Exploiting Structure and Relaxations in Reinforcement Learning and Stochastic Optimal Control

by

Ibrahim El Shar

M.S. of Industrial Engineering, American University of Beirut, 2016

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2022

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Ibrahim El Shar

It was defended on

October 28th 2022

and approved by

Daniel Jiang, Ph.D, Assistant Professor, Department of Industrial Engineering

Lisa Maillart, Ph.D, Professor & Department Chair, Department of Industrial Engineering

Jayant Rajgopal, Ph.D, Professor, Department of Industrial Engineering

Bo Zeng, Ph.D, Associate Professor, Department of Industrial Engineering

Masoud Barati, Ph.D, Asssistant Professor, Department of Electrical and Computer Engineering

Dissertation Director: Daniel Jiang, Ph.D, Assistant Professor, Department of Industrial Engineering

Exploiting Structure and Relaxations in Reinforcement Learning and Stochastic Optimal Control

Ibrahim El Shar, PhD

University of Pittsburgh, 2022

Stochastic optimal control studies the problem of sequential decision making under uncertainty. Dynamic programming (DP) offers a principled approach to solving stochastic optimal control problems. A major drawback of DP methods, however, is that they become quickly intractable in large-scale problems. In this thesis, we show how structural results and various relaxation techniques can be used to obtain good approximations and accelerate learning. First, we propose a new provably convergent variant of Q-learning that leverages upper and lower bounds derived using information relaxation techniques to improve performance in the tabular setting. Second, we study weakly coupled DPs which are a broad class of stochastic sequential decision problems comprised of multiple subproblems coupled by some linking constraints but are otherwise independent. We propose another Q-learning based algorithm that makes use of Lagrangian relaxation to generate upper bounds and improve performance. We also extend our algorithm to the function approximation case using Deep Q-Networks. Finally, we study the problem of spatial dynamic pricing for a fixed number of shared resources that circulate in a network. For the general network, we show that the optimal value function is concave and for a network composed of two locations, we show that the optimal policy enjoys certain monotonicity and bounded sensitivity properties. We use these results to propose a novel heuristic algorithm which we compare against several baselines.

Keywords: Markov decision processes; Approximate dynamic programming; Reinforcement learning.

Table of Contents

List of Tables

List of Figures

1.0 Introduction

Sequential decision-making problems are often modeled as Markov decision processes (MDPs). In this setting actions influence not only immediate rewards but also the future states and consequently the future rewards. The goal is to find an optimal policy that maps states to actions which maximizes the total sum of rewards. DP based approaches such as policy iteration and value iteration are classical methods for solving infinite horizon MDPs. These methods however are only suitable for MDPs with small finite state spaces. In addition, they require the knowledge of the reward function and transition dynamics. In many of the problems to which we wish to find a good policy the state space is huge and combinatorial, and the model may not be available to us. In such cases, the best we can hope for is finding an approximate solution using finite computational resources. Various relaxation techniques and different types of problem specific structures can be used to obtain good approximations and make learning more efficient. In this thesis, we show how reinforcement learning and optimal control can benefit from these techniques through the following contributions:

1. In Chapter 2, we study sequential decision making problems that are stochastic due to exogenous random variables that affects the problem dynamics and rewards. Often case, little is known about the support and distribution of these exogenous variables but the way they affect our problem dynamics is partially known. This is the case in many real-world problems. For example, inventory control problems where the next inventory state is given by a well-specified function of the demand. Other examples include vehicle routing, energy operations, portfolio optimization and dynamic pricing in car-sharing problems. We exploit this partial knowledge of the dynamics by proposing a new Q-learning variant that makes use of information relaxation to learn and solve these problems efficiently.
2. In Chapter 3, we study a broad class of sequential decision making problems called weakly coupled DPs. These problem consist of multiple subproblems that are independent except for a linking constraint on the action space. These problems are hard to solve since they become exponentially larger with the number of subproblems. We propose a Q-

learning algorithm that exploits the weakly coupled DP structure by using Lagrangian relaxation to generate upper bounds that are in turn used to accelerate learning and improve performance. We then extend our algorithm to the function approximation case by utilizing Deep Q-Networks.

3. Finally, we study problems where our actions have indirect influence on resources that circulate in a network structure. These type of problems arise in shared resource systems with a fixed number of resources where a decision maker needs to take actions that influence the distribution of the resources over the network in such away that is appealing from both revenue and logistic perspectives. We analyze the structure of the value function and the policy in these problems and propose an effective heuristic that exploits this structure.

1.1 Lookahead-bounded Q-learning

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 (QL) 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.

1.2 Weakly Coupled Deep Q-Networks

We introduce Weakly Coupled Deep Q-Networks (WCDQN), a novel deep reinforcement learning algorithm that improves the performance of the standard Deep Q-Networks (DQN) algorithm and its sister methods on a broad class of structured problems. WCDQN employs multiple simultaneous DQN agents such that each runs on a separate easier subproblem and when combined they form an upper bound on the action value of the original problem. The upper bound is then used to constrain and guide DQN on the full problem towards optimality. Theoretically, we show that the tabular version of our algorithm called Weakly Coupled Q-learning (WCQL) converges almost surely to the optimal action-value function. Numerical experiments on benchmark problems show that our algorithm exhibits faster convergence when compared to DQN/QL and several related techniques.

1.3 Spatial Dynamic Pricing for Shared Resources Systems

Inspired by the growing popularity of shared transport systems, we then study dynamic pricing of a fixed number of rental units to serve price sensitive customers over a network of locations, in Chapter 4. One of the main challenges faced by these systems is dealing with imbalanced rental units resulting from spatially imbalanced demand. A dynamic pricing framework offers a natural approach to modulate the demand. We formulate the problem as a stochastic dynamic program and analyze the structure of the optimal value and policy functions. For the general problem, we show that the value function is concave in the state. For the specific case, where we only have two locations, we show that the value function enjoys a certain type of discrete convexity and the optimal policy has monotonicity and bounded sensitivity properties. We then propose a heuristic called leave-one-out aggregation that exploits the structure of the optimal policy for the two-locations network. The heuristic decompose an N-location problem into N two-location problem and solve each of them separately to obtain a policy for the full problem.

2.0 Lookahead-Bounded Q-learning

Since its introduction by Watkins in 1989 [74], Q-learning has become one of the most widely-used reinforcement learning (RL) algorithms [63], due to its conceptual simplicity, ease of implementation, and convergence guarantees [33, 68, 9]. 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 [64, 26, 45].

In this chapter, 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 [9] and operations research [53] 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) [14].

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} [42]. For *vehicle routing*, f is often simply the routing decision itself, while w_{t+1} are exogenous demands that require observation [59]. In *energy operations*, a typical setting is to optimize storage that behaves through linear transitions f together with unpredictable renewable supply, w_{t+1} [38]. In *portfolio optimization*, f is the next portfolio, and w_{t+1} represents random prices [55]. In Section 2.4 of this work, we discuss in detail another application domain that follows this paradigm: *repositioning and spatial dynamic pricing for car sharing* [31, 12].

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 section.

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 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 chapter is organized as follows. In the next section, we review the related literature. In Section 2.2, we introduce the notation and review the basic theory of IR. In Section 2.3, we present our algorithm along with its theoretical results. In Section 2.4, we show the numerical results where LBQL is compared to other Q-learning variants. Finally, we state conclusions and future work in Section 2.5.

2.1 Related Literature

Upper and lower bounds on the optimal value have recently been used by optimism-based algorithms, e.g., [22] and [77]. 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) [8], Bounded RTDP [46], Bayesian RTDP [56] and Focused RTDP [61] 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 this work is the work of He et al. [30], 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. [34] use IR dual bounds in a tree search algorithm in order to ignore parts of the tree. Recent work by Chen et al. [17] 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).

2.2 Background

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

2.2.1 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 this section, is taken with respect to $\mathbf{P}$. Our objective is

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

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 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 [54]. 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 [74], is to approximate Q^*.

2.2.2 Information Relaxation Duality

Now let us give a brief review of the theory behind information relaxation duality from Brown et al. [14], which is a way of computing an *upper bound* on the value and action-value functions. This generalizes work by Rogers [55], Haugh and Kogan [28], and Andersen and Broadie [3] 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

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

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

$$z_t^{\pi}(s_t, a_t, w_{t+1} \,|\, \varphi) := \gamma^{t+1}\Big(\varphi(s_{t+1}, \pi(s_{t+1})) - \mathbf{E}\Big[\varphi\big(f(s_t, a_t, w), \pi(f(s_t, a_t, w))\big)\Big]\Big) \tag{2.1}$$

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 and Haugh [13], 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:

$$Q^U(s_0, a_0) = \mathbf{E}\Big[\max_{\mathbf{a}}\Big\{\sum_{t=0}^{\infty}\Big(\gamma^t r(s_t, a_t) - z_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi)\Big)\Big\}\Big], \tag{2.2}$$

where $\mathbf{a} := (a_0, a_1, \ldots)$ is an infinite sequence of actions. Brown and Haugh [13] 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 (2.2) 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 $Q^U(s_0, a_0)$.

2.2.3 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 [54]), 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 (2.1), except we now consider $\varphi \in \mathcal{Q}$:

$$\zeta_t^{\pi}(s_t, a_t, w_{t+1} \,|\, \varphi) := \varphi(s_{t+1}, \pi(s_{t+1})) - \mathbf{E}\Big[\varphi\big(h(s_t, a_t, w), \pi(h(s_t, a_t, w))\big)\Big], \tag{2.3}$$

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 [13].

Proposition 2.2.1 (Duality Results, Proposition 2.2 in [13])**.** *The following duality results are stated for the absorption time formulation of the problem.*

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

$$Q^{\pi}(s_0, a_0) \leq \mathbf{E}\Big[\max_{\mathbf{a}} \sum_{t=0}^{\tau-1} \Big(r(s_t, a_t) - \zeta_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi)\Big)\Big] \tag{2.4}$$

(ii) Strong Duality: It holds that

$$Q^*(s_0, a_0) = \inf_{\varphi \in \mathcal{Q}} \mathbf{E}\Big[\max_{\mathbf{a}} \sum_{t=0}^{\tau-1} \Big(r(s_t, a_t) - \zeta_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi)\Big)\Big], \tag{2.5}$$

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

The DP inside the expectation of the right hand side of (2.4) 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

$$Q_t^U(s_t, a_t) = r(s_t, a_t) - \zeta_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi) + \max_a Q_{t+1}^U(s_{t+1}, a), \tag{2.6}$$

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 $Q_0^U(s_0, a_0)$.

2.2.4 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

$$Q_t^L(s_t, a_t) = r(s_t, a_t) - \zeta_t^{\pi}(s_t, a_t, w_{t+1} \,|\, \varphi) + Q_{t+1}^L(s_{t+1}, \pi(s_{t+1})), \tag{2.7}$$

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.

2.3 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^*.

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.

2.3.1 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

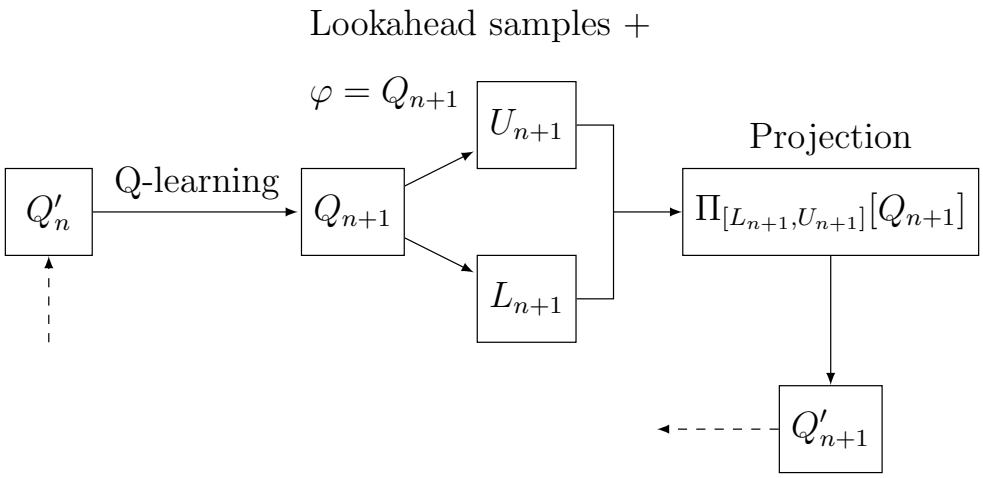


Figure 1: Illustration of LBQL Algorithm at iteration n.

empirical version of (2.3) is simply given by:

$$\hat{\zeta}_t^{\pi}(s_t, a_t, w_{t+1} \,|\, \varphi) := \varphi(s_{t+1}, \pi(s_{t+1})) - \frac{1}{K}\sum_{k=1}^{K} \varphi\big(h(s_t, a_t, w_{t+1}^k), \pi(h(s_t, a_t, w_{t+1}^k))\big), \quad (2.8)$$

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 (2.6) and (2.7) using $\hat{\zeta}_t^{\pi}$, where in (2.7) 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

$$\hat{Q}_t^U(s_t, a_t) = r(s_t, a_t) - \hat{\zeta}_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi) + \max_a \hat{Q}_{t+1}^U(s_{t+1}, a) \quad (2.9)$$

$$\hat{Q}_t^L(s_t, a_t) = r(s_t, a_t) - \hat{\zeta}_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi) + \hat{Q}_{t+1}^L(s_{t+1}, \pi_\varphi(s_{t+1})) \quad (2.10)$$

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 2.3.1. *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.1.1. The proof is similar to that of Proposition 2.3(iv) of [14], 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 $L_0(s,a) \leq Q_0(s,a) \leq U_0(s,a)$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$.

Algorithm 1: Lookahead-Bounded Q-Learning

Input: Initial estimates $L_0 \leq Q_0 \leq U_0$, 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., ϵ-greedy). Observe w_{n+1}. Let

$$Q_{n+1}(s_n,a_n) = Q'_n(s_n,a_n) + \alpha_n(s_n,a_n)\Big[r_n(s_n,a_n) + \gamma \max_a Q'_n(s_{n+1},a) - Q'_n(s_n,a_n)\Big].$$

Set $\varphi = Q_{n+1}$. Using a sample path $\mathbf{w}$, compute $\hat{Q}_0^U(s_n,a_n)$ and $\hat{Q}_0^L(s_n,a_n)$ using (2.9) & (2.10).

Update and enforce upper and lower bounds:

$$U_{n+1}(s_n,a_n) = \Pi_{[-\rho,\infty]}\Big[U_n(s_n,a_n) + \beta_n(s_n,a_n)\left[\hat{Q}_0^U(s_n,a_n) - U_n(s_n,a_n)\right]\Big], \tag{2.11}$$

$$L_{n+1}(s_n,a_n) = \Pi_{[\infty,\rho]}\Big[L_n(s_n,a_n) + \beta_n(s_n,a_n)\left[\hat{Q}_0^L(s_n,a_n) - L_n(s_n,a_n)\right]\Big], \tag{2.12}$$

$$Q'_{n+1}(s_n,a_n) = \Pi_{[L_{n+1}(s_n,a_n),\, U_{n+1}(s_n,a_n)]}\left[Q_{n+1}(s_n,a_n)\right]. \tag{2.13}$$

end for

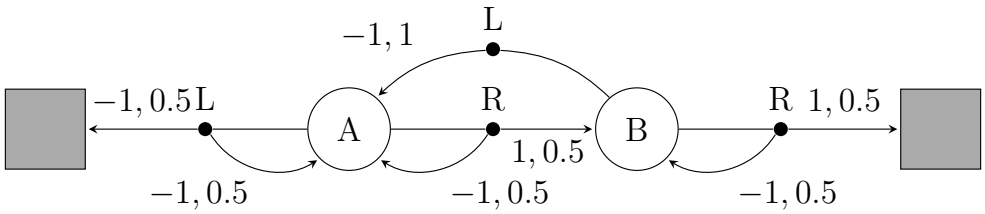


Figure 2: A simple stochastic MDP.

Example 2.1. *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.*

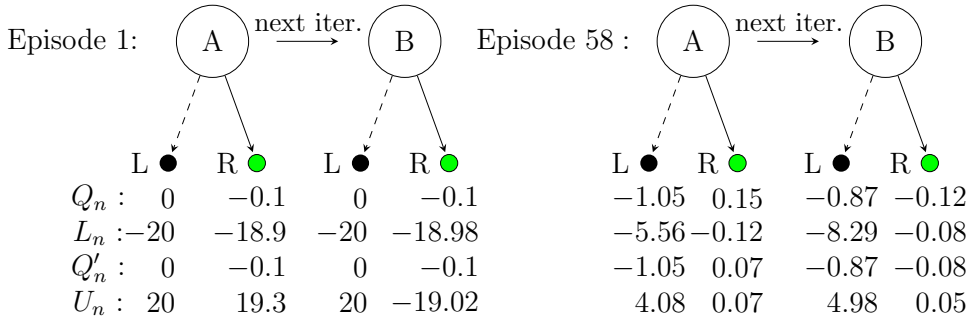


Figure 3: An illustration of LBQL iterates for Example 1.

We let $\alpha_n = 0.1$*,* $\beta_n = 0.05$ *for all* n*. Figure 3 illustrates two iterations from the first and the* 58*th episodes. Initially* $Q_0(s, a) = 0$ *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.*

2.3.2 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.1.

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.1.2 in the Appendix.

Proposition 2.3.2 (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*

$$L_n(s, a) \leq Q^*(s, a) + \eta \quad \textit{and} \quad Q^*(s, a) - \eta \leq U_n(s, a),$$

for all iterations $n \geq n_0$*.*

Proposition 2.3.2(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 2.3.2(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 2.3.2(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 (2.11) and (2.12) 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 2.3.2(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 2.3.1 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.1.3 of the Appendix.

Lemma 2.3.1 (Consistency of Bounds)**.** *If $L_0(s,a) \leq U_0(s,a)$, then $L_n(s,a) \leq U_n(s,a)$ for all iterations n and for all $(s,a) \in \mathcal{S} \times \mathcal{A}$.*

In particular, Lemma 2.3.1 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 2.3.1 follows mainly by the fact that we are using the same sample path to solve the upper and lower bound inner problems, (2.9) and (2.10), respectively. Before stating our convergence results, we first state a typical assumption on the stepsizes and the state visits.

Assumption 2.3.1. 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 2.3.1 (Convergence of LBQL)**.** *Under Assumption 2.3.1, 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 (2.3)*, 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.

2.3.3 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 $\hat{\mathbf{E}}_n[\,.\,]$ 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:

$$\tilde{\zeta}_t^{\pi}(s_t, a_t, w \,|\, \varphi) := \varphi(s_{t+1}, \pi(s_{t+1})) - \hat{\mathbf{E}}_n[\varphi\big(h(s_t, a_t, w), \pi(h(s_t, a_t, w))\big)], \tag{2.14}$$

[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).

and given a sample path $\mathbf{w} = (w_1, w_2, \ldots, w_\tau)$ the inner problems analogous to (2.9) and (2.10) are given by

$$\tilde{Q}_t^U(s_t, a_t) = r(s_t, a_t) - \tilde{\zeta}_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi) + \max_a \tilde{Q}_{t+1}^U(s_{t+1}, a) \tag{2.15}$$

$$\tilde{Q}_t^L(s_t, a_t) = r(s_t, a_t) - \tilde{\zeta}_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi) + \tilde{Q}_{t+1}^L(s_{t+1}, \pi_\varphi(s_{t+1})) \tag{2.16}$$

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 4 in Appendix A.2.

2.3.4 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 2.3.2 and Lemma 2.3.1 still hold when the penalty terms are computed using (2.14).

Lemma 2.3.2. *If at any iteration n, the penalty terms are computed using the estimated distribution $\hat{p}_n$, i.e., as per* (2.14)*, then Proposition 2.3.2 and Lemma 2.3.1 still hold.*

Theorem 2.3.2 (Convergence of LBQL with experience replay)**.** *Under Assumption 2.3.1, the following hold with probability 1:*

(i) $Q'_n(s, a)$ in Algorithm 4 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 4 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 2.3.1, but using the observations collected in the buffer naturally results in an additional bias term in our analysis. The proof of Lemma 2.3.2 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 2.3.1(ii). While both achieve asymptotic convergence of the lower and upper bounds to the optimal action-value function, Theorem 2.3.2(ii) does not require computing the penalty with the true distribution, i.e., using (2.3). This is because in the experience replay version, the distribution of the noise random variables is also learned.

2.4 Numerical Experiments

In our numerical experiments we make slight modifications to Algorithm 4, which help to reduce its computational requirements. A detailed description of all changes is included in Appendix A.3. We also open-source a Python package[3] for LBQL that reproduces all experiments and figures presented in this section. 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) [70, 5, 45]. 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 A.4.

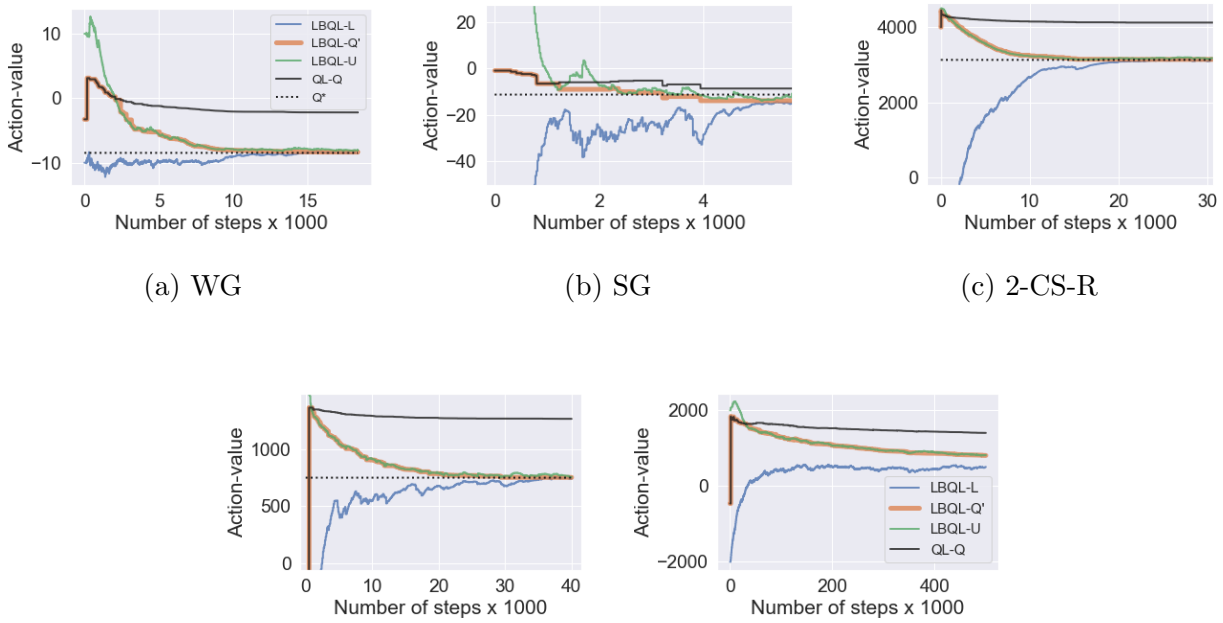
Windy Gridworld (WG). This is a well-known variant of the standard gridworld problem discussed in [63]. There is an *upward wind* with a random intensity. The agent moves extra steps in the wind direction whenever it reaches an affected square. The reward is -1 until the goal state is reached, and the 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.

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 [31]. 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 [12]. 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

[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.

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 2.3.2(ii), suggest that LBQL is converging faster than standard Q-learning.

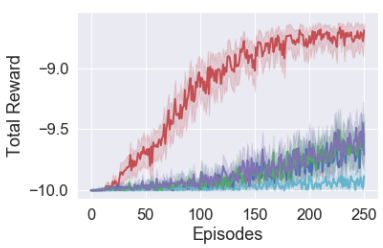
The full results (with 95% confidence intervals) of the numerical experiments are shown in Figures 5 and 6. LBQL drastically outperforms the other algorithms in terms of the performance curve on the gridworld domains, but for the car-sharing problems, double Q-learning is superior in the first 20,000 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 A.4.3). 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

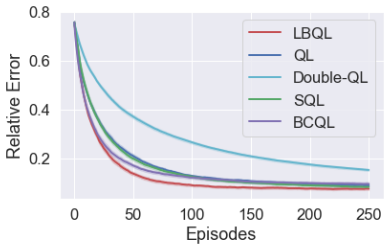
QL in terms of both iterations and CPU time in reaching 20%, 5%, and 1% relative error across all 15 hyperparameter settings that we tested. For the case of 50% relative error, BCQL outperforms LBQL in five out of 15 cases. This indicates that LBQL is significantly more robust to hyperparameter settings 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.

2.5 Conclusion

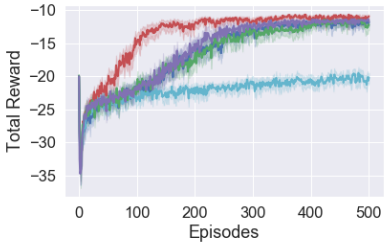
In this chapter, we present the LBQL algorithm and prove its almost sure convergence to the optimal action-value function. We also propose a practical extension of the algorithm that uses an experience replay buffer. Numerical results illustrate the rapid convergence of our algorithm empirically when 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 [47].



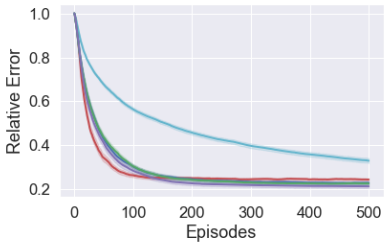
(a) Performance (WG)



(b) Relative Error (WG)

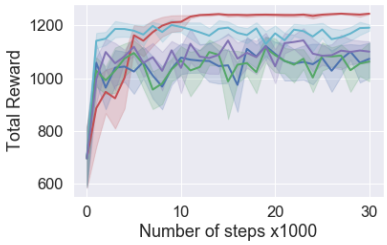


(c) Performance (SG)

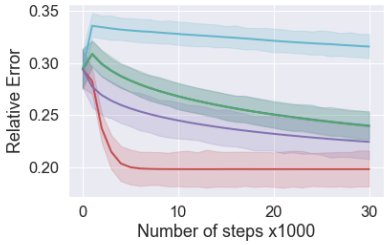


(d) Relative Error (SG)

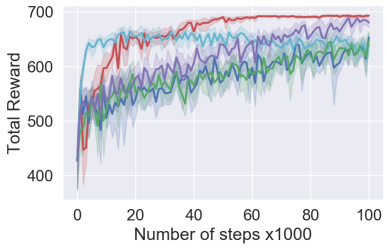
Figure 5: Results from the Gridworld Experiments.



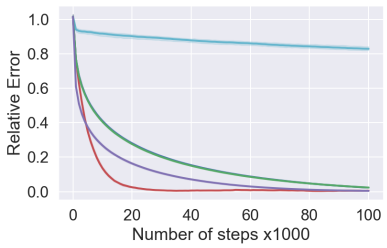
(a) Performance (2-CS-R)



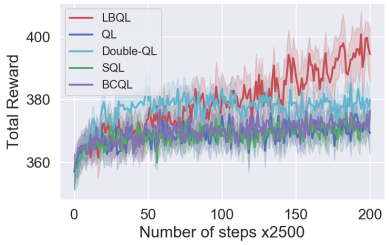
(b) Relative Error (2-CS-R)



(c) Performance (2-CS)



(d) Relative Error (2-CS)



(e) Performance (4-CS)

Figure 6: Results from the CS Experiments.

3.0 Weakly Coupled Deep Q-Networks

Sequential decision-making is a core topic in machine learning, which involves learning how to act in an uncertain environment so as to maximize a numerical reward signal. Reinforcement learning (RL) formalizes this problem by considering an artificial agent that interacts with an environment, gathering experience in form of next states and reward signals, and aims to learn a mapping from states to actions that maximizes the discounted sum of rewards [9, 63]. Notably, the agent's actions have both immediate and future consequences on the reward signals.

Due to its conceptual simplicity, ease of implementation and convergence guarantees, Q-learning by Watkins [74], is one of the most widely-used RL algorithms [33, 68, 9]. Inspired by the success of deep learning in computer vision and natural language processing [44, 41], the *deep Q-networks* (DQN) algorithm of Mnih et al. [47] extends the fundamental ideas behind Q-learning to the case where Q-functions are approximated using deep neural networks. The effectiveness of DQN was famously demonstrated on a set of Atari games, and achieved "human-level" performance. Several extensions of DQN has been proposed in the literature including Double DQN [71] and Dueling DQN [72].

One drawback of both Q-learning and DQN is that scaling to large action spaces can be challenging and often require complex modifications [24, 69], hindering the potential of RL for practical implementations on real-world problems. More generally, Q-learning and DQN typically require a large number of samples, which can be costly to obtain in real environments.

In this chapter, we focus our attention on a class of stochastic sequential decision making problems called *weakly coupled MDPs* and show how, for these problems, one can leverage their inherent structure to alleviate the challenges mentioned above. Weakly coupled MDPs refers to a broad class of stochastic sequential decision-making problems that consist of multiple subproblems that are independent of each other except for a coupling constraint on the action space [29]. This type of problem structure is fairly general, with many practical examples exhibiting this structure, including supply chain management, multiclass queueing

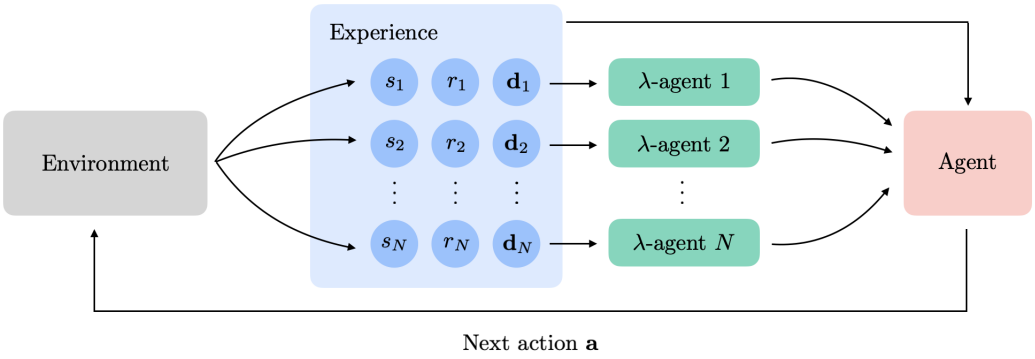


Figure 7: Our WCMDP RL Approach.

networks, auctions, stochastic job scheduling, disaster relief scheduling and a wide variety of bandit-type problems [29, 1]. Such MDPs are computationally challenging to solve using naive techniques, given that their state and action spaces grow exponentially with the number of subproblems [48].

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

1. We propose a novel RL algorithm called *weakly coupled deep Q-networks* (WCDQN), that exploits weakly coupled structure by employing multiple subproblem $\boldsymbol{\lambda}$-*agents* to improve the performance of DQN, Figure 7. Our algorithm makes better use of the collected experience by dynamically estimating upper (and lower) bounds via Lagrangian relaxation, and integrates them into a constrained optimization approach when fitting the Q-networks. We also propose Lagrangian DQN which is a lighter version of our algorithm that does not require computing bounds and still performs well in practice. Neither approach uses any knowledge of the environment dynamics nor do they require restrictive assumptions on the weakly coupled problem such as indexibility [75, 15].

2. We also propose and analyze a tabular version of our algorithm called *weakly coupled Q-learning* (WCQL), which serves to conceptually motivate WCDQN. We show that WCQL converges almost surely to the optimal action-value function.

3. We show numerical experiments on a suite of realistic problems, including electric vehicle charging station operation, multi-product inventory control with a production constraint, and online stochastic ad matching. The results show that our proposed algorithms outperform baselines by a relatively large margin.

In the next section, we review some of the related techniques in the literature and compare it to our work.

3.1 Related Literature

There are a number of papers in the literature that attempt to enhance the learning, stability, and convergence of the DQN algorithm as it was first proposed by Mnih et al. [47]. For example, to overcome the estimation problem and improve stability, Van Hasselt et al. [71] proposes *double DQN*, which adapts the tabular approach of *double Q-learning* from Van Hasselt [70] to the deep RL setting. The main idea is to use a different network for the action selection and evaluation steps. Schaul et al. [57] modifies the experience replay buffer sampling to prioritize certain tuples, and Wang et al. [72] adds a "dueling architecture" to double DQN that combines two components, an estimator for the state value function and an estimator for the state-dependent action advantage function. Another example is *bootstrapped DQN* proposed by Osband et al. [49], which extends standard DQN to learn a distribution over Q-values, allowing the agent to perform temporally-extended exploration by executing a policy based on a sample of the Q-function. The *amortized Q-learning* algorithm of Van de Wiele et al. [69] integrates a learned state-dependent proposal distribution over actions into DQN to avoid the step of maximizing over a large action space.

Our approach, WCDQN (and WCQL), differs from the above works in the sense that while these papers focus on improvements made to certain components of the DQN algorithm (e.g., network architecture, experience replay buffer, exploration strategy), we focus on exploiting the structure of a class of important problems.

More closely related to our work is He et al. [30], which uses a similar constrained optimization approach to enforce upper and lower bounds on the optimal action value function

in the DQN algorithm. Their bounds, however, are derived by exploiting multistep returns in contrast to the dynamically-computed Lagrangian relaxation bounds that we propose in this work. He et al. [30] also does not provide any convergence guarantees for their approach.

In addition, El Shar and Jiang [25] proposed a convergent variant of Q-learning that leverages upper and lower bounds derived using the information relaxation technique of Brown et al. [14] to improve performance of tabular Q-learning. Although our work shares the high-level idea of bounding Q-learning iterates, El Shar and Jiang [25] focused on problems with *partially known transition models* (which are necessary for information relaxation) and the approach did not readily extend to the function approximation setting (i.e., DQN). Besides focusing on a different set of problems (weakly coupled MDPs), our proposed approach is model-free and naturally integrates with DQN.

There are two major ways to decompose weakly coupled DPs: mathematical programming and Lagrangian relaxation. The mathematical programming approach formulates the Bellman equation as a linear program where the value function is approximated as the sum of the subproblem value functions. This approach is due to Schweitzer and Seidmann [58] and was extended to the function approximation case using basis functions by De Farias and Van Roy [23]. Lagrangian relaxation, on the other hand, involves relaxing the linking constraints on the action space by penalizing the objective. This approach was used to generate heuristic index policies for restless bandits by Whittle [75] and further studied by Hawkins [29] and Adelman and Mersereau [1] for weakly coupled DPs. The latter show various results that compare these two relaxations to each other. Lagrangian relaxation has been used by Bertsimas and Mersereau [10] in interactive marketing problems and by Talluri and Van Ryzin [65] and Topaloglu [66] for revenue management. Recent work that involves Lagrangian relaxation includes the work of Nadarajah and Cire [48] and Brown and Zhang [16]. Nadarajah and Cire [48] present a class of network relaxations that embed an exact network encoding of the linking constraints into a linear programming flow model. They develop a procedure to obtain self-adapting relaxations that automatically adjusts to the structure of the linking constraints. Brown and Zhang [16] extends Adelman and Mersereau [1] to provide theoretical justification on the closeness of the bounds obtained by the mathematical programming approach and Lagrangian relaxation and provide conditions under

which the gap between the two relaxations is zero.

Finally, Killian et al. [36] studies restless, multi-armed bandits (RMAB) and propose two Q-learning approaches: the first is based on the Whittle index policy which requires indexibility and restrictive assumptions on the bandit's actions, while the second seeks an upper bound on the value function by minimizing the Lagrangian value function. Follow-up work in Killian et al. [37] focuses on a robust variant of RMAB and a double oracle algorithm that utilizes policy optimization to minimize minimax regret. Our work differs from these papers in several essential ways: first, we study the more general case of weakly coupled MDPs; second, we do not require any restrictive assumptions on our model; and lastly, our methods target the optimal value and policy (with the help of Lagrangian relaxation) rather than setting the Lagrangian value function as the end goal.

3.2 Preliminaries

3.2.1 Weakly Coupled MDPs

We study an infinite horizon weakly coupled DP with finite state space $\mathcal{S} := \mathcal{X} \times \mathcal{W}$ and finite action space $\mathcal{A}$, where $\mathcal{X}$ is the endogenous part and $\mathcal{W}$ is the exogenous part of the full state space. We use the general setup of weakly coupled MDPs from [16]. Suppose that the problem can be decomposed into N subproblems. The state space of subproblem i is denoted by $\mathcal{S}_i := \mathcal{X}_i \times \mathcal{W}$ and the action space is denoted by $\mathcal{A}_i$, such that

$$\mathcal{X} = \otimes_{i=1}^{N} \mathcal{X}_i \quad \text{and} \quad \mathcal{A} = \otimes_{i=1}^{N} \mathcal{A}_i.$$

In each period, the decision maker observes an exogenously and independently evolving state $w \in \mathcal{W}$, along with the endogenous states $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ of the N subproblems. Note that w is shared by all of the subproblems, and this is reflected in the notation we use throughout the paper, where $\mathbf{s} = (\mathbf{x}, w) \in \mathcal{S}$ represents the full state and $s_i = (x_i, w)$ is the state of subproblem i.

The *linking constraints*, which operate across the subproblems, take the form $\sum_{i=1}^{N} \mathbf{d}_i(s_i, a_i) \leq \mathbf{b}(w)$, where $\mathbf{b}(w) \in \mathbb{R}^d$ and $\mathbf{d}_i(s_i, a_i) \in \mathbb{R}^d$ is associated with subproblem i. We also denote by $\mathbf{d}(\mathbf{s}, \mathbf{a}) = \{\mathbf{d}_i(s_i, a_i)\}_{i=1}^{N}$. The set of feasible actions for state $\mathbf{s}$ is given by

$$\mathcal{A}(\mathbf{s}) = \left\{ \mathbf{a} \in \mathcal{A} : \sum_{i=1}^{N} \mathbf{d}_i(s_i, a_i) \leq \mathbf{b}(w) \right\}.$$

After observing state $\mathbf{s} = (\mathbf{x}, w)$, the decision maker selects a feasible action $\mathbf{a} \in \mathcal{A}(\mathbf{s})$.

The transition probabilities for the endogenous component is denoted $p(\mathbf{x}' \,|\, \mathbf{x}, \mathbf{a})$ and we assume that transitions are conditionally independent across subproblems: $p(\mathbf{x}' \,|\, \mathbf{x}, \mathbf{a}) = \Pi_{i=1}^{N} p_i(x_i' \,|\, x_i, a_i)$, where we denote the transition probabilities for subproblem i by $p_i(x_i' \,|\, x_i, a_i)$. The exogenous state transitions according to $q(w' \,|\, w)$. Finally, let $r_i(s_i, a_i)$ be the reward of subproblem i and let $\mathbf{r}(\mathbf{s}, \mathbf{a}) = \{r_i(s_i, a_i)\}_{i=1}^{N}$. The reward of the overall system is additive: $r(\mathbf{s}, \mathbf{a}) = \sum_{i=1}^{N} r_i(s_i, a_i)$.

Given a discount factor $\gamma \in [0, 1)$ and a feasible policy $\pi : \mathcal{S} \to \mathcal{A}$ that maps each state $\mathbf{s}$ to a feasible action $\mathcal{A}(\mathbf{s})$, the value (cumulative discounted reward) of following π when starting in state $\mathbf{s}$ and taking a first action $\mathbf{a}$ is given by the action-value function $Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbf{E}\big[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \,|\, \pi, \mathbf{s}_0 = \mathbf{s}, \mathbf{a}_0 = \mathbf{a}\big]$. Our goal is to find an optimal policy π^*, i.e., one that maximizes $V^\pi(\mathbf{s}) := Q^\pi(\mathbf{s}, \pi(\mathbf{s}))$. We let $Q^*(\mathbf{s}, \mathbf{a}) = \max_\pi Q^\pi(\mathbf{s}, \mathbf{a})$ and $V^*(\mathbf{s}) = \max_\pi V^\pi(\mathbf{s})$ be the optimal action-value and value functions, respectively. It is well-known that the optimal policy selects actions in accordance to $\pi^*(\mathbf{s}) = \arg\max_{\mathbf{a}} Q^*(\mathbf{s}, \mathbf{a})$ and that the Bellman recursion holds:

$$Q^*(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \, \mathbf{E}\Big[\max_{\mathbf{a}' \in \mathcal{A}(\mathbf{s}')} Q^*(\mathbf{s}', \mathbf{a}')\Big], \tag{3.1}$$

where $\mathbf{s}' = (\mathbf{x}', w')$ is specified by $p(\cdot \,|\, \mathbf{x}, \mathbf{a})$ and $q(\cdot \,|\, w)$.

3.2.2 Q-learning and DQN

The Q-learning algorithm of Watkins [74] is a tabular approach that attempts to learn the optimal action-value function Q^* using stochastic approximation on (3.1). Using a learning rate α_n, the update of the approximation Q_n from iteration n to $n+1$ is:

$$Q_{n+1}(\mathbf{s}_n, \mathbf{a}_n) = Q_n(\mathbf{s}_n, \mathbf{a}_n) + \alpha_n(\mathbf{s}, \mathbf{a})\big[y_n - Q_n(\mathbf{s}_n, \mathbf{a}_n)\big],$$

where $y_n = r_n + \gamma \max_{\mathbf{a}'} Q_n(\mathbf{s}_{n+1}, \mathbf{a}')$ is the *target* value, computed using the observed reward r_n at $(\mathbf{s}_n, \mathbf{a}_n)$, the transition to $\mathbf{s}_{n+1}$, and the current Q_n.

The DQN approach of Mnih et al. [47] approximates Q^* via a neural network $Q(\mathbf{s}, \mathbf{a}; \theta)$ with network weights θ. The loss function used to learn θ is directly based on minimizing the discrepancy between the two sides of (3.1):

$$l(\theta) = \mathbf{E}_{\mathbf{s},\mathbf{a}\sim\rho}\Big[\big(y - Q(\mathbf{s}, \mathbf{a}; \theta)\big)^2\Big],$$

where $y = r(\mathbf{s}, \mathbf{a}) + \gamma\, \mathbf{E}\big[\max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}'; \theta^-)\big]$, θ^- are frozen network weights from a previous iteration, and ρ is a *behavioral distribution* [47]. In practice, we sample experience tuples $(\mathbf{s}_n, \mathbf{a}_n, r_n, \mathbf{s}_{n+1})$ from a replay buffer and perform a stochastic gradient update:

$$\theta_{n+1} = \theta_n + \alpha_n\big[y_n - Q(\mathbf{s}_n, \mathbf{a}_n; \theta)\big]\nabla_\theta Q(\mathbf{s}_n, \mathbf{a}_n; \theta),$$

with $y_n = r_n + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}_{n+1}, \mathbf{a}'; \theta^-)$. Note the resemblance of this update to that of Q-learning.

3.2.3 Lagrangian Relaxation

The Lagrangian relaxation approach decomposes weakly coupled DPs by relaxing the linking constraints to obtain separate, easier-to-solve subproblems [1]. The main idea is to dualize the linking constraints $\sum_{i=1}^{N} \mathbf{d}_i(s_i, a_i) \leq \mathbf{b}(w)$ using a penalty vector $\boldsymbol{\lambda} \in \mathbb{R}_+^d$ and optimize an augmented objective consisting of the original objective plus additional terms that penalize constraint violations. The resulting Bellman equation of the relaxed DP in (3.1) is given by:

$$Q^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \boldsymbol{\lambda}^T \Big[\mathbf{b}(w) - \sum_{i=1}^{N} \mathbf{d}_i(s_i, a_i)\Big] + \gamma\, \mathbf{E}\Big[\max_{\mathbf{a}' \in \mathcal{A}} Q^{\boldsymbol{\lambda}}(\mathbf{s}', \mathbf{a}')\Big]. \tag{3.2}$$

Next, we define the following recursion for each subproblem i, when it is considered independently:

$$Q_i^{\boldsymbol{\lambda}}(s_i, a_i) = r_i(s_i, a_i) - \boldsymbol{\lambda}^T \mathbf{d}_i(s_i, a_i) + \gamma\, \mathbf{E}\Big[\max_{a_i' \in \mathcal{A}_i} Q_i^{\boldsymbol{\lambda}}(s_i', a_i')\Big]. \tag{3.3}$$

It is well-known from classical results that any penalty vector $\boldsymbol{\lambda} \geq 0$ produces an MDP whose optimal value function is an upper bound on the $V^*(\mathbf{s})$ [29, 1]. The upcoming proposition is a small extension of these results to the case of action-value functions, which is necessarily for Q-learning.

Proposition 3.2.1. *For any $\boldsymbol{\lambda} \geq 0$ and $\mathbf{s} \in \mathcal{S}$, it holds that:*

(a) $Q^*(\mathbf{s}, \mathbf{a}) \leq Q^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a})$ *for any* $\mathbf{a} \in \mathcal{A}(\mathbf{s})$*;*

(b) The Lagrangian action-value function of (3.2) *satisfies*

$$Q^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a}) = \boldsymbol{\lambda}^T \mathbf{B}(w) + \sum_{i=1}^{N} Q_i^{\boldsymbol{\lambda}}(s_i, a_i) \tag{3.4}$$

where $Q_i^{\boldsymbol{\lambda}}(s_i, a_i)$ is as defined in (3.3) *and and $\mathbf{B}(w)$ satisfies the recursion*

$$\mathbf{B}(w) = \mathbf{b}(w) + \gamma\, \mathbf{E}\big[\mathbf{B}(w')\big], \tag{3.5}$$

where w' follows $q(w' \,|\, w)$.

Proof. See Appendix B.1. □

Part (a) is often referred to as *weak duality* and part (b) shows how the Lagrangian relaxation can be solved by decomposing it across subproblems, dramatically reducing the computational burden. The tightest upper bound[1] is the solution of the *Lagrangian dual problem*, $Q^{\boldsymbol{\lambda}^*}(\mathbf{s}, \mathbf{a}) = \min_{\boldsymbol{\lambda}} Q^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a})$, where $\boldsymbol{\lambda}^*$ is minimizer.

Note that solving the Lagrangian approximation still requires solving a dynamic program, so knowledge of the underlying model (i.e., environment dynamics and reward functions) is required. This is restrictive for applicability in many real-world problems. There exist *approximate dynamic programming* approaches for solving general weakly coupled MDPs, such as the ones discussed in [1], [48], and [16], but none of them naturally accomodate the model-free setting.

3.3 Weakly Coupled Q-learning

We first introduce the tabular version of our RL algorithm, called *weakly coupled Q-learning* (WCQL) and show that our algorithm converges to the optimal action value function. The algorithm is completely model-free and does not require any knowledge of the problem dynamics. In this section, we go introduce the main steps of WCQL, with an illustration summarizing the idea in Figure 8. The full pseudo-code of the WCQL algorithm is available in Appendix B.2.

To approximate the Lagrangian dual problem, we replace the minimization over all $\boldsymbol{\lambda} \geq 0$ by optimization over a finite set $\boldsymbol{\Lambda}$, which we consider as an input to our algorithm. Consider an experience tuple $\boldsymbol{\tau} = (\mathbf{s}, \mathbf{a}, \mathbf{d}, \mathbf{r}, \mathbf{b}, \mathbf{s}')$ and suppose we are at iteration $n + 1$. Also, we let $\tau_i = (s_i, a_i, \mathbf{d}_i, r_i, s_i')$ be the experience relevant to subproblem i; see (3.3). Note that $\mathbf{b}$ is excluded from τ_i because it does not enter subproblem Bellman recursion. The WCQL algorithm can be decomposed into three main steps.

Subproblems and $\boldsymbol{\lambda}$-agents. First, for each subproblem $i \in \{1, 2, \ldots, N\}$ and every $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$, we attempt to learn an approximation of $Q_i^{\boldsymbol{\lambda}}(s_i, a_i)$ from (3.3), which are the Q-values

[1]This optimal Lagrangian action-value function can be used to generate a heuristic policy called the *Lagrangian policy* that performs well in practice, but is not guaranteed to be optimal unless further assumptions are made [75].

of the unconstrained subproblem associated with $\boldsymbol{\lambda}$. We do this by running an instance of Q-learning with learning rate β_n. Letting $Q^{\boldsymbol{\lambda}}_{i,n}$ be the current iterate, the update is given by:

$$Q^{\boldsymbol{\lambda}}_{i,n+1}(s_i, a_i) = Q^{\boldsymbol{\lambda}}_{i,n}(s_i, a_i) + \beta_n(s_i, a_i)\big[y^{\boldsymbol{\lambda}}_{i,n} - Q^{\lambda}_{i,n}(s_i, a_i)\big],$$

where the target value $y^{\boldsymbol{\lambda}}_{i,n}$ is given by

$$y^{\boldsymbol{\lambda}}_{i,n} = r_i(s_i, a_i) - \boldsymbol{\lambda}^T \mathbf{d}_i(s_i, a_i) + \gamma \max_{a'_i} Q^{\lambda}_{i,n}(s'_i, a'_i).$$

Note that although we are running several learning "instances", they all make use of a common experience tuple $\boldsymbol{\tau}$. Each instance operates on a subproblem that is dramatically simpler than the full MDP (since subproblem i has state and action spaces $\mathcal{S}_i$ and $\mathcal{A}_i$). We can think of each subproblem i and each penalty $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ as being associated an agent, which we refer to generically as a $\boldsymbol{\lambda}$-*agent*, that tries to learn the value of $Q^{\boldsymbol{\lambda}}_i$.

Learning the Lagrangian bounds. Next, we combine the approximations of $Q^{\boldsymbol{\lambda}}_{i,n+1}$ to form an estimate of the Lagrangian action-value function $Q^{\boldsymbol{\lambda}}$, as defined in (3.4). However, note that we need to learn an estimate of the resource value $\mathbf{B}(w)$ for $w \in \mathcal{W}$. This can be done using a stochastic approximation step with a learning rate η_n, as follows:

$$\mathbf{B}_{n+1}(w) = \mathbf{B}_n(w) + \eta_n(w)\big[\mathbf{b}(w) + \gamma \mathbf{B}_n(w') - \mathbf{B}_n(w)\big], \tag{3.6}$$

where we recall that w and w' come from $\mathbf{s}$ and $\mathbf{s}'$. Now, using Proposition 3.2.1 (b), we approximate the value of Lagrangian action-value function $Q^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a})$ for each $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$,

$$Q^{\boldsymbol{\lambda}}_{n+1}(\mathbf{s}, \mathbf{a}) = \boldsymbol{\lambda}^T \mathbf{B}_{n+1}(w) + \sum_{i=1}^{N} Q^{\boldsymbol{\lambda}}_{i,n+1}(s_i, a_i). \tag{3.7}$$

Subsequently, we obtain an upper bound on Q^* via

$$Q^{\boldsymbol{\lambda}^*}_{n+1}(\mathbf{s}, \mathbf{a}) = \min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} Q^{\boldsymbol{\lambda}}_{n+1}(\mathbf{s}, \mathbf{a}). \tag{3.8}$$

There is empirical evidence that the Lagrangian policy $\pi^{\boldsymbol{\lambda}^*}$, i.e., policy greedy with respect to the Lagrangian action-value function $Q^{\boldsymbol{\lambda}^*}$, can attain good performance [16, 67]. Therefore, since we have access to the approximation $Q^{\boldsymbol{\lambda}^*}_{n+1}$, we can *approximately evaluate* the

Lagrangian policy and use it as an estimate of *lower bound* on Q^*. To do so, we use the following update:

$$L_{n+1}(\mathbf{s},\mathbf{a}) = L_n(\mathbf{s},\mathbf{a}) + \zeta_n(\mathbf{s},\mathbf{a})\big[r(\mathbf{s},\mathbf{a}) + \gamma L_n(\mathbf{s}', \mathbf{a}^{\boldsymbol{\lambda}^*}_{n+1}) - L_n(\mathbf{s},\mathbf{a})\big],$$

where ζ_n is a learning rate function and $\mathbf{a}^{\boldsymbol{\lambda}^*}_{n+1} = \arg\max_{\mathbf{a}} Q^{\boldsymbol{\lambda}^*}_{n+1}(\mathbf{s}', \mathbf{a})$.

Q-learning guided by Lagrangian bounds. Finally, we have Q-learning on the full problem. Denote the estimate of Q^* at iteration n by Q'_n. We first make a standard update to an intermediate value Q_{n+1} using learning rate α_n:

$$Q_{n+1}(\mathbf{s},\mathbf{a}) = Q'_n(\mathbf{s},\mathbf{a}) + \alpha_n(\mathbf{s},\mathbf{a})\big[y_n - Q'_n(\mathbf{s},\mathbf{a})\big]. \tag{3.9}$$

where $y_n = r(\mathbf{s},\mathbf{a}) + \gamma \max_{\mathbf{a}'} Q'_n(\mathbf{s}',\mathbf{a}')$. To incorporate the bounds that we previously estimated, we then project $Q_{n+1}(\mathbf{s},\mathbf{a})$ to satisfy the estimated upper and lower bounds:

$$Q'_{n+1}(\mathbf{s},\mathbf{a}) = \Pi_{[L_{n+1},\, Q^{\boldsymbol{\lambda}^*}_{n+1}]}[Q_{n+1}(\mathbf{s},\mathbf{a})], \tag{3.10}$$

where $\Pi_{[a,b]}[x] = \max\{\min\{x,b\},a\}$. The agent now takes an action in the environment using a behavioral policy, such as the ϵ-greedy policy on $Q'_{n+1}(\mathbf{s},\mathbf{a})$.

The motivation behind the projection is that since the subproblems are much smaller in terms of state and action spaces than the main problem, the $\boldsymbol{\lambda}$-agents for the subproblems are expected to converge faster than the full problem Q-learning agent. As a result, our lower bound and upper estimates will get better, improving thus the action-value estimate of the main Q-learning agent through the projection step. We note that, especially in settings where the limiting factor is the ability to collect enough experience, these bounds come *nearly for free* in that they only require computing a few additional quantities and importantly, they do not require additional experience. Figure 8 provides a visual summary of WCQL.

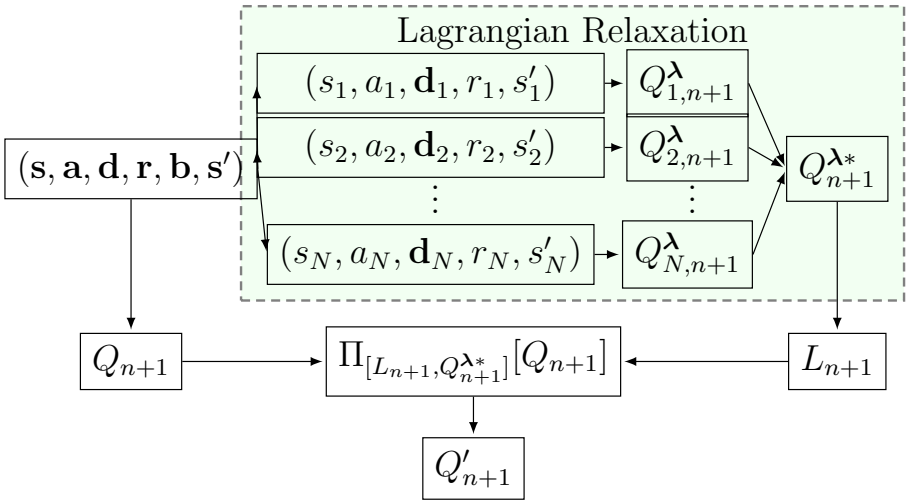


Figure 8: Illustration of WCQL Algorithm.

3.3.1 Convergence Analysis

In this section, we show that WCQL converges to Q^* with probability one ($w.p.1$). First, we state a standard assumption on learning rates and state visitation.

Assumption 3.3.1. We assume the following:

(i) For all $(\mathbf{s}, \mathbf{a}) \in \mathcal{S}$, the sequence $\{\alpha_n(\mathbf{s}, \mathbf{a})\}_n$ satisfies

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

(ii) Analogous statements to (i) hold for $\{\beta_n(s_i, a_i)\}_n$, $\{\eta_n(w)\}_n$, and $\{\zeta_n(\mathbf{s}, \mathbf{a})\}_n$.

(iii) The behavioral policy is such that all state-action pairs $(\mathbf{s}, \mathbf{a})$ are visited infinitely often $w.p.1$.

We now state our main theoretical results.

Theorem 3.3.1. *Under Assumption 3.3.1, the following hold w.p.1.*

(i) For each subproblem i, $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$, and $(\mathbf{s}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}$:

$$\lim_{n \to \infty} Q^{\boldsymbol{\lambda}}_{i,n}(s_i, a_i) = Q^{\boldsymbol{\lambda}}_i(s_i, a_i),$$

$$\lim_{n \to \infty} Q^{\boldsymbol{\lambda}}_n(\mathbf{s}, \mathbf{a}) \geq Q^*(\mathbf{s}, \mathbf{a}).$$

(ii) For all state-action pairs $(\mathbf{s}, \mathbf{a})$*,*

$$\lim_{n\to\infty} Q'_n(\mathbf{s}, \mathbf{a}) = Q^*(\mathbf{s}, \mathbf{a}).$$

Proof. The proof of Theorem 3.3.1 is given in Appendix B.1. □

Theorem 3.3.1 ensures that the $\boldsymbol{\lambda}$-agents for the subproblems converge to the optimal value for any $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$. Furthermore, it shows that asymptotically, the Lagrangian action-value function given by (3.7) will be an upper bound on the optimal action-value function Q^* of the full problem and that our algorithm will converge to Q^*.

3.4 Weakly Coupled DQN

In this section, we propose our main algorithm *weakly coupled DQN* (WCDQN), which integrates the main idea of WCQL into a function approximation setting. WCDQN guides DQN using Lagrangian relaxation bounds, integrated through a constrained optimization approach.

Networks. Analogous to the WCQL, WCDQN has a main network $Q'(\mathbf{s}, \mathbf{a}; \theta)$ that learns the action value of the full problem. In addition to the main network, WCDQN uses a $Q_i^{\boldsymbol{\lambda}}(s_i, a_i; \theta_U)$ network to learn the subproblem action-value functions $Q_i^{\boldsymbol{\lambda}}$. The inputs to this network are $(i, \boldsymbol{\lambda}, s_i, a_i)$, meaning that we can use a single network to learn the action-value function for *all* subproblems and $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ simultaneously. Lastly, WCDQN tracks the parameters of a network $L(\mathbf{s}, \mathbf{a}; \theta_L)$, which aims to learn the value of the Lagrangian policy. The Lagrangian state-action value function is then given by

$$Q^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a}; \theta_U) = \boldsymbol{\lambda}^T \mathbf{B}(w) + \sum_{i=1}^{N} Q_i^{\boldsymbol{\lambda}}(s_i, a_i; \theta_U). \tag{3.11}$$

Loss functions. Before diving into the training process, we describe the loss functions used to train each network, as they are instructive. Consider a behavioral distribution ρ for state-action pairs and a distribution μ over $\boldsymbol{\Lambda}$.

$$l_U(\theta_U) = \mathbf{E}_{\mathbf{s},\mathbf{a}\sim\rho,\boldsymbol{\lambda}\sim\mu}\Big[\sum_{i=1}^{N}\big(y_i^{\boldsymbol{\lambda}} - Q_i^{\boldsymbol{\lambda}}(s_i, a_i; \theta_U)\big)^2\Big], \tag{3.12}$$

where the target value is

$$y_i^{\boldsymbol{\lambda}} = r_i(s_i,a_i) - \boldsymbol{\lambda}^T \mathbf{d}_i(s_i,a_i) + \gamma\, \mathbf{E}\big[\max_{a_i'} Q_i^{\boldsymbol{\lambda}}(s_i',a_i';\theta_U^-)\big], \tag{3.13}$$

and θ_U^- are weights frozen from a previous iteration, as is done in DQN [47]. Denote by $(.)_+ = \max(.,0)$, the second loss function is for θ_L, where we use a soft constraint to encourage $L(\mathbf{s},\mathbf{a};\theta_L)$ to stay below the upper bound.

$$l_L(\theta_L) = \mathbf{E}_{\mathbf{s},\mathbf{a}\sim\rho,\boldsymbol{\lambda}\sim\mu}\Big[\big(y_L - L(\mathbf{s},\mathbf{a};\theta_L)\big)^2 + \kappa_U\big(L(\mathbf{s},\mathbf{a};\theta_L) - y_U\big)_+^2\Big] \tag{3.14}$$

where κ_U is a penalty coefficient for the soft constraint and

$$y_L = r(\mathbf{s},\mathbf{a}) + \gamma \mathbf{E}\big[L(\mathbf{s}',\mathbf{a}^{\boldsymbol{\lambda}^*};\theta_L^-)\big], \tag{3.15}$$

$$y_U = r(\mathbf{s},\mathbf{a}) + \gamma \mathbf{E}\big[\max_{\mathbf{a}'} \min_{\boldsymbol{\lambda}\in\boldsymbol{\Lambda}} Q^{\boldsymbol{\lambda}}(\mathbf{s}',\mathbf{a}';\theta_U^-)\big], \tag{3.16}$$

with $\mathbf{a}^{\boldsymbol{\lambda}^*} = \arg\max_{\mathbf{a}} \min_{\boldsymbol{\lambda}\in\boldsymbol{\Lambda}} Q_n^{\boldsymbol{\lambda}}(\mathbf{s}_{j+1},\mathbf{a};\theta_U)$. For the main network, we propose a loss function that soft penalizes both upper and lower bounds:

$$\begin{aligned} l(\theta) = \mathbf{E}_{\mathbf{s},\mathbf{a}\sim\rho,\boldsymbol{\lambda}\sim\mu}\Big[&\big(y - Q'(\mathbf{s},\mathbf{a};\theta)\big)^2 \\ &+ \kappa_L\big(y_L - Q'(\mathbf{s},\mathbf{a};\theta)\big)_+^2 \\ &+ \kappa_U\big(Q'(\mathbf{s},\mathbf{a};\theta) - y_U\big)_+^2\Big], \end{aligned} \tag{3.17}$$

where κ_L is a penalty coefficient similar to κ_U and

$$y = r(\mathbf{s},\mathbf{a}) + \gamma \mathbf{E}\big[\max_{\mathbf{a}} Q'(\mathbf{s}',\mathbf{a};\theta^-)\big]. \tag{3.18}$$

The penalties *softly* ensures that learned network satisfies the bounds obtained from the Lagrangian relaxation.

Training process. The training process resembles DQN, with a few modifications. At any iteration, we first take an action using an ϵ-greedy policy using the main network, store the obtained transition experience $\boldsymbol{\tau} = (\mathbf{s},\mathbf{a},\mathbf{d},\mathbf{r},\mathbf{b},\mathbf{s}')$ in the buffer, and update the resource value $\mathbf{B}(w)$ following (3.6). Each network is then updated by taking a stochastic gradient descent step on its associated loss function, where the expectations are approximated by sampling minibatches of experience tuples $\boldsymbol{\tau}$ and $\boldsymbol{\lambda}$. Note that, since the $\boldsymbol{\lambda}$s are decoupled from the transitions, we can increase the sample efficiency by using a similar scheme to

Hindsight Experience Replay [4]. Note that the penalty coefficients κ_L and κ_U can be held constant to a positive value or annealed using a schedule throughout the training. The full details are shown in Algorithm 2.

3.4.1 Lagrangian DQN

We also propose a variant of WCDQN, termed *Lagrangian DQN*, that does not require the bounding, making it easier to implement and achieves reasonable performance in practice. The details of Lagrangian DQN are shown in Algorithm 7 in Appendix B.3. This algorithm has only one network, the subproblem $Q_i^{\boldsymbol{\lambda}}$-network. Here, however, the first step is to compute the Lagrangian Q-value using (3.11), and compute $Q^{\boldsymbol{\lambda}^*}$ following (3.8).

3.5 Numerical Experiments

In this section, we evaluate our algorithms on three different weakly coupled MDPs. First, we evaluate WCQL on an electric vehicle (EV) deadline scheduling problem with multiple charging spots and compare its performance with several other tabular algorithms: Q-learning (QL), Double Q-learning (Double-QL), speedy Q-learning (SQL), and bias-corrected Q-learning (BCQL) [70, 5, 45]. Next, we evaluate Lagrangian-DQN and WCDQN on two problems, multi-product inventory control and online stochastic ad matching, and compare against the standard DQN and Double-DQN algorithms. We defer the details of the environments and the algorithmic parameters to Appendix B.4.

EV charging deadline scheduling [76]. In this problem, a decision maker (DM) is responsible for charging electric vehicles (EV) at a charging service center (SC) that consists of $N = 3$ charging spots. An EV enters the system when a charging spot is available and announces the amount of electricity it needs to be charged, denoted B_t, along with the time that it will leave the system, denoted D_t. The DM also faces exogenous, random Markovian processing costs c_t, that affect the SC's ability to process simultaneous charging jobs. At each period, the DM needs to choose which EVs to charge in accordance with the period's capacity

Algorithm 2: Weakly Coupled DQN

Input: Initialized replay buffer $\mathcal{D}$, Lagrangian multipliers set $\mathbf{\Lambda}$, Q-network weights θ & $\theta^- = \theta$, subproblems $Q_i^{\boldsymbol{\lambda}}$-network θ_U & $\theta_U^- = \theta_U$, Lagrangian policy L-network weights θ_L & $\theta_L^- = \theta_L$, and κ_L & κ_U

Output: Approximation $\{Q_n\}$

for $n = 0, 1, 2, \ldots$ **do**

Choose an action $a_n \sim \epsilon$-greedy$(Q_n))$, update $\mathbf{B}_{n+1}(w_n)$ and store transition $\boldsymbol{\tau}_n$ in $\mathcal{D}$

Sample a transitions minibatch $\boldsymbol{\tau}$ from $\mathcal{D}$ along with random $\boldsymbol{\lambda}$

Update subproblems network:

for i= 1, …, N **do**

Compute targets $y_i^{\boldsymbol{\lambda}}$ according (3.13)

Perform a gradient descent step on (3.12)

end for

Find the best upper bound:

For $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ and $\mathbf{a} \in \mathcal{A}(\mathbf{s}_n)$ find $Q_n^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a})$ per (3.11)

Set $Q_n^{\boldsymbol{\lambda}^*}(\mathbf{s}, \mathbf{a}) = \min_{\boldsymbol{\lambda} \in \mathbf{\Lambda}} Q_n^{\boldsymbol{\lambda}}(\mathbf{s}_n, \mathbf{a})$

Compute target y^U as per (3.16)

Update lower bound network:

Set $\mathbf{a}^{\boldsymbol{\lambda}^*} = \arg\max_{\mathbf{a}} Q_n^{\boldsymbol{\lambda}^*}(\mathbf{s}_{j+1}, \mathbf{a})$

Compute target y^L as per (3.15)

Perform a gradient descent step on (3.14)

Update main network:

Compute target y as per (3.18)

Perform a gradient step on (3.17)

end for

constraint. For each unit of power provided to an EV, the service center receives a reward $1 - c_t$. However, if the EV leaves the system with an unfulfilled charge of I units, the service center will incur a penalty of $F(I)$. The goal is to maximize the revenue under penalty costs.

Multi-product inventory control with an exogenous production rate [32]. Consider the problem of resource allocation for a facility that manufactures $K = 5$ products. Each product has an independent exogenous demand given by D_k, $k = 1, \ldots, K$. To meet these demands, the products are made to stock. Limited storage N_k is available for each product, and holding a unit of inventory per period incurs a cost h_k. Unmet demand are backordered at a cost b_k if the number of backorders is less than the maximum number of allowable backorders M_k. Otherwise, it is lost with a penalty cost l_k. The DM needs to allocate a resource level $a_k \in \{0, 1, \ldots, U\}$ for product k from a finite resource quantity in response to changes in the stock of level of each product denoted by $x_k \in X_k = \{-M_k, -M_k+1, \ldots, N_k\}$. A negative stock level corresponds to the number of backorders. Allocating a resource level a_k yields a production rate given by the function $\rho_k(a_k, p_k)$ where p_k is an exogenous random Markovian noise that affects the production rate.

The goal is to minimize the total cost, which consists of holding, back-ordering, and lost sales costs.

Online stochastic ad matching [27]. We study the problem of matching $N = 6$ advertisers (subproblems) to arriving impressions. In each period, an impression of type e_t arrives according to a Markov chain. An action $a_{t,i} \in \{0, 1\}$ assigns impression e_t to advertiser i, with a constraint that exactly one advertiser is selected: $\sum_{i=1}^{N} a_{t,i} = 1$. Advertiser states represent the number of remaining ads to display and evolves according to $s_{t+1,i} = s_{t,i} - a_{t,i}$. The objective is to maximize the discounted sum of expected rewards for all advertisers.

The results of our numerical experiments are shown in Figure 9. We see that in both the tabular and the function approximation cases, our algorithms outperformed the baselines, with WCQL and WCDQN achieving the best average episode total rewards amongst all problems. In Figure 9a, for a given state-action pair in the EV charging problem, we plot the evolution of the upper bound (blue) and lower bounds (red), WCQL Q-value (orange line with 'x' marks indicating the points projected by the bounds), standard Q-learning (green) and the optimal action-value (purple). Notice at the last marker, the bound moved the Q-

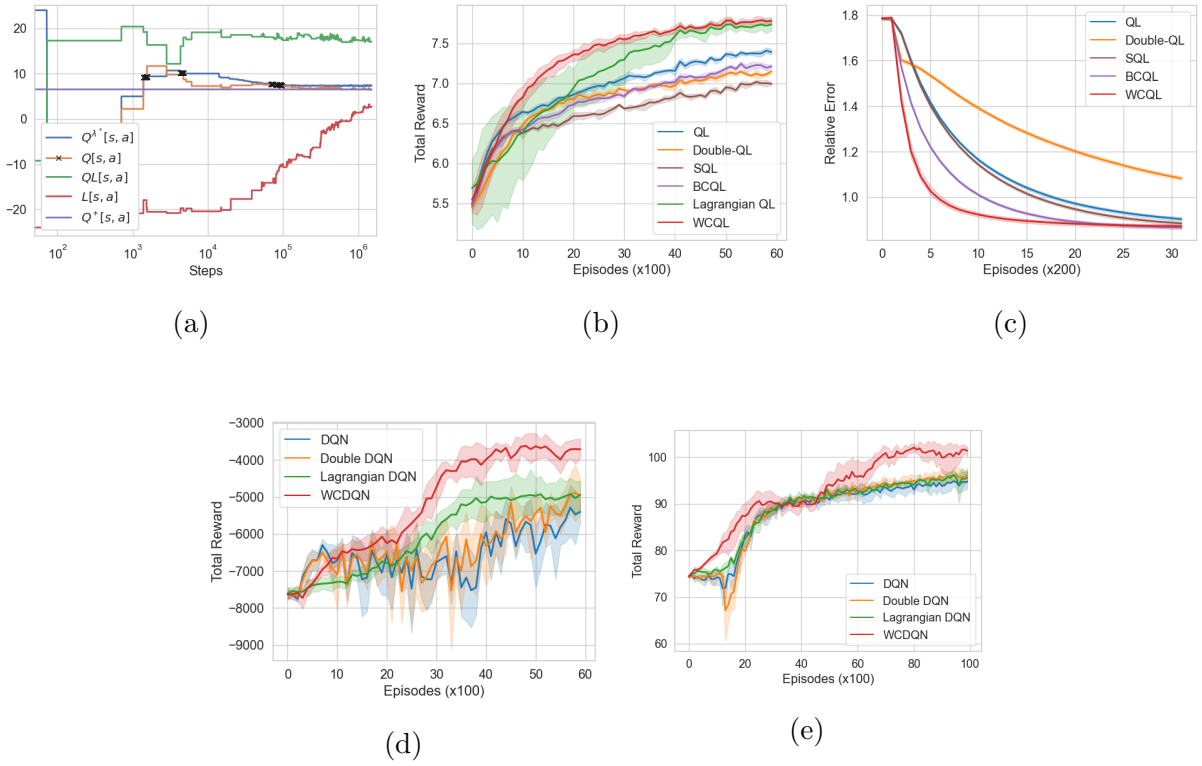


Figure 9: Numerical results: plots showing the bounds behaviour (a), the total rewards and 95% confidence bounds of WCQL and other tabular algorithms (b), and their relative error (c) on the EV charging problem. Plots (d) and (e) show the total rewards for WCDQN and other algorithms on the multi-product inventory control and online stochastic ad matching problems, respectively.

value in orange to a "good" value, relatively nearby the optimal value. WCQL then Q-value evolves on its own and eventually converges. On the other hand, standard QL (green), which follows the same behavior policy as WCQL, is still far from optimal. We also compute the relative error (measures the percent error, in l_2-norm, of the approximate value function with respect to the true optimal value function, i.e., $\|V_n - V^*\|_2/\|V^*\|_2$), see Figure 9c. This figure shows that WCQL has the steepest decline compared to the other algorithms. BCQL and SQL come second and third respectively after WCQL. Note that in Lagrangian QL the upper-bound value is estimated from the Q-values of the subproblems so there is no Q-value for this algorithm to compare to V^*.

From Figure 9b we see that the difference between the performance of Lagrangian QL and WCQL was similar by the end of training. This is not the case however in the general case, where we see a stark difference in the performance of WCDQN and Lagrangian DQN starting from 3000 episodes until the end of training. Moreover, we see that WCQL has a much lower variance compared to Lagrangian QL, which exhibits a high variance in the first 4000 steps of training. We attribute this to the guidance provided by the Lagrangian bounds.

Figure 9d shows a case where DQN and Double DQN initially outperform Lagrangian DQN and WCDQN, but then fall behind by a large margin. Also, we remark that compared to our algorithms, both DQN and Double DQN have a very noisy performance in the multi-product inventory problem. In the online stochastic ad matching problem, WCDQN have outperform the other algorithms while Lagrangian DQN had a comparable performance to DQN and Double-DQN algorithms.

3.6 Limitations and Future Work

A limitation of our work, which can be improved upon, is the requirement to input a finite set of Lagrangian multipliers $\mathbf{\Lambda}$. Note that although we get an upper bound for any value of $\boldsymbol{\lambda} \in \mathbf{\Lambda}$, we could potentially get tighter bounds by optimizing over $\boldsymbol{\lambda}$ directly, for example via subgradient descent [29]. While we believe this can bring an important improvement, we

leave it for future work.

3.7 Conclusion

In this study, we propose WCQL algorithm for learning in weakly coupled problems and we show that our algorithm converges to the optimal action-value function. We then propose WCDQN which extends the idea behind the WCQL algorithm to the function approximation case. Our algorithms are model-free and learn upper and lower bounds using a combination of a Lagrangian relaxations and Q-learning. These bounds are used in a constrained optimization approach to improve performance and make learning more efficient. Our approaches significantly outperformed competing approaches on our benchmark environments.

4.0 Spatial Dynamic Pricing for Shared-Resource Systems

The recent years have seen an increase in the popularity of shared transport platforms. This led to an increase in the study of complex control systems where resources circulate inside a network. In this work, we study dynamic pricing for a system where a fixed number of resources supply incoming price-sensitive demand at different locations in a network.

The crux of such systems is that serving a demand unit at one location causes a supply unit to relocate from the origin to the destination. Dynamic pricing offers a natural approach to modulate demand and help managing the the distribution of resources in the network. At the beginning of each period, the service provider selects a price as well as a supply level for each origin-destination pair in the network. Demand for each origin-destination pair is fulfilled in accordance to the selected supply levels. Otherwise, unfulfilled demands are lost. Both price and supply level decisions, may depend on the overall distribution of resources in the system. Supplied requests reach their destinations by the end of the time period. Note that demands can be non-stationary and depends on the geographical location.

The research question we are trying to answer is how to design a dynamic pricing and allocation strategy that maximizes the expected revenue of the system. This problem is challenging due to demand variability over space and time and in large networks is very difficult to solve because the pricing policy depends on the number of resources in each location.

To address this research question, we model the system as a network and formulate the problem as a stochastic dynamic program (SDP). The demand modeling approach we use allows for bulk demand arrivals with different origins and destinations. Each origin and destination pair represents a different demand *class*. The limited number of resources in the system necessitates using inventory rationing decisions as a control lever in addition to pricing in order to allocate demand classes to resources. This work contributes to the growing body of literature on sharing economy with reusable resources by incorporating inventory rationing decisions in addition to dynamic pricing. The pricing and rationing decisions depend not only on the origin but also on the destination of the trips, permitting

price adjustment flexibility and capturing the spatial granularity of the network.

Theoretically, we analyze the structure of the optimal revenue function in a N-location network and show that it is concave in the number of resources available at each location. In addition, we use the concept of $L^\natural$-concavity to perform structural analysis for the two-region system, which allows for both continuous and discrete decision variables and thus provides a unified approach for continuous and discrete demand. We prove that the optimal policy for pricing and rationing is monotone in the available resources with bounded sensitivity.

For a large number of locations, the stochastic dynamic program is intractable due the "curse of dimensionality" associated with the problem. To overcome this issue, we propose a *leave-one-out aggregation* (LOOA) algorithm that makes use of the structural properties of the optimal policy for the two-region problem. Finally, we conduct numerical experiments to evaluate the performance of our proposed policy. We also compare the performance of our pricing policy with myopic and static based pricing policies.

4.1 Literature Review

Academic research on sharing economy has increased in the recent years due to the growing popularity of shared vehicle systems such as ride-sharing platforms, car-sharing, and bike-sharing platforms. The majority of the literature study repositioning, matching and pricing as control levers to manage the resources that flow over a network of locations.

A body of literature use closed-queueing networks and model the steady state equilibrium of the system dynamics to study the impact of state-independent pricing on the flow of resources in ride sharing systems while optimizing for various objectives such as revenue, throughput, or welfare [73, 7, 50, 11, 12]. Some of these papers use fluid relaxations to get upper bounds on the optimal pricing policy performance which are then used to propose static pricing policies [73, 7].

Papers that study state-dependent pricing in a network include [6], [35], and [18]. Balseiro et al. [6] study a constrained dynamic programming model for dynamic pricing in a star network (also known as hub-and-spoke network) with a fixed number of resources and infinite

horizon. The authors use a Lagrangian relaxation approach and show that their proposed heuristic is asymptotically optimal in terms of the long-run average cost when the number of demands nodes in the network is large. Travel time between nodes is ignored in their theoretical results but later develop a tractable approach to incorporate relocation times into their policies.

Kanoria and Qian [35] consider joint state-dependent control for ride-hailing platform which besides pricing includes admission and matching decisions. Their algorithm does not require knowledge of the demand arrival rates information with transient performance guarantees and asymptotically optimal as well in the large supply regime. Instantaneous relocation times is assumed.

Chen et al. [18] focus on real-time spatial-intertemporal dynamic pricing for ride-hailing platforms with stochastic non-stationary demand over a finite horizon assuming resources relocation travel times are deterministic. The authors propose static and dynamic (node-based and arc-based) pricing heuristics which are all asymptotically optimal but with different optimality gaps in the setting where the demand and supply is large.

All of these papers make the assumption that the time periods are short enough such that for each period there is at most one potential customer, which contradicts the instantaneous arrival assumption. While we assume that trips arrive to their destinations at the end of the period, our demand model does not require short time periods. On the contrary, our bulk demand model allows us to assume that the time periods are large enough such that all relocations are made before the end of the period. The modeling of the events is also different. While the majority of these works assume that customers arrive according to a specific arrival process, e.g., Poisson process, with customers having a private value according to which the demand is either fulfilled or not, we use a more direct approach that links the demand to the price and does not require us to explicitly make assumptions on the arrival process. We also introduce inventory rationing decisions in our model in addition to pricing. Our theoretical analysis is different in that it tackles the properties of the main stochastic dynamic program formulation of the problem rather than alternative relaxations. Our proposed algorithms are novel and makes use of the SDP properties.

4.2 Preliminaries

Let $\mathbb{R}$ denote the real numbers and $\mathbb{R}_+$ the nonnegative reals, $\mathbb{Z}_+$ the set of nonnegative integers and $\mathbb{Z}_{++}$ the set of strictly positive integers. We also define $\mathscr{R} = \mathbb{R} \cap \{\infty\}$. For any $x, y \in \mathbb{R}^n$, let $x \wedge y = \min\{x, y\}$, $x \vee y = \max\{x, y\}$ and $x^+ = \max\{x, 0\}$, where all operations are taken component-wise. In addition, for any $n \in \mathbb{Z}_{++}$, we let $[n] = \{1, \ldots, n\}$. Finally, denote by $\mathbf{e} \in \mathbb{R}^n$ a vector whose all components are all ones and by $\mathbf{e_i} \in \mathbb{R}^n$ a vector whose i-th component is one and all the other elements being zero. A set S of $\mathbb{R}^n$ is called a convex set if for any $x, y \in S, 0 \leq \alpha \leq 1$, we have $\alpha x + (1-\alpha)y \in S$. Moreover, S is called a sublattice of $\mathbb{R}^n$ if $x \wedge y, x \vee y \in S$ for all $x, y \in S$. A function f defined on a convex set S of $\mathbb{R}^n$ is said to be convex if $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$ for all $0 \leq \alpha \leq 1$, and $x, y \in S$ and f is supermodular if S is a sublattice and $f(x) + f(y) \leq f(x \wedge y) + f(x \vee y)$ for all $x, y \in S$. A function f is concave (submodular) if $-f$ is convex (supermodular). We now introduce the concept of $L^\natural$-convexity which can be defined for both integer and continuous variables. We use $\mathscr{F}$ to denote either the real numbers $\mathbb{R}$ or the set of nonnegative integers $\mathbb{Z}_+$ and the notation $\mathscr{F}_+$ to denote the nonnegative elements in $\mathscr{F}$.

Definition 4.2.1 ($L^\natural$-convexity)**.** A function $f : \mathscr{F}^n \to \mathscr{R}$ is $L^\natural$- convex if for any $\mathbf{x}, \mathbf{y} \in \mathscr{F}$, $\alpha \in \mathscr{F}_+$

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f((\mathbf{x} + \alpha\mathbf{e}) \wedge \mathbf{y}) + f(\mathbf{x} \vee (\mathbf{y} - \alpha\mathbf{e})).$$

Note that if $f(\mathbf{x}) = +\infty$ or $f(\mathbf{y}) = +\infty$ then the inequality is assume to hold automatically. If a function f is $L^\natural$-convex then $-f$ is $L^\natural$-concave. A set $\mathscr{V}$ is $L^\natural$-convex if for all $\mathbf{x}, \mathbf{y} \in \mathscr{V}$ and for all $\alpha \in \mathscr{F}_+$, $(\mathbf{x} + \alpha\mathbf{e}) \wedge \mathbf{y} \in \mathscr{V}$ and $\mathbf{x} \vee (\mathbf{y} - \alpha\mathbf{e}) \in \mathscr{V}$. The effective domain of an $L^\natural$-convex function f, denoted by $dom(f) = \{\mathbf{x} \in \mathscr{F}^n | f(\mathbf{x}) < +\infty\}$ is an $L^\natural$-convex set.

A function f is said to be $L^\natural$-convex on a set $\mathscr{V} \subseteq \mathscr{F}^n$ if $\mathscr{V}$ is a $L^\natural$-convex set and the extension of f to the whole space $\mathscr{F}^n$ by defining $f(\mathbf{x}) = +\infty$ for $x \notin \mathscr{V}$ is $L^\natural$-convex. One can also show that an $L^\natural$-convex function restricted to an $L^\natural$-convex set is also $L^\natural$-convex.

Equivalently, a function $f : \mathscr{F}^n \to \mathscr{R}$ is $L^\natural$-convex if and only if $g(\mathbf{x}, \varepsilon) := f(\mathbf{x} - \varepsilon\mathbf{e})$ is submodular on $(x, \varepsilon) \in \mathscr{F}^n \times \mathscr{S}$, where $\mathscr{S}$ is the intersection of $\mathscr{F}$ and any unbounded

interval in $\mathbb{R}$. In addition, the Hessian of an $L^\natural$-convex function is diagonally dominant.

4.3 Problem Formulation

Consider a firm that is responsible for pricing and inventory rationing decisions of a fixed number of resources m, that circulate in a network over either a finite planning horizon consisting of T decision periods or an infinite horizon with discount factor γ. The time periods are denoted by $t \in \mathbb{N}$, with $t = 1$ being the first period in the planning horizon. The network is represented by a directed graph consisting of N nodes indexed by $i = 1, \ldots, N$ that represent the location of the resources. A trip from location i to location j is represented by an arc $i \to j$. Demand requests can be return or one-way trips, where the origin and destination are the same for return trips and different for one-way. We assume that the price for return trips, at each location, are fixed to a nominal price $p_{ii,t} \in \mathcal{P}_{ii,t}$ and we are interested in setting the price for one-way trips $p_{ij,t} \in \mathcal{P}_{ij,t}$ to maximize the expected revenue and ensure that the resources are well distributed over the network. At the beginning of each period, the firm needs to set the price $p_{ij,t} \in \mathcal{P}_{ij,t}$ for the one-way demand from region i to j and the quantity $\theta_{ij,t} \in \Theta_{ij,t}$ of this demand to accept. Demands are nonnegative, independent and depends on the price according to a stochastic non-stationary demand function

$$D_{ij,t}(p_{ij,t}, \epsilon_{ij,t}) := \kappa_{ij,t}(p_{ij,t}) + \epsilon_{ij,t},$$

where $D_{ij,t}(p_{ij,t}, \epsilon_{ij,t})$ is the demand in period t, $\epsilon_{ij,t}$ are random perturbations with zero mean that are revealed at the beginning of period $t+1$ and $\kappa_{ij,t}(p_{ij,t})$ is a deterministic demand function of the price $p_{ij,t}$. The random variables $\epsilon_{ij,t}$ are independent, identically distributed over time, with a bounded support $\underline{\epsilon} \leq 0 \leq \bar{\epsilon}$ such that $D_{ij,t}(\bar{p}_{ij,t}, \underline{\epsilon}_{ij,t}) \geq 0$ for all $i, j \in [N]$. We denote by $F_{ij}(.)$ the distribution function of $\epsilon_{ij,t}$ and we let $\bar{F}_{ij}(.) = 1 - F_{ij}(.)$.

Furthermore, we assume that the expected demand $\mathbf{E}[D_{ij,t}(p_{ij,t}, \epsilon_{ij,t})] = \kappa_{ij,t}(p_{ij,t}) < \infty$ is strictly decreasing in the price $p_{ij,t}$ which is restricted to $\mathcal{P}_{ij,t} := [\underline{p}_{ij,t}, \overline{p}_{ij,t}]$, where $\underline{p}_{ij,t}$ and $\overline{p}_{ij,t}$ are the minimum and the maximum prices, respectively. This assumption implies a one-to-one correspondence between the price $p_{ij,t}$ and the expected demand $d_{ij,t} \in \mathcal{D}_{ij,t} := [\underline{d}_{ij,t}, \overline{d}_{ij,t}]$

for all $p_{ij,t} \in \mathcal{P}_{ij,t}$ where $\underline{d}_{ij,t} = \kappa_{ij,t}(\overline{p}_{ij,t})$ and $\overline{d}_{ij,t} = \kappa_{ij,t}(\underline{p}_{ij,t})$. Denote by $P_{ij,t}(.)$ the inverse of the expected demand function, $\kappa_{ij,t}(.)$. For convenience, we equivalently use the expected demands $d_{ij,t}$ instead of the prices $p_{ij,t}$ as the decision variables in our analysis and charge the price $P_{ij,t}(d_{ij,t})$. For a given expected demand level $d_{ij,t}$ and inventory rationing quantity $\theta_{ij,t}$ between locations i, j, we let $R(d_{ij,t}, \theta_{ij,t}) = P_{ij,t}(d_{ij,t})\mathbf{E}[\min(d_{ij,t} + \epsilon_{ij,t}, \theta_{ij,t})]$ be the one-period expected revenue function. We make the following assumption on the revenue function.

Assumption 4.3.1. $R(d_{ij}, \theta_{ij})$ is continuous and $L^\natural$-concave in $(d_{ij}, \theta_{ij}) \in \mathcal{D}_{ij} \times \Theta_{ij}$.

Assumption 4.3.1, implies that the marginal value of the expected revenue is decreasing in both the demand level and the rationing quantity. Moreover, the demand and rationing quantity are complementary to each other in the sense that increasing the rationing quantity will increase the marginal revenue of increasing the demand level [21]. Next, we state the conditions used by [51] that ensure that Assumption 4.3.1 is satisfied. Specifically, these conditions are,

(C1) $P''_{ij}(d_{ij})d_{ij} + P'_{ij}(d_{ij}) \leq 0$ for all $d_{ij} \in \mathcal{D}_{ij}$ and

(C2) $\varrho_t(d_{ij}, \theta_{ij}) \geq 1$ for all $d_{ij} \in \mathcal{D}_{ij}$ and $\theta_{ij} \geq 0$,

where

$$\varrho(d_{ij}, \theta_{ij}) = d_{ij}\frac{-P_{ij}(d_{ij})}{d_{ij}P'_{ij}(d_{ij})}\frac{F'(\theta_{ij} - d_{ij})}{\bar{F}(\theta_{ij} - d_{ij})},$$

is the lost-sales rate elasticity that measures the relative sensitivity of lost sales probability. i.e., $P(d_{ij} + \epsilon_{ij} > \theta_{ij}) = \bar{F}(\theta_{ij} - d_{ij})$ with respect to rationing quantity and the price. Condition (C1) is satisfied by many common demand models (see Table 1 in [21]). Condition (C2) was first introduced by [40] and it requires the price elasticity of the demand $-P_{ij}(d_{ij})/d_{ij}P'_{ij}(d_{ij})$ and the random shock hazard rate $F'(\theta_{ij} - d_{ij})/\bar{F}(\theta_{ij} - d_{ij})$ be sufficiently large for any $d_{ij} \in \mathcal{D}_{ij}$. These conditions insure that the expected revenue function is concave and supermodular in (d_{ij}, θ_{ij}).

Proposition 4.3.1 (Proposition 3 in [21])**.** *Suppose (C1) and (C2) hold, then $R(d_{ij}, \theta_{ij})$ is $L^\natural$-concave in (d_{ij}, θ_{ij}).*

The problem can be formulated as a Markov decision process (MDP) with state $\mathbf{x} = (x_i)_{i\in[N]}$, which is a vector whose components represent the number of available resources at

each location. The state space is $\mathcal{X} = \Big\{\mathbf{x} = (x_i)_{i\in[N]} \in \mathbb{R}^N : \sum_{i\in[N]} x_i = m\Big\}$ where m is the maximum number of resources in the network. We penalize unmet demands by a lost sales unit cost $\rho_{ij,t}$ which is always larger than the price of trips. The decision vector is $\mathbf{y}_t \in \mathcal{Y}_t(\mathbf{x}_t)$, where $\mathcal{Y}_t(\mathbf{x}_t) = \Big\{\mathbf{d}_t = (d_{ij,t} \in \mathcal{D}_{ij,t})_{ij\in[N]},\, \boldsymbol{\theta}_t = (\theta_{ij,t})_{ij\in[N]} : \sum_{j\in[N]} \theta_{ij,t} = x_{i,t},\, \forall\, i \in [N]\Big\}$. Let $V_t^*(\mathbf{x}_t)$ be the revenue-to-go function with number of available resources $\mathbf{x}_t$ and let $V_{T+1}(\mathbf{x}_{T+1}) = 0$. For $t = 1, \ldots, T$, we have the Bellman recursion:

$$
\begin{aligned}
V_t(\mathbf{x}_t) &= \max_{\mathbf{y}_t \in \mathcal{Y}_t(\mathbf{x}_t)} \left\{ \sum_{ij\in[N]} R(d_{ij,t}, \theta_{ij,t}) + \mathbf{E}\left[J_t(\mathbf{x}_t, \mathbf{d}_t, \boldsymbol{\theta}_t, \boldsymbol{\epsilon}_t)\right] \right\} \\
J_t(\mathbf{x}_t, \mathbf{d}_t, \boldsymbol{\theta}_t, \boldsymbol{\epsilon}_t) &= V_{t+1}(\mathbf{x}_{t+1}) - \sum_{ij\in[N]} \rho_{ij,t}\Big(d_{ij,t} + \epsilon_{ij,t} - w_{ij,t}\Big) \\
\text{s.t.} \quad & w_{ij,t} = \min\left(d_{ij,t} + \epsilon_{ij,t}, \theta_{ij,t}\right) \quad \forall\, i, j \in [N], \\
& x_{i,t+1} = x_{i,t} - \sum_{j\in[N]} w_{ij,t} + \sum_{k\in[N]} w_{ki,t}, \quad \forall\, i \in [N].
\end{aligned}
\tag{4.1}
$$

The quantity $w_{ij,t}$ represents the total fulfilled requests from region i to region j at time t. The following proposition characterizes the structure of the optimal value function. The proofs of all results are included in Appendix C.1.

Proposition 4.3.2 (Concavity of The Value Function)**.** *For $t = 1, \ldots, T+1$,*

$$
\begin{aligned}
J_t(\mathbf{x}_t, \mathbf{d}_t, \boldsymbol{\theta}_t, \boldsymbol{\epsilon}_t) &= \max_{\mathbf{w}_t} \left\{ V_{t+1}(\mathbf{x}_{t+1}) - \sum_{ij\in[N]} \rho_{ij,t}\Big(d_{ij,t} + \epsilon_{ij,t} - w_{ij,t}\Big) \right\} \\
s.t. \quad & w_{ij,t} \le \min\left(d_{ij,t} + \epsilon_{ij,t}, \theta_{ij,t}\right) \quad \forall\, i, j \in [N], \\
& x_{i,t+1} = x_{i,t} - \sum_{j\in[N]} w_{ij,t} + \sum_{k\in[N]} w_{ki,t}, \quad \forall\, i \in [N].
\end{aligned}
\tag{4.2}
$$

and $V_t(\mathbf{x}_t)$ is concave in $\mathbf{x}_t$.

Due to the high dimensionality involved in the state, action, and noise spaces, solving (4.1) is computationally challenging even though the value function is concave in the state. In the next section, we study the case where there is only two locations and we analyze the structure of the optimal value function and optimal policy.

4.4 Dynamic Pricing and Rationing in Two Locations

In this section, we consider a two-location system consisting of regions 1 and 2. We denote the state of the system at period t by x_t which is equal to the number of resources at location 1, i.e., $x_t = x_{1,t}$. Since the number of resources m is fixed, we have $x_{2,t} = m - x_t$. The number of resources available for return trips are then given by $\theta_{11,t} = x_t - \theta_{12,t}$, and $\theta_{22,t} = m - x_t - \theta_{21,t}$ at regions 1 and 2, respectively. The decision vector is given by $\mathbf{y}_t = (d_{12,t}, d_{21,t}, \theta_{12,t}, \theta_{21,t}) \in \mathcal{Y}(x_t)$ where the feasible set $\mathcal{Y}(x_t)$ is given by:

$$\mathcal{Y}(x_t) = \big\{\mathbf{y} : d_{12,t} \in [\underline{d}_{12,t}, \overline{d}_{12,t}], d_{21,t} \in [\underline{d}_{21,t}, \overline{d}_{21,t}], \theta_{12,t} \leq x_t, \theta_{21,t} \leq m - x_t\big\}.$$

The next result characterizes the structure of the value function and the optimal policy.

Theorem 4.4.1. *For the two-locations pricing and rationing problem, we have*

$$\begin{aligned} V_t(x_t) = \max_{\mathbf{y}_t \in \mathcal{Y}(x_t)} \Big\{ & \sum_{ij \in \{1,2\}} P_{ij,t}(d_{ij,t}) \mathbf{E}[\min(d_{ij,t} + \epsilon_{ij,t}, \theta_{ij,t})] + \mathbf{E}[J_t(x_t, \mathbf{d_t}, \boldsymbol{\theta_t}, \boldsymbol{\epsilon_t})] \Big\} \\ J_t(x_t, \mathbf{d_t}, \boldsymbol{\theta_t}, \boldsymbol{\epsilon_t}) = & \max_{\mathbf{w_t}} \Big\{ V_{t+1}(x_{t+1}) - \sum_{ij \in \{1,2\}} \rho_{ij,t}(d_{ij,t} + \epsilon_{ij,t} - w_{ij,t}) \Big\} \\ x_{t+1} = & \, x_t + w_{21,t} - w_{12,t} \\ w_{ij,t} \leq & \min\left(d_{ij,t} + \epsilon_{ij,t}, \theta_{ij,t}\right) \quad \forall \, ij \in \{1,2\} \end{aligned} \tag{4.3}$$

and for $t = 1, \ldots, T$, the functions $V_t(x_t)$ and $J_t(x_t, \mathbf{d_t}, \boldsymbol{\theta_t}, \boldsymbol{\epsilon_t})$ are $L^\natural$-concave in x_t and $(x_t, \mathbf{d_t}, \boldsymbol{\theta_t})$ respectively. Let $(d^*_{12,t}(x_t), d^*_{21,t}(x_t), \theta^*_{12,t}(x_t), \theta^*_{21,t}(x_t))$ be the optimal policy at period t as a function of the state x_t. Then $d^*_{12,t}(x_t), \theta^*_{12,t}(x_t)$ and $d^*_{21,t}(x_t), \theta^*_{21,t}(x_t)$ are non-decreasing, non-increasing respectively in x_t, i.e., the optimal prices $p^*_{12,t}(x_t)$ and $p^*_{21,t}(x_t)$ are non-increasing, non-decreasing respectively in x_t and for any $\delta > 0$, $d^*_{12,t}(x+\delta) \leq d^*_{12,t}(x) + \delta$, $d^*_{21,t}(x+\delta) \geq d^*_{21,t}(x) - \delta$, $\theta^*_{12,t}(x+\delta) \leq \theta^*_{12,t}(x) + \delta$, $\theta^*_{21,t}(x+\delta) \geq \theta^*_{21,t}(x) - \delta$.

This result means that to find the optimal policy via dynamic programming, we do not need to exhaustively evaluate all the actions for a given state. Instead, one can use the optimal action found for state x_t to derive the optimal action at state $x_t + 1$. This limits the number of actions $\mathbf{y}_t$ that we need to evaluate to 8 actions at most.

4.5 The Infinite Horizon Setting

In this section, we show that the concavity of the value function continue to hold for the stationary problem with infinite periods. It is well known that a stationary policy π that uses the same decision rule in each period is optimal for this setting. We denote by $\gamma \in (0,1)$ the discount factor and denote the values of $\rho_{ij,t}$ and $\kappa_{ij,t}$ by ρ_{ij} and κ_{ij}, respectively. We first write the Bellman equation of the optimization problem in the infinite horizon setting as follows:

$$\begin{aligned} V(\mathbf{x}) &= \max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \left\{ \sum_{ij\in[N]} R(d_{ij},\theta_{ij}) + \mathbf{E}\left[J(\mathbf{x},\mathbf{d},\boldsymbol{\theta},\boldsymbol{\epsilon})\right] \right\} \\ J_t(\mathbf{x},\mathbf{d},\boldsymbol{\theta},\boldsymbol{\epsilon}) &= V(f(\mathbf{x},\mathbf{y},\boldsymbol{\epsilon})) - \sum_{ij\in[N]} \rho_{ij}\Big(d_{ij}+\epsilon_{ij}-w_{ij}\Big) \\ \text{s.t.} \quad & \mathbf{w} = \min\left(\mathbf{d}+\boldsymbol{\epsilon},\boldsymbol{\theta}\right) \\ & f(\mathbf{x},\mathbf{y},\boldsymbol{\epsilon}) = \mathbf{x} - \mathbf{we} + \mathbf{w}^T\mathbf{e}. \end{aligned} \tag{4.4}$$

Theorem 4.5.1 (Infinite Horizon: Preservation of the Structural Results)**.** *If Assumption 4.3.1 holds then the results in Proposition 4.3.2 and Theorem 4.4.1 continue to hold for infinite horizon setting.*

Proof. We follow the general approach outlined in [52]. The main idea is based on iterating the structural properties of the one-stage problem. We show that the results for Proposition 4.3.2. The results of Theorem 4.4.1 for the two-location problem follows in a similar way. Denote by $\mathcal{V}^*$ the space of continuous, concave and bounded functions over $\mathcal{X}$. We have that the revenue function is concave (by Proposition 4.3.1) and the lost sales cost is linear so the one-step profit is concave. Moreover, by the induction argument in the proof of Proposition 4.3.2, we have that the one-step structure preservation property holds, that is the next period value function is in $\mathcal{V}^*$, which means that the optimal value function of the next period is also in $\mathcal{V}^*$ [52]. This property and the fact that the set $\mathcal{V}^*$ with the sup-norm is a metric space allows us to apply Corollary 1 of [52] which allows us to conclude that $V \in \mathcal{V}^*$ (other conditions are easy to verify). □

4.6 Leave-One-Out Aggregation Heuristic

Although we characterize the concave structure of the value function of the N locations problem in Proposition 4.3.2, we still cannot directly use convex optimization to solve for the value function or to obtain good policies because of the difficulty involved in the high dimensional state and action spaces. In this section, we propose our heuristic approach that simplifies the N-location problem into a two-location problem using a leave-one-out aggregation (LOOA) strategy. The main idea involves approximating the N-location problem by N two-location problems. The leave-one-out aggregation at location $i \in [N]$, involves approximating the remaining $N-1$ locations into a single location denoted by ϕ_i such that in the new two-location problem the price $p_{ij} = p_{i\phi_i}$ for all $j \neq i$. The demand functions in the two-location problem are given by $D_{k\ell}(p_{k\ell}, \epsilon_{k\ell}) = \kappa_{k\ell}(p_{k\ell}) + \epsilon_{k\ell}$ for $k, \ell \in \{i, \phi_i\}$, where

$$\kappa_{k\ell}(p_{k\ell}) = \begin{cases} \kappa_{ii}(p_{ii}) \text{ for } k = i, \ell = i, p_{ii} \in [\underline{p}_{ii}, \overline{p}_{ii}]; \\ \sum_{j=1:j\neq i}^{N} \kappa_{ij}(p_{i\phi_i}) \text{ for } k = i, \ell = \phi_i, p_{i\phi_i} \in [\underline{p}_{i\phi_i}, \overline{p}_{i\phi_i}]; \\ \sum_{j=1:j\neq i}^{N} \kappa_{ji}(p_{\phi_i,i}) \text{ for } k = \phi_i, \ell = i, p_{\phi_i i} \in [\underline{p}_{\phi_i i}, \overline{p}_{\phi_i i}]; \\ \sum_{r,j:r\neq i,j\neq i} \kappa_{rj}(p_{\phi_i\phi_i}) \text{ for } k = \phi_i, \ell = \phi_i, p_{\phi_i\phi_i} \in [\underline{p}_{\phi_i\phi_i}, \overline{p}_{\phi_i\phi_i}]. \end{cases} \tag{4.5}$$

Denote by $P_{k\ell}(.)$ the inverse function of $\kappa_{k\ell}(.)$, we then have $\underline{p}_{k\ell} = P_{k\ell}(\overline{d}_{k\ell})$ and $\overline{p}_{k\ell} = P_{k\ell}(\underline{d}_{k\ell})$ where $\underline{d}_{i\phi_i} = \sum_{j\neq i} \underline{d}_{ij}$, $\overline{d}_{\phi_i i} = \sum_{j\neq i} \overline{d}_{ji}$, $\underline{d}_{\phi_i i} = \sum_{j\neq i} \underline{d}_{ji}$, $\overline{d}_{\phi_i i} = \sum_{j\neq i} \overline{d}_{ji}$, and $\underline{d}_{\phi_i\phi_i} = \sum_{r,j:r\neq i,j\neq i} \underline{d}_{rj}$, $\overline{d}_{\phi_i\phi_i} = \sum_{r,j:r\neq i,j\neq i} \overline{d}_{rj}$.

Let d^{LOOA} and θ^{LOOA} be the LOOA policies for the demand and resource allocation, respectively, for the full problem. We solve the resulting two-location DP problems, $\{i, \phi_i\}$ for $i \in [N]$, by dynamic programming (4.3) using the monotonicity and bounded sensitivity results (Theorem 4.4.1).

The optimal demand/pricing and resource allocation policies obtained for each location i, denoted by $d^*_{i\phi_i}(x_i)$ and $\theta^*_{i\phi}(x_i)$ in the two-location problems are then disaggregated using the following mechanism to generate a policy for the N-locations problem,

$$d^{LOOA}(\mathbf{x})_{ij} = \Pi_{[\underline{d}_{ij}, \overline{d}_{ij}]}\Big[\tilde{d}^*_{i\phi_i}(x_i)\frac{\alpha_{ij}}{\sum_k \alpha_{ik}}\Big], \tag{4.6}$$

$$\theta^{LOOA}(\mathbf{x})_{ij} = \theta^*_{i\phi_i}(x_i)\frac{\alpha_{ij}}{\sum_k \alpha_{ik}}, \tag{4.7}$$

where $\alpha_{ij} = \frac{d^*_{ij}}{x_j}$ with $d^*_{ij} = \begin{cases} d^*_{i\phi_i}(x_i), & \text{if i} \neq \text{j} \\ d^{\text{fixed}}_{ii}, & \text{otherwise.} \end{cases}$ and $\Pi_{[a,b]}[x] = \max\{\min\{x, b\}, a\}$.

This means that the demand and resources allocated to location j from location i, are proportional to the expected demand proportion from i to j and inversely proportional to the number of resources at location j. This ensures that locations with fewer resources will have higher demand proportion and resources allocated to them. We summarize the steps involved in the leave-one-out heuristic in Algorithm 3.

Algorithm 3: Leave-One-Out Aggregation

Input: demand models between locations $D_{k\ell}$, min and max prices $[\underline{p}_{ij}, \overline{p}_{ij}]$, number of locations N, number of resources m.

Output: pricing and allocation policies: $d^{LOOA}(\mathbf{x})$ and $\theta^{LOOA}(\mathbf{x})$

for $i = 1, 2, \ldots,$ N **do**

Aggregate locations {1,. . . ,N}/{i}:

Compute the demand function at location i and the aggregated location ϕ_i, $\kappa_{k\ell}(p_{k\ell})$, $k, \ell \in \{i, \phi_i\}$ using (4.5).

Solve the resulting two-location problem via DP and record the optimal polices $d^*_{i\phi_i}(x_i)$ and $\theta^*_{i\phi}(x_i)$

end for

Use the resulting optimal policies for each location to obtain $d^{LOOA}(\mathbf{x})$ and $\theta^{LOOA}(\mathbf{x})$ following equations (4.6) and (4.7).

4.7 Computational Experiments

In this section, we report the results of a series of computational experiments to investigate the performance of the leave-one-out aggregation heuristic for the infinite horizon problem. We compare the quality of LOOA policy against the myopic policy that maximizes the revenue of each location separately and the best fixed node and arc based pricing policies. The allocation decisions for these baseline policies are made according to $\theta_{ij}(x_i) = x_i \frac{\alpha_{ij}}{\sum_k \alpha_{ik}}$, where α_{ij} is as defined in Section 4.6. Our simulation study includes five sets of random problems, summarized in Table 1. We assume a linear demand function of the price,

$$\kappa_{ij}(p_{ij}) = a_{ij} - b\, p_{ij},$$

where the parameters, a_{ij} and b_{ij} are generated randomly from uniform distributions $U[7, 15]$ and $U[1, 4]$, respectively. The random noise ϵ_{ij} are sampled from $U[-3, 3]$ for $i, j \in \{1, \ldots, N\}$. The minimum price $\underline{p}_{ij}$ is set to 1 for all $i, j \in \{1, \ldots, N\}$. On the other hand, the maximum price is set to $\frac{a_{ij} - \bar{\epsilon}_{ij}}{b_{ij}}$ such that $\kappa_{ij}(p_{ij}) + \epsilon_{ij} \geq 0$ for all $i, j \in \{1, \ldots, N\}$. The return trips price at each of the location is $p_{ii} = \underline{p}_{ii}$. The lost sales cost for all experiments was set such that $\rho_{ij} = \overline{p}_{ij}$ for all $i \neq j$ and to p_{ii} otherwise.

For each test problem, we compute the following quantities:

$Z^{\text{Node}}_{\text{Fixed}}$: Estimated expected value obtained through simulation of the best fixed *node* based pricing strategy. To evaluate the fixed pricing policy we run a simulation to obtain the value of the discounted sum of rewards. Each run is approximated by truncating the infinite summation of discounted rewards such that the terms after t periods are ignored, where $\gamma^t < 10^{-3}$.

$Z^{\text{Arc}}_{\text{Fixed}}$: Estimated expected value of the best fixed *arc* based pricing strategy obtained using a differential evolution algorithm [62]. The algorithm is implemented in Python's scipy library. We use a population size of $10N$, a differential weight of 0.8, and a crossover probability of 0.7. The solution of the fixed node pricing is used as one of the initial solutions.

Z_{Myopic}: Estimated expected value of the myopic policy that maximizes the revenue at each location.

Table 1: Numerical Results

Problem set (N, m)	γ	$Z^{\text{Node}}_{\text{Fixed}}$	$Z^{\text{Arc}}_{\text{Fixed}}$	Z_{Myopic}	Z_{LOOA}
Problem 1	0.2	10.05	13.17	14.91	**38.06**
(3, 25)	0.5	15.98	17.15	23.68	**60.16**
	0.9	72.37	79.45	105.59	**272.8**
	0.95	143.28	146.17	204.82	**546.88**
	0.99	714.05	721.14	995.99	**2592.85**
Problem 2	0.2	21.94	24.82	31.91	**69.21**
(4, 40)	0.5	33.68	39.82	46.75	**97.38**
	0.9	191.48	194.31	240.76	**487.97**
	0.95	379.47	384.8	481.22	**979**
	0.99	1880.54	1898.37	2450.31	**4905.68**
Problem 3	0.2	142.59	149.89	153.5	**182.34**
(5, 80)	0.5	240.05	242.19	246.8	**291**
	0.9	1188.28	1203.38	1233.11	**1441.56**
	0.95	2354.07	2355.23	2453.36	**2877.13**
	0.99	11766.49	11769.25	12264.52	**14336.1**
Problem 4	0.2	171.35	177.11	180	**230.55**
(6, 100)	0.5	275	276.5	286.97	**364.34**
	0.9	1381.97	1384.18	1452.92	**1805.05**
	0.95	2769.32	2781.27	2884.88	**3606.10**
	0.99	13615.04	13718.61	14391.05	**18035.44**
Problem 5	0.2	278.11	281.44	287.38	**331.7**
(7, 140)	0.5	442.37	448.83	451.04	**525.95**
	0.9	2147.46	2152.12	2202.53	**2587.42**
	0.95	4278.13	4291.85	4467.17	**5170.71**
	0.99	21703.39	21736.16	22527.42	**25834**

Z_{LOOA}: Simulated expected value of the LOOA policy. The aggregated two-location problems are solved by Q-iteration [63].

The expected value of each heuristic is approximated using the average over 100 runs. The heuristics and the simulation experiments were implemented in Python. All the experiments were performed on a shared memory cluster with dual 12-core Skylake CPU (Intel Xeon Gold 6126 2.60 GHz) and 192 GB RAM/node.

Table 1 shows five sets of problems with increasing complexity in terms of number of locations and resources and various values of the discount factor γ. Compared with the other heuristics, LOOA yielded a significantly higher revenue ($p < 0.05$) on all problems. Problems 1 and 2 involve 3 and 4 locations with 25 and 40 cars, respectively. In these problems, LOOA outperformed the myopic policy which in turn outperformed the fixed pricing policies.

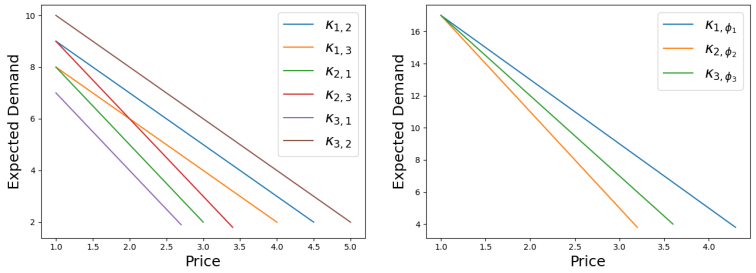
As expected, the performance of the myopic heuristic becomes worse as the discount factor increase, since in this case the long-term impact of the current decision becomes more important. LOOA consistently outperformed the other heuristics on the larger problems as well (Problems 3, 4, and 5). Compared to the best baseline heuristic LOOA achieves up to 18%, 30% , and 16% improvement on Problems 3, 4, and 5, respectively. On average over all problems achieves LOOA 65% improvement, with a max improvement of 167% percent on Problem 1 ($\gamma = 0.95$).

Note that the solution of the myopic policy took significantly more time than the other heuristic for the larger problems. This is because as the number of locations increase the number actions becomes exponentially huge. For example for the 5 locations problem the total number of demand actions is in the order of 1.3 trillion.

The separate and the aggregated expected demand functions for each origin-destination pair of the three location problem, are shown in figures 10a and 10b, respectively.

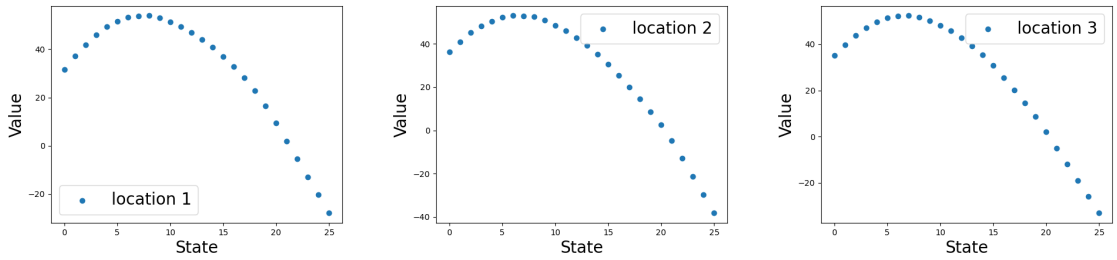
Figures 11 and 12 show the plots of the value functions and policies obtained from running LOOA algorithm on Problem 1 with $\gamma = 0.9$. The plots show the concavity property of the value function in addition to the monotonocity and bounded sensitivity property of the pricing and allocation polices for each of the leave-one-out problems.

Finally, figure 13 shows a representation of the resulting LOOA's demand and pricing policy at location 1. Notice how the demand and allocation levels changes with the state.



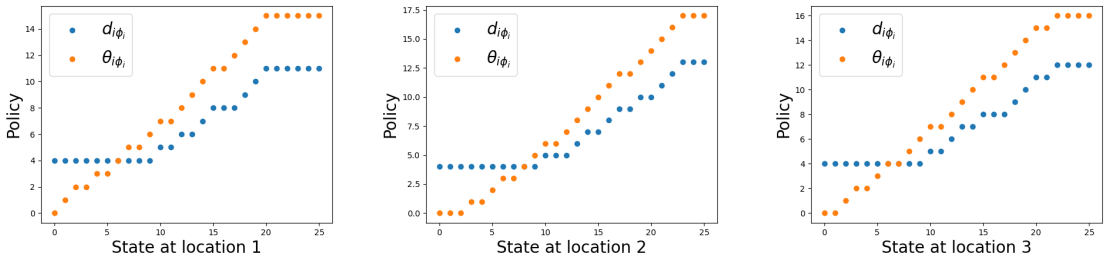
(a) (b)

Figure 10: Plots showing the three location problem origin-destination expected demand functions of the price (a) and the resulting expected demand functions after aggregation (b).



(a) (b) (c)

Figure 11: Individual LOOA value functions for Problem 1, $\gamma = 0.9$.



(a) (b) (c)

Figure 12: Individual LOOA policies for Problem 1, $\gamma = 0.9$.

For example, in figures 13a and 13b, as the number of cars increase at location 1, LOOA's demand levels from location 1 to 2 and 1 to 3 increase as well to hopefully re-balance the resources across all locations.

4.8 Conclusion

In this chapter, we studied the structural properties of value function and policy for a dynamic pricing and allocation problem of price sensitive resources that are shared in a network of locations. We showed that the value function is concave in the state and for the special case where the network consists of two locations, we showed that the value function is $L^\natural$-concave and the policy has monotonicity and bounded sensitivity properties. We developed an efficient heuristic that divides the problem into smaller problems and exploits that theoretical structural results to obtain a good approximate solution to the full problem. Our heuristic consistently outperformed the baselines heuristic by a large margin on a set of test problems.

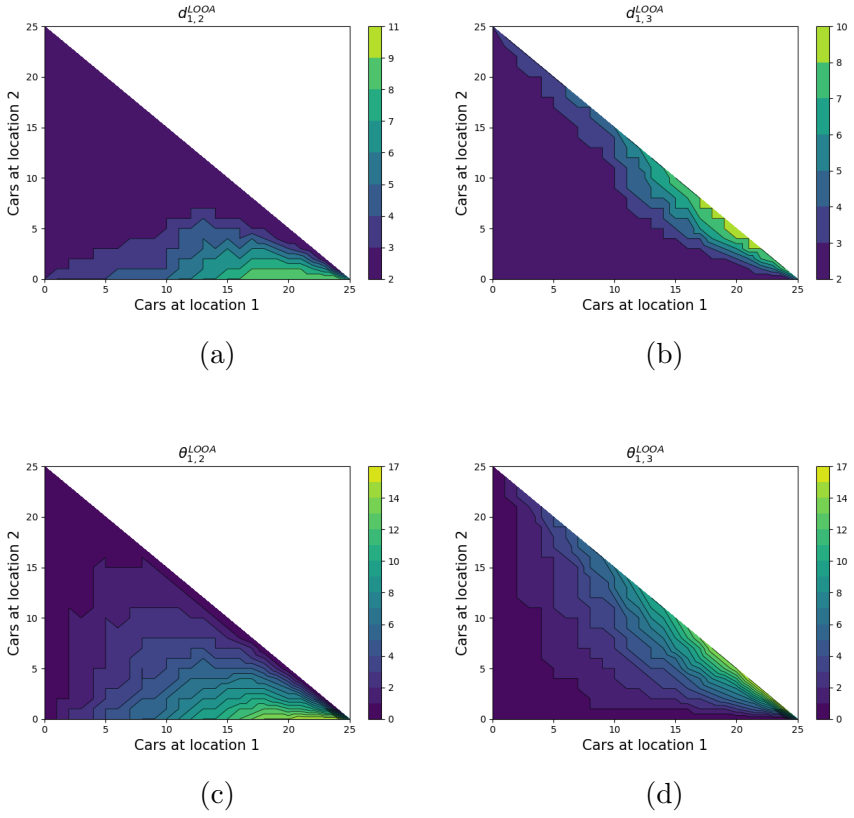


Figure 13: LOOA's demand and allocation policies at location 1 for Problem 1, $\gamma = 0.9$.

5.0 Conclusions and Future Work

In this thesis, we exploit various relaxation techniques and inherent structural properties to efficiently learn and solve sequential decision-making problems. We first study stochastic infinite horizon MDPs where partial knowledge about the environment dynamics is available and the randomness is due to exogenous information that is often observable but otherwise unknown. These problems appear in many optimal control and logistics problems. Examples include inventory management, vehicle routing, dynamic pricing, energy operations, and many more. We develop an algorithm that leverages information relaxation to generate upper and lower bounds on the optimal value to improve the performance of the celebrated Q-learning algorithm. The partial knowledge of the transition function along with the observed outcomes of the exogenous information is used to create and solve relaxed finite-horizon deterministic DP problems. The solutions of these deterministic DPs can be used to obtain a loose upper bound on the optimal value function. To get tighter bounds, we use the idea of information relaxation where a penalty function is used to penalize the usage of future information. Our algorithm takes the current Q-learning Q-value to construct a penalty function and generate the lower and upper bounds. These bounds are then used to improve the current action-value function estimate using a projection step that projects the Q-value between the bounds. The projected values goes through a Q-learning step before it is used again as in the penalty function for bounds generation. We propose two versions of our algorithm that differs in how the samples and sample paths of the exogenous information are obtained to generate the bounds. The first version assumes access to a black-box simulator to generate the exogenous information samples while the other one makes use of a replay-buffer to record the exogenous information obtained from interacting with the environment. In both cases, we show that our algorithm along with the lower and upper bounds converge to the optimal action-value function. One possible extension of our algorithm includes learning the transition function from the experience obtained from interacting with the environment. Another extension would be to extend the algorithm to the function approximation case.

We then study a broad class of problems that have a weakly coupled structure. These

problems consist of multiple subproblems that are independent except for a linking constraint on the action space. We use a Lagrangian relaxation technique to obtain an upper bound on the optimal value. We also learn the value of the Lagrangian policy which serves as a lower bound on Q^*. Unlike the information relaxation approach we developed in Chapter 1, this approach do not require any knowledge of the information dynamics and it is also extendable to the function approximation case which we implement using Deep Q-Networks. The latter works through a soft constrained optimization approach that replaces the hard projection step in the tabular version. We also propose a simpler version of our algorithm that directly learn the Lagrangian policy and does not require computing the bounds. An interesting future work includes learning optimal Lagrangian multipliers through adaptive ways, e.g., using subgradient descent.

Finally, we study the problem of pricing and allocation decisions for a finite number of resources that circulates a network. This problem is inspired from the growing popularity of car-sharing and bike-sharing systems. For the general problem that consists of $N > 2$, we show that the optimal value function is concave in the state and for the two-location problem, we show that value function enjoys a discrete convexity property and that the policy in monotone and has a bounded sensitivity property. We then propose an efficient heuristic that utilizes the structural properties of the two-location problem. It does so by transforming the problem into N two-location problems. The two-location problems consist of one location that is kept unchanged and a single a aggregated location that approximates the remaining $N-1$ locations in the network. The structural properties of the policy allows us to solve these problems efficiently. The resulting policies at each location is then used to generate a single policy for the whole problem. Our heuristic outperforms our baseline heuristics on a different set of problems. Future work includes deriving bounds on the performance of the heuristic and extending our approach to the function approximation case where one could use convex neural networks [2] for example to model the structure of the value function.

Appendix A

A.1 Proofs for Chapter 2

A.1.1 Proof of Proposition 2.3.1

Proof. We provide a proof that is similar to that of Proposition 2.3 (iv) of [14] but for the case of the absorption time formulation of an infinite horizon problem. Here, we define a policy $\pi := \{\pi_t\}_{t\geq 0}$ as a sequence of functions, that maps from $\{w_t\}_{t\geq 1}$ to feasible actions. We may also use stationary policies where π_t is the same for all t and only depends on the current state s_t. Let $\mathbb{G} = \{\mathcal{G}_t\}_{t\geq 0}$ be the perfect information relaxation of the natural filtration $\mathbb{F} = \{\mathcal{F}_t\}_{t\geq 0}$. Under $\mathbb{G}$, we have $\mathcal{G}_t = \mathcal{F}$, i.e., we have access to the entire future uncertainties at each t. Define by $\Pi_{\mathbb{G}}$ the set of policies that includes the policies that have access to future uncertainties in addition to nonanticipative policies. Let $\hat{\mathbb{G}}$ be a relaxation of $\mathbb{G}$ such that in addition to what is known under $\mathbb{G}$ the estimate penalty terms $\hat{\zeta}_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi)$ are revealed at time t.

We first prove $\mathbf{E}[\hat{Q}_0^L(s,a)] \leq Q^*(s,a)$. For an admissible policy π, we have

$$
\begin{aligned}
\mathbf{E}[\hat{Q}_0^L(s,a)] &\overset{\text{(a)}}{=} \mathbf{E}\left[\sum_{t=0}^{\tau-1} r(s_t,\pi(s_t)) - \hat{\zeta}_t^{\pi}(s_t,a_t,w_{t+1}\,|\,\varphi)\,|\,s_0=s,\,a_0=a\right] \\
&\overset{\text{(b)}}{=} \mathbf{E}\left[\sum_{t=0}^{\tau-1} (r(s_t,\pi(s_t)) - \hat{\zeta}_t^{\pi}(s_t,a_t,w_{t+1}\,|\,\varphi))\,\mathbb{1}_{\{\tau<\infty\}}\,|\,s_0=s,\,a_0=a\right] \\
&\overset{\text{(c)}}{=} \sum_{\tau'=1}^{\infty} \mathbf{E}\left[\sum_{t=0}^{\tau'-1} (r(s_t,\pi(s_t)) - \hat{\zeta}_t^{\pi}(s_t,a_t,w_{t+1}\,|\,\varphi))\,\mathbb{1}_{\{\tau=\tau'\}}\,|\,s_0=s,\,a_0=a\right] \\
&\overset{\text{(d)}}{=} \sum_{\tau'=1}^{\infty} \mathbf{E}\left[\sum_{t=0}^{\tau'-1} (r(s_t,\pi(s_t)) - \mathbf{E}[\hat{\zeta}_t^{\pi}(s_t,a_t,w_{t+1}\,|\,\varphi)\,|\,\mathcal{G}_t])\,\mathbb{1}_{\{\tau=\tau'\}}\,|\,s_0=s,\,a_0=a\right] \\
&\overset{\text{(e)}}{=} \sum_{\tau'=1}^{\infty} \mathbf{E}\left[\sum_{t=0}^{\tau'-1} (r(s_t,\pi(s_t)) - \zeta_t^{\pi}(s_t,a_t,w_{t+1}\,|\,\varphi))\,\mathbb{1}_{\{\tau=\tau'\}}\,|\,s_0=s,\,a_0=a\right] \\
&\overset{\text{(f)}}{=} \mathbf{E}\left[\sum_{t=0}^{\tau-1} (r(s_t,\pi(s_t)) - \zeta_t^{\pi}(s_t,a_t,w_{t+1}\,|\,\varphi))\,\mathbb{1}_{\{\tau<\infty\}}\,|\,s_0=s,\,a_0=a\right] \\
&\overset{\text{(g)}}{=} \mathbf{E}\left[\sum_{t=0}^{\tau-1} (r(s_t,\pi(s_t)) - \zeta_t^{\pi}(s_t,a_t,w_{t+1}\,|\,\varphi))\,|\,s_0=s,\,a_0=a\right] \\
&\leq Q^*(s,a)
\end{aligned}
$$

Equality (a) follows from the definition of $\hat{Q}_0(s,a)$ and equality (b) follows since τ has finite mean and r and φ are uniformly bounded. Equalities (c) and (d) follow from the law of total expectations. Equality (e) follows from Lemma A.1 in [14] and from the estimated penalty terms being unbiased, i.e., $\mathbf{E}[\hat{\zeta}_t^{\pi_G^*}(s_t,a_t,w_{t+1}\,|\,\varphi)\,|\,\mathcal{G}_t] = \zeta_t^{\pi_G^*}(s_t,a_t,w_{t+1}\,|\,\varphi)$. Equalities (f) and (g) follow by the law of total expectation and τ being almost surely finite stopping time, respectively. The inequality follows since the expected value of the penalty terms for a feasible policy is zero and the action-value function of a feasible policy, $Q^{\pi}(s,a)$, is less than $Q^*(s,a)$.

Now, we prove $Q^*(s,a) \leq \mathbf{E}[\hat{Q}_0^U(s,a)]$. Let π_G^* be the optimal solution for the dual problem,

$$
\max_{\pi_G \in \Pi_{\mathbb{G}}} \mathbf{E}\left[\sum_{t=0}^{\tau-1} (r(s_t,\pi_G) - \zeta_t^{\pi_G}(s_t,a_t,w_{t+1}\,|\,\varphi))\,|\,s_0=s,\,a_0=a\right]. \tag{A.1}
$$

We have,

$$
\begin{aligned}
\mathbf{E}[\hat{Q}_0^U(s,a)] &\overset{\text{(a)}}{=} \mathbf{E}\left[\max_{\mathbf{a}}\left\{\sum_{t=0}^{\tau-1} r(s_t,a_t) - \hat{\zeta}_t^{\pi_\varphi}(s_t,a_t,w_{t+1}\,|\,\varphi)\right\} \,|\, s_0=s,\, a_0=a\right] \\
&\overset{\text{(b)}}{=} \max_{\pi_G\in\Pi_{\hat{\mathbb{G}}}} \mathbf{E}\left[\sum_{t=0}^{\tau-1}(r(s_t,\pi_G) - \hat{\zeta}_t^{\pi_G}(s_t,a_t,w_{t+1}\,|\,\varphi)) \,|\, s_0=s,\, a_0=a\right] \\
&\geq \mathbf{E}\left[\sum_{t=0}^{\tau-1}(r(s_t,\pi_G^*) - \hat{\zeta}_t^{\pi_G^*}(s_t,a_t,w_{t+1}\,|\,\varphi)) \,|\, s_0=s,\, a_0=a\right] \\
&\overset{\text{(c)}}{=} \mathbf{E}\left[\sum_{t=0}^{\tau-1}(r(s_t,\pi_G^*) - \hat{\zeta}_t^{\pi_G^*}(s_t,a_t,w_{t+1}\,|\,\varphi))\,\mathbb{1}_{\{\tau<\infty\}} \,|\, s_0=s,\, a_0=a\right] \\
&\overset{\text{(d)}}{=} \sum_{\tau'=1}^{\infty}\mathbf{E}\left[\sum_{t=0}^{\tau'-1}(r(s_t,\pi_G^*) - \hat{\zeta}_t^{\pi_G^*}(s_t,a_t,w_{t+1}\,|\,\varphi))\,\mathbb{1}_{\{\tau=\tau'\}} \,|\, s_0=s,\, a_0=a\right] \\
&\overset{\text{(e)}}{=} \sum_{\tau'=1}^{\infty}\mathbf{E}\left[\sum_{t=0}^{\tau'-1}(r(s_t,\pi_G^*) - \mathbf{E}[\hat{\zeta}_t^{\pi_G^*}(s_t,a_t,w_{t+1}\,|\,\varphi)\,|\,\mathcal{G}_t])\,\mathbb{1}_{\{\tau=\tau'\}} \,|\, s_0=s,\, a_0=a\right] \\
&\overset{\text{(f)}}{=} \sum_{\tau'=1}^{\infty}\mathbf{E}\left[\sum_{t=0}^{\tau'-1}(r(s_t,\pi_G^*) - \zeta_t^{\pi_G^*}(s_t,a_t,w_{t+1}\,|\,\varphi))\,\mathbb{1}_{\{\tau=\tau'\}} \,|\, s_0=s,\, a_0=a\right] \\
&\overset{\text{(g)}}{=} \mathbf{E}\left[\sum_{t=0}^{\tau-1}(r(s_t,\pi_G^*) - \zeta_t^{\pi_G^*}(s_t,a_t,w_{t+1}\,|\,\varphi))\,\mathbb{1}_{\{\tau<\infty\}} \,|\, s_0=s,\, a_0=a\right] \\
&\overset{\text{(h)}}{=} \mathbf{E}\left[\sum_{t=0}^{\tau-1}(r(s_t,\pi_G^*) - \zeta_t^{\pi_G^*}(s_t,a_t,w_{t+1}\,|\,\varphi)) \,|\, s_0=s,\, a_0=a\right] \\
&\overset{\text{(i)}}{=} \max_{\pi_G\in\Pi_{\mathbb{G}}} \mathbf{E}\left[\sum_{t=0}^{\tau-1}(r(s_t,\pi_G) - \zeta_t^{\pi_G}(s_t,a_t,w_{t+1}\,|\,\varphi)) \,|\, s_0=s,\, a_0=a\right] \\
&\geq Q^*(s,a).
\end{aligned}
$$

Equality (a) and (b) follow from the definition of $\hat{Q}_0^U(s,a)$ and since $\hat{\mathbb{G}}$ is a relaxation of the perfect information relaxation $\mathbb{G}$, which allows us to interchange the maximum and the expectation. The first inequality follows because $\pi_G^* \in \Pi_{\hat{\mathbb{G}}}$ since $\Pi_{\mathbb{G}} \subseteq \Pi_{\hat{\mathbb{G}}}$. Equality (c) follows since r and φ are uniformly bounded and τ has finite mean. Equalities (d) and (e) follow from the law of total expectations. Equality (f) follows from Lemma A.1 in [14] and from the estimated penalty terms being unbiased, i.e., $\mathbf{E}[\hat{\zeta}_t^{\pi_G^*}(s_t,a_t,w_{t+1}\,|\,\varphi)\,|\,\mathcal{G}_t] = \zeta_t^{\pi_G^*}(s_t,a_t,w_{t+1}\,|\,\varphi)$. Equalities (g) and (h) follow by the law of total expectation and τ being almost surely finite stopping time, respectively. Equality (i) follows since by definition π_G^*

is the optimal solution of (A.1). The last inequality follows by weak duality (Proposition 2.2.1(i)). □

First, we state a technical lemma that is used in the proof of Proposition 2.3.2 and Lemma 2.3.1.

Lemma A.1.1. *For all $n = 1, 2, \ldots$, if $L_{n-1}(s,a) \leq U_{n-1}(s,a)$ and $Q'_{n-1} \in \mathcal{Q}$ then $L_n(s,a) \leq U_n(s,a)$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$.*

Proof. Fix an $(s,a) \in \mathcal{S} \times \mathcal{A}$. Note that the optimal values of the inner problems in (2.9) and (2.10), $\hat{Q}^U_0(s,a)$ and $\hat{Q}^L_0(s,a)$ respectively, are computed using the same sample path $\mathbf{w}$ and for each period within the inner DP, the same batch of samples is used for estimating the expectation in both the upper and lower bound problems. For clarity, let us denote the values of $\hat{Q}^L_0(s,a)$ and $\hat{Q}^U_0(s,a)$ at iteration $n = 1, 2, \ldots$ by $\hat{Q}^L_{n,0}(s,a)$ and $\hat{Q}^U_{n,0}(s,a)$, respectively. Assume $\alpha_n(s,a) \leq 1$ for all n. We provide a proof by induction. For $n = 1$, we have:

$$Q_1(s_0,a_0) = Q'_0(s_0,a_0) + \alpha_0(s_0,a_0)\Big[r(s_0,a_0) + \gamma \max_a Q'_0(s_1,a) - Q'_0(s_0,a_0)\Big].$$

Since the rewards $r(s,a)$ are uniformly bounded, $0 < \gamma < 1$ and $|Q'_0(s,a)| \leq \rho$ then Q_1 is bounded. Set $\varphi = Q_1$, since the actions selected by the policy π_{Q_1} are feasible in (2.9), we have

$$\hat{Q}^U_{1,0}(s,a) - \hat{Q}^L_{1,0}(s,a) \geq 0,$$

and with $L_0(s,a) \leq U_0(s,a)$, it follows that $L_1(s,a) \leq U_1(s,a)$. A similar proof can be used to show the inductive case also holds at iteration n,

$$\begin{aligned} Q_n(s_{n-1},a_{n-1}) &= Q'_{n-1}(s_{n-1},a_{n-1}) \\ &\quad + \alpha_{n-1}(s_{n-1},a_{n-1})\Big[r(s_{n-1},a_{n-1}) + \gamma \max_a Q'_{n-1}(s_n,a) - Q'_{n-1}(s_{n-1},a_{n-1})\Big]. \end{aligned}$$

By the inductive hypothesis, we have $Q'_{n-1} \in \mathcal{Q}$ and $L_{n-1}(s,a) \leq U_{n-1}(s,a)$. Then, similar to the base case, we have $Q_n \in \mathcal{Q}$ and $\hat{Q}^U_{n,0}(s,a) - \hat{Q}^L_{n,0}(s,a) \geq 0$. Therefore, $L_n(s,a) \leq U_n(s,a)$. Since our choice of (s,a) was arbitrary then the result follows for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. □

A.1.2 Proof of Proposition 2.3.2

Proof. Part (i): First, note that by (2.11) and (2.12) the upper and lower bound estimates $U_n(s,a)$ and $L_n(s,a)$ are bounded below and above by ρ and $-\rho$ respectively for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ and for all n, where $\rho = R_{\max}/(1-\gamma)$. We assume in this proof that $\alpha_n(s,a) \leq 1$ for all n. Let $\tilde{L}_n = \max_{(s,a)} L_n(s,a)$ and $\tilde{U}_n = \max_{(s,a)} U_n(s,a)$. We claim that for every iteration n, we have that for all (s,a),

$$\bar{L}_n \leq Q_n(s,a) \leq \bar{U}_n \quad \text{and} \quad \bar{L}'_n \leq Q'_n(s,a) \leq \bar{U}'_n \tag{A.2}$$

where

$$\bar{L}_n = \min\left\{\tilde{U}_{n-1}(1+\gamma), \ldots, \tilde{U}_1 \textstyle\sum_{i=0}^{n-1}\gamma^i, -M\sum_{i=0}^{n}\gamma^i\right\} \tag{A.3}$$

$$\bar{U}_n = \max\left\{\tilde{L}_{n-1}(1+\gamma), \ldots, \tilde{L}_1 \textstyle\sum_{i=0}^{n-1}\gamma^i, M\sum_{i=0}^{n}\gamma^i\right\}, \tag{A.4}$$

$$\bar{L}'_n = \min\left\{\tilde{U}_n, \tilde{U}_{n-1}(1+\gamma), \ldots, \tilde{U}_1 \textstyle\sum_{i=0}^{n-1}\gamma^i, -M\sum_{i=0}^{n}\gamma^i\right\}, \tag{A.5}$$

$$\bar{U}'_n = \max\left\{\tilde{L}_n, \tilde{L}_{n-1}(1+\gamma), \ldots, \tilde{L}_1 \textstyle\sum_{i=0}^{n-1}\gamma^i, M\sum_{i=0}^{n}\gamma^i\right\}, \tag{A.6}$$

and M is a finite positive scalar defined as $M = \max\left\{R_{\max},\ \max_{(s,a)} Q_0(s,a)\right\}$.

The result follows from the claim in (A.2). To see this note that at any iteration n, $\bar{L}_n$ and $\bar{L}'_n$ are bounded below by $-\rho\sum_{i=0}^{n}\gamma^i$ since each term inside the minimum of (A.3) and (A.5) is bounded below by $-\rho\sum_{i=0}^{n}\gamma^i$. As $n \to \infty$, we have

$$\frac{-\rho}{1-\gamma} \leq \liminf_{n\to\infty} \bar{L}_n \quad \text{and} \quad \frac{-\rho}{1-\gamma} \leq \liminf_{n\to\infty} \bar{L}'_n. \tag{A.7}$$

An analogous argument yields

$$\limsup_{n\to\infty} \bar{U}_n \leq \frac{\rho}{1-\gamma} \quad \text{and} \quad \limsup_{n\to\infty} \bar{U}'_n \leq \frac{\rho}{1-\gamma}. \tag{A.8}$$

Boundedness of $Q_n(s,a)$ and $Q'_n(s,a)$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ follows from (A.2), (A.7) and (A.8).

Now, we prove our claim in (A.2) by induction. Since Algorithm 1 is asynchronous, at the nth iteration, the updates for the action-value iterates for (s,a), $Q_{n+1}(s,a)$ and $Q'_{n+1}(s,a)$, are either according to (5) and (2.13) (case 1) or set equal to $Q_n(s,a)$ and $Q'_n(s,a)$ respectively (case 2).

We first focus on $Q'_n(s,a) \leq \bar{U}_n$ part of (A.2), since $\bar{L}'_n \leq Q'_n(s,a)$ and $\bar{L}_n \leq Q_n(s,a) \leq \bar{U}_n$ proceed in an analogous manner. For $n = 1$, we have $Q'_0(s,a) = Q_0(s,a)$, so if the update is carried out as in case 1,

$$\begin{aligned} Q_1(s,a) &= (1-\alpha_0(s,a))\, Q'_0(s,a) + \alpha_0(s,a)\,[r(s,a) + \gamma \max_a Q'_0(s',a)] \\ &\leq (1-\alpha_0(s,a))M + \alpha_0(s,a)M + \alpha_0(s,a)\gamma M \\ &\leq M(1+\gamma) \end{aligned}$$

so $Q'_1(s,a) \leq \max\{L_1(s,a), \min\{U_1(s,a), M(1+\gamma)\}\}$. Now consider the case where $U_1(s,a) \leq M(1+\gamma)$. Since Q_0 is bounded by ρ and $L_0(s,a) \leq U_0(s,a)$ then by Lemma A.1.1, we have $L_1(s,a) \leq U_1(s,a)$, so

$$Q'_1(s,a) \leq U_1(s,a) \leq M(1+\gamma). \tag{A.9}$$

Otherwise, if $U_1(s,a) \geq M(1+\gamma)$, we then have

$$Q'_1(s,a) \leq \max\{L_1(s,a), M(1+\gamma)\}. \tag{A.10}$$

From (A.9) and (A.10), we have

$$\begin{aligned} Q'_1(s,a) &\leq \max\{L_1(s,a), M(1+\gamma)\} \\ &\leq \max\{\tilde{L}_1, M(1+\gamma)\}. \end{aligned} \tag{A.11}$$

If the update is carried out as in case 2, we have,

$$\begin{aligned} Q'_1(s,a) &= Q'_0(s,a) \\ &\leq M \\ &< M(1+\gamma) \\ &\leq \max\{\tilde{L}_1, M(1+\gamma)\}. \end{aligned}$$

Thus $\bar{U}'_n(s,a)$ part of (A.2) is true for $n = 1$. Suppose that it is true for $n = 1, 2, \ldots, k$. We will show it for $n = k+1$. Consider first the instance where the update is carried out according to case 1. We do casework on the inequality

$$Q'_k(s,a) \leq \max\left\{\tilde{L}_k, \tilde{L}_{k-1}(1+\gamma), \ldots, \tilde{L}_1 \textstyle\sum_{i=0}^{k-1} \gamma^i, M \sum_{i=0}^{k} \gamma^i\right\}, \tag{A.12}$$

which holds for all (s, a). First, let us consider the case where the right-hand-side of (A.12) is equal to $\tilde{L}_{k'} \sum_{i=0}^{k-k'} \gamma^i$ for some k' such that $1 \leq k' \leq k$. Then, we have

$$\begin{aligned} Q_{k+1}(s,a) &= (1-\alpha_k(s,a))Q'_k(s,a) + \alpha_k(s,a)[r(s,a) + \gamma \max_a Q'_k(s',a)] \\ &\leq (1-\alpha_k(s,a))\tilde{L}_{k'} \textstyle\sum_{i=0}^{k-k'} \gamma^i + \alpha_k(s,a)M + \alpha_k(s,a)\gamma\tilde{L}_{k'} \sum_{i=0}^{k-k'} \gamma^i \\ &\leq (1-\alpha_k(s,a))\tilde{L}_{k'} \textstyle\sum_{i=0}^{k-k'} \gamma^i + \alpha_k(s,a)\tilde{L}_{k'} + \alpha_k(s,a)\tilde{L}_{k'} \sum_{i=1}^{k-k'+1} \gamma^i \\ &= (1-\alpha_k(s,a))\tilde{L}_{k'} \textstyle\sum_{i=0}^{k-k'} \gamma^i + \alpha_k(s,a)\tilde{L}_{k'} \sum_{i=0}^{k-k'} \gamma^i + \alpha_k(s,a)\tilde{L}_{k'}\gamma^{k-k'+1} \\ &\leq \tilde{L}_{k'} \textstyle\sum_{i=0}^{k-k'} \gamma^i + \tilde{L}_{k'}\gamma^{k-k'+1} \\ &= \tilde{L}_{k'} \textstyle\sum_{i=0}^{k-k'+1} \gamma^i \end{aligned} \tag{A.13}$$

The first inequality holds by the induction assumption (A.12). The second inequality holds since in this case we have the right-hand-side of (A.12) is equal to $\tilde{L}_{k'}(1 + \gamma + \ldots + \gamma^{k-k'})$. It follows that

$$\tilde{L}_{k'}(1 + \gamma + \ldots + \gamma^{k-k'}) \geq M(1 + \gamma + \ldots + \gamma^k),$$

which implies that $\tilde{L}_{k'} \geq M$. Finally, the third inequality holds by the assumption that $\alpha_n(s,a) \leq 1$ for all n. We have

$$\begin{aligned} Q'_{k+1}(s,a) &= \max\{L_{k+1}(s,a), \min\{U_{k+1}(s,a), Q_{k+1}(s,a)\}\} \\ &\leq \max\{L_{k+1}(s,a), \min\{U_{k+1}(s,a), \tilde{L}_{k'}(1 + \gamma + \ldots + \gamma^{k-k'+1})\}\}. \end{aligned}$$

Now, consider the case where $U_{k+1}(s,a) \leq \tilde{L}_{k'}(1 + \gamma + \ldots + \gamma^{k-k'+1})$. By the induction assumption, $Q'_n(s,a)$ is bounded below by $-\rho \sum_{i=0}^{n} \gamma^i$ and above by $\rho \sum_{i=0}^{n} \gamma^i$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ and all $n = 1, 2, \ldots, k$. Since $L_0(s,a) \leq U_0(s,a)$, Lemma A.1.1 can be applied iteratively on $n = 1, \ldots, k+1$ to obtain that $L_{K+1}(s,a) \leq U_{k+1}(s,a)$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. Thus, we have

$$Q'_{k+1}(s,a) \leq U_{k+1}(s,a) \leq \tilde{L}_{k'}(1 + \gamma + \ldots + \gamma^{k-k'+1}). \tag{A.14}$$

Otherwise, if $U_{k+1}(s,a) \geq \tilde{L}_{k'}(1 + \gamma + \ldots + \gamma^{k-k'+1})$, we have

$$\begin{aligned} Q'_{k+1}(s,a) &\leq \max\{L_{k+1}(s,a), \tilde{L}_{k'}(1 + \gamma + \ldots + \gamma^{k-k'+1})\} \\ &\leq \max\{\tilde{L}_{k+1}, \tilde{L}_{k'}(1 + \gamma + \ldots + \gamma^{k-k'+1})\}. \end{aligned} \tag{A.15}$$

Moving on to the case where the right-hand-side of (A.12) is equal to $M(1+\gamma+\ldots+\gamma^k)$:

$$\begin{aligned} Q_{k+1}(s,a) &= (1-\alpha_k(s,a))\, Q'_k(s,a) + \alpha_k(s,a)\,[r(s,a) + \gamma \max_a Q'_k(s',a)] \\ &\leq (1-\alpha_k(s,a))M \textstyle\sum_{i=0}^{k} \gamma^i + \alpha_k(s,a)M + \alpha_k(s,a)\gamma M \sum_{i=0}^{k} \gamma^i \\ &= (1-\alpha_k(s,a))M \textstyle\sum_{i=0}^{k} \gamma^i + \alpha_k(s,a)M + \alpha_k(s,a)M \sum_{i=1}^{k+1} \gamma^i \\ &\leq M \textstyle\sum_{i=0}^{k} \gamma^i - \alpha_k(s,a)M \sum_{i=0}^{k} \gamma^i + \alpha_k(s,a)M \sum_{i=0}^{k} \gamma^i + M\gamma^{k+1} \\ &= M(1+\gamma+\ldots+\gamma^{k+1}). \end{aligned} \tag{A.16}$$

We have

$$\begin{aligned} Q'_{k+1}(s,a) &= \max\{L_{k+1}(s,a), \min(U_{k+1}(s,a), Q_{k+1}(s,a))\} \\ &\leq \max\{L_{k+1}(s,a), \min(U_{k+1}(s,a), M(1+\gamma+\ldots+\gamma^{k+1})\}\}. \end{aligned}$$

Now if $U_{k+1}(s,a) \leq M(1+\gamma+\ldots+\gamma^{k+1})$, then by applying Lemma A.1.1 as before,

$$Q'_{k+1}(s,a) \leq U_{k+1}(s,a) \leq M(1+\gamma+\ldots+\gamma^{k+1}). \tag{A.17}$$

Otherwise, if $U_{k+1}(s,a) \geq M(1+\gamma+\ldots+\gamma^{k+1})$, we have

$$\begin{aligned} Q'_{k+1}(s,a) &\leq \max\{L_{k+1}(s,a), M(1+\gamma+\ldots+\gamma^{k+1})\} \\ &\leq \max\{\tilde{L}_{k+1}, M(1+\gamma+\ldots+\gamma^{k+1})\}. \end{aligned} \tag{A.18}$$

Now, if the update is carried out according to case 2,

$$\begin{aligned} Q'_{k+1}(s,a) &= Q'_K(s,a) \\ &\leq \max\{\tilde{L}_k, \tilde{L}_{k-1}(1+\gamma), \ldots, \tilde{L}_1 \textstyle\sum_{i=0}^{k-1} \gamma^i, M \sum_{i=0}^{k} \gamma^i\} \\ &\leq \max\{\tilde{L}_{k+1}, (1+\gamma)\tilde{L}_k, \ldots, \tilde{L}_1 \textstyle\sum_{i=0}^{k} \gamma^i, M \sum_{i=0}^{k+1} \gamma^i\}. \end{aligned} \tag{A.19}$$

By (A.14), (A.15), (A.17), (A.18) and (A.19), we have $Q'_{k+1}(s,a) \leq \bar{U}'_{k+1}$. A similar argument can be made to show $\bar{L}_n \leq Q'_n(s,a)$ and $\bar{L}_n \leq Q_n(s,a) \leq \bar{U}_n$, which completes the inductive proof. □

Proof. Part (ii): Fix an $(s, a) \in \mathcal{S} \times \mathcal{A}$. By part (i) we have the action-value iterates Q_n and Q'_n are bounded for all n. We denote the "sampling noise" term using

$$\xi_n^L(s,a) = \hat{Q}_{n,0}^L(s,a) - \mathbf{E}[\hat{Q}_{n,0}^L(s,a)].$$

We also define an accumulated noise process started at iteration ν by $W_{\nu,\nu}^L(s,a) = 0$, and

$$W_{n+1,\nu}^L(s,a) = (1 - \alpha_n(s,a))\, W_{n,\nu}^L(s,a) + \alpha_n(s,a)\, \xi_{n+1}^L(s,a) \quad \forall\, n \geq \nu,$$

which averages noise terms together across iterations. Note that τ is an almost surely finite stopping time, the rewards $r(s,a)$ are uniformly bounded, and Q_{n+1} is also bounded (by part (i)). Then, $\hat{Q}_{n,0}^L$ is bounded by some random variable and so is the conditional variance of $\xi_n^L(s,a)$. Hence, Corollary 4.1 in [9] applies and it follows that

$$\lim_{n\to\infty} W_{n,\nu}^L(s,a) = 0 \quad \forall\, \nu \geq 0.$$

Let $\tilde{\nu}$ be large enough so that $\alpha_n(s,a) \leq 1$ for all $n \geq \tilde{\nu}$. We also define

$$\begin{aligned} Y_{\tilde{\nu}}^L(s,a) &= \rho, \\ Y_{n+1}^L(s,a) &= (1 - \alpha_n(s,a))\, Y_n^L(s,a) + \alpha_n(s,a)\, Q^*(s,a), \quad \forall\, n \geq \tilde{\nu}. \end{aligned}$$

It is easy to see that the sequence $Y_n^L(s,a) \to Q^*(s,a)$. We claim that for all iterations $n \geq \tilde{\nu}$, it holds that

$$L_n(s,a) \leq \min\{\rho,\, Y_n^L(s,a) + W_{n,\tilde{\nu}}^L(s,a)\}.$$

To prove this claim, we proceed by induction on n. For $n = \tilde{\nu}$, we have

$$Y_{\tilde{\nu}}^L(s,a) = \rho \quad \text{and} \quad W_{\tilde{\nu},\tilde{\nu}}^L(s,a) = 0,$$

so it is clear that the statement is true for the base case. We now show that it is true for $n+1$ given that it holds at n:

$$\begin{aligned} L_{n+1}(s,a) &= \min\{\rho,\, (1 - \alpha_n(s,a))\, L_n(s,a) + \alpha_n(s,a)\, (\hat{Q}_{n,0}^L(s,a) - \mathbf{E}[\hat{Q}_{n,0}^L(s,a)] + \mathbf{E}[\hat{Q}_{n,0}^L(s,a)])\} \\ &= \min\{\rho,\, (1 - \alpha_n(s,a))\, L_n(s,a) + \alpha_n(s,a)\, \xi_n^L(s,a) + \alpha_n(s,a)\, \mathbf{E}[\hat{Q}_{n,0}^L(s,a)]\} \\ &\leq \min\{\rho,\, (1 - \alpha_n(s,a))\, (Y_n^L(s,a) + W_{n,\nu_k}^L(s,a)) + \alpha_n(s,a)\, \xi_n^L(s,a) + \alpha_n(s,a)\, Q^*(s,a)\} \\ &\leq \min\{\rho,\, Y_{n+1}^L(s,a) + W_{n+1,\tilde{\nu}}^L(s,a)\}, \end{aligned}$$

where the first inequality follows by the induction hypothesis and $\mathbf{E}[\hat{Q}^L_{n,0}(s,a)] \leq Q^*(s,a)$ follows by Proposition 2.3.1. Next, since $Y^L_n(s,a) \to Q^*(s,a)$, $W^L_{n,\nu_k}(s,a) \to 0$ and $Q^*(s,a) \leq \rho$, we have

$$\limsup_{n\to\infty} L_n(s,a) \leq Q^*(s,a).$$

Therefore, since our choice of (s,a) was arbitrary, it follows that for every $\eta > 0$, there exists some time n' such that $L_n(s,a) \leq Q^*(s,a) + \eta$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ and $n \geq n'$.

Using Proposition 2.3.1, $Q^*(s,a) \leq \mathbf{E}[\hat{Q}^U_{n,0}(s,a)]$, a similar argument as the above can be used to establish that

$$Q^*(s,a) \leq \liminf_{n\to\infty} U_n(s,a).$$

Hence, there exists some time n'' such that $Q^*(s,a) - \eta \leq U_n(s,a)$ for all (s,a) and $n \geq n''$. Take n_0 to be some time greater than n' and n'' and the result follows. □

A.1.3 Proof of Lemma 2.3.1

Proof. We use induction on n. Since for all $(s,a) \in \mathcal{S} \times \mathcal{A}$, $L_0(s,a) \leq U_0(s,a)$ and $-\rho \leq Q'_0(s,a) \leq \rho$ then $L_1(s,a) \leq U_1(s,a)$ by Lemma A.1.1. Suppose that $L_n(s,a) \leq U_n(s,a)$ holds for all (s,a) for all $n = 1, \ldots, k$. For all $(s,a) \in \mathcal{S} \times \mathcal{A}$, we have $Q'_k(s,a)$ is bounded since by Proposition 2.3.2(i) $Q'_n(s,a)$ is bounded for all n. We also have $L_k(s,a) \leq U_k(s,a)$ for all (s,a) by the induction assumption. Applying Lemma A.1.1 again at $n = k+1$ yields $L_{k+1}(s,a) \leq U_{k+1}(s,a)$ and the inductive proof is complete. □

A.1.4 Proof of Theorem 2.3.1

Proof. We first prove part (i). We start by writing Algorithm 1 using DP operator notation. Define a mapping H such that

$$(HQ')(s,a) = r(s,a) + \gamma \mathbf{E}\left[\max_{a'} Q'(s',a')\right],$$

where $s' = f(s,a,w)$. It is well-known that the mapping H is a γ-contraction in the maximum norm. We also define a random noise term

$$\xi_n(s,a) = \gamma \max_{a'} Q'_n(s',a') - \gamma \mathbf{E}\left[\max_{a'} Q'_n(s',a')\right]. \tag{A.20}$$

The main update rules of Algorithm 1 can then be written as

$$
\begin{aligned}
Q_{n+1}(s,a) &= (1-\alpha_n(s,a))\, Q'_n(s,a) + \alpha_n(s,a)\, \left[(HQ'_n)(s,a) + \xi_{n+1}(s,a)\right], \\
U_{n+1}(s,a) &= \Pi_{[-\rho,\infty]} \left[(1-\beta_n(s,a))\, U_n(s,a) + \beta_n(s,a)\, \hat{Q}_0^U(s,a)\right], \\
L_{n+1}(s,a) &= \Pi_{[\infty,\rho]} \left[(1-\beta_n(s,a))\, L_n(s,a) + \beta_n(s,a)\, \hat{Q}_0^L(s,a)\}\right], \\
Q'_{n+1}(s,a) &= \Pi_{[L_{n+1}(s,a),\, U_{n+1}(s,a)]} \left[Q_{n+1}(s,a)\right]. \qquad \text{(A.21)}
\end{aligned}
$$

Assume without loss of generality that $Q^*(s,a) = 0$ for all state-action pairs (s,a). This can be established by shifting the origin of the coordinate system. Note that by (A.21) at any iteration n and for all (s,a), we have $L_n(s,a) \leq Q'_n(s,a) \leq U_n(s,a)$.

We proceed via induction. First, note that by Propostion 2.3.2(i) the iterates of Algorithm 1 $Q'_n(s,a)$ are bounded in the sense that there exists a constant D_0 such that $|Q'_n(s,a)| \leq D_0$ for all (s,a) and iterations n. Define the sequence $D_{k+1} = (\gamma + \epsilon)\, D_k$, such that $\gamma + \epsilon < 1$ and $\epsilon > 0$. Clearly, $D_k \to 0$. Suppose that there exists some time n_k such that for all (s,a),

$$\max\{-D_k, L_n(s,a)\} \leq Q'_n(s,a) \leq \min\{D_k, U_n(s,a)\}, \quad \forall n \geq n_k.$$

We will show that this implies the existence of some time n_{k+1} such that

$$\max\{-D_{k+1}, L_n(s,a)\} \leq Q'_n(s,a) \leq \min\{D_{k+1}, U_n(s,a)\} \quad \forall\, (s,a),\, n \geq n_{k+1}.$$

This implies that $Q'_n(s,a)$ converges to $Q^*(s,a) = 0$ for all (s,a). We also assume that $\alpha_n(s,a) \leq 1$ for all (s,a) and n. Define an accumulated noise process started at n_k by $W_{n_k,n_k}(s,a) = 0$, and

$$W_{n+1,n_k}(s,a) = (1-\alpha_n(s,a))\, W_{n,n_k}(s,a) + \alpha_n(s,a)\, \xi_{n+1}(s,a), \quad \forall\, n \geq n_k, \qquad \text{(A.22)}$$

where $\xi_n(s,a)$ is as defined in (A.20). We now use Corollary 4.1 in [9] which states that under the assumptions of Theorem 2.3.1 on the step size $\alpha_n(s,a)$, and if $\mathbf{E}[\xi_n(s,a) \,|\, \mathcal{F}_n] = 0$ and $\mathbf{E}[\xi_n^2(s,a) \,|\, \mathcal{F}_n] \leq A_n$, where the random variable A_n is bounded with probability 1, the

sequence $W_{n+1,n_k}(s,a)$ defined in (A.22) converges to zero, with probability 1. From our definition of the stochastic approximation noise $\xi_n(s,a)$ in (A.20), we have

$$\mathbf{E}[\xi_n(s,a)\,|\,\mathcal{F}_n] = 0 \quad \text{and} \quad \mathbf{E}[\xi_n^2(s,a)\,|\,\mathcal{F}_n] \leq C(1+\max_{s',a'} Q_n^{'2}(s',a')),$$

where C is a constant. Then, it follows that

$$\lim_{n\to\infty} W_{n,n_k}(s,a) = 0 \quad \forall\,(s,a),\ n_k.$$

Now, for the sake of completeness, we restate a lemma from [9] below, which we will use to bound the accumulated noise.

Lemma A.1.2 (Lemma 4.2 in [9])**.** *For every $\delta > 0$, and with probability one, there exists some n' such that $|W_{n,n'}(s,a)| \leq \delta$, for all $n \geq n'$.*

Using the above lemma, let $n_{k'} \geq n_k$ such that, for all $n \geq n_{k'}$ we have

$$|W_{n,n_{k'}}(s,a)| \leq \gamma\epsilon D_k < \gamma D_k.$$

Furthermore, by Proposition 2.3.2(ii) let $\nu_k \geq n_{k'}$ such that, for all $n \geq \nu_k$ we have

$$L_n(s,a) \leq \gamma D_k - \gamma\epsilon D_k \quad \text{and} \quad \gamma\epsilon D_k - \gamma D_k \leq U_n(s,a).$$

Define another sequence Y_n that starts at iteration ν_k.

$$Y_{\nu_k}(s,a) = D_k \quad \text{and} \quad Y_{n+1}(s,a) = (1-\alpha_n(s,a))\,Y_n(s,a) + \alpha_n(s,a)\,\gamma\,D_k \tag{A.23}$$

Note that it is easy to show that the sequence $Y_n(s,a)$ in (A.23) is decreasing, bounded below by γD_k, and converges to γD_k as $n \to \infty$. Now we state the following lemma.

Lemma A.1.3. *For all state-action pairs (s,a) and iterations $n \geq \nu_k$, it holds that:*

(1) $Q'_n(s,a) \leq \min\{U_n(s,a), Y_n(s,a) + W_{n,\nu_k}(s,a)\}$,

(2) $\max\{L_n(s,a), -Y_n(s,a) + W_{n,\nu_k}(s,a)\} \leq Q'_n(s,a)$.

Proof. We focus on part (1). For the base case $n = \nu_k$, the statement holds because $Y_{\nu_k}(s,a) = D_k$ and $W_{\nu_k,\nu_k}(s,a) = 0$. We assume it is true for n and show that it continues to hold for $n+1$:

$$\begin{aligned}
Q_{n+1}(s,a) &= (1-\alpha_n(s,a))Q'_n(s,a) + \alpha_n(s,a)\,\left[(HQ'_n)(s,a) + \xi_{n+1}(s,a)\right] \\
&\leq (1-\alpha_n(s,a))\min\{U_n(s,a), Y_n(s,a) + W_{n,\nu_k}(s,a)\} \\
&\qquad\qquad + \alpha_n(s,a)\,(HQ'_n)(s,a) + \alpha_n(s,a)\,\xi_{n+1}(s,a) \\
&\leq (1-\alpha_n(s,a))\,(Y_n(s,a) + W_{n,\nu_k}(s,a)) + \alpha_n(s,a)\,\gamma D_k + \alpha_n(s,a)\,\xi_{n+1}(s,a) \\
&\leq Y_{n+1}(s,a) + W_{n+1,\nu_k}(s,a),
\end{aligned}$$

where we used $(HQ'_n) \leq \gamma\|Q'_n\| \leq \gamma D_k$. Now, we have

$$\begin{aligned}
Q'_{n+1}(s,a) &= \Pi_{[L_{n+1}(s,a),\,U_{n+1}(s,a)]}\,[Q_{n+1}(s,a)] \\
&\leq \Pi_{[L_{n+1}(s,a),\,U_{n+1}(s,a)]}\,[Y_{n+1}(s,a) + W_{n+1,\nu_k}(s,a)] \\
&\leq \min\{U_{n+1}(s,a), Y_{n+1}(s,a) + W_{n+1,\nu_k}(s,a)\}.
\end{aligned}$$

The first inequality holds because

$$Q_{n+1}(s,a) \leq Y_{n+1}(s,a) + W_{n+1,\nu_k}(s,a).$$

The second inequality holds because $Y_{n+1}(s,a) + W_{n+1,\nu_k}(s,a) \geq \gamma D_k - \gamma\epsilon D_k$, $L_n(s,a) \leq \gamma D_k - \gamma\epsilon D_k$, and $L_n(s,a) \leq U_n(s,a)$ by Lemma 2.3.1. Symmetrically, it can be shown that

$$\max\{L_{n+1}(s,a), -Y_{n+1}(s,a) + W_{n+1,\nu_k}(s,a)\} \leq Q'_{n+1}(s,a),$$

which completes the proof. □

Since $Y_n(s,a) \to \gamma D_k$ and $W_{n,\nu_k}(s,a) \to 0$, we have

$$\limsup_{n\to\infty} \|Q'_n\| \leq \gamma D_k < D_{k+1}.$$

Therefore, there exists some time n_{k+1} such that

$$\max\{-D_{k+1}, L_n(s,a)\} \leq Q'_n(s,a) \leq \min\{D_{k+1}, U_n(s,a)\} \quad \forall\,(s,a),\ n \geq n_{k+1},$$

completing thus the induction.

For part (ii) of the theorem: we fix (s,a) and focus on the convergence analysis of $U_n(s,a)$ to $Q^*(s,a)$. A similar analysis can be done to show $L_n(s,a) \to Q^*(s,a)$ almost surely. First note that we can write (2.11) the update equation of $U_n(s,a)$ as:

$$U_{n+1}(s_n,a_n) = \Pi_{[-\rho,\infty]}\left[U_n(s_n,a_n) + \beta_n(s_n,a_n)\left[\psi_n(U_n(s_n,a_n), Q_{n+1}(s_n,a_n))\right]\right]$$

where $\psi_n(U_n(s_n,a_n), Q_{n+1}(s_n,a_n))$ is the stochastic gradient and in this case is equal to $\hat{Q}_0^U(s_n,a_n) - U_n(s_n,a_n)$. We define the noise terms

$$\bar{\epsilon}_{n+1}(s_n,a_n) = \psi_n(U_n(s_n,a_n), Q^*(s_n,a_n)) - \mathbf{E}[\psi_n(U_n(s_n,a_n), Q^*(s_n,a_n))] \tag{A.24}$$

$$\bar{\varepsilon}_{n+1}(s_n,a_n) = \psi_n(U_n(s_n,a_n), Q_{n+1}(s_n,a_n)) - \psi_n(U_n(s_n,a_n), Q^*(s_n,a_n)). \tag{A.25}$$

Note here that $\bar{\epsilon}_{n+1}(s_n,a_n)$ represents the error that the sample gradient deviates from its mean when computed using the optimal action-value Q^* and $\bar{\varepsilon}_{n+1}(s_n,a_n)$ is the error between the two evaluations of ψ_n due only to the difference between $Q_{n+1}(s_n,a_n)$ and $Q^*(s_n,a_n)$. Thus, we have

$$\psi_n(U_n(s_n,a_n), Q_{n+1}(s_n,a_n)) = \mathbf{E}[\psi_n(U_n(s_n,a_n), Q^*(s_n,a_n))] + \bar{\epsilon}_{n+1}(s_n,a_n) + \bar{\varepsilon}_{n+1}(s_n,a_n).$$

Since $Q_n \to Q^*$ by part (i) of the Theorem, then $\bar{\varepsilon}_n(s_n,a_n) \to 0$ almost surely. It is now convenient to view $U_n(s,a)$ as a stochastic process in n, adapted to the filtration $\{\mathcal{F}_n\}_{n\geq 0}$. By definition of $\bar{\epsilon}_{n+1}(s,a)$, we have that

$$\mathbf{E}[\bar{\epsilon}_{n+1}(s,a)|\mathcal{F}_n] = 0 \quad a.s.$$

Since $\bar{\epsilon}_{n+1}(s,a)$ is unbiased and $\bar{\varepsilon}_{n+1}(s,a)$ converges to zero, we can apply Theorem 2.4 of [43], a standard stochastic approximation convergence result, to conclude that $U_n(s,a) \to Q^*(s,a)$ almost surely. Since our choice of (s,a) was arbitrary, this convergence holds for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. □

A.1.5 Proof of Lemma 2.3.2

Proof. First note that Lemma A.1.1 still holds in this case. To see this, note that if the requirements of the lemma are satisfied (i.e., if we are at iteration $n = 1, 2, \ldots$, and in the previous iteration, we had $L_{n-1}(s,a) \leq U_{n-1}(s,a)$ for all (s,a) and $Q'_{n-1} \in \mathcal{Q}$), then Q_n is bounded using the same argument as before. Since the rewards $r(s,a)$ are uniformly bounded and τ is an almost surely finite stopping time, then $\tilde{Q}^L_{n,0}$ and $\tilde{Q}^U_{n,0}$ are finite. Moreover, since $\tilde{Q}^L_{n,0}$ and $\tilde{Q}^U_{n,0}$ are computed using the same sample path $\mathbf{w}$, it follows that

$$\tilde{Q}^U_{n,0}(s,a) - \tilde{Q}^L_{n,0}(s,a) \geq 0, \quad \text{for all } (s,a) \in \mathcal{S} \times \mathcal{A}.$$

This can be easily seen if we subtract (2.16) from (2.15). Notice that the reward and the penalty will both cancel out and we have $\tilde{Q}^U_{n,t} - \tilde{Q}^L_{n,t} \geq 0$ for all $t = 0, 1, \ldots, \tau - 1$. With $L_{n-1}(s,a) \leq U_{n-1}(s,a)$ and $\tilde{Q}^U_{n,0}(s,a) \geq \tilde{Q}^L_{n,0}(s,a)$ it follows that $L_n(s,a) \leq U_n(s,a)$ for all (s,a).

Now, we prove Proposition 2.3.2 again for the experience replay buffer case.

For part (i): our original proof still holds since Lemma A.1.1 still holds.

For part (ii): first note that since the experience replay buffer is updated with a new observation of the noise at every iteration, then by Borel's law of large numbers, we have our probability estimate $\hat{p}(w)$ for the noise converges to the true noise distribution $p(w)$ as $n \to \infty$, i.e.,

$$\lim_{n\to\infty} \hat{p}_n(w) = p(w) \text{ for all } w \in \mathcal{W}. \tag{A.26}$$

Fix an $(s,a) \in \mathcal{S} \times \mathcal{A}$. By part (i) we have the action-value iterates Q_n and Q'_n are bounded for all n. Now we write the iterate $\tilde{Q}^L_{n,0}(s,a)$ in terms of a noise term and a bias term as follows,

$$\tilde{Q}^L_{n,0}(s,a) = \underbrace{\tilde{Q}^L_{n,0}(s,a) - \mathbf{E}[\tilde{Q}^L_{n,0}(s,a)]}_{\text{noise}} + \underbrace{\mathbf{E}[\tilde{Q}^L_{n,0}(s,a)] - \mathbf{E}[Q^L_{n,0}(s,a)]}_{\text{bias}} + \mathbf{E}[Q^L_{n,0}(s,a)].$$

Now, we define the noise term using

$$\xi^L_n(s,a) = \tilde{Q}^L_{n,0}(s,a) - \mathbf{E}[\tilde{Q}^L_{n,0}(s,a)].$$

Also, similar to the original proof we define an accumulated noise process started at iteration ν by $W^L_{\nu,\nu}(s,a) = 0$, and

$$W^L_{n+1,\nu}(s,a) = (1-\alpha_n(s,a))\, W^L_{n,\nu}(s,a) + \alpha_n(s,a)\, \xi^L_{n+1}(s,a) \quad \forall\ n \geq \nu,$$

which averages noise terms together across iterations. We have $\mathbf{E}[\tilde{Q}^L_{n,0}(s,a) - \mathbf{E}[\tilde{Q}^L_{n,0}(s,a)]|\mathcal{F}_n] = 0$, so Corollary 4.1 applies and it follows that

$$\lim_{n\to\infty} W^L_{n,\nu}(s,a) = 0 \quad \forall\, \nu \geq 0.$$

Let $\tilde{\nu}$ be large enough so that $\alpha_n(s,a) \leq 1$ for all $n \geq \tilde{\nu}$. We denote the bias term by

$$\chi_n(s,a) = \mathbf{E}[\tilde{Q}^L_{n,0}(s,a)] - \mathbf{E}[Q^L_{n,0}(s,a)].$$

Since as $n \to \infty$, we have $\hat{p}_n(w) \to p(w)$, the bias due to sampling from the experience buffer $\chi_n(s,a) \to 0$. Let $\eta > 0$ and $\bar{\nu} \geq \tilde{\nu}$ be such that $|\chi(s,a)| \leq \frac{\eta}{2}$ for all $n \geq \bar{\nu}$ and all (s,a). We also define

$$\begin{aligned} Y^L_{\tilde{\nu}}(s,a) &= \rho, \\ Y^L_{n+1}(s,a) &= (1-\alpha_n(s,a))\, Y^L_n(s,a) + \alpha_n(s,a)\, Q^*(s,a) + \alpha_n(s,a)\, \frac{\eta}{2}, \quad \forall\, n \geq \bar{\nu}. \end{aligned}$$

It is easy to see that the sequence $Y^L_n(s,a) \to Q^*(s,a) + \frac{\eta}{2}$. Now we show that the following claim holds. Claim: for all iterations $n \geq \bar{\nu}$, it holds that

$$L_n(s,a) \leq \min\{\rho,\, Y^L_n(s,a) + W^L_{n,\bar{\nu}}(s,a)\}.$$

To prove this claim, we proceed by induction on n. For $n = \bar{\nu}$, we have

$$Y^L_{\bar{\nu}}(s,a) = \rho \quad \text{and} \quad W^L_{\bar{\nu},\bar{\nu}}(s,a) = 0,$$

so the statement is true for the base case. We now show that it is true for $n+1$ given that it holds at n:

$$\begin{aligned}
L_{n+1}(s,a) &= \min\{\rho,\ (1-\alpha_n(s,a))\, L_n(s,a) + \alpha_n(s,a)\, (\tilde{Q}^L_{n,0}(s,a) - \mathbf{E}[\tilde{Q}^L_{n,0}(s,a)] \\
&\qquad + \mathbf{E}[\tilde{Q}^L_{n,0}(s,a)] - \mathbf{E}[Q^L_{n,0}(s,a)] + \mathbf{E}[Q^L_{n,0}(s,a)])\} \\
&= \min\{\rho,\ (1-\alpha_n(s,a))\, L_n(s,a) + \alpha_n(s,a)\, \xi^L_n(s,a) + \alpha_n(s,a)\, \chi_n(s,a) \\
&\qquad + \alpha_n(s,a)\, \mathbf{E}[Q^L_{n,0}(s,a)]\} \\
&\leq \min\{\rho,\ (1-\alpha_n(s,a))\, (Y^L_n(s,a) + W^L_{n,\nu_k}(s,a)) + \alpha_n(s,a)\, \xi^L_n(s,a) \\
&\qquad + \alpha_n(s,a)\, \frac{\eta}{2} + \alpha_n(s,a)\, Q^*(s,a)\} \\
&\leq \min\{\rho,\ Y^L_{n+1}(s,a) + W^L_{n+1,\tilde{\nu}}(s,a)\},
\end{aligned}$$

where the first inequality follows by the induction hypothesis and $\mathbf{E}[Q^L_{n,0}(s,a)] \leq Q^*(s,a)$. Next, since $Y^L_n(s,a) \to Q^*(s,a) + \frac{\eta}{2}$, $W^L_{n,\nu_k}(s,a) \to 0$, then if $Q^*(s,a) + \frac{\eta}{2} \leq \rho$, we have

$$\limsup_{n\to\infty}\ L_n(s,a) \leq Q^*(s,a) + \frac{\eta}{2}.$$

Otherwise, if $\rho < Q^*(s,a) + \frac{\eta}{2}$, then

$$\limsup_{n\to\infty}\ L_n(s,a) \leq \rho < Q^*(s,a) + \frac{\eta}{2}.$$

Therefore, since our choice of (s,a) was arbitrary, it follows that for every $\eta > 0$, there exists some time n' such that $L_n(s,a) \leq Q^*(s,a) + \eta$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ and $n \geq n'$.

Using Proposition 2.2.1(i), $Q^*(s,a) \leq \mathbf{E}[Q^U_{n,0}(s,a)]$, a similar argument as the above can be used to establish that

$$Q^*(s,a) - \frac{\eta}{2} \leq \liminf_{n\to\infty}\ U_n(s,a).$$

Hence, there exists some time n'' such that $Q^*(s,a) - \eta \leq U_n(s,a)$ for all (s,a) and $n \geq n''$. Take n_0 to be some time greater than n' and n'' and the result follows.

The proof of Lemma 2.3.1 when using an experience buffer is similar to that given in appendix A.1.3 so it is omitted. □

A.1.6 Proof of Theorem 2.3.2

Proof. The proof of both parts (i) and (ii) are similar to that of Theorem 2.3.1, so we omit them. □

A.2 LBQL with Experience Replay Algorithm

Algorithm 4: LBQL with Experience Replay

Input: Initial estimates $L_0 \leq Q_0 \leq U_0$, batch size K, stepsize rules $\alpha_n(s,a)$, $\beta_n(s,a)$, and noise buffer $\mathcal{B}$.

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., ϵ-greedy). Observe w_{n+1}.

Store w_{n+1} in $\mathcal{B}$ and update $\hat{p}_n(w_{n+1})$.

Perform a standard Q-learning update:

$$Q_{n+1}(s_n,a_n) = Q'_n(s_n,a_n) + \alpha_n(s_n,a_n)\Big[r_n(s_n,a_n) + \gamma \max_a Q'_n(s_{n+1},a) - Q'_n(s_n,a_n)\Big].$$

Sample randomly a sample path $\mathbf{w} = (w_1, w_2, \ldots, w_\tau)$ from $\mathcal{B}$, where $\tau \sim \text{Geom}(1-\gamma)$.

Set $\varphi = Q_{n+1}$. Using $\mathbf{w}$ and the current $\hat{p}_n$ compute $\tilde{Q}_0^U(s_n,a_n)$ and $\tilde{Q}_0^L(s_n,a_n)$, using (2.15) and (2.16), respectively.;

Update and enforce upper and lower bounds:

$$U_{n+1}(s_n,a_n) = \Pi_{[-\rho,\infty]}\Big[U_n(s_n,a_n) + \beta_n(s_n,a_n)\left[\tilde{Q}_0^U(s_n,a_n) - U_n(s_n,a_n)\right]\Big],$$
$$L_{n+1}(s_n,a_n) = \Pi_{[\infty,\rho]}\Big[L_n(s_n,a_n) + \beta_n(s_n,a_n)\left[\tilde{Q}_0^L(s_n,a_n) - L_n(s_n,a_n)\right]\Big],$$

$$Q'_{n+1}(s_n,a_n) = \Pi_{[L_{n+1}(s_n,a_n),\, U_{n+1}(s_n,a_n)]}\left[Q_{n+1}(s_n,a_n)\right]$$

end for

A.3 Implementation Details of LBQL with Experience Replay

We use a noise buffer $\mathcal{B}$ of size κ to record the noise values w that have been previously observed. The buffer $\mathcal{B}$ is used to generate the sample path $\mathbf{w}$ and the batch sample $\{w_1, \ldots, w_K\}$ used to estimate the expectation in the penalty function. Here, it is not necessary that the noise space $\mathcal{W}$ is finite. This is also convenient in problems with a large noise support such as the car-sharing problem with four stations where we have two sources of noise. Specifically, the noise due to the distribution of the rentals among the stations has a very large support.

In order to reduce the computational requirements of LBQL, the lower and upper bounds updates are done every m steps and only if the difference between the current values of the bounds is greater than some threshold δ.

Since we can easily obtain inner DP results for all (s, a) each time the DP is solved, we perform the upper and lower bound updates for all (s, a) whenever an update is performed (as opposed to just at the current state-action pair). However, only the action-value of the current (s, a) is projected between the lower and upper bounds, so the algorithm is still asynchronous. The pseudo-code of LBQL with experience replay with these changes, is shown in Algorithm 5.

Algorithm 5: LBQL with Experience Replay (Full Details)

Input: Initial estimates $L_0 \leq Q_0 \leq U_0$, batch size K, stepsize rules $\alpha_n(s,a)$, $\beta_n(s,a)$, noise buffer $\mathcal{B}$ of size κ, number of steps between bound updates m, and threshold δ.

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., ϵ-greedy). Observe w_{n+1}.;

Store w_{n+1} in $\mathcal{B}$.;

Perform a standard Q-learning update:

$$Q_{n+1}(s_n, a_n) = Q'_n(s_n, a_n) + \alpha_n(s_n, a_n)\Big[r_n(s_n, a_n) + \gamma \max_a Q'_n(s_{n+1}, a) - Q'_n(s_n, a_n)\Big].$$

if $n \geq \kappa$ and n **mod** $m = 0$ and $|U_n(s_n, a_n) - L(s_n, a_n)| > \delta$ **then**

Sample randomly a batch $\mathcal{D} = \{w_1, w_2, \ldots, w_K\}$ and a sample path $\mathbf{w} = \{w_1, w_2, \ldots, w_\tau\}$ from $\mathcal{B}$, where $\tau \sim \text{Geom}(1-\gamma)$.;

Set $\varphi = Q_{n+1}$. Using $\mathbf{w}$ and $\mathcal{D}$, compute $\hat{Q}_0^U(s,a)$ and $\hat{Q}_0^L(s,a)$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$, using (2.9) and (2.10), respectively.;

For all $(s,a) \in \mathcal{S} \times \mathcal{A}$, update upper and lower bounds:

$$U_{n+1}(s_n, a_n) = \Pi_{[-\rho, \infty]}\left[U_n(s_n, a_n) + \beta_n(s_n, a_n)\left[\hat{Q}_0^U(s_n, a_n) - U_n(s_n, a_n)\right]\right],$$
$$L_{n+1}(s_n, a_n) = \Pi_{[\infty, \rho]}\left[L_n(s_n, a_n) + \beta_n(s_n, a_n)\left[\hat{Q}_0^L(s_n, a_n) - L_n(s_n, a_n)\right]\right],$$

end if

Enforce upper and lower bounds:

$$Q'_{n+1}(s_n, a_n) = \Pi_{[L_{n+1}(s_n,a_n), U_{n+1}(s_n,a_n)]}\left[Q_{n+1}(s_n, a_n)\right]$$

end for

A.4 Numerical Experiments Details

Let $\nu(s,a)$ and $\nu(s)$ be the number of times state-action pair (s,a) and state s, have been visited, respectively. For all algorithms, a polynomial learning rate $\alpha_n(s,a) = 1/\nu_n(s,a)^r$ is used, where $r = 0.5$. Polynomial learning rates have been shown to have a better performance than linear learning rates [70].

We use a discount factor of $\gamma = 0.95$ for the pricing car-sharing/stormy gridworld problems, $\gamma = 0.9$ for the windy gridworld problem and $\gamma = 0.99$ for the repositioning problem. Moreover, we use an ϵ-greedy exploration strategy such that $\epsilon(s) = 1/\nu(s)^e$, where e is 0.4 for the four-stations pricing car-sharing problem and 0.5 for all the other problems. For the car-sharing/windy gridworld problems, the initial state-action values are chosen randomly such that $L_0(s,a) \leq Q_0(s,a) \leq U_0(s,a)$ where

$$L_0(s,a) = -R_{\max}/(1-\gamma) \quad \text{and} \quad U_0(s,a) = R_{\max}/(1-\gamma)$$

for all (s,a). For the stormy gridworld problem, we set the initial state-action values to zero (we find that a random initialization caused all algorithms except LBQL to perform extremely poorly).

We report LBQL parameters used in our numerical experiments in Table 2. Note that for a fair comparison, the parameter K of bias-corrected Q-learning algorithm is taken equal to K of LBQL in all experiments. In addition, the κ steps used to create the buffer for LBQL are included in the total number of steps taken. Results of the gridworld and car-sharing problems are averaged over 50 and 10 runs, respectively. All experiments were run on a 3.5 GHz Intel Xeon processor with 32 GB of RAM workstation.

Table 2: LBQL parameters.

	Parameter				
Problem	β	κ	K	m	δ
2-CS-R	0.01	40	20	10	0.01
2-CS	0.01	40	20	15	0.01
4-CS	0.01	1000	20	200	0.01
WG	0.2	100	10	10	0.01
SG	0.2	500	20	20	0.05

A detailed description of the environments is given in the next two sections.

A.4.1 Gridworld Examples

First we consider, windy gridworld, a well-known variant of the standard gridworld problem discussed in [63]. Then we introduce, stormy gridworld, a new environment that is more complicated than windy gridworld. The environments are summarized below.

Windy Gridworld. The environment is a 10×7 gridworld, with a cross wind pointing *upward*, [63]. The default *wind* values corresponding to each of the 10 columns are $\{0, 0, 0, 1, 1, 1, 2, 2, 1, 0\}$. Allowable actions are {up, right, down, left}. If the agent happens to be in a column whose wind value is different from zero, the resulting next states are shifted upward by the "wind" whose intensity is stochastic, varying from the given values in each column by $\{-1, 0, 1\}$ with equal probability. Actions that corresponds to directions that takes the agent off the grid leave the location of the agent unchanged. The start and goal states are $(3, 1)$ and $(3, 8)$, respectively. The reward is -1 until the goal state is reached, where the reward is 0 thereafter.

Stormy Gridworld. Consider the stochastic windy gridworld environment. Now, however, we allow the wind to blow half the time as before and the other half it can blow from any of the three other directions. The horizontal wind values corresponding to each row

from top to bottom are given by $\{0, 0, 1, 1, 1, 1, 0\}$. Also, it can randomly rain with equal probability in any of the central states that are more than two states away from the edges of the grid. The start and goal states are $(3, 1)$ and $(3, 10)$ respectively. Rain creates a puddle which affects the state itself and all of its neighboring states. The reward is as before except when the agent enters a puddle state the reward is -10.

A.4.2 Car-sharing Benchmark Examples

In this section, we give the detailed formulations of the two variants of the car-sharing benchmark, repositioning and pricing. The essential difference is that in the pricing version, the decision maker "repositions" by setting prices to induce directional demand (but does not have full control since this demand is random).

A.4.2.1 Repositioning Benchmark for Car-sharing

We consider the problem of repositioning cars for a two stations car-sharing platform, [31]. The action is the number of cars to be repositioned from one station to the other, before random demand is realized. Since repositioning in both directions is never optimal, we use $r > 0$ to denote the repositioned vehicles from station 1 to 2 and $r < 0$ to denote repositioning from station 2 to 1. The stochastic demands at time t are $D_{1,t}$ and $D_{2,t}$ for stations 1 and 2 respectively, are i.i.d., discrete uniform, each supported on $\{3, \ldots, 9\}$. The rental prices are $p_1 = 3.5$ and $p_2 = 4$ for stations 1 and 2, respectively. All rentals are one-way (i.e., rentals from station 1 end up at station 2, and vice-versa). The goal is to maximize profit, where unmet demands are charged a lost sales cost $\rho_1 = \rho_2 = 2$ and repositioning cost $c_1 = 1$ for cars reposition from station 1 to 2 and $c_2 = 1.5$ for cars repositioned from station 2 to 1. We assume a total of $\bar{s} = 12$ cars in the system and formulate the problem as an MDP, with state $s_t \in \mathcal{S} = \{0, 1, \ldots, 12\}$ representing the number of cars at station 1 at beginning of period t. We denote by $V^*(s_t)$ the optimal value function starting from state

s_t. The Bellman recursion is:

$$
\begin{aligned}
V^*(s_t) = \max_{s_t - \bar{s} \le r_t \le s_t} \mathbf{E}\Bigg[& \sum_{i \in \{1,2\}} p_i\, \omega_{it}(D_{i,t+1}) - \sum_{i \in \{1,2\}} \rho_i \Big(D_{i,t+1} - \omega_{it}(D_{i,t+1}) \Big) \\
& - c_1 \max(r_t, 0) + c_2 \min(r_t, 0) + \gamma V^*(s_{t+1}) \Bigg], \\
\omega_{1t}(D_{1,t+1}) &= \min(D_{1,t+1}, s_t - r_t), \\
\omega_{2t}(D_{2,t+1}) &= \min(D_{2,t+1}, \bar{s} - s_t + r_t), \\
s_{t+1} &= s_t - r_t + \omega_{2t}(D_{2,t+1}) - \omega_{1t}(D_{1,t+1}),
\end{aligned}
\tag{A.27}
$$

where $\gamma \in (0, 1)$ is a discount factor. The repositioning problem for two stations is illustrated in Figure 14a. The nodes represent stations, solid arcs represent fulfilled demands, and dashed arcs represent repositioned vehicles.

A.4.2.2 Pricing Benchmark for Car-sharing

Suppose that a vehicle sharing manager is responsible for setting the rental price for the vehicles at the beginning of each period in an infinite planning horizon. We model a car sharing system with N stations. The goal is to optimize the prices to set for renting a car at each of the N stations; let the price at station i and time t be p_{it} for $i \in [N] := \{1, 2, \ldots, N\}$. Demands are nonnegative, independent and depends on the vehicle renting price according to a stochastic demand function

$$D_{it}(p_{it}, \epsilon_{i,t+1}) := \kappa_i(p_{it}) + \epsilon_{i,t+1},$$

where $D_{it}(p_{it}, \epsilon_{i,t+1})$ is the demand in period t, $\epsilon_{i,t+1}$ are random perturbations that are revealed at time $t + 1$ and $\kappa_i(p_{it})$ is a deterministic demand function of the price p_{it} that is set at the beginning of period t at station $i \in [N]$. The random variables $\epsilon_{i,t+1}$ are independent with $\mathbf{E}[\epsilon_{i,t+1}] = 0$ without loss of generality. Furthermore, we assume that the expected demand $\mathbf{E}[D_{it}(p_{it}, \epsilon_{i,t+1})] = \kappa_i(p_{it}) < \infty$ is strictly decreasing in the rental price p_{it} which is restricted to a set of feasible price levels $[\underline{p}_i, \overline{p}_i]$ for all $i \in [N]$, where $\underline{p}_i, \overline{p}_i$ are the minimum and the maximum prices that can be set at station i, respectively. This assumption

implies a one-to-one correspondence between the rental price p_{it} and the expected demand $d_{it} \in \mathfrak{D} := [\underline{d}_i, \overline{d}_i]$ for all $p_{it} \in [\underline{p}_i, \overline{p}_i]$ where $\underline{d}_i = \kappa_i(\overline{p}_i)$ and $\overline{d}_i = \kappa_i(\underline{p}_i)$.

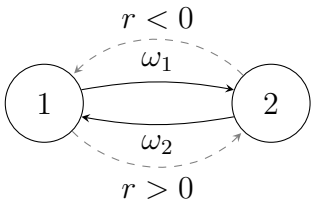
The problem can be formulated as an MDP with state $\mathbf{s}_t$, which is a vector whose components represent the number of available cars at each of the N stations at beginning of period t. The state space is $\mathcal{S}^{N-1}$ with $\mathcal{S} = \{0, 1, \ldots, \bar{s}\}$ and $\bar{s}$ is the maximum number of cars in the vehicle sharing system. We assume that a customer at station i goes to station j with probability ϕ_{ij} for all $i, j \in [N]$. Let $Y_{ik,t+1}$ be a random variable taking values in $[N]$ that represents the random destination of customer k at station i, which is only observed at the beginning of period $t+1$. We have $Y_{ik,t+1} = j$ with probability ϕ_{ij}, so $Y_{ik,t+1}$ are i.i.d. for each customer k. Denote by l_{ij} the distance from station i to j, for all $i, j \in [N]$. We penalize unmet demands by a lost sales unit cost ρ_i, $i \in [N]$. The decision vector is $\mathbf{p}_t = \{p_{it} \in [\underline{p}_i, \overline{p}_i], \forall i \in [N]\}$. Let $V^*(\mathbf{s}_t)$ be the revenue-to-go function with number of available vehicles $\mathbf{s}_t$. Thus, we have the Bellman recursion

$$
\begin{aligned}
V^*(\mathbf{s}_t) = \max_{\mathbf{p}_t} \; & \mathbf{E}\Big[\sum_{i\in[N]} p_{it} \sum_{j\in[N]} l_{ij}\, \omega_{ijt}(\epsilon_{i,t+1}) - \sum_{i\in[N]} \rho_i \Big(\kappa_i(p_{it}) + \epsilon_{i,t+1} - \omega_{it}(\epsilon_{i,t+1}) \Big) + \gamma V^*(\mathbf{s}_{t+1}) \Big] \\
& \omega_{it}(\epsilon_{i,t+1}) = \min\left(\kappa_i(p_{it}) + \epsilon_{i,t+1}, s_{it}\right) \quad \forall\, i \in [N], \\
& \omega_{ijt}(\epsilon_{i,t+1}) = \sum_{k=1}^{\omega_{it}(\epsilon_{i,t+1})} \mathbb{1}_{\{Y_{ik,t+1}=j\}} \quad \forall\, i, j \in [N], \\
& s_{i,t+1} = s_{it} + \sum_{j\in[N]} \omega_{jit}(\epsilon_{i,t+1}) - \omega_{it}(\epsilon_{i,t+1}), \quad \forall\, i \in [N],
\end{aligned}
\tag{A.28}
$$

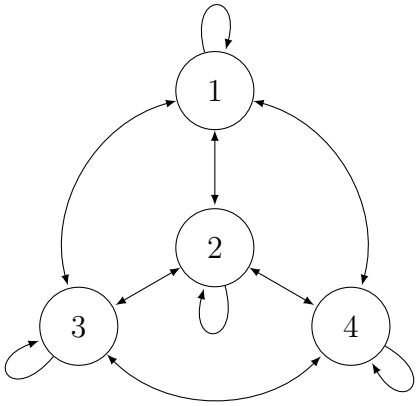
where $\gamma \in (0, 1)$ is a discount factor. Note that the MDP in (A.28) can be reformulated using the action-value function $Q(\mathbf{s}_t, p_t)$ instead of $V(\mathbf{s}_t)$. The quantity $\omega_{it}(\epsilon_{i,t+1})$ represents the total fulfilled customer trips from station i at time t for a given realization of the noise $\epsilon_{i,t+1}$. Notice that in (A.28) there are two sources of randomness: the noise due to stochastic demand represented by ϵ_i, for all $i \in [N]$ and the noise due to the random distribution of fulfilled rentals between the stations, i.e., due to the random variables $Y_{i1}, \ldots, Y_{i\omega_{it}(\epsilon_{i,t+1})}$ for all $i \in [N]$. Due to the high dimensionality involved in the state, action, and noise spaces, solving (A.28) is computationally challenging.

Spatial Pricing in Two-Location Car-sharing. We first consider the pricing problem on two stations and 12 cars in total. The state space is $\mathcal{S} = \{0, 1, \ldots, 12\}$ representing the number of cars at station 1. All rentals are one-way. The prices, at each period t, are restricted to $p_{1t} \in [1, 6]$ and $p_{2t} \in [1, 7]$. The stochastic demand functions at period t are given by: $D_{1t}(p_{1t}, \epsilon_{1,t+1}) := 9 - p_{1t} + \epsilon_{1,t+1}$ and $D_{2t}(p_{2t}, \epsilon_{2,t+1}) := 10 - p_{2t} + \epsilon_{2,t+1}$ for stations 1 and 2 respectively. The random variables $\epsilon_{1,t+1}$ and $\epsilon_{2,t+1}$ are independent, discrete uniform, each supported on $\{-3, -2, \ldots, 3\}$. We use the discretized expected demands, as our actions: $d_{1t} \in \{3, \ldots, 8\}$ and $d_{2t} \in \{3, \ldots, 9\}$. The lost sales cost is 2 at both stations.

Spatial Pricing in Four-Location Car-sharing. Consider the car-sharing problem for four stations with $\bar{s} = 20$ cars and $d_{it} \in \{3, 4\}$ for each station. In total there are 1771 states and 16 actions. The random variables $\epsilon_{i,t+1}$ are independent, discrete uniform, each supported on $\{-3, -2, \ldots, 3\}$. We consider both one way and return trips at each station. Figure 14b shows an illustration of the stations (nodes) and the rentals between the stations (arcs). The probabilities $\phi_{ij} = 0.25$ for all $i, j \in \{1, 2, 3, 4\}$ and the lost sales costs (ρ_i) are $1.7, 1.2, 1.5, 2$ at stations $1, 2, 3, 4$, respectively. The distance between the stations are taken such that $l_{ij} = 1$ if $i = j$, and the other distances being symmetrical, meaning $l_{ij} = l_{ji}$ with $l_{12} = 1.8$, $l_{13} = 1.5$, $l_{14} = 1.4$, $l_{23} = 1.6$, $l_{24} = 1.1$, and $l_{34} = 1.2$.



(a) Repositioning problem with 2 stations.



(b) Pricing problem with 4 stations.

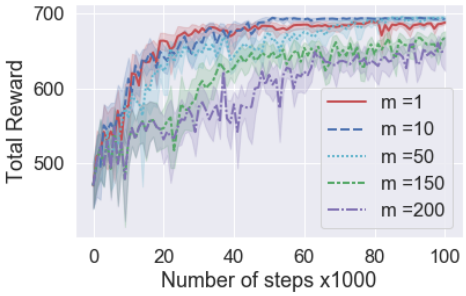
Figure 14: Illustrations of the repositioning and pricing car-sharing problems.

A.4.3 Sensitivity Analysis

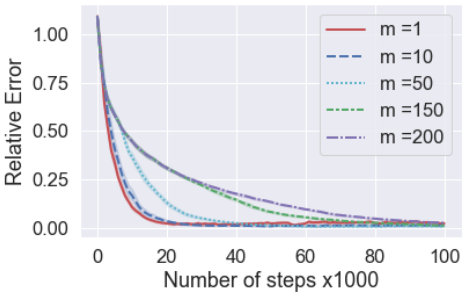
We also perform sensitivity analysis on the five algorithms with respect to the learning rate and exploration parameters r and e for the car-sharing problem with two stations. Here, r controls the polynomial learning rate defined by, $\alpha_n(s,a) = 1/\nu_n(s,a)^r$ and e controls the ϵ-greedy exploration strategy, where ϵ is annealed according to $\epsilon(s) = 1/\nu(s)^e$. We use $\nu(s,a)$ and $\nu(s)$ to denote the number of times a state-action pair (s,a) and state s, have been visited, respectively. We report our results in Table 3. These results show the average number of iterations and CPU time until each algorithm first reach 50%, 20%, 5%, 1% relative error for each case of the parameters e and r while keeping all other parameters as before. The "-" indicates that the corresponding % relative error for the corresponding case was not achieved during the course of training. The values in the table are obtained by averaging five independent runs for each case. Except for the few cases where BCQL performs slightly better, LBQL once again drastically outperforms the other algorithms and exhibits *robustness* against the learning rate and exploration parameters, an important practical property that the other algorithms seem to lack.

The effect of varying parameters m and K of LBQL is presented in Figure 15. These plots are obtained by tuning parameters $m \in \{1, 10, 50, 150, 200\}$ and $K \in \{1, 5, 10, 100, 1000\}$ of LBQL algorithm in the car-sharing problem with two stations. All other parameters are kept the same as before. Figures 15a and 15c show the mean total reward with a 95% CI. Figures 15b and 15d show the mean and 95% CI of the relative error given by: $\|V_n - V^*\|_2/\|V^*\|_2$. The results are obtained from 10 independent runs. Using larger values of m reduces the strength of LBQL in both the performance and relative error metrics, as shown in Figures 15a and 15b. This is expected since the effect of the bounds fades as we update the bounds less frequently. Interestingly, we can see from the performance plot that $m = 10$ strikes a good balance between how often to do the bounds and Q-learning updates and achieves a performance that is slightly better and more stable than that of $m = 1$ after about half of the training process (50,000 steps). In terms of the sample size K, Figures 15c and 15d clearly show that larger values of K improve the performance of LBQL in terms of performance and relative error measures. This is not unexpected because a larger sample yields a better

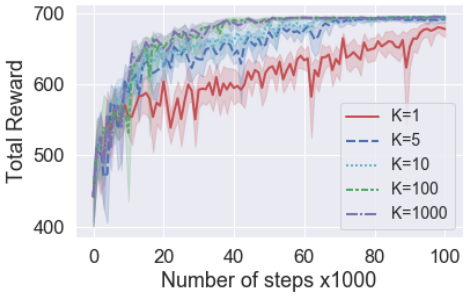
approximation of the penalty.



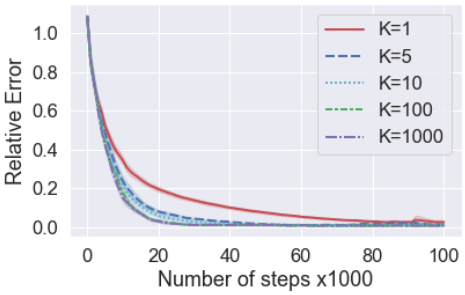
(a) Performance (2-CS)



(b) Relative Error (2-CS)



(c) Performance (2-CS)



(d) Relative Error (2-CS)

Figure 15: Plots showing the effect of tuning the parameters m and K of LBQL algorithm.

Table 3: Computational results for different exploration & learning rate parameters. Bold numbers indicate the best performing algorithm.

	e	r	% Relative error							
			50%		20%		5%		1%	
			n	t (s)	n	t (s)	n	t (s)	n	t (s)
LBQL	0.4	0.5	3,672.6	1.9	**9,323.6**	**4.6**	**18,456.6**	**8.9**	**33,054.0**	**15.7**
		0.6	**3,632.0**	1.7	**9,147.4**	**4.4**	**18,270.0**	**8.6**	**39,624.8**	**18.6**
		0.7	**3,725.2**	**1.8**	**9,087.4**	**4.3**	**18,217.8**	**8.6**	**41,941.2**	**19.7**
		0.8	**3,698.2**	**1.8**	**9,321.8**	**4.4**	**20,860.0**	**9.9**	**53,752.6**	**25.6**
		0.9	**3,992.2**	**1.9**	**10,119.2**	**4.9**	**23,070.2**	**11.0**	**80,252.8**	**38.4**
	0.5	0.5	3,316.0	1.6	**8,040.2**	**3.8**	**15,050.2**	**7.2**	**27,912.8**	**13.2**
		0.6	3,514.4	1.7	**8,529.4**	**4.0**	**16,595.6**	**7.9**	**36,100.2**	**17.0**
		0.7	**3,531.8**	**1.7**	**8,712.8**	**4.1**	**17,835.6**	**8.6**	**46,010.0**	**21.9**
		0.8	**3,449.2**	**1.6**	**8,571.8**	**4.0**	**18,152.6**	**8.5**	**65,007.6**	**30.4**
		0.9	**3,398.4**	**1.6**	**8,346.4**	**3.9**	**18,844.8**	**8.8**	**99,820.2**	**46.6**
	0.6	0.5	2,877.4	1.3	**7,129.0**	**3.3**	**13,046.0**	**6.2**	**23,822.0**	**11.2**
		0.6	3,182.8	1.5	**8,066.0**	**3.8**	**15,421.4**	**7.3**	**33,286.0**	**15.6**
		0.7	**2,979.4**	1.4	**7,625.6**	**3.6**	**15,414.6**	**7.2**	**34,238.0**	**15.9**
		0.8	**3,272.6**	**1.6**	**8,431.0**	**4.1**	**17,809.2**	**8.5**	**114,032.8**	**54.2**
		0.9	**3,185.6**	**1.5**	**8,480.0**	**4.1**	**19,242.4**	**9.2**	**123,331.2**	**58.8**
	0.4	0.5	**3,200.8**	**1.0**	22,455.0	7.0	65,329.0	20.3	107,785.6	33.7
		0.6	4,618.2	**1.5**	43,724.6	13.6	159,662.6	49.6	292,421.0	34.9

(continued on next page)

Table 3: *(continued)*

e	r	% Relative error							
		50%		20%		5%		1%	
		n	t (s)	n	t (s)	n	t (s)	n	t (s)
	0.7	8,059.4	2.5	123,484.2	38.4	-	-	-	-
	0.8	17,287.0	5.3	-	-	-	-	-	-
	0.9	67,162.2	20.9	-	-	-	-	-	-
0.5	0.5	**2,209.6**	**0.7**	15,604.0	4.9	48,715.6	15.3	80,317.4	25.2
	0.6	**3,274.2**	**1.0**	31,422.8	9.8	124,319.6	38.7	243,101.4	75.6
	0.7	5,619.6	1.8	89,857.0	27.8	-	-	-	-
	0.8	11,417.0	3.6	-	-	-	-	-	-
	0.9	42,605.4	13.1	-	-	-	-	-	-
0.6	0.5	**1,830.4**	**0.6**	11,639.6	3.6	35,763.0	11.1	61,249.0	19.0
	0.6	**2,612.4**	**0.8**	23,571.6	7.4	92,101.4	28.8	177,127.6	55.5
	0.7	4,371.2	**1.3**	66,526.0	20.5	-	-	-	-
	0.8	9,028.2	2.8	297,368.6	17.9	-	-	-	-
	0.9	31,673.6	9.8	-	-	-	-	-	-
0.4	0.5	7,750.6	1.9	37,889.8	9.1	93,820.0	22.5	141,171.0	33.8
	0.6	11,329.4	2.8	75,364.0	18.3	233,422.0	56.4	-	-
	0.7	20,131.4	4.8	212,767.0	51.0	-	-	-	-
	0.8	46,986.8	11.3	-	-	-	-	-	-
	0.9	182,890.0	43.8	-	-	-	-	-	-

(continued on next page)

Table 3: *(continued)*

e	r	% Relative error							
		50%		20%		5%		1%	
		n	t (s)	n	t (s)	n	t (s)	n	t (s)
0.5	0.5	6,122.2	1.5	30,944.8	7.4	79,167.4	19.0	120,527.8	29.0
	0.6	9,166.6	2.2	62,540.6	14.9	201,822.4	48.2	-	-
	0.7	15,835.6	3.8	174,233.6	42.0	-	-	-	-
	0.8	36,548.8	8.7	-	-	-	-	-	-
	0.9	157,029.0	37.7	-	-	-	-	-	-
0.6	0.5	4,984.0	1.2	24,989.0	6.0	64,605.4	15.4	98,554.6	23.6
	0.6	7,396.2	1.8	50,282.4	12.0	165,574.2	39.8	-	-
	0.7	13,018.8	3.1	143,142.6	34.1	-	-	-	-
	0.8	29,201.0	6.9	-	-	-	-	-	-
	0.9	122,335.6	29.2	-	-	-	-	-	-
0.4	0.5	7,743.0	1.7	38,114.2	8.2	93,303.4	20.2	136,851.4	29.6
	0.6	11,644.0	2.5	76,625.0	16.6	232,679.4	50.9	-	-
	0.7	20,181.6	4.4	212,401.4	46.3	-	-	-	-
	0.8	45,987.2	10.1	-	-	-	-	-	-
	0.9	191,442.2	41.9	-	-	-	-	-	-
0.5	0.5	6,143.6	1.3	30,996.2	6.8	78,131.8	16.9	116,361.2	25.3
	0.6	9,331.6	2.0	63,998.2	13.9	204,593.6	44.6	-	-

(continued on next page)

TQ

Table 3: *(continued)*

	e	r	% Relative error							
			50%		20%		5%		1%	
			n	t (s)	n	t (s)	n	t (s)	n	t (s)
		0.7	16,247.0	3.5	178,842.8	38.4	-	-	-	-
		0.8	38,297.0	8.2	-	-	-	-	-	-
		0.9	165,835.8	35.7	-	-	-	-	-	-
	0.6	0.5	5,005.2	1.1	24,877.2	5.4	63,777.8	13.7	96,402.0	20.8
		0.6	7,547.0	1.9	51,369.4	13.1	166,179.2	42.3	289,882.8	46.1
		0.7	13,288.2	3.1	144,318.2	33.1	-	-	-	-
		0.8	30,172.6	6.5	-	-	-	-	-	-
		0.9	139,952.6	30.3	-	-	-	-	-	-
	0.4	0.5	224,490.2	51.2	-	-	-	-	-	-
		0.6	-	-	-	-	-	-	-	-
		0.7	-	-	-	-	-	-	-	-
		0.8	-	-	-	-	-	-	-	-
		0.9	-	-	-	-	-	-	-	-
	0.5	0.5	-	-	-	-	-	-	-	-
		0.6	-	-	-	-	-	-	-	-
		0.7	-	-	-	-	-	-	-	-
		0.8	-	-	-	-	-	-	-	-
		0.9	-	-	-	-	-	-	-	-
Double-QL										

(continued on next page)

Table 3: *(continued)*

e	r	% Relative error							
		50%		20%		5%		1%	
		n	t (s)	n	t (s)	n	t (s)	n	t (s)
0.6	0.5	-	-	-	-	-	-	-	-
	0.6	-	-	-	-	-	-	-	-
	0.7	-	-	-	-	-	-	-	-
	0.8	-	-	-	-	-	-	-	-
	0.9	-	-	-	-	-	-	-	-

Appendix B

B.1 Proofs for Chapter 4

B.1.1 Proof of Proposition 3.2.1

Proof. We prove part (a) by induction. First, define

$$Q_0^*(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) \quad \text{and} \quad Q_0^{\boldsymbol{\lambda}}(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \boldsymbol{\lambda}^T \left[\mathbf{b}(w) - \textstyle\sum_{i=1}^N \mathbf{d}_i(s_i,a_i)\right],$$

and suppose we run value iteration for both systems:

$$\begin{aligned} Q_{t+1}^*(\mathbf{s},\mathbf{a}) &= r(\mathbf{s},\mathbf{a}) + \gamma\, \mathbf{E}\big[\max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} Q_t^*(\mathbf{s}',\mathbf{a}')\big], \\ Q_{t+1}^{\boldsymbol{\lambda}}(\mathbf{s},\mathbf{a}) &= r(\mathbf{s},\mathbf{a}) + \boldsymbol{\lambda}^T \left[\mathbf{b}(w) - \textstyle\sum_{i=1}^N \mathbf{d}_i(s_i,a_i)\right] + \gamma\, \mathbf{E}\big[\max_{\mathbf{a}'\in\mathcal{A}} Q_t^{\boldsymbol{\lambda}}(\mathbf{s}',\mathbf{a}')\big]. \end{aligned}$$

It is well-known that, by the value iteration algorithm's convergence,

$$Q^*(\mathbf{s},\mathbf{a}) = \lim_{t\to\infty} Q_t^*(\mathbf{s},\mathbf{a}) \quad \text{and} \quad Q^{\boldsymbol{\lambda}}(\mathbf{s},\mathbf{a}) = \lim_{t\to\infty} Q_t^{\boldsymbol{\lambda}}(\mathbf{s},\mathbf{a}).$$

Consider a state $\mathbf{s}\in\mathcal{S}$ and a feasible action $\mathbf{a}\in\mathcal{A}(\mathbf{s})$. We have,

$$Q_0^{\boldsymbol{\lambda}}(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \boldsymbol{\lambda}^T \left[\mathbf{b}(w) - \textstyle\sum_{i=1}^N \mathbf{d}_i(s_i,a_i)\right] \geq r(\mathbf{s},\mathbf{a}) = Q_0^*(\mathbf{s},\mathbf{a}).$$

Suppose $Q_t^{\boldsymbol{\lambda}}(\mathbf{s},\mathbf{a}) \geq Q_t^*(\mathbf{s},\mathbf{a})$ holds for all $\mathbf{s}\in\mathcal{S}$ and $\mathbf{a}\in\mathcal{A}(\mathbf{s})$ for some $t>0$ (induction hypothesis). Then,

$$\begin{aligned} Q_{t+1}^{\boldsymbol{\lambda}}(\mathbf{s},\mathbf{a}) &= r(\mathbf{s},\mathbf{a}) + \boldsymbol{\lambda}^T \left[\mathbf{b}(w) - \textstyle\sum_{i=1}^N \mathbf{d}_i(s_i,a_i)\right] + \gamma\, \mathbf{E}\big[\max_{\mathbf{a}'\in\mathcal{A}} Q_t^{\boldsymbol{\lambda}}(\mathbf{s}',\mathbf{a}')\big] \\ &\geq r(\mathbf{s},\mathbf{a}) + \boldsymbol{\lambda}^T \left[\mathbf{b}(w) - \textstyle\sum_{i=1}^N \mathbf{d}_i(s_i,a_i)\right] + \gamma\, \mathbf{E}\big[\max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} Q_t^*(\mathbf{s}',\mathbf{a}')\big] \\ &\geq r(\mathbf{s},\mathbf{a}) + \gamma\, \mathbf{E}\big[\max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} Q_t^*(\mathbf{s}',\mathbf{a}')\big] = Q_{t+1}^*(\mathbf{s},\mathbf{a}). \end{aligned}$$

Thus, it follows that $Q^{\boldsymbol{\lambda}}(\mathbf{s},\mathbf{a}) \geq Q^*(\mathbf{s},\mathbf{a})$.

For the proof of part (b), define

$$\mathbf{B}_0(w) = \mathbf{b}(w) \quad \text{and} \quad \mathbf{B}_{t+1}(w) = \mathbf{b}(w) + \gamma\, \mathbb{E}\big[\mathbf{B}_t(w')\big].$$

We use an induction proof. We have for all $(\mathbf{s}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}$,

$$\begin{aligned} Q_0^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a}) &= r(\mathbf{s}, \mathbf{a}) + \boldsymbol{\lambda}^T \left[\mathbf{b}(w) - \sum_{i=1}^N \mathbf{d}_i(s_i, a_i) \right] \\ &= \sum_{i=1}^N \Big[r_i(s_i, a_i) - \boldsymbol{\lambda}^T \mathbf{d}_i(s_i, a_i) \Big] + \boldsymbol{\lambda}^T \mathbf{b}(w) = \sum_{i=1}^N Q_{0,i}^{\boldsymbol{\lambda}}(s_i, a_i) + \boldsymbol{\lambda}^T \mathbf{B}_0(w), \end{aligned}$$

where $Q_{0,i}^{\boldsymbol{\lambda}}(s_i, a_i) = r_i(s_i, a_i) - \boldsymbol{\lambda}^T \mathbf{d}_i(s_i, a_i)$. Similarly, for all $(\mathbf{s}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}$,

$$\begin{aligned} Q_1^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a}) &= r(\mathbf{s}, \mathbf{a}) + \boldsymbol{\lambda}^T \left[\mathbf{b}(w) - \sum_{i=1}^N \mathbf{d}_i(s_i, a_i) \right] + \gamma \, \mathbf{E}\big[\max\nolimits_{\mathbf{a}' \in \mathcal{A}} Q_0^{\boldsymbol{\lambda}}(\mathbf{s}', \mathbf{a}') \big] \\ &= \sum_{i=1}^N \Big[r_i(s_i, a_i) - \boldsymbol{\lambda}^T \mathbf{d}_i(s_i, a_i) \Big] + \boldsymbol{\lambda}^T \mathbf{b}(w) + \gamma \, \mathbf{E} \left[\max_{\mathbf{a}' \in \mathcal{A}} \left\{ \sum_{i=1}^N Q_{0,i}^{\boldsymbol{\lambda}}(s_i', a_i') + \boldsymbol{\lambda}^T \mathbf{B}_0(w') \right\} \right] \\ &= \sum_{i=1}^N \Big[r_i(s_i, a_i) - \boldsymbol{\lambda}^T \mathbf{d}_i(s_i, a_i) + \gamma \, \mathbf{E}\big[\max\nolimits_{a_i' \in \mathcal{A}_i} Q_{0,i}^{\boldsymbol{\lambda}}(s_i', a_i') \big] \Big] + \boldsymbol{\lambda}^T \Big(\mathbf{b}(w) + \gamma \, \mathbf{E}\big[\mathbf{B}_0(w') \big] \Big) \\ &= \sum_{i=1}^N Q_{1,i}^{\boldsymbol{\lambda}}(s_i, a_i) + \boldsymbol{\lambda}^T \mathbf{B}_1(w). \end{aligned}$$

If we continue in this manner, we arrive at $Q_t^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a}) = \sum_{i=1}^N Q_{t,i}^{\boldsymbol{\lambda}}(s_i, a_i) + \boldsymbol{\lambda}^T \mathbf{B}_t(w)$. Finally, we have

$$\begin{aligned} Q^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a}) &= \lim_{t \to \infty} Q_t^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a}) \\ &= \lim_{t \to \infty} \sum_{i=1}^N Q_{t,i}^{\boldsymbol{\lambda}}(s_i, a_i) + \boldsymbol{\lambda}^T \mathbf{B}_t(w) = \sum_{i=1}^N Q_i^{\boldsymbol{\lambda}}(s_i, a_i) + \boldsymbol{\lambda}^T \mathbf{B}(w), \end{aligned}$$

which follows by the convergence of value iteration. □

B.1.2 Proof of Theorem 3.3.1

Proof. We first prove part (i). For a fixed $\boldsymbol{\lambda}$, we can define a new MDP with a reward function given by $r_i(s_i, a_i) - \boldsymbol{\lambda}^T \mathbf{d}_i(s_i, a_i)$. It follows that the Q-learning algorithm on this new MDP converges almost surely to the optimal action value function $Q^{\boldsymbol{\lambda},*}$, see for example [9].

We now prove the second result: $\lim_{n\to\infty} Q_n^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a}) \geq Q^*(\mathbf{s}, \mathbf{a})$. Recall that

$$Q_n^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a}) = \boldsymbol{\lambda}^T \mathbf{B}_n(w) + \sum_{i=1}^{N} Q_{i,n}^{\boldsymbol{\lambda}}(s_i, a_i).$$

By part (i), we have that $Q_{i,n}^{\boldsymbol{\lambda}}(s_i, a_i) \to Q_i^{\boldsymbol{\lambda},*}(s_i, a_i)$ as $n \to \infty$. Also by standard stochastic approximation theory, we have $\lim_{n\to\infty} \mathbf{B}_n(w) = \mathbf{B}(w)$ for all w [43]. Thus, we have $\lim_{n\to\infty} Q_n^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a}) = Q^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a})$ for all $(\mathbf{s}, \mathbf{a})$. The result follows by Proposition 3.2.1(a) which states that $Q^*(\mathbf{s}, \mathbf{a}) \leq Q^{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{a})$ for any $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$.

For part (ii), we provide a sketch of the proof. The idea is similar to the proof of Theorem 1 of [25]. Assume without loss of generality that $Q^*(\mathbf{s}, \mathbf{a}) = 0$ for all state-action pairs $(\mathbf{s}, \mathbf{a})$. First, note that the iterates $Q'_n(\mathbf{s}, \mathbf{a})$ are bounded in the sense that there exists a constant $D_0 = R_{\max}/(1-\gamma)$, $R_{\max} = \max_{(\mathbf{s},\mathbf{a})} |r(\mathbf{s}, \mathbf{a})|$, such that $|Q'_n(\mathbf{s}, \mathbf{a})| \leq D_0$ for all $(\mathbf{s}, \mathbf{a})$ and iterations n [26]. Next, define the sequence $D_{k+1} = (\gamma + \epsilon) D_k$, such that $\gamma + \epsilon < 1$ and $\epsilon > 0$. Clearly, $D_k \to 0$.

We then proceed by induction. The goal is now to show that there exists some time n_k such that for all $(\mathbf{s}, \mathbf{a})$,

$$\max\{-D_k, L_n(\mathbf{s}, \mathbf{a})\} \leq Q'_n(\mathbf{s}, \mathbf{a}) \leq \min\{D_k, Q_n^{\boldsymbol{\lambda}^*}(\mathbf{s}, \mathbf{a})\}, \quad \forall n \geq n_k. \tag{B.1}$$

This would impliy that $Q'_n(\mathbf{s}, \mathbf{a})$ converges to $Q^*(\mathbf{s}, \mathbf{a}) = 0$ for all $(\mathbf{s}, \mathbf{a})$.

To show inequality (B.1), we have that as $n \to \infty$, the L-iterate will converge to the Lagrangian policy's action-value function, which gives a lower bound on Q^*. Also, $Q^*(\mathbf{s}, \mathbf{a}) \leq Q^{\boldsymbol{\lambda},*}(\mathbf{s}, \mathbf{a})$ by part (i). Thus by construction of Q'_n (projection step), we can establish that there exist some n'_k, such that for all $n \geq n'_k$ we have

$$L_n(\mathbf{s}, \mathbf{a}) \leq Q'_n(\mathbf{s}, \mathbf{a}) \leq Q_n^{\boldsymbol{\lambda}^*}(\mathbf{s}, \mathbf{a}).$$

Finally, given the above inequality, and combined with standard theory (see [9]), we can establish that there must be some n_k for which (B.1) holds for all $n \geq n_k$.

□

B.2 Weakly Coupled Q-learning Algorithm

Algorithm 6: Weekly Coupled Q-learning

Input: Initial estimates L_0, Q_0, and $Q_{i,0}$, and stepsize rules α_n, β_n, η_n, and ζ_n.

Output: Approximations $\{L_n\}$, $\{Q'_n\}$, and $\{Q_{i,n}\}$.

Set $Q'_0 = Q_0$ and choose an initial state s_0.

for $n = 0, 1, 2, \ldots$ **do**

Choose an action $\mathbf{a}_n$ via some behavior policy (e.g., ϵ-greedy). Let

$$Q_{n+1}(\mathbf{s}_n, \mathbf{a}_n) = Q'_n(\mathbf{s}_n, \mathbf{a}_n) + \alpha_n \Big[r_n(\mathbf{s}_n, \mathbf{a}_n) + \gamma \max_{\mathbf{a}} Q'_n(\mathbf{s}_{n+1}, \mathbf{a}) - Q'_n(\mathbf{s}_n, \mathbf{a}_n) \Big].$$

Run a separate Q-learning for each $\lambda \in \Lambda$, for each subproblem $i \in \{1, \ldots, N\}$

$$Q^{\boldsymbol{\lambda}}_{i,n+1}(s_{i,n}, a_{i,n}) = Q^{\boldsymbol{\lambda}}_{i,n}(s_{i,n}, a_{i,n}) + \beta_n \Big[r_i(s_{i,n}, a_{i,n}) - \boldsymbol{\lambda}^T \mathbf{d}_i(s_{i,n}, a_{i,n}) + \gamma \max_{a'_i} Q^{\lambda}_{i,n}(s_{i,n+1}, a'_i) - Q^{\lambda}_n(s_{i,n}, a_{i,n}) \Big].$$

Update $\mathbf{B}_{n+1}(w_n)$ according to equation (3.6).

Find the best upper bound:

For $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ and $\mathbf{a} \in \mathcal{A}(\mathbf{s}_n)$ compute $Q^{\boldsymbol{\lambda}}_{n+1}(\mathbf{s}_n, \mathbf{a})$ using (3.7).

Set $Q^{\boldsymbol{\lambda}^*}_{n+1}(\mathbf{s}_n, \mathbf{a}) = \min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} Q^{\boldsymbol{\lambda}}_{n+1}(\mathbf{s}_n, \mathbf{a})$

Update lower bound:

Set $\mathbf{a}^{\boldsymbol{\lambda}^*}_{n+1} = \arg\min_{\mathbf{a}} Q^{\boldsymbol{\lambda}^*}_{n+1}(\mathbf{s}_{n+1}, \mathbf{a})$

$$L_{n+1}(\mathbf{s}_n, \mathbf{a}_n) = L_n(\mathbf{s}_n, \mathbf{a}_n) + \zeta_n \Big[r_n(\mathbf{s}_n, \mathbf{a}_n) + \gamma L_n(\mathbf{s}_{n+1}, \mathbf{a}^{\boldsymbol{\lambda}^*}_{n+1}) - L_n(\mathbf{s}_n, \mathbf{a}_n) \Big].$$

Project to satisfy the bounds using (2.13).

end for

B.3 Lagrangian DQN Algorithm

Algorithm 7: Lagrangian DQN

Input: Initialized replay buffer $\mathcal{D}$, Lagrangian multipliers set $\mathbf{\Lambda}$, subproblems $Q_i^{\boldsymbol{\lambda}}$-network θ_U & $\theta_U^- = \theta_U$

Output: Approximation $\{Q_n^{\boldsymbol{\lambda}}\}$

for $n = 0, 1, 2, \ldots$ **do**

Find the best upper bound:

For $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ and $\mathbf{a} \in \mathcal{A}(\mathbf{s}_n)$ find $Q_n^{\boldsymbol{\lambda}}(\mathbf{s}_n, \mathbf{a})$ per (3.11).

Choose an action $\mathbf{a}_n$ via some behavior policy (e.g., ϵ-greedy($Q_n^{\boldsymbol{\lambda}^*}$)), observe the transition experience and store $(\mathbf{s}_n, \mathbf{a}_n, \mathbf{d}_n, \mathbf{r}_n, \mathbf{b}_n, \mathbf{s}_{n+1})$ in $\mathcal{D}$.

Update $\mathbf{B}_{n+1}(w)$ according to equation (3.6).

Update subproblems network:

Sample a minibatch of transitions $\boldsymbol{\tau}$ from $\mathcal{D}$ along with random $\boldsymbol{\lambda}$.

for i= 1, ..., N **do**

Compute targets y_i as per (3.13).

Perform a gradient descent step on (3.12).

end for

end for

B.4 Numerical Experiments Details

A discount factor of 0.9 is used for the EV charging problem and 0.99 for the multi-product inventory and online stochastic ad matching problems. In the tabular setting, we use a polynomial learning rate that depends on the state-action pairs visitation given by $\alpha_n(\mathbf{s}, \mathbf{a}) = 1/\nu_n(\mathbf{s}, \mathbf{a})^r$, where $\nu_n(\mathbf{s}, \mathbf{a})$ represent the number of times $(\mathbf{s}, \mathbf{a})$ has been visited up to iteration n, and $r = 0.4$. We also use an ϵ-greedy exploration policy given by $\epsilon(s) = 1/\nu(\mathbf{s})^e$, where $\nu(\mathbf{s})$ is the number of time the state $(\mathbf{s})$ has been visited and $e = 0.4$. In the function approximation setting, we an ϵ-greedy policy that decays ϵ from 1 to 0.05 after

200,000 steps. All state-action value functions are initialized randomly. Experiments were ran on a shared memory cluster with dual 12-core Skylake CPU (Intel Xeon Gold 6126 2.60 GHz) and 192 GB RAM/node.

B.4.1 EV Charging with Exogenous Electricity Cost

In this problem, there are in total three charging spots $N = 3$. Each spot represents a subproblem with state $(c_t, B_{i,t}, D_{i,t})$, where $c_t \in \{0.2, 0.5, 0.8\}$ is the exogenous electric cost, $B_{i,t} \leq 2$ is the amount of charge required and $D_{i,t} \leq 4$ is the remaining time until the EV leaves the system. The state space size is 36 for each subproblem. At a given period t, the action of each subproblem is whether to charge an EV occupying the charging spot $a_{i,t} = 1$ or not $a_{i,t} = 0$. A feasible action is given by $\sum_{i=1}^{N} a_{i,t} \leq b(c_t)$, where $b(0.2) = 3, b(0.5) = 2$, and $b(0.8) = 1$. The reward of each subproblem is given by the reward function,

$$r_i((c_t, B_{i,t}, D_{i,t}), a_{i,t}) = \begin{cases} (1 - c_t)a_{i,t} \text{ if } B_{i,t} > 0, D_{i,t} > 1, \\ (1 - c_t)a_{i,t} - F(B_{i,t} - a_{i,t}) \text{ if } B_{i,t} > 0, D_{i,t} = 1, \\ 0, \quad \text{otherwise,} \end{cases}$$

where $F(B_{i,t} - a_{i,t}) = 0.2(B_{i,t} - a_{i,t})^2$ is a penalty function for failing to complete the charging of the EV before the deadline. The endogenous state of each subproblem evolves such that $(B_{i,t+1}, D_{i,t+1}) = (B_{i,t} - a_{i,t}, D_{i,t} - 1)$ if $D_{i,t} > 1$, and $(B_{i,t+1}, D_{i,t+1}) = (B, D)$ with probability $q(D, B)$ if $D_t \leq 1$, where $q(0, 0) = 0.3$ and $q(B, D) = 0.7/11$ for all $B > 0$ and $D > 0$. On the other hand, the exogenous state c_t evolves following the transition probabilities given by:

$$P(c_{t+1}|c_t) = \begin{pmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.5 & 0.3 \\ 0.6 & 0.2 & 0.2 \end{pmatrix}.$$

B.4.2 Multi-product Inventory Control with an Exogenous Production Rate

We consider manufacturing $K = 5$ products. The exogenous demand D_k for each product $k \in \{1, 2, \ldots, 5\}$ follows a Poisson distribution with mean value μ_k. The maximum storage capacity and the maximum number of allowable backorders (after which lost sales incur) for product k are given by N_k and M_k, respectively. The state for subproblem k is given by (p_k, x_k), where $x_k \in X_k = \{-M_k, -M_k + 1, \ldots, N_k\}$ is the continuous inventory level for product k, and p_k is an exogenous random Markovian noise with support $\{0.8, 0.85, 0.9, 0.95, 1., 1.05, 1.1, 1.15, 1.2\}$. A negative stock level corresponds to the number of backorders. For subproblem k, the action a_k is the number of resources allocated to the product k. The maximum number of resources available for all products is $U = 4$ so a feasible action is such that $\sum_k a_k \leq 4$. Allocating a resource level a_k yields a production rate $\rho_k(a_k, p_k) = p_k(10\, a_k)/(5.971 + a_k)$. The cost function for product k, $c_k(p_k, x_k, a_k)$, is the sum of the holding, backorders, and lost sales costs. We let h_k, b_k, and l_k denote the holding, backorder, and lost sale costs per unit respectively. The cost function $c_k(p_k, x_k, a_k)$ is given by,

$$\begin{aligned} c_k(p_k, x_k, a_k) = h_k(x_k + \rho_k(a_k, p_k))_+ + b_k(-x_k - \rho_k(a_k, p_k))_+ \\ + l_k((D_k - x_k - \rho_k(a_k, p_k))_+ - M_k)_+, \end{aligned}$$

where $(.)_+ = \max(., 0)$. We summarize the cost parameters and the mean demand for each product in Table 4. Finally, the transition function for subproblem k is given by

$$f(p_t, x_{k,t}, a_{k,t}) = (p_{t+1}, \max(\min(x_{k,t} + \rho_k(a_{k,t}, p_t) - D_{k,t}, N_{k,t}), -M_{k,t})),$$

where the exogenous noise p_t evolves according to a transition matrix sampled from a Dirichlet distribution whose parameters are each sampled from a uniform $U(1, 20)$ distribution.

Table 4: Multi-product inventory environment parameters

Product k	**1**	**2**	**3**	**4**	**5**
Storage capacity $\mathbf{N}_k$	10	11	16	20	25
Maximum backorders $\mathbf{M}_k$	5	10	7	15	12
Mean demand $\boldsymbol{\mu}_k$	1.3	1.2	1.4	1.5	1.6
Holding cost $\mathbf{h}_k$	1.3	1.2	1.1	1.4	5
Backorders cost $\mathbf{b}_k$	6	5	15	12	8
Lost sales cost $\mathbf{l}_k$	12	11	25	30	10

B.4.3 Online Stochastic Ad. Matching

In this problem, a DM needs to match $N = 6$ advertisers to arriving impressions [27]. An impression $e_t \in E := \{1, 2, \ldots, 5\}$ arrives according to a discrete time Markov chain with transition probabilities given by $P_E(e_{t+1}|e_t)$, where each row of the transition matrix P_E is sampled from a Dirichlet distribution whose parameters are sampled from a uniform distribution $U(1, 20)$. The action $a_{t,i} \in \{0, 1\}$ is whether to assign impression e_t to advertiser i or not. The DM can assign an impression to at most one advertiser, $\sum_{i=1}^{N} a_{i,t} = 1$. The state of advertiser i, $x_{i,t}$ gives the number of remaining ads to display and evolves according to $x_{i,t+1} = s_{i,t} - a_{i,t}$. The initial state is $x_0 = (10, 11, 12, 10, 14, 9)$. The reward obtained from advertiser i in state $s_{i,t} = (e_t, x_{i,t})$ is $r_i(e_t, x_{i,t}, a_{i,t}) = l_{i,e_t} \min(x_{i,t}, a_{i,t})$, where the parameters $l_{i,e_t} \sim U(1, 4)$.

Training settings. We train the algorithms for 6000 episodes on the EV charging and the multi-product inventory control problems and for 10000 episodes on the online stochastic ad matching problem. Each episode consists of 50 time steps except for the online ad stochastic ad matching problem which consists of 80 steps. The result of each algorithm is averaged over 5 runs. We use a neural network architecture that consists of two hidden layers, with 64 and 32 hidden units respectively, for all algorithms. A rectified linear unit

(ReLU) is used as the activation function for each hidden layer. All algorithms were trained using Adam with a learning rate of 0.0001 [39]. For Lagrangian DQN and WCDQN, we use a Lagrangian multiplier $\lambda \in [0, 10]$, with a 0.1 discretization. We also used an experience buffer of size 500,000 and initialized it with 100,000 experience tuples that were obtained using a random policy. For the WCDQN algorithm, we set the penalty coefficients τ_L and τ_U to 1 and 10, respectively.

Appendix C

C.1 Proofs for Chapter 3

C.1.1 Proof of Proposition 4.3.2

Proof. Let $f_t(\mathbf{w}_t) = V_{t+1}(\mathbf{x}_{t+1}) - \sum_{ij\in[N]} \rho_{ij,t}(d_{ij,t} + \epsilon_{ij,t} - w_{ij,t})$ for a given $\boldsymbol{\epsilon_t}$, $t = 1, \ldots, T$. We first show that the solution $w^*_{ij,t} = \min(d_{ij,t} + \epsilon_{ij,t}, \theta ij, t)$ for all $i, j \in [N]$, is optimal for the maximization problem in (4.2) confirming thus that $J_t(x_t, \mathbf{d}_t, \boldsymbol{\theta}_t, \boldsymbol{\epsilon}_t) = f_t(\mathbf{w}^*_t)$, for $t = 1, \ldots, T$.

Suppose, on the contrary, for some t, there exist an optimal solution $w'_{kl,t} < \min(d_{kl,t} + \epsilon_{kl,t}, \theta_{kl,t})$ for some $k, l \in [N]$, w.l.o.g. Denote by $\mathbf{x}'_{t+1}$ the state in period $t+1$ under $\mathbf{w}'_t$. Let $\varepsilon = \min(d_{kl,t} + \epsilon_{kl,t}, \theta_{kl,t}) - w'_{kl,t} > 0$, then the solution constructed by $w_{kl,t} = w'_{kl,t} + \varepsilon$ and $w_{ij,t} = w'_{ij,t}$ for any $ij \neq kl$ is still feasible solution to (4.2) and

$$\begin{aligned} f_t(\mathbf{w}_t) &= V_{t+1}(\mathbf{x}_{t+1}) - \sum_{ij\in[N]} \rho_{ij,t}(d_{ij,t} + \epsilon_{ij,t} - w_{ij,t}) \\ &= V_{t+1}(\mathbf{x}_{t+1}) - \sum_{ij\in[N]} \rho_{ij,t}(d_{ij,t} + \epsilon_{ij,t} - w'_{ij,t}) + \rho_{kl,t}\varepsilon. \end{aligned} \tag{C.1}$$

Note here if $k = l$ then $\mathbf{x}'_{t+1} = \mathbf{x}_{t+1}$, i.e., $V_{t+1}(\mathbf{x}_{t+1}) = V_{t+1}(\mathbf{x}'_{t+1})$ therefore $f_t(\mathbf{w}_t) \geq f_t(\mathbf{w}'_t)$. We claim that $f_t(\mathbf{w}_t) \geq f_t(\mathbf{w}'_t)$ still holds for the case when $k \neq l$, for all $t = 1, \ldots, T$. We give a proof by induction. At time T, we have

$$\begin{aligned} f_T(\mathbf{w}_T) &= V_{T+1}(\mathbf{x}_{T+1}) - \sum_{ij\in[N]} \rho_{ij,T}(d_{ij,T} + \epsilon_{ij,T} - w'_{ij,T}) + \rho_{kl,T}\varepsilon_T \\ &= -\sum_{ij\in[N]} \rho_{ij,T}(d_{ij,T} + \epsilon_{ij,T} - w'_{ij,T}) + \rho_{kl,T}\varepsilon_T \\ &= f_T(\mathbf{w}'_T) + \rho_{kl,T}\varepsilon_T \\ &\geq f_T(\mathbf{w}'_T). \end{aligned}$$

Thus, our claim is true at time T. Suppose it is true at time $t+1$, i.e., $f_{t+1}(\mathbf{w}_{t+1}) \geq f_{t+1}(\mathbf{w}'_{t+1})$. Then, at time t, we have

$$\begin{aligned}
f_t(\mathbf{w}_t) &= V_{t+1}(\mathbf{x}_{t+1}) - \sum_{ij\in[N]} \rho_{ij,t}(d_{ij,t} + \epsilon_{ij,t} - w'_{ij,t}) + \rho_{kl,t}\varepsilon_t \\
&= \max_{\mathbf{y}_{t+1}\in\mathcal{Y}(\mathbf{x}_{t+1})} \left\{ \sum_{ij\in[N]} P_{ij,t+1}(d_{ij,t+1})\mathbf{E}[(\mathbf{w}_{t+1})_{ij}] + \mathbf{E}[f_{t+1}(\mathbf{w}_{t+1})] \right\} \\
&- \sum_{ij\in[N]} \rho_{ij,t+1}(d_{ij,t+1} + \epsilon_{ij,t} - w'_{ij,t}) + \rho_{kl,t}\varepsilon_t \\
&\geq \max_{\mathbf{y}_{t+1}\in\mathcal{Y}(\mathbf{x}_{t+1})} \left\{ \sum_{ij\in[N]} P_{ij,t+1}(d_{ij,t+1})\mathbf{E}[(\mathbf{w}'_{t+1})_{ij}] + \mathbf{E}[f_{t+1}(\mathbf{w}'_{t+1})] \right\} \\
&- \sum_{ij\in[N]} \rho_{ij,t+1}(d_{ij,t+1} + \epsilon_{ij,t} - w'_{ij,t}) + \rho_{kl,t}\varepsilon_t \\
&= V_{t+1}(\mathbf{x}'_{t+1}) - \sum_{ij\in[N]} \rho_{ij,t+1}(d_{ij,t+1} + \epsilon_{ij,t} - w'_{ij,t}) + \rho_{kl,t}\varepsilon_t \\
&= f_t(\mathbf{w}'_t) + \rho_{kl,t}\varepsilon_t \\
&\geq f_t(\mathbf{w}'_t),
\end{aligned}$$

where the second equality follows by definition of $V_{t+1}(\mathbf{x}_{t+1}), \mathbf{w}_{t+1}$ and $f_{t+1}(\mathbf{w}_{t+1})$. The first inequality follows because $\mathbf{w}_{t+1} \geq \mathbf{w}'_{t+1}$ and $f_{t+1}(\mathbf{w}_{t+1}) \geq f_{t+1}(\mathbf{w}'_{t+1})$ by the induction hypothesis and because expectation is a linear operator. Accordingly, $\mathbf{w}_t$ is optimal, for $t = 1, \ldots, T$.

We now turn to prove the concavity of the value function. Clearly, $V_{T+1}(\mathbf{x}_{T+1})$ is concave. Suppose that $V_{t+1}(\mathbf{x}_{t+1})$ is concave for some $t < T$. We have the objective in (4.2) is jointly concave in $\mathbf{x}_t, \mathbf{d}_t, \boldsymbol{\theta}_t$ and $\mathbf{w}_t$. Since the constraint set is convex, then for any $\boldsymbol{\epsilon}_t$, we have $J_t(\mathbf{x}_t, \mathbf{d}_t, \boldsymbol{\theta}_t, \boldsymbol{\epsilon}_t)$ is jointly concave in $\mathbf{x}_t, \mathbf{d}_t$, and $\boldsymbol{\theta}_t$ by Proposition 2.1.15 in [60]. By the linearity of the expectation, we have $\mathbf{E}[J_t(\mathbf{x}_t, \mathbf{d}_t, \boldsymbol{\theta}_t, \boldsymbol{\epsilon}_t)]$ is jointly concave in $\mathbf{x}_t, \mathbf{d}_t$, and $\boldsymbol{\theta}_t$. Since the rest of the terms in the objective of (4.1) is concave and the constraint set is again a convex set, we have $V_t(\mathbf{x}_t)$ is concave in $\mathbf{x}_t$. □

C.1.2 Proof of Theorem 4.4.1

Proof. The proof that $w^*_{ij,t} = \min\left(d_{ij,t} + \epsilon_{ij,t}, \theta ij, t\right) \forall\, ij \in \{1,2\}$, is optimal for the maximization problem in (4.3) is similar to the one given in Proposition 4.3.2.

For the second part of the theorem, we will first do the following variable transformation to (4.3):
Let $\hat{d}_{12,t} = x_t - d_{12,t}, \tilde{d}_{21,t} = -d_{21,t}, \hat{w}_{12,t} = x_t - w_{12,t}, \tilde{w}_{21,t} = -w_{21,t}, \tilde{\theta}_{21,t} = -\theta_{21,t}, \hat{\theta}_{12,t} = x_t - \theta_{12,t}$, and let $\dot{\mathbf{d}}_\mathbf{t}, \dot{\boldsymbol{\theta}}_\boldsymbol{t}$ and $\dot{\mathbf{y}}_t = (\hat{d}_{12,t}, \tilde{d}_{21,t}, \hat{\theta}_{12,t}, \tilde{\theta}_{21,t}) \in \dot{\mathcal{Y}}(x_t)$ denote the transformed decision vectors where the transformed feasible set $\dot{\mathcal{Y}}(x_t)$ is given by:

$$\dot{\mathcal{Y}}(x_t) = \left\{\dot{\mathbf{y}} : \hat{d}_{12,t} - x_t \le -\underline{d}_{12,t},\ x_t - \hat{d}_{12,t} \le \bar{d}_{12,t},\ \tilde{d}_{21,t} \le -\underline{d}_{21,t},\ \tilde{d}_{21,t} \ge -\bar{d}_{21,t}, \hat{\theta}_{12,t} \ge 0, \tilde{\theta}_{21,t} \ge x_t - \bar{x}\right\}.$$

Equivalently, we can write (4.3) as,

$$\begin{aligned}
V^*_t(x_t) = \max_{\dot{\mathbf{y}}_t \in \dot{\mathcal{Y}}(x_t)} \Big\{ & P_{11,t}(d_{11,t})\mathbf{E}[\min(d_{11,t} + \epsilon_{11,t}, \hat{\theta}_{12,t})] \\
& + P_{22,t}(d_{22,t})\mathbf{E}[\min(d_{22,t} + \epsilon_{22,t}, \bar{x} - x_t + \tilde{\theta}_{21,t})] \\
& + P_{12,t}(x_t - \hat{d}_{12,t})\mathbf{E}[\min(x_t - \hat{d}_{12,t} + \epsilon_{12,t}, x_t - \hat{\theta}_{12,t})] \\
& + P_{21,t}(-\tilde{d}_{21,t})\mathbf{E}[\min(-\tilde{d}_{21,t} + \epsilon_{21,t}, -\tilde{\theta}_{21,t})] \\
& + \mathbf{E}[J_t(x_t, \dot{\mathbf{d}}_\mathbf{t}, \dot{\boldsymbol{\theta}}_\boldsymbol{t}, \boldsymbol{\epsilon}_\boldsymbol{t})] \Big\} \\
J_t(x_t, \dot{\mathbf{d}}_\mathbf{t}, \dot{\boldsymbol{\theta}}_\boldsymbol{t}, \boldsymbol{\epsilon}_\boldsymbol{t}) = \max_{\hat{w}_{12,t}, \tilde{w}_{21,t} \in \dot{\mathcal{W}}_t} \Big\{ & V_{t+1}(\hat{w}_{12,t} - \tilde{w}_{21,t}) - \sum_{ij \in \{1,2\}, i=j} \rho_{ij,t+1}(d_{ij,t+1} + \epsilon_{ij,t} - w_{ij,t}) \\
& - \rho_{12,t}(-\hat{d}_{12,t} + \epsilon_{12,t} + \hat{w}_{12,t}) - \rho_{21,t}(-\tilde{d}_{21,t} + \epsilon_{21,t} + \tilde{w}_{21,t}) \Big\}
\end{aligned}$$

where,

$$\begin{aligned}
\dot{\mathcal{W}}_t = \{\dot{\mathbf{w}} :\ & \hat{\theta}_{12,t} - \hat{w}_{12,t} \le 0, \hat{d}_{12,t} - \hat{w}_{12,t} \le \epsilon_{12,t}, \tilde{d}_{21,t} - \tilde{w}_{21,t} \le \epsilon_{21,t}, \tilde{\theta}_{21,t} - \tilde{w}_{21,t} \le 0, \\
& w_{11,t} - d_{11,t} \le \epsilon_{11,t}, w_{22,t} - d_{22,t} \le \epsilon_{22,t}\}
\end{aligned} \tag{C.2}$$

We now show by induction that $L^\natural$-concavity is preserved in the dynamic program recursion. Clearly $V_{T+1}(x_{t+1})$ is $L^\natural$-concave. Assume that it holds for $t+1$, then using Lemma 1 in [21] $V_{t+1}(\hat{w}_{12,t} - \tilde{w}_{21,t})$ is $L^\natural$-concave in $(x_t, \hat{\theta}_{12,t}, \tilde{\theta}_{21,t}, \hat{d}_{12,t}, \tilde{d}_{21,t}, \hat{w}_{21,t}, \tilde{w}_{21,t})$ for $t \le T$. The $L^\natural$-concavity of the rest of terms is straight-forward to verify. By Proposition 1. part (f) in [19]

the constraint set in $\dot{\mathcal{W}}$ is $L^\natural$-convex then by Lemma 2 in [21] $J_t(x_t, \dot{\mathbf{d}}_\mathbf{t}, \dot{\boldsymbol{\theta}}_\boldsymbol{t}, \boldsymbol{\epsilon}_\boldsymbol{t})$ is $L^\natural$-concave in $(x_t, \hat{\theta}_{12,t}, \tilde{\theta}_{21,t}, \hat{d}_{12,t}, \tilde{d}_{21,t}, \hat{w}_{21,t}, \tilde{w}_{21,t})$. Since $L^\natural$-concavity is preserved by expectation and the rest of the terms in the objective of (C.2) are $L^\natural$-concave, (see Proposition C.1.1), then the objective function is also $L^\natural$-concave in $(x_t, \hat{\theta}_{12,t}, \tilde{\theta}_{21,t}, \hat{d}_{12,t}, \tilde{d}_{21})$. By Proposition 1. part (f) in [19] the constraint set $\dot{\mathcal{Y}}$ is $L^\natural$-convex. Finally, by Lemma 2 in [21] we have $V_t(x_t)$ is $L^\natural$-concave in x_t. By Proposition 1. of [20], the optimal solution $\hat{\theta}^*_{12,t}(x_t)$, $\hat{d}^*_{12,t}(x_t)$, $\tilde{\theta}^*_{21,t}(x_t)$, and $\tilde{d}^*_{21,t}(x_t)$ are all nondecreasing in x_t and for any $\omega > 0$, the following inequalities hold, $\hat{d}^*_{12,t}(x+\omega) \leq \hat{d}^*_{12,t}(x)+\omega$, $\tilde{d}^*_{21,t}(x+\omega) \leq \tilde{d}^*_{21,t}(x)+\omega$, $\hat{\theta}^*_{12,t}(x+\omega) \leq \hat{\theta}^*_{12,t}(x)+\omega$, $\tilde{\theta}^*_{21,t}(x+\omega) \leq \tilde{\theta}^*_{21,t}(x) + \omega$. Since we have $\hat{d}_{12,t} = x_t - d_{12,t}, \tilde{d}_{21,t} = -d_{21,t}, \tilde{\theta}_{21,t} = -\theta_{21,t}, \hat{\theta}_{12,t} = x_t - \theta_{12,t}$, it follows that $\theta^*_{12,t}(x_t)$, $d^*_{12,t}(x_t)$ and $\theta^*_{21}(x_t), d^*_{21,t}(x_t)$ are nondecreasing and nonincreasing respectively in x_t. Moreover, for any $\omega > 0$, the following inequalities hold. $d^*_{12,t}(x+\omega) \leq d^*_{12,t}(x)+\omega$, $d^*_{21,t}(x+\omega) \geq d^*_{21,t}(x)-\omega$, $\theta^*_{12,t}(x+\omega) \leq \theta^*_{12,t}(x)+\omega$, $\theta^*_{21,t}(x+\omega) \geq \theta^*_{21,t}(x)-\omega$. Q.E.D. □

Proposition C.1.1. *Under conditions (C1) and (C2),* $R_{11}(\hat{\theta}_{12}) = P_{11}(d_{11})\mathbf{E}[\min(d_{11} + \epsilon_{11}, \hat{\theta}_{12})]$,
$R_{22}(x, \tilde{\theta}_{21}) = P_{22}(d_{22})\mathbf{E}[\min(d_{22}+\epsilon_{22}, x_0 - x + \tilde{\theta}_{21})]$, $R_{12}(\hat{d}_{12}, x, \hat{\theta}_{12}) = P_{12}(x-\hat{d}_{12})\mathbf{E}[\min(x - \hat{d}_{12} + \epsilon_{12}, x - \hat{\theta}_{12})]$ *and* $R_{21}(\tilde{d}_{21}, \tilde{\theta}_{21}) = P_{21}(-\tilde{d}_{21})\mathbf{E}[\min(-\tilde{d}_{21} + \epsilon_{21}, -\tilde{\theta}_{21})]$ *are* $L^\natural$*-concave in their respective variables.*

Proof. First we start by showing that $R_{11}(\hat{\theta}_{12}) = P_{11}(d_{11})\mathbf{E}[\min(d_{11}+\epsilon_{11}, \hat{\theta}_{12})]$ is $L^\natural$-concave in $\hat{\theta}_{12}$. Note that since we fix d_{11} then we only need to show that R_{11} is $L^\natural$-concave in $\hat{\theta}_{12}$. It is enough to show that $R_{11}(\hat{\theta}_{12} - \zeta)$ is supermodular in $(\hat{\theta}_{12}, \zeta)$ for $\zeta \in \{\zeta : 0 \leq \zeta \leq \hat{\theta}_{12}\}$. Note that we can write $R_{11}(\hat{\theta}_{12} - \zeta)$ as $P_{11}(d_{11})d_{11} + P_{11}(d_{11})\mathbf{E}[\min(\epsilon_{11}, \hat{\theta}_{12} - \zeta - d_{11})]$.

$$\begin{aligned}\frac{\partial^2 R_{11}}{\partial\hat{\theta}_{12}\partial\zeta} &= P_{11}(d_{11})F'(\hat{\theta}_{12} - \zeta - d_{11}) \\ &\geq -P'_{11}(d_{11})\bar{F}(\hat{\theta}_{12} - \zeta - d_{11}) \\ &\geq 0\end{aligned}$$

Where the first inequality follows from $(C2)$ and the second inequality follow from $P'_{11} \leq 0$.

Similarly, we fix d_{22} and study the $L^\natural$-concavity of $R_{22}(x, \tilde{\theta}_{21}) = P_{22}(d_{22})\mathbf{E}[\min(d_{22} + \epsilon_{22}, x_0 - x + \tilde{\theta}_{21})]$. We need to show that $R_{22}(x - \zeta, \tilde{\theta}_{21} - \zeta)$ is supermodular in $(x, \tilde{\theta}_{21}, \zeta)$. But $R_{22}(x-\zeta, \tilde{\theta}_{21} - \zeta) = R_{22}(x, \tilde{\theta}_{21})$, so it is enough to show that $R_{22}(x, \tilde{\theta}_{21})$ is supermodular in $(x, \tilde{\theta}_{12})$.

Note that we can write $R_{22}(x, \tilde{\theta}_{21})$ as $P_{22}(d_{22})d_{22} + P_{22}(d_{22})\mathbf{E}[\min(\epsilon_{22}, x_0 - d_{22} - x + \tilde{\theta}_{21})]$.

$$\begin{aligned}
\frac{\partial^2 R_{22}}{\partial x \partial \tilde{\theta}_{21}} &= P_{22}(d_{22})F'(x_0 - d_{22} - x + \tilde{\theta}_{21}) \\
&\geq -P'_{22}(d_{22})\bar{F}(x_0 - d_{22} - x + \tilde{\theta}_{21}) \\
&\geq 0.
\end{aligned}$$

Where the first inequality follows from condition $(C2)$ and the second inequality follow from $P'_{22} \leq 0$.

To show that $R_{12}(\hat{d}_{12}, x, \hat{\theta}_{12}) = P_{12}(x - \hat{d}_{12})\mathbf{E}[\min(x - \hat{d}_{12} + \epsilon_{12}, x - \hat{\theta}_{12})]$ is $L^\natural$-concave in $(\hat{d}_{12}, x, \hat{\theta}_{12})$, it is enough to show that $R_{12}(\hat{d}_{12} - \zeta, x - \zeta, \hat{\theta}_{12} - \zeta)$ is supermodular in $(\hat{d}_{12}, x, \zeta)$. But $R_{12}(\hat{d}_{12} - \zeta, x - \zeta, \hat{\theta}_{12} - \zeta) = R_{12}(\hat{d}_{12}, x, \hat{\theta}_{12})$, so we show that $R_{12}(\hat{d}_{12}, x, \hat{\theta}_{12})$ is supermodular in $(\hat{d}_{12}, x, \hat{\theta}_{12})$. Note that we can write $R_{12}(\hat{d}_{12}, x, \hat{\theta}_{12})$ as $P_{12}(x - \hat{d}_{12})(x - \hat{d}_{12}) + P_{12}(x - \hat{d}_{12})\mathbf{E}[\min(\epsilon_{12}, \hat{d}_{12} - \hat{\theta}_{12})]$.

$$\begin{aligned}
\frac{\partial^2 R_{12}}{\partial \hat{d}_{12} \partial x} &= -2P'_{12}(x - \hat{d}_{12}) - P''_{12}(x - \hat{d}_{12})(x - \hat{d}_{12}) - P''_{12}(x - \hat{d}_{12})\mathbf{E}[\min(\epsilon_{12}, \hat{d}_{12} - \hat{\theta}_{12})] \\
&\quad + P'_{12}(x - \hat{d}_{12})\bar{F}(\hat{d}_{12} - \hat{\theta}_{12})
\end{aligned}$$

If $P''_{12}(d) \leq 0$ then,

$$\begin{aligned}
\frac{\partial^2 R_{12}}{\partial \hat{d}_{12} \partial x} &= -2P'_{12}(x - \hat{d}_{12}) - P''_{12}(x - \hat{d}_{12})\mathbf{E}[\min(x - \hat{d}_{12} + \epsilon_{12}, x - \hat{\theta}_{12})] \\
&\quad + P'_{12}(x - \hat{d}_{12})\bar{F}(\hat{d}_{12} - \hat{\theta}_{12}) \\
&\geq -2P'_{12}(x - \hat{d}_{12}) + P'_{12}(x - \hat{d}_{12})\bar{F}(\hat{d}_{12} - \hat{\theta}_{12}) \\
&\geq 0
\end{aligned}$$

Where the first inequality follows since $\mathbf{E}[\min(x - \hat{d}_{12} + \epsilon_{12}, x - \hat{\theta}_{12})] \geq 0$ and the second inequality follow from $P'_{12} \leq 0$.

If $P''_{12}(d) > 0$ then,

$$\begin{aligned}\frac{\partial^2 R_{12}}{\partial \hat{d}_{12}\partial x} &= -2P'_{12}(x-\hat{d}_{12}) - P''_{12}(x-\hat{d}_{12})(x-\hat{d}_{12}) - P''_{12}(x-\hat{d}_{12})\mathbf{E}[\min(\epsilon_{12}, \hat{d}_{12}-\hat{\theta}_{12})] \\ &\quad + P'_{12}(x-\hat{d}_{12})\bar{F}(\hat{d}_{12}-\hat{\theta}_{12}) \\ &\geq -2P'_{12}(x-\hat{d}_{12}) - P''_{12}(x-\hat{d}_{12})(x-\hat{d}_{12}) + P'_{12}(x-\hat{d}_{12})\bar{F}(\hat{d}_{12}-\hat{\theta}_{12}) \\ &\geq -(P'_{12}(x-\hat{d}_{12}) + P''^{AB}(x-\hat{d}_{12})(x-\hat{d}_{12})) \\ &\geq 0\end{aligned}$$

The first inequality follows from $\mathbf{E}[\min(\epsilon_{12}, \hat{d}_{12}-\hat{\theta}_{12})] \leq \mathbf{E}[\epsilon_{12}] = 0$. The second inequality follows since $-P'_{12}(x-\hat{d}_{12}) + P'_{12}(x-\hat{d}_{12})\bar{F}(\hat{d}_{12}-\hat{\theta}_{12}) = -P'_{12}(x-\hat{d}_{12})F(\hat{d}_{12}-\hat{\theta}_{12}) \geq 0$ and the last inequality follows from condition (C1).

Deriving with respect to $\hat{d}_{12}$ and $\hat{\theta}_{12}$,

$$\begin{aligned}\frac{\partial^2 R_{12}}{\partial \hat{d}_{12}\partial \hat{\theta}_{12}} &= P'_{12}(x-\hat{d}_{12})\bar{F}(\hat{d}_{12}-\hat{\theta}_{12}) + P_{12}(x-\hat{d}_{12})F'(\hat{d}_{12}-\hat{\theta}_{12}) \\ &= -P'_{12}(x-\hat{d}_{12})\bar{F}(\hat{d}_{12}-\hat{\theta}_{12})[-1+\varrho(x-\hat{d}_{12}, x-\hat{\theta}_{12})] \\ &\geq 0\end{aligned}$$

Where the last inequality follows from condition (C2).

Deriving with respect to $\hat{\theta}_{12}$ and x,

$$\begin{aligned}\frac{\partial^2 R_{12}}{\partial \hat{\theta}_{12}\partial x} &= -P'_{12}(x-\hat{d}_{12})\bar{F}(\hat{d}_{12}-\hat{\theta}_{12}) \\ &\geq 0\end{aligned}$$

Finally, we prove the $L^\natural$-concavity of $R_{21}(\tilde{d}_{21}, \tilde{\theta}_{21}) = P_{21}(-\tilde{d}_{21})\mathbf{E}[\min(-\tilde{d}_{21}+\epsilon_{21}, -\tilde{\theta}_{21})]$ in $(\tilde{d}_{21}, \tilde{\theta}_{21})$. We do that by showing $R_{21}(\tilde{d}_{21}-\zeta, \tilde{\theta}_{21}-\zeta)$ is supermodular in $(\tilde{d}_{21}, \tilde{\theta}_{21}, \zeta)$ for $\zeta \in \{\zeta : 0 \leq \zeta \leq \hat{\theta}_{21}\}$.

Note that we can write $R_{21}(\tilde{d}_{21}-\zeta, \tilde{\theta}_{21}-\zeta)$ as $P_{21}(-\tilde{d}_{21}+\zeta)(-\tilde{d}_{21}+\zeta) + P_{21}(-\tilde{d}_{21}+\zeta)\mathbf{E}[\min(\epsilon_{21}, -\tilde{\theta}_{21}+\tilde{d}_{21})]$.

$$\begin{aligned}\frac{\partial^2 R_{21}}{\partial \tilde{d}_{21}\partial \zeta} &= -2P'_{21}(\zeta-\tilde{d}_{21}) - P''_{21}(\zeta-\tilde{d}_{21})(\zeta-\tilde{d}_{21}) - P''_{21}(\zeta-\tilde{d}_{21})\mathbf{E}[\min(\epsilon_{21}, \tilde{d}_{21}-\tilde{\theta}_{21})] \\ &\quad + P'_{21}(\zeta-\tilde{d}_{21})\bar{F}(\tilde{d}_{21}-\tilde{\theta}_{21})\end{aligned}$$

If $P''_{21}(d) \leq 0$ then,

$$\begin{aligned}
\frac{\partial^2 R_{21}}{\partial \tilde{d}_{21} \partial \zeta} &= -2P'_{21}(\zeta - \tilde{d}_{21}) - P''_{21}(\zeta - \tilde{d}_{21})\mathbf{E}[\min(\zeta - \tilde{d}_{21} + \epsilon_{21}, \zeta - \tilde{\theta}_{21})] \\
&\quad + P'_{21}(\zeta - \tilde{d}_{21})\bar{F}(\tilde{d}_{21} - \tilde{\theta}_{21}) \\
&\geq -2P'_{21}(\zeta - \tilde{d}_{21}) + P'_{21}(\zeta - \tilde{d}_{21})\bar{F}(\tilde{d}_{21} - \tilde{\theta}_{21}) \\
&\geq 0
\end{aligned}$$

Where the first inequality follows since $\mathbf{E}[\min(\zeta - \tilde{d}_{21} + \epsilon_{21}, \zeta - \tilde{\theta}_{21})] \geq 0$ and the second inequality follow from $P'_{21} \leq 0$.

If $P''_{21}(d) > 0$ then,

$$\begin{aligned}
\frac{\partial^2 R_{21}}{\partial \tilde{d}_{21} \partial \zeta} &= -2P'_{21}(\zeta - \tilde{d}_{21}) - P''_{21}(\zeta - \tilde{d}_{21})(\zeta - \tilde{d}_{21}) - P''_{21}(\zeta - \tilde{d}_{21})\mathbf{E}[\min(\epsilon_{21}, \tilde{d}_{21} - \tilde{\theta}_{21})] \\
&\quad + P'_{21}(\zeta - \tilde{d}_{21})\bar{F}(\tilde{d}_{21} - \tilde{\theta}_{21}) \\
&\geq -2P'_{21}(\zeta - \tilde{d}_{21}) - P''_{21}(\zeta - \tilde{d}_{21})(\zeta - \tilde{d}_{21}) + P'_{21}(\zeta - \tilde{d}_{21})\bar{F}(\tilde{d}_{21} - \tilde{\theta}_{21}) \\
&\geq -(P'_{21}(\zeta - \tilde{d}_{21}) + P''^{BA}(\zeta - \tilde{d}_{21})(\zeta - \tilde{d}_{21})) \\
&\geq 0
\end{aligned}$$

The first inequality follows from $\mathbf{E}[\min(\epsilon_{21}, \tilde{d}_{21} - \tilde{\theta}_{21})] \leq \mathbf{E}[\epsilon_{21}] = 0$. The second inequality follows since $-P'_{21}(\zeta - \tilde{d}_{21}) + P'_{21}(\zeta - \tilde{d}_{21})\bar{F}(\tilde{d}_{21} - \tilde{\theta}_{21}) = -P'_{21}(\zeta - \tilde{d}_{21})F(\tilde{d}_{21} - \tilde{\theta}_{21}) \geq 0$ and the last inequality follows from condition (C1).

Deriving with respect to $\tilde{d}_{21}$ and $\tilde{\theta}_{21}$,

$$\begin{aligned}
\frac{\partial^2 R_{21}}{\partial \tilde{d}_{21} \partial \hat{\theta}_{21}} &= P'_{21}(\zeta - \tilde{d}_{21})\bar{F}(\tilde{d}_{21} - \tilde{\theta}_{21}) + P_{21}(\zeta - \tilde{d}_{21})F'(\tilde{d}_{21} - \tilde{\theta}_{21}) \\
&= -P'_{21}(\zeta - \tilde{d}_{21})\bar{F}(\tilde{d}_{21} - \tilde{\theta}_{21})[-1 + \varrho(\zeta - \tilde{d}_{21}, \zeta - \tilde{\theta}_{21})] \\
&\geq 0
\end{aligned}$$

Where the last inequality follows from condition (C2).

Deriving with respect to $\tilde{\theta}_{21}$ and ζ,

$$\begin{aligned}
\frac{\partial^2 R_{21}}{\partial \tilde{\theta}_{21} \partial \zeta} &= -P'_{21}(\zeta - \tilde{d}_{21})\bar{F}(\tilde{d}_{21} - \tilde{\theta}_{21}) \\
&\geq 0
\end{aligned}$$

□

Bibliography

[1] Daniel Adelman and Adam J Mersereau. Relaxations of weakly coupled stochastic dynamic programs. *Operations Research*, 56(3):712–727, 2008.

[2] Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *International Conference on Machine Learning*, pages 146–155. PMLR, 2017.

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

[4] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

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

[6] Santiago R. Balseiro, David B. Brown, and Chen Chen. Dynamic Pricing of Relocating Resources in Large Networks. *Management Science*, page mnsc.2020.3735, 2020.

[7] Siddhartha Banerjee, Daniel Freund, and Thodoris Lykouris. Pricing and optimization in shared vehicle systems: An approximation framework. *arXiv preprint arXiv:1608.06819*, 2016.

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

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

[10] Dimitris Bertsimas and Adam J Mersereau. A learning approach for interactive marketing to a customer segment. *Operations Research*, 55(6):1120–1135, 2007.

[11] Omar Besbes, Francisco Castro, and Ilan Lobel. Surge pricing and its spatial supply response. *Management Science*, 67(3):1350–1367, 2021.

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

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

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

[15] David B Brown and James E Smith. Index policies and performance bounds for dynamic selection problems. *Management Science*, 66(7):3029–3050, 2020.

[16] David B Brown and Jingwei Zhang. On the strength of relaxations of weakly coupled stochastic dynamic programs. *Operations Research*, 2022.

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

[18] Qi (George) Chen, Yanzhe (Murray) Lei, and Stefanus Jasin. Real-Time Spatial-Intertemporal Dynamic Pricing for Balancing Supply and Demand in a Ride-Hailing Network. SSRN Scholarly Paper ID 3610517, Social Science Research Network, Rochester, NY, 2020.

[19] Xin Chen, Xiangyu Gao, and Zhenyu Hu. A new approach to two-location joint inventory and transshipment control via l-convexity. *Operations Research Letters*, 43(1):65–68, 2015.

[20] Xin Chen, Xiangyu Gao, and Zhan Pang. Preservation of structural properties in optimization with decisions truncated by random variables and its applications. *Operations Research*, 66(2):340–357, 2018.

[21] Xin Chen, Zhan Pang, and Limeng Pan. Coordinating inventory control and pricing strategies for perishable products. *Operations Research*, 62(2):284–300, 2014.

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

[23] Daniela Pucci De Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations research*, 51(6):850–865, 2003.

[24] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

[25] Ibrahim El Shar and Daniel Jiang. Lookahead-bounded q-learning. In *International Conference on Machine Learning*, pages 8665–8675. PMLR, 2020.

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

[27] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and Shan Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 117–126. IEEE, 2009.

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

[29] Jeffrey Thomas Hawkins. *A Langrangian decomposition approach to weakly coupled dynamic optimization problems and its applications*. PhD thesis, Massachusetts Institute of Technology, 2003.

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

[31] Long He, Zhenyu Hu, and Meilin Zhang. Robust repositioning for vehicle sharing. *Manufacturing & Service Operations Management*, 2019.

[32] David J Hodge and Kevin D Glazebrook. Dynamic resource allocation in a multi-product make-to-stock production system. *Queueing Systems*, 67(4):333–364, 2011.

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

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

[35] Yash Kanoria and Pengyu Qian. Blind Dynamic Resource Allocation in Closed Networks via Mirror Backpressure. *arXiv:1903.02764 [math]*, 2020.

[36] Jackson A Killian, Arpita Biswas, Sanket Shah, and Milind Tambe. Q-learning lagrange policies for multi-action restless bandits. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 871–881, 2021.

[37] Jackson A Killian, Lily Xu, Arpita Biswas, and Milind Tambe. Robust restless bandits: Tackling interval uncertainty with deep reinforcement learning. *arXiv preprint arXiv:2107.01689*, 2021.

[38] Jae Ho Kim and Warren B Powell. Optimal energy commitments with storage and intermittent supply. *Operations Research*, 59(6):1347–1360, 2011.

[39] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[40] Ayşe Kocabıyıkoğlu and Ioana Popescu. An elasticity approach to the newsvendor with price-sensitive demand. *Operations research*, 59(2):301–312, 2011.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

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

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

[44] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

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

[46] H. B. McMahan, M. Likhachev, and G. J. Gordon. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In

Proceedings of the 22nd international conference on Machine learning, pages 569–576. ACM, 2005.

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

[48] Selvaprabu Nadarajah and Andre Augusto Cire. Self-adapting network relaxations for weakly coupled markov decision processes. *Available at SSRN*, 2021.

[49] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

[50] Erhun Özkan. Joint pricing and matching in ride-sharing systems. *European Journal of Operational Research*, 287(3):1149–1160, 2020.

[51] Zhan Pang. Optimal dynamic pricing and inventory control with stock deterioration and partial backordering. *Operations Research Letters*, 39(5):375–379, 2011.

[52] Evan L Porteus. On the optimality of structured policies in countable stage decision processes. *Management Science*, 22(2):148–157, 1975.

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

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

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

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

[57] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[58] Paul J Schweitzer and Abraham Seidmann. Generalized polynomial approximations in markovian decision processes. *Journal of mathematical analysis and applications*, 110(2):568–582, 1985.

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

[60] David Simchi-Levi, Xin Chen, and Julien Bramel. *The Logic of Logistics*. Springer Series in Operations Research and Financial Engineering. Springer New York, New York, NY, 2014.

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

[62] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

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

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

[65] Kalyan Talluri and Garrett Van Ryzin. An analysis of bid-price controls for network revenue management. *Management science*, 44(11-part-1):1577–1593, 1998.

[66] Huseyin Topaloglu. Using lagrangian relaxation to compute capacity-dependent bid prices in network revenue management. *Operations Research*, 57(3):637–649, 2009.

[67] Huseyin Topaloglu and Sumit Kunnumkal. Approximate dynamic programming methods for an inventory allocation problem under uncertainty. *Naval Research Logistics (NRL)*, 53(8):822–841, 2006.

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

[69] Tom Van de Wiele, David Warde-Farley, Andriy Mnih, and Volodymyr Mnih. Q-learning in enormous action spaces via amortized approximate maximization. *arXiv preprint arXiv:2001.08116*, 2020.

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

[71] H. van Hasselt, A. R. Guez, and D. Silver. Deep reinforcement learning with double Q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[72] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[73] Ariel Waserhole and Vincent Jost. Pricing in vehicle sharing systems: Optimization in queuing networks with product forms. *EURO Journal on Transportation and Logistics*, 5(3):293–320, 2016.

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

[75] Peter Whittle. Restless bandits: Activity allocation in a changing world. *Journal of applied probability*, 25(A):287–298, 1988.

[76] Zhe Yu, Yunjian Xu, and Lang Tong. Deadline scheduling as restless bandits. *IEEE Transactions on Automatic Control*, 63(8):2343–2358, 2018.

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