Lookahead-Bounded Q-Learning

Ibrahim El Shar [1] **Daniel R. Jiang** [1]

Abstract

We introduce the lookahead-bounded Q-learning (LBQL) algorithm, a new, provably convergent variant of Q-learning that seeks to improve the performance of standard Q-learning in stochastic environments through the use of "lookahead" upper and lower bounds. To do this, LBQL employs previously collected experience and each iteration's state-action values as dual feasible penalties to construct a sequence of sampled information relaxation problems. The solutions to these problems provide estimated upper and lower bounds on the optimal value, which we track via stochastic approximation. These quantities are then used to constrain the iterates to stay within the bounds at every iteration. Numerical experiments on benchmark problems show that LBQL exhibits faster convergence and more robustness to hyperparameters when compared to standard Q-learning and several related techniques. Our approach is particularly appealing in problems that require expensive simulations or real-world interactions.

1 Introduction

Since its introduction by Watkins (1989), Q-learning has become one of the most widely-used reinforcement learning (RL) algorithms (Sutton & Barto, 2018), due to its conceptual simplicity, ease of implementation, and convergence guarantees (Jaakkola et al., 1994; Tsitsiklis, 1994; Bertsekas & Tsitsiklis, 1996). However, practical, real-world applications of Q-learning are difficult due to the often high cost of obtaining data and experience from real environments, along with other issues such as overestimation bias (Szepesvári, 1998; Even-Dar & Mansour, 2003; Lee & Powell, 2019).

In this paper, we address these challenges for a specific class of problems with *partially known* transition models. We write the system dynamics as $s_{t+1} = f(s_t, a_t, w_{t+1})$, where s_t and a_t are the current state and action, s_{t+1} is the next state, w_{t+1} is random noise, and f is the *transition function*. We focus on problems where f is known, but the noise w_{t+1} can only be observed through interactions with the environment. This type of model is the norm in the control (Bertsekas & Tsitsiklis, 1996) and operations research (Powell, 2007) communities. In this work, we propose and analyze a new RL algorithm called *lookahead-bounded Q-learning* (LBQL), which exploits knowledge of the transition function f to improve the efficiency of Q-learning and address overestimation bias. It does so by making better use of the observed data through estimating upper and lower bounds using a technique called *information relaxation* (IR) (Brown et al., 2010).

Indeed, there are abundant real-world examples that fall into this subclass of problems, as we now illustrate with a few examples. In *inventory control*, the transition from one inventory state to the next is a well-specified function f given knowledge of a stochastic demand w_{t+1} (Kunnumkal & Topaloglu, 2008). For *vehicle routing*, f is often simply the routing decision itself, while w_{t+1} are exogenous demands that require observation (Secomandi, 2001). In *energy operations*, a typical setting is to optimize storage that behaves through linear transitions f together with unpredictable renewable supply, w_{t+1} (Kim & Powell, 2011). In *portfolio optimization*, f is the next portfolio, and w_{t+1} represents random prices (Rogers, 2002). In Section 5 of this paper, we discuss in detail another application domain that follows this paradigm: *repositioning and spatial dynamic pricing for car sharing* (He et al., 2019; Bimpikis et al., 2019).

Although we specialize to problems with partially known transition dynamics, this should not be considered restrictive: in fact, our proposed algorithm can be integrated with the framework of model-based RL to handle the standard model-free RL setting, where f is constantly being learned. We leave this extension to future work.

Main Contributions. We make the following methodological and empirical contributions in this paper.

1. We propose a novel algorithm that takes advantage of IR theory and Q-learning to generate upper and lower bounds on the optimal value. This allows our algorithm to mitigate the effects of maximization bias, while making better use of the collected experience and

[1]Department of Industrial Engineering, University of Pittsburgh, PA, USA. Correspondence to: Ibrahim El Shar <ije8@pitt.edu>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

the transition function f. A variant of the algorithm based on experience replay is also given.

2. We prove that our method converges almost surely to the optimal action-value function. The proof requires a careful analysis of several interrelated stochastic processes (upper bounds, lower bounds, and the Q-factors themselves).
3. Numerical experiments on five test problems show superior empirical performance of LBQL compared to Q-learning and other widely-used variants. Moreover, sensitivity analysis shows that LBQL is more robust to learning rate and exploration parameters.

The rest of the paper is organized as follows. In the next section, we review the related literature. In Section 3, we introduce the notation and review the basic theory of IR. In Section 4, we present our algorithm along with its theoretical results. In Section 5, we show the numerical results where LBQL is compared to other Q-learning variants. Finally, we state conclusions and future work in Section 6.

2 Related Literature

Upper and lower bounds on the optimal value have recently been used by optimism-based algorithms, e.g., Dann et al. (2018) and Zanette & Brunskill (2019). These papers focus on finite horizon problems, while we consider the infinite horizon case. Their primary use of the lower and upper bounds is to achieve better exploration, while our work is focused on improving the action-value estimates by mitigating overestimation and enabling data re-use.

In the context of real-time dynamic programming (RTDP) (Barto et al., 1995), Bounded RTDP (McMahan et al., 2005), Bayesian RTDP (Sanner et al., 2009) and Focused RTDP (Smith & Simmons, 2006) propose extensions of RTDP where a lower bound heuristic and an upper bound are maintained on the value function. These papers largely use heuristic approaches to obtain bounds, while we use the principled idea of IR duality.

More closely related to our paper is the work of He et al. (2016), which exploits multistep returns to construct bounds on the optimal action-value function, before utilizing constrained optimization to enforce those bounds. However, unlike our work, no theoretical guarantees are provided. To the best of our knowledge, we provide the first asymptotic proof of convergence to the general approach of enforcing dynamically computed (noisy) bounds.

There are also two papers that utilize IR bounds in the related setting of *finite horizon* dynamic programming. Jiang et al. (2020) use IR dual bounds in a tree search algorithm in order to ignore parts of the tree. Recent work by Chen et al. (2020) uses IR duality in a duality-based dynamic programming algorithm that converges monotonically to the optimal value function through a series of "subsolutions" under more restrictive assumptions (e.g., knowledge of probability distributions).

3 Background

In this section, we first introduce some definitions and concepts from Markov decision process (MDP) theory. Then, we describe the basic theory of information relaxations and duality, which is the main tool used in our LBQL approach.

3.1 MDP Model

Consider a discounted, infinite horizon MDP with a finite state space $\mathcal{S}$, and a finite action space $\mathcal{A}$, and a disturbance space $\mathcal{W}$. Let $\{w_t\}$ be a sequence of independent and identically distributed (i.i.d.) random variables defined on a probability space $(\Omega, \mathcal{F}, \mathbf{P})$, where each w_t is supported on the set $\mathcal{W}$. Let $s_t \in \mathcal{S}$ be the state of the system at time t. We also define a state transition function $f : \mathcal{S} \times \mathcal{A} \times W \to \mathcal{S}$, such that if action a_t is taken at time t, then the next state is governed by $s_{t+1} = f(s_t, a_t, w_{t+1})$. This "transition function" model of the MDP is more convenient for our purposes, but we note that it can easily be converted to the standard model used in RL, where the transition probabilities, $p(s_{t+1} \,|\, s_t, a_t)$, are modeled directly. For simplicity and ease of notation, we assume that w is independent[1] from (s, a). Let $r(s_t, a_t)$ be the expected reward when taking action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$. We assume that the rewards $r(s_t, a_t)$ are uniformly bounded by $R_{\max}$ and for simplicity in notation, that the feasible action set $\mathcal{A}$ does not depend on the current state. As usual, a deterministic Markov policy $\pi \in \Pi$ is a mapping from states to actions, such that $a_t = \pi(s_t)$ whenever we are following policy π. We let Π be the set of all possible policies (or the set of all "admissible" policies).

Given a discount factor $\gamma \in (0, 1)$ and a policy $\pi \in \Pi$, the *value* and the *action-value* functions are denoted respectively by

$$V^\pi(s) = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \;\middle|\; \pi,\, s_0 = s\right] \quad \text{and}$$

$$Q^\pi(s, a) = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \;\middle|\; s_0 = s, a_0 = a, \pi\right],$$

where the notation of "conditioning on π" refers to actions a_t selected by $\pi(s_t)$. The expectation $\mathbf{E}$, here and throughout the paper, is taken with respect to $\mathbf{P}$. Our objective is to find a policy $\pi \in \Pi$ such that from any initial state s, it achieves the optimal expected discounted cumulative reward. The value of an optimal policy π^* for a state s is called the optimal value function and is denoted

[1]However, we can also allow for (s, a)-dependent w with essentially no fundamental changes to our approach.

by $V^*(s) = \max_\pi V^\pi(s)$. Specifically, it is well-known that an optimal policy selects actions according to $\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s,a)$, where $Q^*(s,a) = \max_\pi Q^\pi(s,a)$ is the optimal action-value function (Puterman, 2014). The Bellman optimality equation gives the following recursion:

$$Q^*(s_t, a_t) = r(s_t, a_t) + \gamma\, \mathbf{E}\Big[\max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})\Big].$$

The goal in many RL algorithms, including Q-learning (Watkins, 1989), is to approximate Q^*.

3.2 Information Relaxation Duality

Now let us give a brief review of the theory behind information relaxation duality from Brown et al. (2010), which is a way of computing an *upper bound* on the value and action-value functions. This generalizes work by Rogers (2002), Haugh & Kogan (2004), and Andersen & Broadie (2004) on pricing American options. Note that any feasible policy provides a lower bound on the optimal value, but computing an upper bound is less straightforward. The *information relaxation* approach proposes to relax the "non-anticipativity" constraints on policies, i.e., it allows them to depend on realizations of future uncertainties when making decisions. Naturally, optimizing in the class of policies that can "see the future" provides an upper bound on the best admissible policy. We focus on the special case of *perfect information relaxation*, where full knowledge of the future uncertainties, i.e., the sample path $(w_1, w_2, \ldots)$, is used to create upper bounds. The naive version of the perfect information bound is simply given by

$$V^*(s_0) \le \mathbf{E}\left[\max_{\mathbf{a}} \left\{ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right\}\right],$$

which, in essence, is an interchange of the expectation and max operators; the interpretation here is that an agent who is allowed to adjust her actions *after* the uncertainties are realized achieves higher reward than an agent who acts sequentially. As one might expect, perfect information can provide upper bounds that are quite loose.

The central idea of the information relaxation approach to strengthen these upper bounds is to simultaneously (1) allow the use of future information but (2) also penalize the agent for doing so by assessing a penalty on the reward function in each period. A penalty function is said to be *dual feasible* if it does not penalize any admissible policy $\pi \in \Pi$ in expectation. Let $s_{t+1} = f(s_t, a_t, w_{t+1})$ be the next state, $\varphi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ be a bounded function, and w have the same distribution as w_{t+1}. Then, penalties involving terms of the form

$$\begin{aligned} z_t^\pi(s_t, a_t, w_{t+1} \,|\, \varphi) := \gamma^{t+1}\Big(&\varphi(s_{t+1}, \pi(s_{t+1})) \\ &- \mathbf{E}\Big[\varphi\big(f(s_t,a_t,w), \pi(f(s_t,a_t,w))\big)\Big]\Big) \end{aligned} \tag{1}$$

are dual feasible because

$$\mathbf{E}\left[\sum_{t=0}^{\infty} z_t^\pi(s_t, a_t, w_{t+1} \,|\, \varphi)\right] = 0.$$

This is a variant of the types of penalties introduced in Brown & Haugh (2017), extended to the case of action-value functions. Intuitively, if φ is understood to be an estimate of the optimal action-value function Q^* and π an estimate of the optimal policy π^*, then z_t^π can be thought of as the one-step value of future information (i.e., knowing w_{t+1} versus taking an expectation over its distribution).

These terms, however, may have negative expected value for policies that violate non-anticipativity constraints. Let π_φ be the policy that is greedy with respect to the bounded function φ (considered to be an approximate action-value function). Consider the problem constructed by subtracting the penalty term from the reward in each period and relaxing non-anticipativity constraints by interchanging maximization and expectation:

$$\begin{aligned} Q^U(s_0, a_0) = \mathbf{E}\Big[\max_{\mathbf{a}} \Big\{ \sum_{t=0}^{\infty} \Big(&\gamma^t r(s_t, a_t) \\ &- z_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi)\Big)\Big\}\Big], \end{aligned} \tag{2}$$

where $\mathbf{a} := (a_0, a_1, \ldots)$ is an infinite sequence of actions. Brown & Haugh (2017) shows that the objective value of this problem, $Q^U(s_0, a_0)$, is an upper bound on $Q^*(s_0, a_0)$. Our new approach to Q-learning will take advantage of this idea, with φ and π being continuously updated. Notice that in principle, it is possible to estimate the problem in (2) using Monte Carlo simulation. To do this, we generate infinitely long sample paths of the form $\mathbf{w} = (w_1, w_2, \ldots)$, and for each fixed $\mathbf{w}$, we solve the inner deterministic dynamic programming (DP) problem. Averaging over the results produces an estimate of the upper bound of $Q^U(s_0, a_0)$.

3.3 Absorption Time Formulation

In practice, however, we cannot simulate infinitely long sample paths $\mathbf{w}$. One solution is to use an *equivalent* formulation with a finite, but random, horizon (see for e.g. Proposition 5.3.1 of Puterman 2014), where instead of discounting, a new absorbing state $\tilde{s}$ with zero reward is added to the state space $\mathcal{S}$. This new state $\tilde{s}$ can be reached from every state and for any feasible action with probability $1-\gamma$. We define a new state transition function h, which transitions to $\tilde{s}$ with probability $1-\gamma$ from every (s,a), but conditioned on not absorbing (i.e., with probability γ), h is identical to f. We refer to this as the *absorption time formulation*, where the horizon length $\tau := \min\{t : s_t = \tilde{s}\}$ has a geometric distribution with parameter $1-\gamma$ and the state transitions are governed by the state transition function h instead of f. Let $\mathcal{Q}$ be the set of bounded functions φ such that $\varphi(\tilde{s}, a) = 0$ for all $a \in \mathcal{A}$. The penalty terms for the absorption time formulation are defined in a similar way as (1), except we now consider $\varphi \in \mathcal{Q}$:

$$\begin{aligned}\zeta_t^\pi(s_t, a_t, w_{t+1} \,|\, \varphi) &:= \varphi(s_{t+1}, \pi(s_{t+1})) \\ &- \mathbf{E}\Big[\varphi\big(h(s_t, a_t, w), \pi(h(s_t, a_t, w))\big)\Big],\end{aligned} \tag{3}$$

where $s_{t+1} = h(s_t, a_t, w_{t+1})$. We now state a proposition that summarizes the information relaxation duality results, which is a slight variant of results in Proposition 2.2 of Brown & Haugh (2017).

Proposition 1 (Duality Results, Proposition 2.2 in Brown & Haugh (2017))**.** *The following duality results are stated for the absorption time formulation of the problem.*

(i) Weak Duality: For any $\pi \in \Pi$ and $\varphi \in \mathcal{Q}$,

$$\begin{aligned}Q^\pi(s_0, a_0) \le \mathbf{E}\Big[\max_{\mathbf{a}} \sum_{t=0}^{\tau-1} \Big(& r(s_t, a_t) \\ & - \zeta_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi)\Big)\Big]\end{aligned} \tag{4}$$

(ii) Strong Duality: It holds that

$$\begin{aligned}Q^*(s_0, a_0) = \inf_{\varphi \in \mathcal{Q}} \mathbf{E}\Big[\max_{\mathbf{a}} \sum_{t=0}^{\tau-1} \Big(& r(s_t, a_t) \\ & - \zeta_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi)\Big)\Big],\end{aligned} \tag{5}$$

with the infimum attained at $\varphi = Q^*$.

The DP inside the expectation of the right hand side of (4) is called the *inner DP problem*. Weak duality tells us that by using a dual feasible penalty, we can get an estimated upper bound on the optimal action-value function $Q^*(s_0, a_0)$ by simulating multiple sample paths and averaging the optimal value of the resulting inner problems. Strong duality suggests that the information gained from accessing the future is perfectly cancelled out by the optimal dual feasible penalty.

For a given sample path $\mathbf{w} = (w_1, w_2, \ldots, w_\tau)$, each of the inner DP problems can be solved via the backward recursion

$$\begin{aligned}Q_t^U(s_t, a_t) &= r(s_t, a_t) - \zeta_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi) \\ &+ \max_a Q_{t+1}^U(s_{t+1}, a),\end{aligned} \tag{6}$$

for $t = \tau - 1, \tau - 2, \ldots, 0$ with $s_{t+1} = h(s_t, a_t, w_{t+1})$ and $Q_\tau^U \equiv 0$ (as there is no additional reward after entering the absorbing state $\tilde{s}$). The optimal value of the inner problem is given by $Q_0^U(s_0, a_0)$.

3.4 Lower Bounds using IR

The penalty function approach also allows for using a feasible policy to estimate a *lower bound* on the optimal value, such that when using a common sample path, this lower bound is guaranteed to be less than the corresponding estimated upper bound, a crucial aspect of our theoretical analysis. Specifically, given a sample path $(w_1, w_2, \ldots, w_\tau)$, the inner problem used to evaluate a feasible policy $\pi \in \Pi$ is given by

$$\begin{aligned}Q_t^L(s_t, a_t) &= r(s_t, a_t) - \zeta_t^\pi(s_t, a_t, w_{t+1} \,|\, \varphi) \\ &+ Q_{t+1}^L(s_{t+1}, \pi(s_{t+1})),\end{aligned} \tag{7}$$

for $t = 0, \ldots, \tau - 1$, with $s_{t+1} = h(s_t, a_t, w_{t+1})$ and $Q_\tau^L \equiv 0$. It follows that $\mathbf{E}\big[Q_0^L(s_0, a_0)\big] = Q^\pi(s_0, a_0)$, as the penalty terms $\zeta_t^\pi(s_t, a_t, w_{t+1} \,|\, \varphi)$ have zero mean.

4 QL with Lookahead Bounds

We now introduce our proposed approach, which integrates the machinery of IR duality with Q-learning in a unique way. An outline of the essential steps is given below.

1. On a given iteration, we first experience a realization of the exogenous information w_{t+1} and make a standard Q-learning update.
2. We then set φ to be the newly updated Q-iterate and compute noisy upper and lower bounds on the true Q^*, which are then tracked and averaged using a stochastic approximation step.
3. Finally, we project the Q-iterate so that it satisfies the averaged upper and lower bounds and return to Step 1.

Figure 1 shows an illustration of each of these steps at a given iteration of the algorithm. Since we are setting φ to be the current Q-iterate at every iteration, the information relaxation bounds are computed using a *dynamic sequence of penalty functions* and averaged together using stochastic approximation. The idea is that as our approximation of Q^* improves, our upper and lower bounds also improve. As the upper and lower bounds improve, the projection step further improves the Q-iterates. It is this back-and-forth feedback between the two processes that has the potential to yield rapid convergence toward the optimal Q^*.

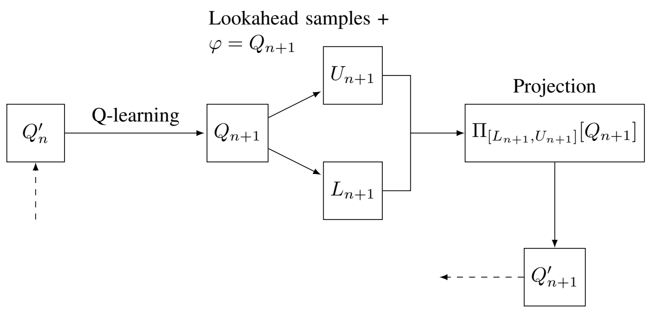


Figure 1: Illustration of LBQL Algorithm at iteration n.

The primary drawback of our approach is that in the computation of the information relaxation dual bounds, expectations need to be computed. We first show an *idealized* version of the algorithm where these expectations are estimated using unbiased samples of w_{t+1} from a black-box simulator. Later, we relax the need for a black-box simulator and show how our algorithm can be implemented with a replay-buffer. Both versions are analyzed theoretically and convergence results are provided.

4.1 An Idealized Algorithm

Let $\{w_{t+1}^1, w_{t+1}^2, \ldots, w_{t+1}^K\}$ be a *batch* (as opposed to a sample path) of K samples from the distribution of the exogenous information w_{t+1} (i.e., from a black-box simulator). An empirical version of (3) is simply given by:

$$\hat{\zeta}_t^\pi(s_t, a_t, w_{t+1} \,|\, \varphi) := \varphi(s_{t+1}, \pi(s_{t+1})) - \frac{1}{K}\sum_{k=1}^{K} \varphi\big(h(s_t, a_t, w_{t+1}^k), \pi(h(s_t, a_t, w_{t+1}^k))\big), \tag{8}$$

where $s_{t+1} = h(s_t, a_t, w_{t+1})$. Given a sample path $\mathbf{w} = (w_1, w_2, \ldots, w_\tau)$ of the absorption time formulation of the problem, analogues to (6) and (7) using $\hat{\zeta}_t^\pi$, where in (7) we set $\pi = \pi_\varphi$ (i.e., the lower bound on the optimal value is constructed by approximately evaluating the feasible policy π_φ) are given by

$$\hat{Q}_t^U(s_t, a_t) = r(s_t, a_t) - \hat{\zeta}_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi) + \max_a \hat{Q}_{t+1}^U(s_{t+1}, a) \tag{9}$$

$$\hat{Q}_t^L(s_t, a_t) = r(s_t, a_t) - \hat{\zeta}_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi) + \hat{Q}_{t+1}^L(s_{t+1}, \pi_\varphi(s_{t+1})) \tag{10}$$

for $t = 0, 1, \ldots, \tau - 1$, where $s_{t+1} = h(s_t, a_t, w_{t+1})$, $\hat{Q}_\tau^U \equiv \hat{Q}_\tau^L \equiv 0$, and we assume that each call to $\hat{\zeta}_t^\pi$ uses a fresh batch of K samples.

Proposition 2. *The valid upper and lower bound properties continue to hold in the empirical case:*

$$\mathbf{E}[\hat{Q}_0^L(s,a)] \le Q^*(s,a) \le \mathbf{E}[\hat{Q}_0^U(s,a)],$$

for any state-action pair (s, a).

We include the proof in Appendix A.1. The proof is similar to that of Proposition 2.3(iv) of Brown et al. (2010), except extended to the infinite horizon setting with the absorption time formulation. A detailed description of the LBQL algorithm is given in Algorithm 1, where we use 'n' for the iteration index in order to avoid confusion with the 't' used in the inner DP problems. We use $\Pi_{[a,b]}[x]$ to denote x projected onto $[a, b]$, i.e., $\Pi_{[a,b]}[x] = \max\{\min\{x, b\}, a\}$, where either a or b could be ∞. Let $\rho = R_{\max}/(1-\gamma)$, the initial lower and upper bounds estimates are set such that $L_0(s,a) = -\rho$ and $U_0(s,a) = \rho$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. The initial action-value Q_0 is set arbitrarily such that $L_0(s,a) \le Q_0(s,a) \le U_0(s,a)$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$.

Algorithm 1 Lookahead-Bounded Q-Learning

```
Input: Initial estimates $L_0 \le Q_0 \le U_0$, batch size $K$, and stepsize rules $\alpha_n(s,a)$, $\beta_n(s,a)$.
Output: Approximations $\{L_n\}$, $\{Q'_n\}$, and $\{U_n\}$.
Set $Q'_0 = Q_0$ and choose an initial state $s_0$.
for $n = 0, 1, 2, \ldots$ do
    Choose an action $a_n$ via some behavior policy (e.g., $\epsilon$-greedy). Observe $w_{n+1}$. Let
        $Q_{n+1}(s_n, a_n) = Q'_n(s_n, a_n) + \alpha_n(s_n, a_n)\Big[r_n(s_n, a_n) + \gamma \max_a Q'_n(s_{n+1}, a) - Q'_n(s_n, a_n)\Big]$.   (11)
    Set $\varphi = Q_{n+1}$. Using one sample path $\mathbf{w}$, compute $\hat{Q}_0^U(s_n, a_n)$ and $\hat{Q}_0^L(s_n, a_n)$ using (9) and (10).
    Update and enforce upper and lower bounds:
        $U_{n+1}(s_n, a_n) = \Pi_{[-\rho, \infty]}\Big[U_n(s_n, a_n) + \beta_n(s_n, a_n)\,[\hat{Q}_0^U(s_n, a_n) - U_n(s_n, a_n)]\Big]$,   (12)
        $L_{n+1}(s_n, a_n) = \Pi_{[\infty, \rho]}\Big[L_n(s_n, a_n) + \beta_n(s_n, a_n)\,[\hat{Q}_0^L(s_n, a_n) - L_n(s_n, a_n)]\Big]$,   (13)
        $Q'_{n+1}(s_n, a_n) = \Pi_{[L_{n+1}(s_n,a_n),\, U_{n+1}(s_n,a_n)]}\,[Q_{n+1}(s_n, a_n)]$.   (14)
end for
```

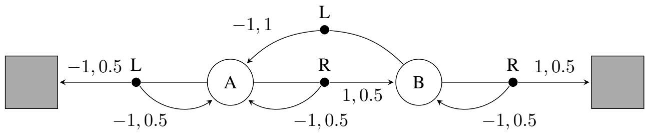


Figure 2: A simple stochastic MDP.

Example 1. *We demonstrate the idealized LBQL algorithm using the simple MDP shown in Figure 2. The MDP has two non-terminal states* `A` *and* `B`*. Each episode starts in state* `A`*, with a choice of two actions: right and left denoted by* `R` *and* `L` *respectively. The rewards and transition probabilities of taking an action in each state are shown on the edges in the figure. Assume that the transitions are governed by the outcome of a fair coin. If the outcome is* `Head` *then we transition in the direction of our chosen action and in the opposite direction for a* `Tail` *outcome. For a discount factor* $\gamma = 0.95$*, the optimal policy is to go right at both* `A` *and* `B`*. The optimal action-values are given by* $Q^*(A,R) = Q^*(B,R) = 0$, $Q^*(A,L) = Q^*(B,L) = -1$*. Consider applying the idealized version of LBQL described in Algorithm 1.*

We let $\alpha_n = 0.1$, $\beta_n = 0.05$ *for all* n*. Figure 3 illustrates two iterations from the first and the* 58th *episodes. Initially*

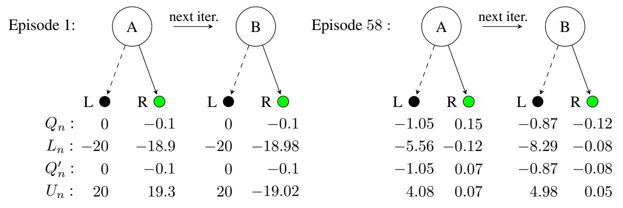

Episode 1:
A
next iter.
B
Episode 58 :
A
next iter.
B
L
R
L
R
L
R
L
R
Q_n : 0 −0.1 0 −0.1 −1.05 0.15 −0.87 −0.12
L_n : −20 −18.9 −20 −18.98 −5.56 −0.12 −8.29 −0.08
Q'_n : 0 −0.1 0 −0.1 −1.05 0.07 −0.87 −0.08
U_n : 20 19.3 20 −19.02 4.08 0.07 4.98 0.05

Figure 3: An illustration of LBQL iterates for Example 1.

$Q_0(s,a) = 0$ and $\rho = 20$. After one episode the bounds are still loose, so we have $Q_1(A,R) = Q'_1(A,R) = -0.1$. At episode 58 (281 iterations): learning has occurred for the lower and upper bounds values for the right action at A *and* B*. We see that the bounds are effective already in keeping the Q-iterate close to Q^*. Interestingly, the upper bound is enforced at* A*, while the lower bound is enforced at* B*. Note that these are the results of a real simulation.*

4.2 Analysis of Convergence

In this section, we analyze the convergence of the idealized version of the LBQL algorithm to the optimal action-value function Q^*. We start by summarizing and developing some important technical results that will be used in our analysis. All proofs are presented in Appendix A.

The following proposition establishes the boundedness of the action-value iterates and asymptotic bounds on the L_n and U_n iterates of Algorithm 1, which are needed in our proof of convergence. The proof of this proposition is presented in Section A.2 in the Appendix.

Proposition 3 (Boundedness)**.** *For all $(s,a) \in \mathcal{S} \times \mathcal{A}$, we have the following:*

(i) The iterates $Q_n(s,a)$ and $Q'_n(s,a)$, remains bounded for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ and for all n.

(ii) For every $\eta > 0$, and with probability one, there exists some finite iteration index n_0 such that
$L_n(s,a) \le Q^*(s,a)+\eta$ and $Q^*(s,a)-\eta \le U_n(s,a)$, for all iterations $n \ge n_0$.

Proposition 3(i) ensures that at each iteration n the action-value iterates Q_n and Q'_n are bounded. This allows us to set $\varphi = Q_{n+1}$ at each iteration of Algorithm 1 and is required to establish convergence in general. The proof is based on showing an inductive relationship that connects Q_n and Q'_n to the previous lower and upper bound iterates. Specifically, we show that both action-value iterates are bounded below by the preceding upper bound iterates and above by the preceding lower bound iterates. Proposition 3(ii) ensures that there exists a finite iteration after which the lower and upper bound iterates L_n and U_n are lower and upper bounds on the optimal action-value function Q^* with an error margin of at most an arbitrary amount $\eta > 0$. In the proof of Proposition 3(ii), we bound the lower and upper bound iterates by a noise process and another sequence that converges to Q^*. We show that the noise process possesses some properties that help to eliminate the effect of the noise asymptotically. With the effects of the noise terms vanishing, the boundedness of the lower and upper bound iterates by Q^* is achieved. Examining the update equations (12) and (13) for U_{n+1} and L_{n+1} in Algorithm 1, we remark that they are not "standard" stochastic approximation or stochastic gradient updates because $\hat{Q}_0^U$ and $\hat{Q}_0^L$ are computed with iteration-dependent penalty functions generated by $\varphi = Q_{n+1}$. In other words, the noiseless function itself is changing over time. The proof of Proposition 3(ii) essentially uses the fact that even though these updates are being performed with respect to different underlying functions, as long as we can apply Proposition 2 in every case, then after the noise is accounted for, the averaged values U_{n+1} and L_{n+1} are eventually bounded below and above by Q^*, respectively. The following lemma derives some guarantees on the lower and upper bound iterates of Algorithm 1, whose proof appears in Section A.3 of the Appendix.

Lemma 1 (Consistency of Bounds)**.** *If $L_0(s,a) \le U_0(s,a)$, then $L_n(s,a) \le U_n(s,a)$ for all iterations n and for all $(s,a) \in \mathcal{S} \times \mathcal{A}$.*

In particular, Lemma 1 shows that the upper and lower bound iterates do not interchange roles and become inconsistent. This is an important property; otherwise, the projection step of Algorithm 1 loses its meaning and would require additional logic to handle inconsistent bounds. The results of Lemma 1 follows mainly by the fact that we are using the same sample path to solve the upper and lower bound inner problems, (9) and (10), respectively. Before stating our convergence results, we first state a typical assumption on the stepsizes and the state visits.

Assumption 1. We assume that:

(i) $\sum_{n=0}^{\infty} \alpha_n(s,a) = \infty$, $\sum_{n=0}^{\infty} \alpha_n^2(s,a) < \infty$, $\sum_{n=0}^{\infty} \beta_n(s,a) = \infty$, $\sum_{n=0}^{\infty} \beta_n^2(s,a) < \infty$,

(ii) Each state $s \in \mathcal{S}$ is visited infinitely often with probability one.

We now state one of our main theoretical results.

Theorem 1 (Convergence of LBQL)**.** *Under Assumption 1, the following hold with probability 1:*

(i) $Q'_n(s,a)$ in Algorithm 1 converges to the optimal action-value function $Q^*(s,a)$ for all state-action pairs (s,a).

(ii) If the penalty terms are computed exactly, i.e. as per (3), then the iterates $L_n(s,a), Q'_n(s,a), U_n(s,a)$ in Algorithm 1 converge to the optimal action-value function $Q^*(s,a)$ for all state-action pairs (s,a).

Due to the interdependent feedback between Q, U, and L, it is not immediately obvious that the proposed scheme does not diverge. The primary challenge in the analysis for this theorem is to handle this unique aspect of the algorithm.

4.3 LBQL with Experience Replay

We now introduce a more practical version of LBQL that uses experience replay in lieu of a black-box simulator. Here, we use a noise buffer $\mathcal{B}$ to record the unique noise values w that are observed at every iteration. We further assume that the noise space $\mathcal{W}$ is finite, a reasonable assumption for a finite MDP. The buffer $\mathcal{B}$ is used in two ways: (1) to generate the sample path $\mathbf{w}$ and (2) to estimate the expectation in the penalty function. Here, we track and update the *distribution* of the noise w after every iteration and directly compute the expectation under this distribution instead of sampling a batch of size K, as we did previously. To illustrate how this can be done, suppose $\mathcal{W} = \{w_a, w_b, w_c, w_d\}$ and that at iteration n we observe $w_{n+1} = w_a$. Let p_a denote the probability of observing w_a, and $N_n(w_a)$ the number of times w_a is observed in the first n iterations, then the empirical estimate of p_a is given by $\hat{p}_n(w_a) = N_n(w_a)/n$.[2] We denote by $\hat{\mathbf{E}}_n[\,\cdot\,]$ the expectation computed using the empirical distribution $\hat{p}_n$. To differentiate the penalty and the action-values (solutions to the inner problems) that are computed from the buffer from those defined in the idealized version of the algorithm, we define:

$$\begin{aligned}\tilde{\zeta}_t^{\pi}(s_t, a_t, w \,|\, \varphi) &:= \varphi(s_{t+1}, \pi(s_{t+1})) \\ &\quad - \hat{\mathbf{E}}_n[\varphi(h(s_t, a_t, w), \pi(h(s_t, a_t, w)))],\end{aligned} \tag{15}$$

and given a sample path $\mathbf{w} = (w_1, w_2, \ldots, w_\tau)$ the inner problems analogous to (9) and (10) are given by

$$\begin{aligned}\tilde{Q}_t^U(s_t, a_t) &= r(s_t, a_t) - \tilde{\zeta}_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi) \\ &\quad + \max_a \tilde{Q}_{t+1}^U(s_{t+1}, a)\end{aligned} \tag{16}$$

$$\begin{aligned}\tilde{Q}_t^L(s_t, a_t) &= r(s_t, a_t) - \tilde{\zeta}_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi) \\ &\quad + \tilde{Q}_{t+1}^L(s_{t+1}, \pi_\varphi(s_{t+1}))\end{aligned} \tag{17}$$

for $t = 0, 1, \ldots, \tau - 1$, where $s_{t+1} = h(s_t, a_t, w_{t+1})$ and $\tilde{Q}_\tau^U \equiv \tilde{Q}_\tau^L \equiv 0$. The pseudo-code of LBQL with experience replay is shown in Algorithm 2 in Appendix B.

4.4 Convergence of LBQL with Experience Replay

In this section, we prove that the version of LBQL with experience replay also converges to the optimal action-value function. We start by stating a lemma that confirms Proposition 3 and Lemma 1 still hold when the penalty terms are computed using (15).

Lemma 2. *If at any iteration n, the penalty terms are computed using the estimated distribution $\hat{p}_n$, i.e., as per (15), then Proposition 3 and Lemma 1 still hold.*

[2]Note that LBQL could, in principle, be adapted to the case of of continuous noise (i.e., where w is continuous random variable) using methods like kernel density estimation (KDE).

Theorem 2 (Convergence of LBQL with experience replay)**.** *Under Assumption 1, the following hold with probability 1:*

(i) $Q'_n(s, a)$ in Algorithm 2 converges to the optimal action-value function $Q^*(s, a)$ for all state-action pairs (s, a).

(ii) The iterates $L_n(s, a), Q'_n(s, a)$, $U_n(s, a)$ in Algorithm 2 converge to the optimal action-value function $Q^*(s, a)$ for all state-action pairs (s, a).

The proof is similar to that of Theorem 1, but using the observations collected in the buffer naturally results in an additional bias term in our analysis. The proof of Lemma 2 shows that as we learn the distribution of the noise, this bias term goes to zero and our original analysis in the unbiased case continues to hold.

Notice that the results in part (ii) of the theorem are, in a sense, stronger than that of Theorem 1(ii). While both achieve asymptotic convergence of the lower and upper bounds to the optimal action-value function, Theorem 2(ii) does not require computing the penalty with the true distribution, i.e., using (3). This is because in the experience replay version, the distribution of the noise random variables is also learned.

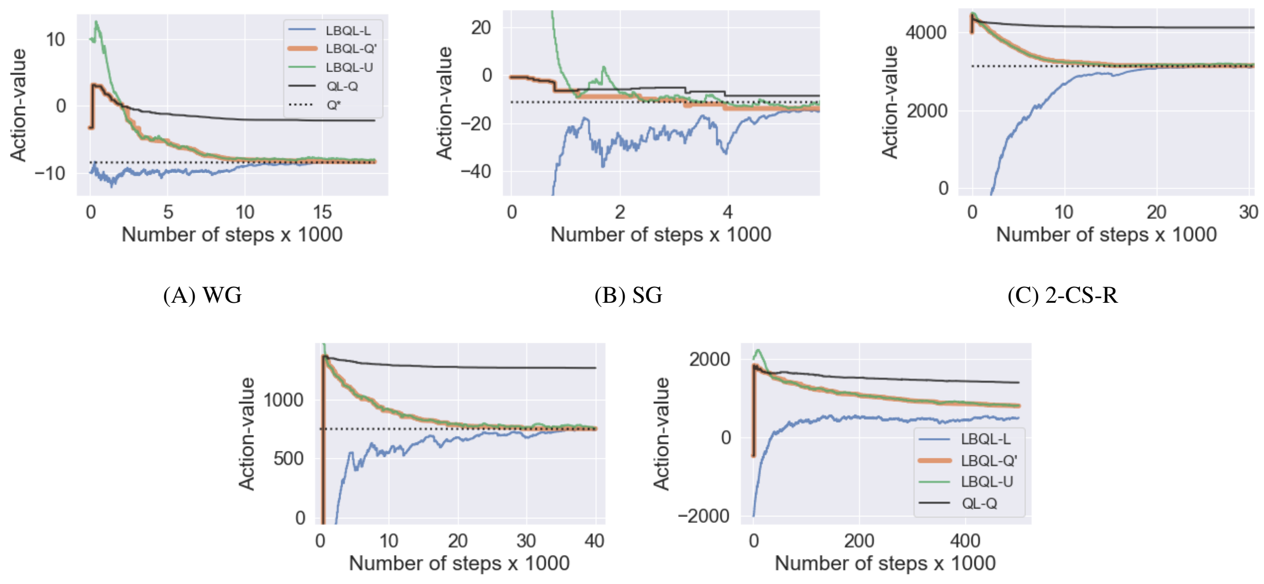
5 Numerical Experiments

In our numerical experiments we make slight modifications to Algorithm 2, which help to reduce its computational requirements. A detailed description of all changes is included in Appendix C. We also open-source a Python package[3] for LBQL that reproduces all experiments and figures presented in this paper. We compare LBQL with experience replay with several algorithms: Q-learning (QL), double Q-learning (Double-QL), speedy Q-learning (SQL), and bias-corrected Q-learning (BCQL) (van Hasselt, 2010; Azar et al., 2011; Lee & Powell, 2019). The environments that we consider are summarized below. Detailed description of the environments, the parameters used for the five algorithms, and sensitivity analysis are deferred to Appendix D.

Windy Gridworld (WG). This is a well-known variant of the standard gridworld problem discussed in Sutton & Barto (2018). There is an *upward wind* with a random intensity. The agent moves extra steps in the wind direction whenever it reaches an affected square. The reward is -1 until the goal state is reached, and the reward is 0 thereafter.

Stormy Gridworld (SG). We then consider a new domain that adds the additional complexity of rain and *multi-directional* wind to windy gridworld. The location of the rain is random and when it occurs, puddles that provide negative rewards are created. The reward is similar to that of WG, except that puddle states provide a reward of -10.

[3]https://github.com/ibrahim-elshar/LBQL_ICML2020.



(A) WG

(B) SG

(C) 2-CS-R

(D) 2-CS

(E) 4-CS

Figure 4: Illustration of LBQL Upper and Lower Bounds.

Repositioning in Two-Location Car-sharing (2-CS-R). Our next benchmark is a synthetic problem of balancing an inventory of cars by repositioning them in a car-sharing platform with two stations (He et al., 2019). The actions are to decide on the number of cars to be repositioned from one station to the other before random demand is realized. All rentals are one-way (i.e., rentals from station A end up at B, and vice-versa). The goal is to maximize revenue for a fixed rental price subject to lost sales and repositioning costs.

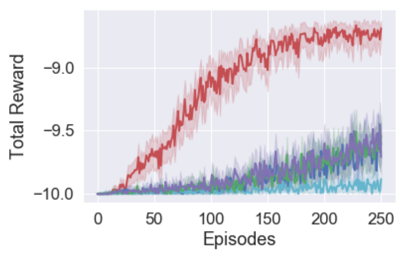
Pricing in Two-Location Car-sharing (2-CS). Here, we consider the benchmark problem of spatial dynamic pricing on a car-sharing platform with two stations, motivated partially by (Bimpikis et al., 2019). The actions are to set a price at each station, which influence the station's (stochastic) demand for rentals. Rentals are one-way and the goal is to maximize revenue under lost sales cost.

Pricing in Four-Location Car-sharing (4-CS). The final benchmark that we consider is a variant of the above pricing problem with four stations. Now, however, we consider both one way and return trips at each station. In this case, we have two sources of randomness: the noise due to stochastic demand and the noise due to the random distribution of fulfilled rentals between the stations.

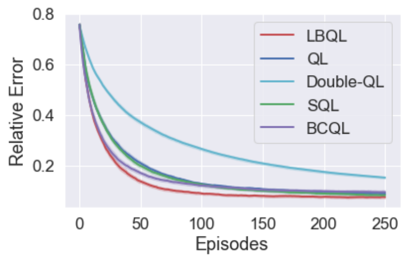
First, we illustrate conceptually in Figure 4 how the upper and lower bounds of LBQL can "squeeze" the Q-learning results toward the optimal value (the plots show a particular state-action pair (s, a) for illustrative reasons). For example, in Figure 4A, we observe that the LBQL iterates (orange) match the Q-learning iterates (solid black) initially, but as the upper bound (green) becomes better estimated, the LBQL iterates are pushed toward the optimal value (dotted black). We see that even though the same hyperparameters are used between LBQL and QL, the new approach is able to quickly converge. In the 4-CS example, Figure 4E, Q^* is not shown since it is computationally difficult to obtain, but the gap between the upper and lower bounds, along with Theorem 2(ii), suggest that LBQL is converging faster than standard Q-learning.

The full results (with 95% confidence intervals) of the numerical experiments are shown in Figure 5. LBQL drastically outperforms the other algorithms in terms of the performance curve on the gridworld domains, but for the car-sharing problems, double Q-learning is superior in the first 20,000 steps. Afterwards, LBQL catches up and remains the best performing algorithm. From the relative error plots (which measure the percent error, in l_2-norm, of the approximate value function with the true optimal value function, i.e., $\|V_n - V^*\|_2/\|V^*\|_2$), we see that LBQL has the steepest initial decline. In windy gridworld and car-sharing, LBQL outperforms other algorithms in terms of relative error, but BCQL and SQL achieve slightly lower relative error than LBQL for stormy gridworld.

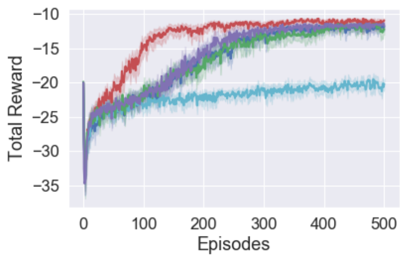
We also conducted a set of sensitivity analysis experiments, where we varied the learning rate and exploration hyperparameters across all algorithms (results are given in Appendix D.3). We examine the number of iterations and CPU time needed to reach 50%, 20%, 5%, and 1% relative error. The results show that LBQL outperforms BCQL, SQL, and



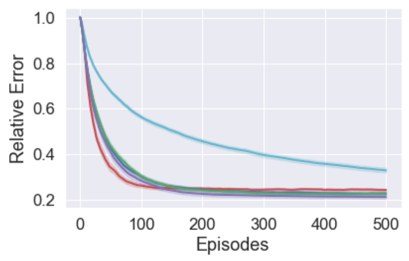
(A) Performance (WG)



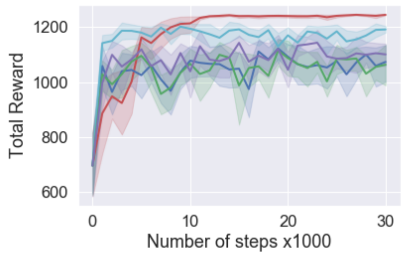
(B) Relative Error (WG)



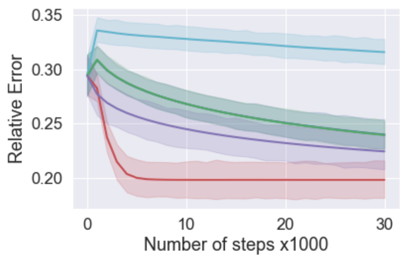
(C) Performance (SG)



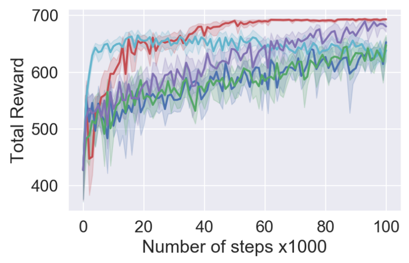
(D) Relative Error (SG)



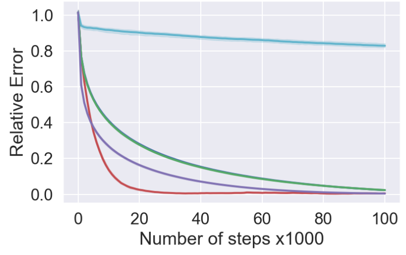
(E) Performance (2-CS-R)



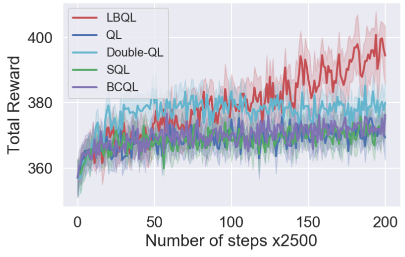
(F) Relative Error (2-CS-R)



(G) Performance (2-CS)



(H) Relative Error (2-CS)



(I) Performance (4-CS)

Figure 5: Results from Numerical Experiments.

QL in terms of both iterations and CPU time in reaching 20%, 5%, and 1% relative error across all 15 hyperparameter settings that we tested. For the case of 50% relative error, BCQL outperforms LBQL in five out of 15 cases. This indicates that LBQL is significantly more robust to hyperparameter settings than the other algorithms. Roughly speaking, this robustness might be attributed to the "approximate planning" aspect of the algorithm, where lower and upper bounds are computed.

6 Conclusion

In this paper, we present the LBQL algorithm and prove its almost sure convergence to the optimal action-value function. We also propose a practical extension of the algorithm that uses an experience replay buffer. Numerical results illustrate the rapid convergence of our algorithm empirically when compared to a number of well-known variants of Q-learning on five test problems. LBQL is shown to have superior performance, robustness against learning rate and exploration strategies, and an ability to mitigate maximization bias.

Interesting future work is the extension of our new framework to the model-based RL setting, where the transition function f is learned while the policy is optimized. Other interesting future work includes looking beyond the tabular case and adapting our algorithm to the setting of value function approximations, such as DQN (Mnih et al., 2013).

References

Andersen, L. and Broadie, M. Primal-dual simulation algorithm for pricing multidimensional American options. *Management Science*, 50(9):1222–1234, 2004.

Azar, M. G., Munos, R., Ghavamzadaeh, M., and Kappen, H. J. Speedy Q-learning. In *Advances in Neural Information Processing Systems 24*, 2011.

Barto, A. G., Bradtke, S. J., and Singh, S. P. Learning to act using real-time dynamic programming. *Artificial intelligence*, 72(1-2):81–138, 1995.

Bertsekas, D. P. and Tsitsiklis, J. N. *Neuro-dynamic programming*, volume 5. Athena Scientific Belmont, MA, 1996.

Bimpikis, K., Candogan, O., and Saban, D. Spatial pricing in ride-sharing networks. *Operations Research*, 2019.

Brown, D. B. and Haugh, M. B. Information relaxation bounds for infinite horizon markov decision processes. *Operations Research*, 65(5):1355–1379, 2017.

Brown, D. B., Smith, J. E., and Sun, P. Information relaxations and duality in stochastic dynamic programs. *Operations Research*, 58(4-part-1):785–801, 2010.

Chen, N., Ma, X., Liu, Y., and Yu, W. Information relaxation and a duality-driven algorithm for stochastic dynamic programs. *arXiv preprint arXiv:2007.14295*, 2020.

Dann, C., Li, L., Wei, W., and Brunskill, E. Policy certificates: Towards accountable reinforcement learning. *arXiv preprint arXiv:1811.03056*, 2018.

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

Haugh, M. B. and Kogan, L. Pricing American options: a duality approach. *Operations Research*, 52(2):258–270, 2004.

He, F. S., Liu, Y., Schwing, A. G., and Peng, J. Learning to play in a day: Faster deep reinforcement learning by optimality tightening. *arXiv preprint arXiv:1611.01606*, 2016.

He, L., Hu, Z., and Zhang, M. Robust repositioning for vehicle sharing. *Manufacturing & Service Operations Management*, 2019.

Jaakkola, T., Jordan, M. I., and Singh, S. P. Convergence of stochastic iterative dynamic programming algorithms. In *Advances in Neural Information Processing Systems*, pp. 703–710, 1994.

Jiang, D. R., Al-Kanj, L., and Powell, W. B. Optimistic Monte Carlo tree search with sampled information relaxation dual bounds. *Operations Research (forthcoming)*, 2020.

Kim, J. H. and Powell, W. B. Optimal energy commitments with storage and intermittent supply. *Operations Research*, 59(6):1347–1360, 2011.

Kunnumkal, S. and Topaloglu, H. Using stochastic approximation methods to compute optimal base-stock levels in inventory control problems. *Operations Research*, 56(3): 646–664, 2008.

Kushner, H. and Yin, G. G. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.

Lee, D. and Powell, W. B. Bias-corrected Q-learning with multistate extension. *IEEE Transactions on Automatic Control*, 2019.

McMahan, H. B., Likhachev, M., and Gordon, G. J. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In *Proceedings of the 22nd international conference on Machine learning*, pp. 569–576. ACM, 2005.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

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

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

Rogers, L. C. G. Monte Carlo valuation of American options. *Mathematical Finance*, 12(3):271–286, 2002.

Sanner, S., Goetschalckx, R., Driessens, K., and Shani, G. Bayesian real-time dynamic programming. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

Secomandi, N. A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research*, 49(5):796–802, 2001.

Smith, T. and Simmons, R. Focused real-time dynamic programming for mdps: Squeezing more out of a heuristic. In *AAAI*, pp. 1227–1232, 2006.

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

Szepesvári, C. The asymptotic convergence-rate of Q-learning. In *Advances in Neural Information Processing Systems*, pp. 1064–1070, 1998.

Tsitsiklis, J. N. Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16(3):185–202, 1994.

van Hasselt, H. Double Q-learning. In *Advances in Neural Information Processing Systems*, pp. 2613–2621, 2010.

Watkins, C. J. C. H. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989.

Zanette, A. and Brunskill, E. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. *arXiv preprint arXiv:1901.00210*, 2019.