and Powell (2003), we can show that the Q-factors for this problem satisfy

$$
Q^u((i,\kappa)) \le Q^u((i+1,\kappa)) \qquad \text{for all } i \in \{0,\ldots,2K-1\},\ \kappa \in \{0,\ldots,2K\},\ u \in \{0,1\} \tag{32}
$$

$$
Q^u((i,\kappa)) \le Q^u((i,\kappa+1)) \qquad \text{for all } i \in \{0,\ldots,2K\},\ \kappa \in \{0,\ldots,2K-1\},\ u \in \{0,1\}. \tag{33}
$$

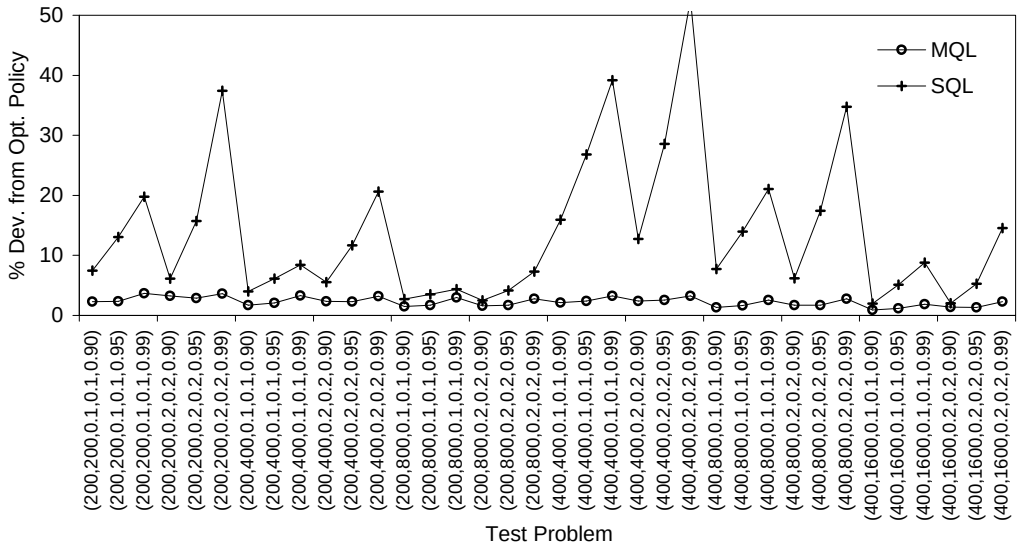

Figure 6: Performances of the greedy policies obtained by MQL and SQL after 1,000,000 iterations for different test problems.

Unfortunately, MQL cannot impose both of these properties on the Q-factor approximations obtained during the intermediate iterations and we arbitrarily choose to impose the first property. That is, letting $\{Q_t^u((i,\kappa)) : i,\kappa \in \{0,\ldots,2K\},\ u \in \{0,1\}\}$ be the Q-factor approximations obtained at iteration t, $Q_t^u(\kappa)$ be the vector $\{Q_t^u((i,\kappa)) : i \in \{0,\ldots,2K\}\}$ and $\mathcal{P}^{-C,C}$ be the set $\{y \in \mathbb{R}^{2K+1} : -C \leq y(0) \leq y(1) \leq \ldots \leq y(2K) \leq C\}$ for some large scalar C, we project $Q_t^u(\kappa)$ onto $\mathcal{P}^{-C,C}$ when $Q_t^u(\kappa) \notin \mathcal{P}^{-C,C}$ for some $\kappa \in \{0,\ldots,2K\}$ and $u \in \{0,1\}$.

We assume that the numbers of product arrivals of type 1 and type 2 at each time period are independent and have geometric distributions respectively with parameters ρ_1 and ρ_2. We let $h_1 = 1$ and $h_2 = 0.5$ throughout, and vary K, R, ρ_1, ρ_2 and λ to obtain different test problems. Figure 6 shows the performances of the greedy policies obtained by MQL and SQL after 1,000,000 iterations for different test problems. We label each test problem by $(K, R, \rho_1, \rho_2, \lambda)$ on the horizontal axis. The performance measure that we use in this figure is the same as the one in Figure 4 with obvious modifications to accommodate the two-dimensional state variable.

For all of the test problems, the greedy policy obtained by MQL performs better than the greedy policy obtained by SQL and the performance gap becomes more pronounced as the discount factor approaches 1. The performance gap between MQL and SQL can be as large as 40% for the test problems where the discount factor is close to 1.

To give a feel for the computational effort required to solve the test problems through conventional dynamic programming tools, Figure 7 shows the runtimes to compute the value functions by using the value iteration algorithm. We run the value iteration algorithm until the maximum change in

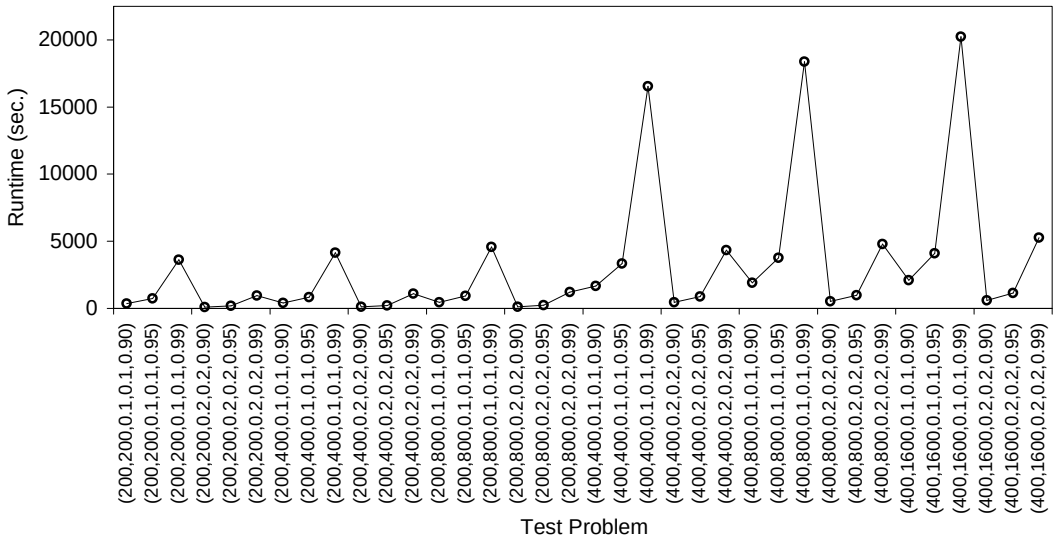

Figure 7: Runtimes of the value iteration algorithm for different test problems.

the value functions in two successive iterations is less than 0.1% and use the Gauss-Seidel variant (see Puterman (1994)). As expected, the runtimes increase with the size of the state space, but more interestingly, the discount factor also plays a significant role in the runtimes. In particular, the discount factor affects the number of iterations until termination. For comparison purpose, we note that 1,000,000 iterations for both MQL and SQL can be carried out in a few seconds. Nevertheless, we emphasize that the Q-learning algorithm should generally not be used when all problem parameters are known and the size of the state space is small enough to allow using conventional dynamic programming tools. The power of the Q-learning algorithm lies in its nonparametric and sampling-based nature.

6 Conclusions

In this paper, we proposed a new variant of the Q-learning algorithm applicable to problems whose Q-factors are known to be monotone in the state. We established the convergence of this algorithm and experimentally showed that it can significantly improve the convergence behavior of the standard version of the Q-learning algorithm. The idea of exploiting the known structural properties of Markov decision problems to improve the performances of the learning algorithms forms a rich area of research. Extending our work to problems whose Q-factors satisfy other properties, such as convexity and K-convexity, is a line of research we are pursuing. For problems with multi-dimensional state variables, projecting the Q-factor approximations onto a certain set can be computationally prohibitive. Finding efficient remedies for this difficulty is another avenue of investigation. Finally, imposing multiple properties on the Q-factor approximations can improve the convergence behavior

of the monotone Q-learning algorithm even further. For example, for the two-product batch service problem, we know that the Q-factors satisfy the properties in (32) and (33), but we only imposed the property in (32) on the Q-factor approximations. Clearly, we can project the Q-factor approximations onto the set

$$\{Q \in \mathbb{R}^{2K\times 2K\times|\mathcal{U}|}:$$
$$L \leq Q^u((0,\kappa)) \leq Q^u((1,\kappa)) \leq \ldots \leq Q^u((2K,\kappa)) \leq U \quad \text{for all } \kappa \in \{0,\ldots,2K\},\ u \in \{0,1\}$$
$$L \leq Q^u((i,0)) \leq Q^u((i,1)) \leq \ldots \leq Q^u((i,2K)) \leq U \quad \text{for all } i \in \{0,\ldots,2K\},\ u \in \{0,1\}\}.$$

However, it is not clear that a projection operator onto this set satisfies (A.5) and the proof of Proposition 3 does not immediately apply. Finding ways of imposing multiple properties on the Q-factor approximations remains a challenging research question.

References

Andradottir, S. (1995), 'A stochastic approximation algorithm with varying bounds', *Operations Research* **43**(6), 1037–1048.

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

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.

Ding, X. (2002), Estimation and optimization in discrete inventory models, PhD thesis, The University of British Columbia, Vancouver, British Columbia.

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.

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

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

Neveu, J. (1975), *Discrete Parameter Martingales*, North Holland, Amsterdam.

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., 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.

Schweitzer, P. and Seidmann, A. (1985), 'Generalized polynomial approximations in Markovian decision processes', *Journal of Mathematical Analysis and Applications* **110**, 568–582.

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. and Powell, W. B. (2006), 'Dynamic programming approximations for stochastic, time-staged integer multicommodity flow problems', *INFORMS Journal on Computing* **18**(1), 31–42.

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

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

Watkins, C. J. C. H. (1989), Learning from Delayed Rewards, PhD thesis, Cambridge University, Cambridge, England.

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

7 Online Supplement

In this section, we prove Lemmas 2 and 5. Our proof of Lemma 2 uses the following generalization of the supermartingale convergence theorem (see, for example, Neveu (1975)).

Proposition 7 *Let $\{X_t\}, \{Y_t\}$ and $\{Z_t\}$ be sequences of positive random variables adopted to the filtration $\{\mathcal{G}_t\}$. Assume that $\mathbb{E}\{X_{t+1} \,|\, \mathcal{G}_t\} \leq X_t - Y_t + Z_t$ for all $t = 0, 1, \ldots$ and $\sum_{t=0}^{\infty} Z_t < \infty$ w.p.1. Then, $\{X_t\}$ converges to a positive random variable and we have $\sum_{t=0}^{\infty} Y_t < \infty$ w.p.1.*

We now prove Lemma 2.

Proof of Lemma 2 Letting R_t and $G_t \in \mathbb{R}^{n \times n}$ be the diagonal matrices whose diagonal components are respectively $\{\rho_t(i) : i \in \mathcal{S}\}$ and $\{\gamma_t(i) : i \in \mathcal{S}\}$, (9) can be written in vector notation as $\hat{y}_t = y_t + R_t\, G_t\big[\, y^* + \theta_t - y_t\big]$. By the nonexpansiveness of the projection operator $\Pi^{L,U}$, we have

$$\|y_{t+1} - y^*\|_2^2 \leq \|\hat{y}_t - y^*\|_2^2 = \|y_t - y^*\|_2^2 - 2\big\langle y_t - y^*, R_t\, G_t\big[\, y_t - y^*\big]\big\rangle + 2\big\langle y_t - y^*, R_t\, G_t\, \theta_t\big\rangle + \|R_t\, G_t\big[\, y^* + \theta_t - y_t\big]\|_2^2. \quad (34)$$

Letting P_t be the diagonal matrix whose diagonal components are $\{\mathbb{P}\{i_t = i \,|\, \mathcal{G}_t\} : i \in \mathcal{S}\}$, and noting the fact that y_t is $\mathcal{G}_t$-measurable and $\mathbb{E}\big\{\rho_t(i)\ \gamma_t(i) \,|\, \mathcal{G}_t\big\} = \mathbb{P}\{i_t = i \,|\, \mathcal{G}_t\}\ \gamma_t(i)$, we have

$$\mathbb{E}\big\{\big\langle y_t - y^*, R_t\, G_t\big[\, y_t - y^*\big]\big\rangle \,|\, \mathcal{G}_t\big\} = \big\langle y_t - y^*, \mathbb{E}\big\{R_t\, G_t \,|\, \mathcal{G}_t\big\}[y_t - y^*]\big\rangle = \big\langle y_t - y^*, P_t\, G_t\big[\, y_t - y^*\big]\big\rangle. \quad (35)$$

Noting (10) and (14), we have

$$\mathbb{E}\big\{\big\langle y_t - y^*, R_t\, G_t\, \theta_t\big\rangle \,|\, \mathcal{G}_t, i_t\big\} = \mathbb{E}\big\{\big[\, y_t(i_t) - y^*(i_t)\big]\gamma_t(i_t)\, \theta_t(i_t) \,|\, \mathcal{G}_t, i_t\big\} = \big[\, y_t(i_t) - y^*(i_t)\big]\gamma_t(i_t)\, \mathbb{E}\big\{\theta_t(i_t) \,|\, \mathcal{G}_t, i_t\big\} = 0 \qquad \text{w.p.1,}$$

which implies that

$$\mathbb{E}\big\{\big\langle y_t - y^*, R_t\, G_t\, \theta_t\big\rangle \,|\, \mathcal{G}_t\big\} = \mathbb{E}\Big\{\mathbb{E}\big\{\big\langle y_t - y^*, R_t\, G_t\, \theta_t\big\rangle \,|\, \mathcal{G}_t, i_t\big\} \,|\, \mathcal{G}_t\Big\} = 0 \qquad \text{w.p.1.} \quad (36)$$

Since y_t is the projection of $\hat{y}_{t-1}$ onto $\mathcal{P}^{L,U}$, we have $y_t \in \mathcal{P}^{L,U}$. Then, (14), (15) and the fact that $y^* \in \mathcal{P}^{L,U}$ imply that

$$\begin{aligned}
\mathbb{E}\big\{\|R_t\, G_t\big[\, y^* + \theta_t - y_t\big]\|_2^2 \,|\, \mathcal{G}_t, i_t\big\} &= \mathbb{E}\big\{\|R_t\, G_t\big[\, y^* - y_t\big]\|_2^2 \,|\, \mathcal{G}_t, i_t\big\} \\
&\quad + 2\,\mathbb{E}\big\{\big\langle R_t\, G_t\big[\, y^* - y_t\big], R_t\, G_t\, \theta_t\big\rangle \,|\, \mathcal{G}_t, i_t\big\} + \mathbb{E}\big\{\|R_t\, G_t\, \theta_t\|_2^2 \,|\, \mathcal{G}_t, i_t\big\} \\
&= [\gamma_t(i_t)]^2\big[\, y^*(i_t) - y_t(i_t)\big]^2 + 2\,[\gamma_t(i_t)]^2\big[\, y^*(i_t) - y_t(i_t)\big]\,\mathbb{E}\big\{\theta_t(i_t) \,|\, \mathcal{G}_t, i_t\big\} + [\gamma_t(i_t)]^2\,\mathbb{E}\big\{[\theta_t(i_t)]^2 \,|\, \mathcal{G}_t, i_t\big\} \\
&\leq [\gamma_t(i_t)]^2\,\big\{4\,[|L| \vee |U|]^2 + A\big\} \qquad \text{w.p.1,}
\end{aligned}$$

where we use $a \vee b = \max\{a, b\}$. Taking the expectation of the expression above conditional on $\mathcal{G}_t$, we have

$$\mathbb{E}\big\{\|R_t\, G_t\big[\, y^* + \theta_t - y_t\big]\|_2^2 \,|\, \mathcal{G}_t\big\} \leq \mathbb{E}\big\{[\gamma_t(i_t)]^2 \,|\, \mathcal{G}_t\big\}\, \big\{4\,[|L| \vee |U|]^2 + A\big\} \qquad \text{w.p.1.} \tag{37}$$

Taking the expectation of (34) conditional on $\mathcal{G}_t$ and using (35)-(37), we obtain

$$\begin{aligned}\mathbb{E}\big\{\|y_{t+1} - y^*\|_2^2 \,|\, \mathcal{G}_t\big\} \\ \leq \|y_t - y^*\|_2^2 - 2\big\langle y_t - y^*, P_t\, G_t\big[\, y_t - y^*\big]\big\rangle + \mathbb{E}\big\{[\gamma_t(i_t)]^2 \,|\, \mathcal{G}_t\big\}\, \big\{4\,[|L| \vee |U|]^2 + A\big\}\,.\end{aligned} \tag{38}$$

On the other hand, using (13) and appealing to the monotone convergence theorem, we have

$$\begin{aligned}\mathbb{E}\left\{\sum_{t=0}^{\infty} \mathbb{E}\big\{[\gamma_t(i_t)]^2 \,|\, \mathcal{G}_t\big\}\right\} &= \sum_{t=0}^{\infty} \mathbb{E}\big\{[\gamma_t(i_t)]^2\big\} \\ &= \sum_{t=0}^{\infty} \mathbb{E}\left\{\sum_{i=1}^{n} \rho_t(i)\,[\gamma_t(i)]^2\right\} = \sum_{i=1}^{n}\sum_{t=0}^{\infty} \mathbb{E}\big\{\rho_t(i)\,[\gamma_t(i)]^2\big\} < \infty,\end{aligned}$$

which implies that $\sum_{t=0}^{\infty} \mathbb{E}\big\{[\gamma_t(i_t)]^2 \,|\, \mathcal{G}_t\big\} < \infty$ w.p.1. Therefore, noting (38), we can use Proposition 7 to conclude that the sequence $\{\|y_t - y^*\|_2^2\}$ converges w.p.1 and

$$\sum_{t=0}^{\infty} \big\langle y_t - y^*, P_t\, G_t\big[\, y_t - y^*\big]\big\rangle < \infty \qquad \text{w.p.1.}$$

Then, we have $\sum_{t=0}^{\infty}[\min_{i\in\mathcal{S}} \gamma_t(i)]\, \big\langle y_t - y^*, P_t\big[\, y_t - y^*\big]\big\rangle \leq \sum_{t=0}^{\infty} \big\langle y_t - y^*, P_t\, G_t\big[\, y_t - y^*\big]\big\rangle < \infty$ w.p.1. Therefore, (11) and (12) imply that there exists a subset of iterations $\mathcal{T}$ such that the sequence $\{y_t - y^*\}_{t\in\mathcal{T}}$ converges to 0 w.p.1. Since $\{\|y_t - y^*\|_2^2\}$ converges w.p.1, the whole sequence $\{y_t - y^*\}$ converges to 0 w.p.1. □

We now turn to Lemma 5. Letting $v(0) = L$ and $v(n+1) = U$, the result of the projection $\Pi^{L,U}\hat{y}$ is the optimal solution to the problem

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{i=1}^{n} [v(i) - \hat{y}(i)]^2 \\ \text{subject to} \quad & v(i) \leq v(i+1) \qquad \text{for all } i \in \{0, \ldots, n\}. \end{aligned} \tag{39}$$

Associating the nonnegative Lagrange multipliers $\{\lambda(i) : i \in \{0, \ldots, n\}\}$ with the constraints, the Karush-Kuhn-Tucker conditions for this problem are

$$\hat{y}(i) - \lambda(i) + \lambda(i-1) = v(i) \qquad \text{for all } i \in \{1, \ldots, n\} \tag{40}$$

$$\lambda(i)\,[v(i+1) - v(i)] = 0 \qquad \text{for all } i \in \{0, \ldots, n\}. \tag{41}$$

The following result is useful when proving Lemma 5.

Lemma 8 *Assume that $y \in \mathcal{P}^{L,U}$. Fix $i^* \in \mathcal{S}$ and let $\hat{y} \in \mathbb{R}^n$ be obtained by*

$$\hat{y}(i) = \begin{cases} y(i) + \alpha\,[A - y(i)] & \text{if } i = i^* \\ y(i) & \text{otherwise,} \end{cases}$$

where A is a scalar and $\alpha \in [0,1]$. Let $v = \Pi^{L,U}\hat{y}$ be computed by solving problem (39) and $\{\lambda(i) : i \in \{0,\ldots,n\}\}$ be the corresponding optimal Lagrange multipliers. Then, one of the following cases holds.

1) *Either we have $\lambda(i) = 0$ for all $i \in \{0,\ldots,n\}$.*

2) *Or there exist κ and $\mu \in \{0,\ldots,n\}$ with $\kappa \leq \mu$ such that $\lambda(i) > 0$ if and only if $i \in \{\kappa,\ldots,\mu\}$.*

Proof of Lemma 8 To reach a contradiction, we assume that there exist $\kappa_1, \mu_1, \kappa_2, \mu_2, \ldots, \kappa_k$ and $\mu_k \in \{0,\ldots,n\}$ with $k \geq 2$ such that $\kappa_1 < \mu_1 + 1 < \kappa_2 < \mu_2 + 1 < \ldots < \kappa_k < \mu_k + 1$ and $\lambda(i) > 0$ if and only if $i \in \{\kappa_1,\ldots,\mu_1\} \cup \{\kappa_2,\ldots,\mu_2\} \cup \ldots \cup \{\kappa_k,\ldots,\mu_k\}$. We consider four cases.

Case 1 – Assume that $\kappa_1 > 0$ and $\mu_2 < n$. Since $\lambda(i) > 0$ for all $i \in \{\kappa_1,\ldots,\mu_1\}$, (41) implies that $v(\kappa_1) = v(\kappa_1 + 1) = \ldots = v(\mu_1 + 1)$. Since $\lambda(\kappa_1) > 0$, $\lambda(\mu_1) > 0$ and $\lambda(\kappa_1 - 1) = \lambda(\mu_1 + 1) = 0$, (40) implies that $v(\kappa_1) = \hat{y}(\kappa_1) - \lambda(\kappa_1) < \hat{y}(\kappa_1)$ and $v(\mu_1 + 1) = \hat{y}(\mu_1 + 1) + \lambda(\mu_1) > \hat{y}(\mu_1 + 1)$. Therefore, we obtain, $\hat{y}(\kappa_1) > v(\kappa_1) = v(\mu_1 + 1) > \hat{y}(\mu_1 + 1)$. On the other hand, since $y \in \mathcal{P}^{L,U}$, we have $y(\kappa_1) \leq y(\mu_1 + 1)$. Therefore, since $\hat{y}$ differs from y only in the i^*-th component, we must have $i^* = \kappa_1$ or $i^* = \mu_1 + 1$. By using a similar argument, we can obtain $\hat{y}(\kappa_2) > \hat{y}(\mu_2 + 1)$, which implies that we must also have $i^* = \kappa_2$ or $i^* = \mu_2 + 1$. Since we cannot have both $i^* \in \{\kappa_1, \mu_1 + 1\}$ and $i^* \in \{\kappa_2, \mu_2 + 1\}$, we reach a contradiction.

Case 2 – Assume that $\kappa_1 = 0$ and $\mu_2 < n$. Since $\lambda(i) > 0$ for all $i \in \{0,\ldots,\mu_1\}$ and $v(0) = L$, (41) implies that $L = v(1) = v(2) = \ldots = v(\mu_1 + 1)$. Then, since $\lambda(\mu_1) > 0$ and $\lambda(\mu_1 + 1) = 0$, (40) implies that $L = v(\mu_1 + 1) = \hat{y}(\mu_1 + 1) + \lambda(\mu_1) > \hat{y}(\mu_1 + 1)$. On the other hand, since $y \in \mathcal{P}^{L,U}$, we have $y(\mu_1 + 1) \geq L$. Therefore, since $\hat{y}$ differs from y only in the i^*-th component, we must have $i^* = \mu_1 + 1$. Following an argument similar to the one in the previous case, we must also have $i^* = \kappa_2$ or $i^* = \mu_2 + 1$. Since we cannot have both $i^* = \mu_1 + 1$ and $i^* \in \{\kappa_2, \mu_2 + 1\}$, we reach a contradiction.

The cases where we have $\kappa_1 > 0$ and $\mu_2 = n$, or $\kappa_1 = 0$ and $\mu_2 = n$ can be handled using similar arguments. □

We now prove Lemma 5.

Proof of Lemma 5 Let v be computed by solving problem (39) and $\{\lambda(i) : i \in \{0,\ldots,n\}\}$ be the corresponding optimal Lagrange multipliers. We show that there exists $\ell \in \mathcal{S}$ that satisfies (23)-(25). Before we begin, we note that $v \in \mathcal{P}^{L,U}$ since v is a feasible solution to problem (39).

If $\lambda(i) = 0$ for all $i \in \{0, \ldots, n\}$, then (40) implies that $v(i) = \hat{y}(i)$ for all $i \in \{1, \ldots, n\}$. In this case, we set $\ell = i^*$. Since $v \in \mathcal{P}^{L,U}$, we have $L \leq v(i^*) = \hat{y}(i^*) \leq U$, which implies that $v(i^*) = \hat{y}(i^*) = [\hat{y}(i^*) \wedge U] \vee L$. Therefore, (23) is satisfied. Since $\ell = i^*$, (24) and (25) do not need to be shown.

We now assume that there exists at least one strictly positive Lagrange multiplier. Let κ and μ with $\kappa \leq \mu$ be such that $\lambda(i) > 0$ if and only if $i \in \{\kappa, \ldots, \mu\}$. The existence of such κ and μ is guaranteed by Lemma 8. We consider four cases.

Case 1 – Assume that $\kappa > 0$ and $\mu < n$. In this case, following an argument similar to the one in Case 1 in the proof of Lemma 8, we obtain $v(\kappa) = v(\kappa+1) = \ldots = v(\mu+1)$ and we must have $i^* = \kappa$ or $i^* = \mu + 1$. By adding (40) for all $i \in \{\kappa, \ldots, \mu + 1\}$, we obtain

$$v(\kappa) = v(\kappa + 1) = \ldots = v(\mu + 1) = \frac{\hat{y}(\kappa) + \ldots + \hat{y}(\mu + 1)}{\mu - \kappa + 2}. \tag{42}$$

Since $\lambda(i) = 0$ for all $i \notin \{\kappa, \ldots, \mu\}$, (40) implies that

$$v(i) = \hat{y}(i) \qquad \text{for all } i \notin \{\kappa, \ldots, \mu + 1\}. \tag{43}$$

Finally, since $\lambda(\kappa) > 0$, $\lambda(\mu) > 0$ and $\lambda(\kappa-1) = \lambda(\mu+1) = 0$, (40) implies that $v(\kappa) = \hat{y}(\kappa) - \lambda(\kappa) < \hat{y}(\kappa)$ and $v(\mu + 1) = \hat{y}(\mu + 1) + \lambda(\mu) > \hat{y}(\mu + 1)$. Then, using (42), we obtain

$$\hat{y}(\mu + 1) < v(\mu + 1) = \frac{\hat{y}(\kappa) + \ldots + \hat{y}(\mu + 1)}{\mu - \kappa + 2} = v(\kappa) < \hat{y}(\kappa). \tag{44}$$

Since we must have $i^* = \kappa$ or $i^* = \mu + 1$, we consider the following two subcases.

Case 1.a – Assume that $i^* = \kappa$. In this case, we set $\ell = \mu + 1$ and we have $\ell > i^*$. Since $v \in \mathcal{P}^{L,U}$, (42) implies that

$$v(i^*) = v(i^* + 1) = \ldots = v(\ell) = \frac{\hat{y}(i^*) + \ldots + \hat{y}(\ell)}{\ell - i^* + 1} \leq U,$$

from which we obtain

$$v(i) = \frac{\hat{y}(i^*) + \ldots + \hat{y}(\ell)}{\ell - i^* + 1} \wedge U \qquad \text{for all } i \in \{i^*, \ldots, \ell\}.$$

By (43), we also have $v(i) = \hat{y}(i)$ for all $i \notin \{i^*, \ldots, \ell\}$. Therefore, (23) is satisfied. Since $y \in \mathcal{P}^{L,U}$ and $\hat{y}$ differs from y only in the i^*-th component, we have $\hat{y}(i^*+1) \leq \hat{y}(i^*+2) \leq \ldots \leq \hat{y}(\ell) = \hat{y}(\mu+1)$. Then, noting (44) shows that (24) is satisfied. Since $\ell > i^*$, (25) does not need to be shown.

Case 1.b – Assume that $i^* = \mu + 1$. In this case, we set $\ell = \kappa$ and we have $\ell < i^*$. Since $v \in \mathcal{P}^{L,U}$, (42) implies that

$$v(\ell) = v(\ell + 1) = \ldots = v(i^*) = \frac{\hat{y}(\ell) + \ldots + \hat{y}(i^*)}{i^* - \ell + 1} \geq L,$$

from which we obtain

$$v(i) = \frac{\hat{y}(\ell) + \ldots + \hat{y}(i^*)}{i^* - \ell + 1} \vee L \qquad \text{for all } i \in \{\ell, \ldots, i^*\}.$$

By (43), we also have $v(i) = \hat{y}(i)$ for all $i \notin \{\ell, \ldots, i^*\}$. Therefore, (23) is satisfied. Since $y \in \mathcal{P}^{L,U}$ and $\hat{y}$ differs from y only in the i^*-th component, we have $\hat{y}(\kappa) = \hat{y}(\ell) \leq \hat{y}(\ell+1) \leq \ldots \leq \hat{y}(i^*-1)$. Then, noting (44) shows that (25) is satisfied. Since $\ell < i^*$, (24) does not need to be shown.

Case 2 – Assume that $\kappa = 0$ and $\mu < n$. In this case, following an argument similar to the one in Case 2 in the proof of Lemma 8, we obtain $L = v(1) = v(2) = \ldots = v(\mu+1)$, $\hat{y}(\mu+1) < L$ and we must have $i^* = \mu + 1$. By adding (40) for all $i \in \{1, \ldots, \mu+1\}$, we obtain

$$L = v(1) = v(2) = \ldots = v(\mu+1) = \frac{\hat{y}(1) + \ldots + \hat{y}(\mu+1) + \lambda(0)}{\mu+1} > \frac{\hat{y}(1) + \ldots + \hat{y}(\mu+1)}{\mu+1}. \tag{45}$$

Since $\lambda(i) = 0$ for all $i \notin \{0, \ldots, \mu\}$, (40) implies that

$$v(i) = \hat{y}(i) \qquad \text{for all } i \notin \{1, \ldots, \mu+1\}. \tag{46}$$

Since we must have $i^* = \mu + 1$, we consider the following two subcases.

Case 2.a – Assume that $\mu > 0$. In this case, we have $i^* > 1$, we set $\ell = 1$ and we obtain $\ell < i^*$. By (45), we have

$$v(i) = L = \frac{\hat{y}(\ell) + \ldots + \hat{y}(i^*)}{i^* - \ell + 1} \vee L \qquad \text{for all } i \in \{\ell, \ldots, i^*\}.$$

By (46), we also have $v(i) = \hat{y}(i)$ for all $i \notin \{\ell, \ldots, i^*\}$. Therefore, (23) is satisfied. Since $y \in \mathcal{P}^{L,U}$ and $\hat{y}$ differs from y only in the i^*-th component, we have $L \leq \hat{y}(\ell) \leq \hat{y}(\ell+1) \leq \ldots \leq \hat{y}(i^*-1)$. We also have $\hat{y}(i^*) = \hat{y}(\mu+1) < L$. Therefore, we have $\hat{y}(i^*) < \hat{y}(\ell) \leq \hat{y}(\ell+1) \leq \ldots \leq \hat{y}(i^*-1)$, which implies that $\hat{y}(i^*) < [\hat{y}(\ell) + \ldots + \hat{y}(i^*)]/[i^* - \ell + 1]$. Noting (45), we obtain

$$\hat{y}(i^*) < \frac{\hat{y}(\ell) + \ldots + \hat{y}(i^*)}{i^* - \ell + 1} < L \leq \hat{y}(\ell) \leq \ldots \leq \hat{y}(i^*-1),$$

which shows that (25) is satisfied. Since $\ell < i^*$, (24) does not need to be shown.

Case 2.b – Assume that $\mu = 0$. In this case, we have $i^* = 1$, we set $\ell = 1$ and we obtain $\ell = i^*$. Since $\hat{y}(1) = \hat{y}(\mu+1) < L \leq U$, (45) implies that $v(i^*) = v(1) = L = [\hat{y}(1) \wedge U] \vee L = [\hat{y}(i^*) \wedge U] \vee L$. By (46), we also have $v(i) = \hat{y}(i)$ for all $i \notin \{i^*\}$. Therefore, (23) is satisfied. Since $\ell = i^*$, (24) and (25) do not need to be shown.

The cases where we have $\kappa > 0$ and $\mu = n$, or $\kappa = 0$ and $\mu = n$ can be handled using similar arguments. □