AN ABSTRACT OF THE DISSERTATION OF

Majid Alkaee Taleghan for the degree of Doctor of Philosophy in Computer Science presented on January 3, 2017.

Title: Simulator-Defined MDP Planning with Applications in Natural Resource Management

Abstract approved: _____

Thomas G. Dietterich

This work is inspired by problems in natural resource management centered on the challenge of invasive species. Computing optimal management policies for maintaining ecosystem sustainable is challenging. Many ecosystem management problems can be formulated as MDP (Markov Decision Process) planning problems. In a simulator-defined MDP, the Markovian dynamics and rewards are provided in the form of a simulator from which samples can be drawn. Simulators in natural resource management can be very expensive to execute, so that the time required to solve such MDPs is dominated by the number of calls to the simulator. This thesis studies MDP planning algorithms that attempt to minimize the number of simulator calls before terminating and outputting a policy that is approximately optimal with high probability. This thesis addresses three questions on unconstrained MDPs: (a) what confidence interval should be employed to bound the optimality of the policy and (b) how should samples be drawn to shrink the confidence interval as quickly as possible? (c) how can we find the optimal policy with high probability efficiently? Many computational sustainability problems involving MDPs must also be concerned with catastrophic outcomes such as species extinction. These problems can be formulated as constrained MDPs. We define the downside risk as the probability of reaching catastrophic states and then constrain the MDP solution to bound the probability of entering such states. We then develop the first PAC-Safe-RL algorithm for constrained MDPs. We evaluate our algorithms on an invasive species problem as well as on standard reinforcement learning benchmarks.

Simulator-Defined MDP Planning with Applications in Natural Resource Management

by

Majid Alkaee Taleghan

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented January 3, 2017
Commencement June 2017

Doctor of Philosophy dissertation of Majid Alkaee Taleghan presented on January 3, 2017.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Majid Alkaee Taleghan, Author

ACKNOWLEDGEMENTS

My journey through my Ph.D. studies would not have been possible without the help of many people.

First and foremost, I would like to thank my advisor, Professor Thomas Dietterich, for his kindness, guidance, time, insights, profound thinking, and continuous support over the years. I was fortunate to have had the opportunity to work with him. I learned not only how to do ethical research, but also lessons for my life. I could not have wished for a better mentor, and I am deeply grateful for that.

I would also like to thank Kim Hall, Professor H. Jo Albers, and Mark Crowley for their collaboration on the invasive species project. It has been a pleasure working together, and I have learned a lot from you.

Next, I want to thank my Ph.D. committee, Professors Prasad Tadepalli, Xioali Fern, Rebecca Hutchinson, and John Dilles, for their constructive comments on my research.

I would like to acknowledge all the members of Professor Dietterich's group, who provided me with useful thoughts and ideas throughout my graduate studies. Special thanks to Sean McGregor, Liping Liu, Jesse Hostetler, and Shahed Sorower.

I would also like to thank all of the staff and faculty at the EECS department who have helped me in countless ways. My special thanks to Mike Sanders for always accommodating my needs to run my experiments.

I wish to acknowledge the support of the US National Science Foundation, who funded much of this work under Computational Sustainability Grant 0832804 and Grant 1331932.

I am indebted to many friends and colleagues for their companionship during these years. Special thanks to Amir Taherkordi, Javad Azimi, Hashem Baktash, Behrouz Behmardi, and Soroush Ghorashi.

I'd like to thank all my family (every one of you included!), and express my deep gratitude to my parents for their unconditional support and encouragement. I am especially grateful to my mother, who has always believed in me and motivated me to successfully complete my Ph.D.

Finally, I want to express my appreciation for my dear wife, Sarah Ghasedi, who has been closest to me all these years and shared happy moments with me. Her inspiration and love in the toughest moments and her smart advice helped me get through this. I am lucky to have her in my life.

CONTRIBUTION OF AUTHORS

The author would like to thank Kim Hall, H. Jo Albers, and Mark Crowley for their collaboration on the invasive species project. They have contributed to Chapter 2.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF ALGORITHMS

Dedicated to my inspiring and loving parents

Chapter 1: Introduction

Many natural resource management problems involve promoting or preventing spread through spatial networks. Preventing the spread of invasives [7, 31], the recovery of an endangered species [59], and encouraging the spread of low-intensity ground fires while preventing high-intensity crown fires that destroy the habitat of critically-endangered species [34] are examples of ecosystem management problems. The motivation of this work is the spread of tamarisk in river networks [7, 31]. The tamarisk plant (*Tamarix* spp.) is a native of the Middle East. It has become an invasive plant in the dryland rivers and streams of the western US [13, 61]. It out-competes native vegetation primarily by producing large numbers of seeds. Given an ongoing tamarisk invasion, a manager must repeatedly decide how and where to fight the invasion (e.g., eradicate tamarisk plants? plant native plants? upstream? downstream?). Computing optimal management policies for maintaining sustainable ecosystems is challenging, because they exhibit spatial and temporal interactions at multiple scales [11].

These kinds of problems can be formulated as Markov Decision Problems (MDPs). Since it is not feasible to explicitly specify the mathematical formulations of the system or the MDP model parameters, the system dynamics are represented by simulation models [9]. In a simulator-defined MDP, the Markovian dynamics and rewards are provided in the form of a simulator from which samples can be drawn. Simulators in natural resource management can be very expensive to execute, so that the time required to solve such MDPs is dominated by the number of calls to the simulator. This is particularly true when the simulator was developed for some other purpose where fidelity was much more important than computational efficiency. One example, reported by Houtman et al. [34], concerns the FARSITE [25] model for simulating wildfires. It was initially developed for tactical (same-day) fire fighting, but Houtman, et al. apply it for policy evaluation over 100-year horizons. It takes FARSITE anywhere from several minutes to an hour to simulate a single wildfire, so simulating a 100-year trajectory containing hundreds of fires is very expensive.

A naive approach to solving simulator-defined MDP planning problems is to invoke the simulator a sufficiently large number of times in every state-action pair to estimate the transition probability matrix and then apply standard MDP planning algorithms to compute a PAC-optimal

policy. While this is required in the worst case (c.f., [3]), there are two sources of constraint that algorithms can exploit to reduce simulator calls. First, the transition probabilities in the MDP may be sparse so that only a small fraction of states are directly reachable from any given state. Second, in MDP planning problems, there is a designated starting state s_0, and the goal is to find an optimal policy for acting in that state and in all states *reachable* from that state. In the case where the optimality criterion is cumulative *discounted* reward, an additional constraint is that the algorithm only need to consider states that are reachable within a fixed horizon, because rewards far in the future have no significant impact on the value of the starting state.

To create a good MDP-planning algorithm, there are two critical design questions: (a) what confidence interval should be employed to bound the optimality of the policy? and (b) how should samples be drawn to shrink this confidence interval as quickly as possible. Most existing work [22, 17, 62, 35] answers the first question by computing a "local" confidence interval for each state-action pair and then combining these intervals via value iteration. Specifically, Even-Dar, Mannor, and Mansour [17] apply the Hoeffding bound at each state. Strehl and Littman [62] apply a multinomial confidence interval developed by Weissman et al. [73] at each state. One can easily apply the empirical Bernstein bound [2] at each state as well.

To answer the second question, most existing work [62, 35] uses the upper confidence bound $\overline{Q}(s, a)$ for each state-action pair to compute the upper confidence bound policy π^{UCB} and then draws samples by executing that policy along trajectories. This "optimism under uncertainty" has been shown both theoretically and experimentally to give good exploration. However, there is no particular reason to think that exploration should take place along trajectories. If we have a "strong" simulator—that is, a simulator that can provide samples from arbitrary state-action pairs in any order—then there could be advantages to focusing sampling on particular parts of the state-action space.

An alternative to local confidence intervals are "global" (or "trajectory-wise") confidence intervals. Given a set of trajectories generated according to a fixed policy π, we can compute the return from each trajectory and then apply either the Hoeffding bound [32] or the empirical Bernstein bound [2] to compute a confidence interval on the expected return of the fixed policy π. Such intervals can be much tighter, because, unlike the local intervals, they do not need to propagate the error by dynamic programming nor do they need to invoke a union bound to ensure that each of the individual local intervals holds simultaneously. Chapter 3 studies theoretical and experimental analysis of local and global confidence intervals methods.

For modest-sized instances of these problems, model-based MDP planning algorithms can

be applied. During the planning phase, the algorithm can invoke a simulator to obtain samples of the transitions and rewards. Simulators in these problems typically model the system to high fidelity and, hence, are very expensive to execute. Consequently, the time required to solve such MDPs is dominated by the number of calls to the simulator. Although approximate or heuristic solutions would be useful, our collaborating economists tell us that our policy recommendations will carry more weight if they are provably optimal with high probability.

In addition to above motivations for unconstrained MDPs, in many practical scenarios, such as natural resource management, a desirable policy needs to satisfy certain constraints imposed by decision makers. In these scenarios, maximizing the expected reward does not necessarily avoid rare catastrophic or dangerous situations. For example, in conservation problems, catastrophic outcomes include species extinction, long-term establishment of an invasive species, and severe wildfires. A standard approach to finding policies that avoid catastrophic states is to assign a large negative reward to those states [26, 27]. This is equivalent to a so-called Big M method for establishing a lexicographic preference for policies that do not enter catastrophic states. However, this approach does not quantify the risk (probability) of entering a catastrophic state, nor does it determine whether there are policies that control this risk. A better approach is to adopt the Constrained MDP (C-MDP) formalism [1], which seeks to maximize one objective (e.g., economic value) while satisfying one or more constraints probabilistically. For example, in invasive species management, we can define a C-MDP to minimize the economic cost of invasive species management while ensuring that the probability of native species extinction is less than a specified threshold.

Recently, Geibel and Wysotzki [27] developed a model-free Q-learning algorithm for C-MDPs. Their formulation is applicable to episodic tasks with a combination of absorbing catastrophic and goal states. As Geramifard [28] pointed out, the Geibel, et al., work does not provide a performance guarantee on the result.

An alternative to constrained MDPs is to consider risk-sensitive objectives such as variance penalties, value at risk (VaR), and conditional value at risk (CVaR) [26, 1]. Var and CVar optimize the α-quantile of the expected return, and CVaR has favorable mathematical properties. While these are all very interesting approaches, we find the constrained MDP formulation easier to understand and explain to stakeholders, and for this reason, we focus our efforts on C-MDPs.

A drawback of C-MDPs is that the optimal policy can be stochastic in some cases. Specifically, if there are c constraints, then the optimal policy may be stochastic in up to c states. From the perspective of our stakeholders, this stochastic behavior is confusing and undesirable. Hence,

in this thesis, we aim to find a stationary deterministic policy that satisfies a downside risk constraint as well as maximizing the discounted reward. We seek to do this while economizing on the number of calls to the simulator and while providing PAC guarantees both that the constraints are satisfied and that the resulting policy is within a fixed bound of optimality. This provides the first PAC-RL algorithm for deterministic policies in C-MDPs.

1.1 Contribution

Chapter 2 proposes the following five improvements:

1. Instead of exploring along trajectories, we take advantage of the fact that our simulators can be invoked for any state-action pair in any order. Hence, our algorithms perform fine-grained exploration where they iteratively select the state-action pair that they believe will be most informative.

2. By not exploring along trajectories (rooted at the start state), we could potentially lose the guarantee that the algorithm only explores states that are reachable from the start state. We address this by maintaining an estimate of the discounted state occupancy measure. This measure is non-zero only for states reachable from the start state. We also use the occupancy measure in our exploration heuristics.

3. We adopt an extension to the termination condition introduced by Even-Dar et al. [15, 17], which is the width of a confidence interval over the optimal value of the start state. We halt when the width of the confidence interval is less than ϵ, the desired accuracy bound.

4. We replace the Hoeffding-bound confidence intervals employed by Fiechter [22] (and others) with the multinomial confidence intervals of Weissman, Ordentlich, Seroussi, Verdu, and Weinberger [73] employed in the MBIE algorithm of Strehl and Littman [62].

5. To take advantage of sparse transition functions, we incorporate an additional confidence interval for the Good-Turing estimate of the "missing mass" (the total probability of all unobserved outcomes for a given state-action pair). This confidence interval can be easily combined with the Weissman et al. interval.

Chapter 3 makes three main contributions:

1. It develops optimal sampling strategies for Monte Carlo policy evaluation using both local and trajectory-wise forms of the Hoeffding and empirical Bernstein bounds.

2. It provides experimental evidence that for policy evaluation the trajectory-wise bounds generally out-perform the local bounds except in a few special cases.

3. It introduces two new MDP planning algorithms, LGCV and LLGCV, for simulator-defined MDPs. These algorithms combine trajectory-wise confidence intervals with local confidence intervals to reduce the number of samples needed to achieve target levels of accuracy. The LGCV algorithm combines a local upper bound with a global lower bound. Although this performs well on some problems, it exhibits poor performance on others. The LLGCV algorithm extends LGCV by intersecting both the local and global lower bounds. Although this requires computing additional confidence intervals, it eliminates LGCV's failure cases. Hence, LLGCV provides a robust method that can provide significant improvements over previous methods.

Chapter 4 makes two main contributions:

1. It finds a stationary deterministic policy that satisfies a downside risk constraint as well as maximizing the discounted reward. It does this while economizing on the number of calls to the simulator and while providing PAC guarantees both that the constraints are satisfied and that the resulting policy is within a fixed bound of optimality.

2. It provides the first PAC-Safe-RL algorithm for deterministic policies in C-MDPs.

1.2 Organization

The thesis is composed of three manuscripts; the first one has been published the Journal of Machine Learning Research. The second and third manuscripts are ready to be submitted to relevant machine learning journals and conferences. The first and second manuscripts focus on sample-efficient algorithms for unconstrained MDPs. The first manuscript proposes a sampling heuristic for local confidence intervals. The second manuscript makes a theoretical analysis of Monte-Carlo policy evaluation, and proposes a new PAC-optimal algorithm combining global and local confidence bounds. The third manuscript extends the ideas in the first two manuscripts to C-MDPs.

PAC Optimal MDP Planning with Application to Invasive Species Management

Majid Alkaee Taleghan, Thomas G. Dietterich, Mark Crowley, Kim Hall, H. Jo Albers
Journal of Machine Learning Research
16 (Dec), 3877-3903

Chapter 2: PAC Optimal MDP Planning with Application to Invasive Species Management

In a simulator-defined MDP, the Markovian dynamics and rewards are provided in the form of a simulator from which samples can be drawn. This chapter studies MDP planning algorithms that attempt to minimize the number of simulator calls before terminating and outputting a policy that is approximately optimal with high probability. The chapter introduces two heuristics for efficient exploration and an improved confidence interval that enables earlier termination with probabilistic guarantees. We prove that the heuristics and the confidence interval are sound and produce with high probability an approximately optimal policy in polynomial time. Experiments on two benchmark problems and two instances of an invasive species management problem show that the improved confidence intervals and the new search heuristics yield reductions of between 8% and 47% in the number of simulator calls required to reach near-optimal policies.[1]

2.1 Introduction

The motivation for this chapter is the area of ecosystem management in which a manager seeks to maintain the healthy functioning of an ecosystem by taking actions that promote the persistence and spread of endangered species or actions that fight the spread of invasive species, fires, and disease. Most ecosystem management problems can be formulated as MDP (Markov Decision Process) planning problems with separate planning and execution phases. During the planning phase, the algorithm can invoke a simulator to obtain samples of the transitions and rewards. Simulators in these problems typically model the system to high fidelity and, hence, are very expensive to execute. Consequently, the time required to solve such MDPs is dominated by the number of calls to the simulator. A good MDP planning algorithm minimizes the number of calls to the simulator and yet terminates with a policy that is approximately optimal with high probability. This is referred to as being PAC-RL [22].

Because of the separation between the exploration phase (where the simulator is invoked

[1]Portions of this work appeared in Proceedings of Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI-2013)

and a policy is computed) and the exploitation phase (where the policy is executed in the actual ecosystem), we refer to these ecosystem management problems as problems of *MDP Planning* rather than of *Reinforcement Learning*. In MDP planning, we do not need to resolve the exploration-exploitation tradeoff.

Another aspect of these MDP planning problems that distinguishes them from reinforcement learning is that the planning algorithm must decide when to terminate and output a PAC-optimal policy. Many reinforcement learning algorithms, such as Sparse Sampling [39], FSSS [72], MBIE [62], and UCRL2 [35] never terminate. Instead, their performance is measured in terms of the number of "significantly non-optimal actions" (known as PAC-MDP, [36]) or cumulative regret [35].

A final aspect of algorithms for ecosystem management problems is that they must produce an explicit policy in order to support discussions with stakeholders and managers to convince them to adopt and execute the policy. Hence, receding horizon search methods, such as Sparse Sampling and FSSS, are not appropriate because they do not compute an explicit policy.

A naive approach to solving simulator-defined MDP planning problems is to invoke the simulator a sufficiently large number of times in every state-action pair and then apply standard MDP planning algorithms to compute a PAC-optimal policy. While this is required in the worst case (c.f., [3]), there are two sources of constraint that algorithms can exploit to reduce simulator calls. First, the transition probabilities in the MDP may be sparse so that only a small fraction of states are directly reachable from any given state. Second, in MDP planning problems, there is a designated starting state s_0, and the goal is to find an optimal policy for acting in that state and in all states *reachable* from that state. In the case where the optimality criterion is cumulative *discounted* reward, an additional constraint is that the algorithm only need to consider states that are reachable within a fixed horizon, because rewards far in the future have no significant impact on the value of the starting state.

It is interesting to note that the earliest PAC-optimal algorithm published in the reinforcement learning community was in fact an MDP planning algorithm: the method of Fiechter [22] addresses exactly the problem of making a polynomial number of calls to the simulator and then outputting a policy that is approximately correct with high probability. Fiechter's method works by exploring a series of trajectories, each of which begins at the start state and continues to a fixed-depth horizon. By exploring along trajectories, this algorithm ensures that only reachable states are explored. And by terminating the exploration at a fixed horizon, it exploits discounting.

Our understanding of reinforcement learning has advanced considerably since Fiechter's

work. This chapter can be viewed as applying these advances to develop "modern" MDP planning algorithms. Specifically, we introduce the following five improvements:

1. Instead of exploring along trajectories, we take advantage of the fact that our simulators can be invoked for any state-action pair in any order. Hence, our algorithms perform fine-grained exploration where they iteratively select the state-action pair that they believe will be most informative.

2. By not exploring along trajectories (rooted at the start state), we could potentially lose the guarantee that the algorithm only explores states that are reachable from the start state. We address this by maintaining an estimate of the discounted state occupancy measure. This measure is non-zero only for states reachable from the start state. We also use the occupancy measure in our exploration heuristics.

3. We adopt an extension to the termination condition introduced by Even-Dar et al. [15, 17], which is the width of a confidence interval over the optimal value of the start state. We halt when the width of the confidence interval is less than ϵ, the desired accuracy bound.

4. We replace the Hoeffding-bound confidence intervals employed by Fiechter (and others) with the multinomial confidence intervals of Weissman, Ordentlich, Seroussi, Verdu, and Weinberger [73] employed in the MBIE algorithm of Strehl and Littman [62].

5. To take advantage of sparse transition functions, we incorporate an additional confidence interval for the Good-Turing estimate of the "missing mass" (the total probability of all unobserved outcomes for a given state-action pair). This confidence interval can be easily combined with the Weissman et al. interval.

This chapter is organized as follows. Section 2 introduces our notation. Section 3 describes a particular ecosystem management problem—control of the invasive plant tamarisk—and its formulation as an MDP. Section 4 reviews previous work on sample-efficient MDP planning and describes in detail the algorithms against which we will evaluate our new methods. Section 5 presents the technical contributions of the chapter. It introduces our improved confidence intervals, proves their soundness, and presents experimental evidence that they enable earlier termination than existing methods. It then describes two new exploration heuristics, proves that they achieve polynomial sample size, and presents experimental evidence that they are more effective than previous heuristics. Section 6 concludes the chapter.

2.2 Definitions

We employ the standard formulation of an infinite horizon discounted Markov Decision Process (MDP; Bellman 4, Puterman 55) with a designated start state distribution. Let the MDP be defined by $\mathcal{M} = \langle S, A, P, R, \gamma, P_0 \rangle$, where S is a finite set of (discrete) states of the world; A is a finite set of possible actions that can be taken in each state; $P : S \times A \times S \mapsto [0, 1]$ is the conditional probability of entering state s' when action a is executed in state s; $R(s, a)$ is the (deterministic) reward received after performing action a in state s; $\gamma \in (0, 1)$ is the discount factor, and P_0 is the distribution over starting states. It is convenient to define a special starting state s_0 and action a_0 and define $P(s|s_0, a_0) = P_0(s)$ and $R(s_0, a_0) = 0$. We assume that $0 \leq R(s, a) \leq R_{max}$ for all s, a. Generalization of our methods to (bounded) stochastic rewards is straightforward.

A *strong simulator* (also called a *generative model*) is a function $F : S \times A \mapsto S \times \Re$ that given (s, a) returns (s', r) where s' is sampled according to $P(s'|s, a)$ and $r = R(s, a)$.

A (deterministic) policy is a function from states to actions, $\pi : S \mapsto A$. The value of a policy π at the starting state is defined as $V^\pi(s_0) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))]$, where the expectation is taken with respect to the stochastic transitions. The maximum possible $V^\pi(s_0)$ is denoted $V_{max} = R_{max}/(1 - \gamma)$. An optimal policy π^* maximizes $V^\pi(s_0)$, and the corresponding value is denoted by $V^*(s_0)$. The action-value of state s and action a under policy π is defined as $Q^\pi(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a)V^\pi(s')$. The optimal action-value is denoted $Q^*(s, a)$.

Define $pred(s)$ to be the set of states s^- such that $P(s|s^-, a) > 0$ for at least one action a and $succ(s, a)$ to be the set of states s' such that $P(s'|s, a) > 0$.

Definition 1 *Fiechter [22]. A learning algorithm is PAC-RL*[2] *if for any discounted MDP defined by* $\langle S, A, P, R, \gamma, P_0 \rangle$, $\epsilon > 0$, $1 > \delta > 0$, *and* $0 \leq \gamma < 1$, *the algorithm halts and outputs a policy* π *such that*

$$\mathbb{P}[|V^*(s_0) - V^\pi(s_0)| \leq \epsilon] \geq 1 - \delta,$$

in time polynomial in $|S|$, $|A|$, $1/\epsilon$, $1/\delta$, $1/(1 - \gamma)$, *and* R_{max}.

As a learning algorithm explores the MDP, it collects the following statistics. Let $N(s, a)$ be

[2] In retrospect, it would have been better if Fiechter had called this PAC-MDP, because he is doing MDP planning. In turn, PAC-MDP has come to refer to *reinforcement learning* algorithms with polynomial time or regret bounds, which would be more appropriately called PAC-RL algorithms. At some point, the field should swap the meaning of these two terms.

the number of times the simulator has been called with state-action pair (s, a). Let $N(s, a, s')$ be the number of times that s' has been observed as the result. Let $R(s, a)$ be the observed reward.

2.3 Managing Tamarisk Invasions in River Networks

The tamarisk plant (*Tamarix* spp.) is a native of the Middle East. It has become an invasive plant in the dryland rivers and streams of the western US [13, 61]. It out-competes native vegetation primarily by producing large numbers of seeds. Given an ongoing tamarisk invasion, a manager must repeatedly decide how and where to fight the invasion (e.g., eradicate tamarisk plants? plant native plants? upstream? downstream?).

A stylized version of the tamarisk management problem can be formulated as an MDP as follows. The state of the MDP consists of a tree-structured river network in which water flows from the leaf nodes toward the root (see Figure 2.1). The network contains E edges. Each edge in turn has H slots at which a plant can grow. Each slot can be in one of three states: empty, occupied by a tamarisk plant, or occupied by a native plant. In this stylized model, because the exact physical layout of the H slots within each edge is unimportant, the state of the edge can be represented using only the number of slots that are occupied by tamarisk plants and the number of slots occupied by native plants. The number of empty slots can be inferred by subtracting these counts from H. Hence, each edge can be in one of $(H + 1)(H + 2)/2$ states. Consequently, the total number of states in the MDP is $E^{(H+1)(H+2)/2}$.

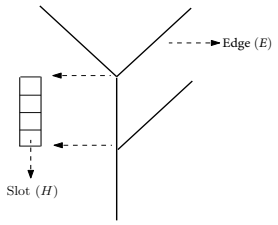


Figure 2.1: Tamarisk structure

The dynamics are defined as follows. In each time step, each plant (tamarisk or native) dies with probability 0.2. The remaining plants each produce 100 seeds. The seeds then disperse according to a spatial process such that downstream spread is much more likely than upstream spread. We employ the dispersal model of Muneepeerakul et al. [48, Appendix B] with an upstream parameter of 0.1 and a downstream parameter of 0.5. An important aspect of the dispersal model is that there is a non-zero probability for a propagule to travel from any edge to any other edge. Each propagule that arrives at an edge lands in a slot chosen uniformly at random. Hence, after dispersal, each propagule has landed in one of the slots in the river network. The seeds that arrive at an occupied slot die and have no effect. The seeds that arrive

at an empty slot compete stochastically to determine which one will occupy the site and grow. In the MDPs studied in this chapter, this competition is very simple: one of the arriving seeds is chosen uniformly at random to occupy the slot.

Many variations of the model are possible. For example, we can allow the tamarisk plants to be more fecund (i.e., produce more seeds) than the native plants. The seeds can have differential competitive advantage. The plants can have differential mortality, and so on. One variation that we will employ in one of our experiments is to include "exogenous arrivals" of tamarisk seeds. This models the process by which new seeds are introduced to the river network from some external source (e.g., fishermen transporting seeds on their clothes or equipment). Specifically, in the exogenous arrivals condition, in addition to the seeds that arrive at an edge via dispersal, up to 10 additional seeds of each species arrive in each edge. These are sampled by taking 10 draws from a Bernoulli distribution for each species. For tamarisk, the Bernoulli parameter is 0.1; for the native seeds, the Bernoulli parameter is 0.4.

The dynamics can be represented as a very complex dynamic Bayesian network (DBN). However, inference in this DBN is intractable, because the induced tree width is immense. One might hope that methods from the factored MDP literature could be applied, but the competition between the seeds that arrive at a given slot means that every slot is a parent of every other slot, so there is no sparseness to be exploited. An additional advantage of taking a simulation approach is that our methods can be applied to any simulator-defined MDP. We have therefore constructed a simulator that draws samples from the DBN. Code for the simulator can be obtained from `http://2013.rl-competition.org/domains/invasive-species`.

The actions for the management MDP are defined as follows. At each time step, one action can be taken in each edge. The available actions are "do nothing", "eradicate" (attempt to kill all tamarisk plants in all slots in the edge), and "restore" (attempt to kill all tamarisk plants in all slots in the edge and then plant native plants in every empty slot). The effects are controlled by two parameters: the probability that killing a tamarisk plant succeeds ($\chi = 0.85$) and the probability that planting a native plant in an empty slot succeeds ($\beta = 0.65$). Taken together, the probability that the "restore" action will change a slot from being occupied by a tamarisk plant to being occupied by a native plant is the product $\chi \times \beta = 0.5525$. Because these actions can be taken in each edge, the total number of actions for the MDP is 3^E. However, we will often include a budget constraint that makes it impossible to treat more than one edge per time step.

The reward function assigns costs as follows. There is a cost of 1.0 for each edge that is invaded (i.e., that has at least one slot occupied by a tamarisk plant) plus a cost of 0.1 for each

slot occupied by a tamarisk plant. The cost of applying an action to an edge is 0.0 for "do nothing", 0.5 for "eradicate", and 0.9 for "restore".

The optimization objective is to minimize the infinite horizon discounted sum of costs. However, for notational consistency we will describe our algorithms in terms of maximizing the discounted sum of rewards throughout the chapter.

It is important to note that in real applications, all of the parameters of the cost function and transition dynamics may be only approximately known, so another motivation for developing sample-efficient algorithms is to permit experimental analysis of the sensitivity of the optimal policy to the values of these parameters. The techniques employed in this chapter are closely-related to those used to compute policies that are robust to these uncertainties [44, 67].

Now that we have described our motivating application problem, we turn our attention to developing efficient MDP planning algorithms. We start by summarizing previous research.

2.4 Previous Work on Sample-Efficient MDP Planning

Fiechter [22] first introduced the notion of PAC reinforcement learning in Definition 1 and presented the PAC-RL algorithm shown in Figure 1. Fiechter's algorithm defines a measure of uncertainty $\tilde{d}_h^\pi(s)$, which with high probability is an upper bound on the difference $|V_h^*(s) - V_h^\pi(s)|$ between the value of optimal policy and the value of the "maximum likelihood" policy that would be computed by value iteration using the current transition probability estimates. The subscript h indicates the depth of state s from the starting state. Fiechter avoids dealing with loops in the MDP by computing a separate transition probability estimate for each combination of state, action and depth (s, a, h) up to $h \leq H$, where H is the maximum depth ("horizon") at which estimates are needed. Hence, the algorithm maintains separate counts $N_h(s, a, s')$ and $N_h(s, a)$ to record the results of exploration for each depth h. To apply this algorithm in practice, Fiechter [23] modifies the algorithm to drop the dependency of the related statistics on h.

Fiechter's algorithm explores along a sequence of trajectories. Each trajectory starts at state s_0 and depth 0 and follows an exploration policy π^e until reaching depth H. The exploration policy is the optimal policy for an "exploration MDP" whose transition function is $P_h(s'|s, a)$ but whose reward function for visiting state s at depth h is equal to

$$R_h(s, a) = \frac{6}{\epsilon} \frac{V_{max}}{1 - \delta} \sqrt{\frac{2 \ln 4H|S||A| - 2 \ln \delta}{N_h(s, a)}}.$$

This reward is derived via an argument based on the Hoeffding bound. The transition probabilities $P_h(s'|s,a)$ are computed from the observed counts.

The quantity $d^{\pi^e}(s)$ is the value function corresponding to π^e. Because the MDP is stratified by depth, π^e and d^{π_e} can be computed in a single sweep starting at depth H and working backward to depth 0. The algorithm alternates between exploring along a single trajectory and recomputing π^e and d^{π^e}. It halts when $d_0^{\pi^e}(s_0) \leq 2/(1-\gamma)$. By exploring along π^e, the algorithm seeks to visit a sequence of states whose total uncertainty is maximized in expectation.

Algorithm 1 Fiechter($s_0, \gamma, F, \epsilon, \delta$)

Input: s_0: start state; γ: discount rate; F: a simulator

Initialization:

- $H = \left\lceil \frac{1}{1-\gamma}\left(\ln V_{max} + \ln \frac{6}{\epsilon}\right)\right\rceil$ // horizon depth
- **for** $s, s' \in S, a \in A(s), h = 0, \ldots, H-1$ **do**
  - $N_h(s,a) = 0$
  - $N_h(s,a,s') = 0$
  - $R_h(s,a,s') = 0$
  - $\pi_h^e(s) = a_1$

Exploration:

- **while** $d_0^{\pi^e}(s_0) > 2/(1-\gamma)$ **do**
  - **reset** $h = 0$ and $s = s_0$
  - **while** $h < H$ **do**
    - $a = \pi_h^e(s)$
    - $(r, s') \sim F(s,a)$ // draw sample
    - **update** $N_h(s,a)$, $N_h(s,a,s')$, and $R_h(s,a,s')$
    - $h = h + 1$
    - $s = s'$
  - Compute new policy π^e (and values d^{π^e}) using the following dynamic program $d_{max} = (12V_{max})/(\epsilon(1-\gamma))$
  - $P_h(s'|s,a) = N_h(s,a,s')/N_h(s,a)$
  - $d_H^{\pi^e}(s) = 0, \forall s \in S$
  - **for** $h = H-1, \ldots, 0$ **do**
    - $e_h^{\pi^e}(s,a) = \min\left\{d_{max}, \frac{6}{\epsilon}\frac{V_{max}}{1-\delta}\sqrt{\frac{2\ln 4H|S||A| - 2\ln\delta}{N_h(s,a)}} + \gamma\sum_{s' \in succ(s,a)} P_h(s'|s,a) d_{h+1}^{\pi^e}(s')\right\}$
    - $\pi_h^e(s) = \text{argmax}_{a \in A(s)}\, e_h^{\pi^e}(s,a)$
    - $d_h^{\pi^e}(s) = e_h^{\pi^e}(s, \pi_h^e(s))$

Compute policy π, and return it.

A second important inspiration for our work is the Model-Based Action Elimination (MBAE) algorithm of Even-Dar et al. [15, 17]. Their algorithm maintains confidence intervals $Q(s,a) \in [Q_{lower}(s,a), Q_{upper}(s,a)]$ on the action-values for all state-action pairs in the MDP. These confidence intervals are computed via "extended value iteration" that includes an additional term derived from the Hoeffding bounds:

$$Q_{upper}(s,a) = R(s,a) + \gamma \sum_{s'} \hat{P}(s'|s,a) V_{upper}(s') + V_{max} \sqrt{\frac{\ln ct^2|S||A| - \ln \delta}{|N(s,a)|}} \quad (2.1)$$

$$V_{upper}(s) = \max_a Q_{upper}(s,a) \quad (2.2)$$

$$Q_{lower}(s,a) = R(s,a) + \gamma \sum_{s'} \hat{P}(s'|s,a) V_{lower}(s') - V_{max} \sqrt{\frac{\ln ct^2|S||A| - \ln \delta}{|N(s,a)|}} \quad (2.3)$$

$$V_{lower}(s) = \max_a Q_{lower}(s,a). \quad (2.4)$$

In these equations, t is a counter of the number of times that the confidence intervals have been computed and c is an (unspecified) constant. Even-Dar et al. prove that the confidence intervals are sound. Specifically, they show that with probability at least $1-\delta$, $Q_{lower}(s,a) \leq Q^*(s,a) \leq Q_{upper}(s,a)$ for all s, a, and iterations t.

Their MBAE algorithm does not provide a specific exploration policy. Instead, the primary contribution of their work is to demonstrate that these confidence intervals can be applied as a termination rule. Specifically, if for all (s,a), $|Q_{upper}(s,a) - Q_{lower}(s,a)| < \frac{\epsilon(1-\gamma)}{2}$, then the policy that chooses actions to minimize $Q_{lower}(s,a)$ is ϵ-optimal with probability at least $1-\delta$. Note that the iteration over s' in these equations only needs to consider the observed transitions, as $\hat{P}(s'|s,a) = 0$ for all transitions where $N(s,a,s') = 0$.

An additional benefit of the confidence intervals is that any action a' can be eliminated from consideration in state s if $Q_{upper}(s,a') < Q_{lower}(s,a)$. Even-Dar et al. demonstrate experimentally that this can lead to faster learning than standard Q learning (with either uniform random action selection or ϵ-greedy exploration).

The third important source of ideas for our work is the Model-Based Interval Estimation (MBIE) algorithm of Strehl and Littman [62]. MBIE maintains an upper confidence bound on the action-value function, but unlike Fiechter and Even-Dar et al., this bound is based on a confidence region for the multinomial distribution developed by Weissman et al. [73].

Let $\hat{P}(s'|s,a) = N(s,a,s')/N(s,a)$ be the maximum likelihood estimate for $P(s'|s,a)$,

and let $\hat{P}$ and $\tilde{P}$ denote $\hat{P}(\cdot|s,a)$ and $\tilde{P}(\cdot|s,a)$. Define the confidence set CI as

$$CI(\hat{P}|N(s,a),\delta) = \left\{ \tilde{P} \;\middle|\; \|\tilde{P} - \hat{P}\|_1 \leq \omega(N(s,a),\delta) \right\}, \tag{2.5}$$

where $\|\cdot\|_1$ is the L_1 norm and $\omega(N(s,a),\delta) = \sqrt{\frac{2[\ln(2^{|S|}-2)-\ln\delta]}{N(s,a)}}$. The confidence interval is an L_1 "ball" of radius $\omega(N(s,a),\delta)$ around the maximum likelihood estimate for P. Weissman et al. [73] prove that with probability $1-\delta$, $P(\cdot|s,a) \in CI(\hat{P}(\cdot|s,a)|N(s,a),\delta)$.

Given confidence intervals for all visited (s,a), MBIE computes an upper confidence bound on Q and V as follows. For any state where $N(s,a) = 0$, define $Q_{upper}(s,a) = V_{max}$. Then iterate the following dynamic programming equations to convergence:

$$Q_{upper}(s,a) = R(s,a) + \max_{\tilde{P}(s,a)\in CI(P(s,a),\delta_1)} \gamma \sum_{s'} \tilde{P}(s'|s,a) \max_{a'} Q_{upper}(s',a') \quad \forall s,a \tag{2.6}$$

At convergence, define $V_{upper}(s) = \max_a Q_{upper}(s,a)$. Strehl and Littman [62] prove that this converges.

Strehl and Littman provide Algorithm UPPERP (Algorithm 2) for solving the optimization over $CI(P(s,a),\delta_1)$ in (2.6) efficiently. If the radius of the confidence interval is ω, then we can solve for $\tilde{P}$ by shifting $\Delta\omega = \omega/2$ of the probability mass from outcomes s' for which $V_{upper}(s') = \max_{a'} Q_{upper}(s',a')$ is low ("donor states") to outcomes for which it is maximum ("recipient states"). This will result in creating a $\tilde{P}$ distribution that is at L_1 distance ω from $\hat{P}$. The algorithm repeatedly finds a pair of successor states $\underline{s}$ and $\overline{s}$ and shifts probability from one to the other until it has shifted $\Delta\omega$. Note that in most cases, $\overline{s}$ will be a state for which $N(s,a,\overline{s}) = 0$—that is, a state we have never visited. In such cases, $V_{upper}(\overline{s}) = V_{max}$.

As with MBAE, UPPERP only requires time proportional to the number of transitions that have been observed to have non-zero probability.

The MBIE algorithm works as follows. Given the upper bound Q_{upper}, MBIE defines an exploration policy based on the optimism principle [6]. Specifically, at each state s, it selects the action a that maximizes $Q_{upper}(s,a)$. It then performs that action in the MDP simulator to obtain the immediate reward r and the resulting state s'. It then updates its statistics $N(s,a,s')$, $R(s,a)$, and $N(s,a)$ and recomputes Q_{upper}.

MBIE never terminates. However, it does compute a constant m such that if $N(s,a) > m$, then it does not draw a new sample from the MDP simulator for (s,a). Instead, it samples a

Algorithm 2 UPPERP(s, a, δ, M_0)

Input: s, a
δ: Confidence parameter
M_0: missing mass limit
Lines marked by **GT:** are for the Good-Turing extension
$N(s,a) := \sum_{s'} N(s,a,s')$
$\hat{P}(s'|s,a) := N(s,a,s')/N(s,a)$ for all s'
$\tilde{P}(s'|s,a) := \hat{P}(s'|s,a)$ for all s'
$\Delta\omega := \omega(N(s,a),\delta)/2$
GT: $N_0(s,a) := \{s'|N(s,a,s') = 0\}$
GT: $\Delta\omega := \min\left(\omega(N(s,a),\delta/2)/2, (1+\sqrt{2})\sqrt{\frac{ln(2/\delta)}{N(s,a)}}\right)$
while $\Delta\omega > 0$ **do**
  $S' := \{s' : \hat{P}(s'|s,a) < 1\}$ recipient states
GT: **if** $M_0 = 0$ **then** $S' := S' \setminus N_0(s,a)$
  $\underline{s} := \text{argmin}_{s':\tilde{P}(s'|s,a)>0} V_{upper}(s')$ donor state
  $\overline{s} := \text{argmax}_{s'\in S', \tilde{P}(s'|s,a)<1} V_{upper}(s')$ recipient state
  $\xi := \min\{1 - \tilde{P}(\overline{s}|s,a), \tilde{P}(\underline{s}|s,a), \Delta\omega\}$
  $\tilde{P}(\underline{s}|s,a) := \tilde{P}(\underline{s}|s,a) - \xi$
  $\tilde{P}(\overline{s}|s,a) := \tilde{P}(\overline{s}|s,a) + \xi$
  $\Delta\omega := \Delta\omega - \xi$
GT: **if** $\overline{s} \in N_0(s,a)$ **then** $M_0 := M_0 - \xi$
return $\tilde{P}$

next state according to its transition probability estimate $\hat{P}(s'|s,a)$. Hence, in an ergodic[3] or unichain[4] MDP, it will eventually stop drawing new samples, because it will have invoked the simulator on all actions a in all non-transient states s at least m times.

Because MBIE does not terminate, it cannot be applied directly to MDP planning. However, we can develop an MDP planning version by using the horizon time H computed by Fiechter's method and forcing MBIE to jump back to s_0 each time it has traveled H steps away from the start state. Algorithm 3 provides the pseudo-code for this variant of MBIE, which we call MBIE-reset.

Now that we have described the application goal and previous research, we present the novel contributions of this chapter.

[3] An ergodic MDP is an MDP where every state can be accessed in a finite number of steps from any other state
[4] In unichain MDP, every policy in an MDP result in a single ergodic class

Algorithm 3 MBIE-reset($s_0, \gamma, F, H, \epsilon, \delta$)

Input: s_0:start state, γ: discount rate, F: a simulator, H : horizon,ϵ, δ: accuracy and confidence parameters

$N(s, a, s') = 0$ for all (s, a, s')

$m = c\left[\frac{|S|}{\epsilon^2(1-\gamma)^4} + \frac{1}{\epsilon^2(1-\gamma)^4} \ln \frac{|S||A|}{\epsilon(1-\gamma)\delta}\right]$

repeat forever

- $s = s_0$
- $h = 1$
- **while** $h \leq H$ **do**
  - **update** Q_{upper} and V_{upper} by iterating equation 2.6 to convergence
  - $a = \text{argmax}_a\, Q_{upper}(s)$
  - **if** $N(s, a) < m$ **then**
    - $(r, s') \sim F(s, a)$ // draw sample
    - **update** $N(s, a, s')$, $N(s, a)$, and $R(s, a)$
  - **else**
    - $s' \sim \hat{P}(s'|s, a)$
    - $r = R(s, a)$
  - $h = h + 1$

2.5 Improved Model-Based MDP Planning

We propose a new algorithm, which we call DDV. Algorithm 4 presents the general schema for the algorithm. For each state-action (s, a) pair that has been explored, DDV maintains upper and lower confidence limits on $Q(s, a)$ such that $Q_{lower}(s, a) \leq Q^*(s, a) \leq Q_{upper}(s, a)$ with high probability. From these, we compute a confidence interval on the value of the start state s_0 according to $V_{lower}(s_0) = \max_a Q_{lower}(s_0, a)$ and $V_{upper}(s_0) = \max_a Q_{upper}(s_0, a)$. Consequently, $V_{lower}(s_0) \leq V^*(s_0) \leq V_{upper}(s_0)$ with high probability. The algorithm terminates when the width of this confidence interval, which we denote by $\Delta V(s_0) = V_{upper}(s_0) - V_{lower}(s_0)$, is less than ϵ.

The confidence intervals for Q_{lower} and Q_{upper} are based on an extension of the Weissman, et al. confidence interval of Equation (2.5), which we will refer to as $CI^{GT}(P(s, a), \delta_1)$ (which will be described below). The confidence intervals are computed by iterating the following

equations to convergence:

$$Q_{lower}(s,a) = R(s,a) + \min_{\tilde{P}(s,a)\in CI^{GT}(P(s,a),\delta_1)} \gamma \sum_{s'} \tilde{P}(s'|s,a) \max_{a'} Q_{lower}(s',a') \quad \forall (2.7)$$

$$Q_{upper}(s,a) = R(s,a) + \max_{\tilde{P}(s,a)\in CI^{GT}(P(s,a),\delta_1)} \gamma \sum_{s'} \tilde{P}(s'|s,a) \max_{a'} Q_{upper}(s',a') \quad \forall (2.8)$$

The Q values are initialized as follows: $Q_{lower}(s,a) = 0$ and $Q_{upper}(s,a) = V_{max}$. At convergence, define $V_{lower}(s) = \max_a Q_{lower}(s,a)$ and $V_{upper}(s) = \max_a Q_{upper}(s,a)$.

Lemma 1 *If $\delta_1 = \delta/(|S||A|)$, then with probability $1-\delta$, $Q_{lower}(s,a) \leq Q^*(s,a) \leq Q_{upper}(s,a)$ for all (s,a) and $V_{lower}(s) \leq V^*(s) \leq V_{upper}$ for all s.*

Proof 1 *Strehl and Littman [62] prove this for Q_{upper} and V_{upper} by showing that it is true at the point of initialization and that Equation (2.8) is a contraction. Hence, it remains true by induction on the number of iterations of value iteration. The proof for Q_{lower} and V_{lower} is analogous.*

The exploration heuristic for DDV is based on exploring the state-action pair (s,a) that maximizes the expected decrease in $\Delta V(s_0)$. We write this quantity as $\Delta\Delta V(s_0|s,a)$, because it is a change (Δ) in the confidence interval width $\Delta V(s_0|s,a)$. Below, we will describe two different heuristics that are based on two different approximations to $\Delta\Delta V(s_0|s,a)$.

We now present the improved confidence interval, CI^{GT}, and evaluate its effectiveness experimentally. Then we introduce our two search heuristics, analyze them, and present experimental evidence that they improve over previous heuristics.

2.5.1 Tighter Statistical Analysis for Earlier Stopping

The first contribution of this chapter is to improve the confidence intervals employed in equation (2.6). In many real-world MDPs, the transition probability distributions are sparse in the sense that there are only a few states s' such that $P(s'|s,a) > 0$. A drawback of the Weissman et al. confidence interval is that $\omega(N,\delta)$ scales as $O(\sqrt{|S|/N})$, so the intervals are very wide for large state spaces. We would like a tighter confidence interval for sparse distributions.

Our approach is to intersect the Weissman et al. confidence interval with a confidence interval based on the Good-Turing estimate of the missing mass [30].

Algorithm 4 DDV $(s_0, \gamma, F, \epsilon, \delta)$

Input: s_0:start state
γ: discount rate
F: a simulator
ϵ, δ: accuracy and confidence parameters
$m = c\left[\frac{|S|}{\epsilon^2(1-\gamma)^4} + \frac{1}{\epsilon^2(1-\gamma)^4}\ln\frac{|S||A|}{\epsilon(1-\gamma)^\delta}\right]$
$\delta' = \delta/(|S||A|m)$
$\tilde{S} = \{s_0\}$ // observed and/or explored states
$N(s, a, s') = 0$ for all (s, a, s')
repeat forever
  update $Q_{upper}, Q_{lower}, V_{upper}, V_{lower}$ by iterating equations 2.7 and 2.8 using δ' to compute the confidence intervals
  if $V_{upper}(s_0) - V_{lower}(s_0) \leq \epsilon$ **then**
    // compute a good policy and terminate
    $\pi_{lower}(s) = \arg\max_a Q_{lower}(s, a)$
    return π_{lower}
  forall the *explored or observed states* s **do**
    forall the *actions* a **do**
      compute $\Delta\Delta V(s_0|s, a)$
  compute $(s, a) := \text{argmax}_{(s,a)}\, \Delta\Delta V(s_0|s, a)$
  $(r, s') \sim F(s, a)$ // draw sample
  $\tilde{S} := \tilde{S} \cup \{s'\}$ // update the set of discovered states
  update $N(s, a, s')$, $N(s, a)$, and $R(s, a)$

Definition 2 *For a given state-action pair (s, a), let $N_k(s, a) = \{s'|N(s, a, s') = k\}$ be the set of all result states s' that have been observed exactly k times. We seek to bound the total probability of those states that have never been observed: $M_0(s, a) = \sum_{s' \in N_0(s,a)} P(s'|s, a)$. The Good-Turing estimate of $M_0(s, a)$ is*

$$\widehat{M_0}(s, a) = \frac{|N_1(s, a)|}{N(s, a)}.$$

In words, Good and Turing count the number of successor states that have been observed exactly once and divide by the number of samples. The following lemma follows directly from [38], [46], and [45].

Lemma 2 *With probability* $1-\delta$*,*

$$M_0(s,a) \leq \widehat{M}_0(s,a) + (1+\sqrt{2})\sqrt{\frac{\ln(1/\delta)}{N(s,a)}}. \tag{2.9}$$

Proof 2 *Let* $S(M_0(s,a),x)$ *be the Chernoff "entropy", defined as*

$$S(M_0(s,a),x) = \sup_{\beta} x\beta - \ln Z(M_0(s,a),\beta),$$

where $Z(M_0(s,a),\beta) = \mathbb{E}[e^{\beta M_0(s,a)}]$*. McAllester and Ortiz [45, Theorem 16] prove that*

$$S(M_0(s,a), \mathbb{E}[M_0(s,a)] + \epsilon) \geq N(s,a)\epsilon^2.$$

From Lemmas 12 and 13 of McAllester and Schapire [46],

$$\mathbb{E}[M_0(s,a)] \leq \widehat{M}_0(s,a) + \sqrt{\frac{2\log 1/\delta}{N(s,a)}}.$$

Combining these results yields

$$S\left(M_0(s,a), \widehat{M}_0(s,a) + \sqrt{\frac{2\log 1/\delta}{N(s,a)}} + \epsilon\right) \geq N(s,a)\epsilon^2. \tag{2.10}$$

Chernoff [10] proves that

$$P(M_0(s,a) \geq x) \leq e^{-S(M_0(s,a),x)}.$$

Plugging in (2.10) gives

$$P\left(M_0(s,a) \geq \widehat{M}_0(s,a) + \sqrt{\frac{2\log 1/\delta}{N(s,a)}} + \epsilon\right) \leq e^{-N(s,a)\epsilon^2}. \tag{2.11}$$

Setting $\delta = e^{-N(s,a)\epsilon^2}$ *and solving for* ϵ *gives* $\epsilon = \sqrt{(\log 1/\delta)/N(s,a)}$*. Plugging this into (2.11) and simplifying gives the result.*

Define $CI^{GT}(\hat{P}|N(s,a),\delta)$ to be the set of all distributions $\tilde{P} \in CI(\hat{P}|N(s,a),\delta/2)$ such

that $\sum_{s' \in N_0(s,a)} \tilde{P}(s'|s,a) < \widehat{M}_0(s,a) + (1+\sqrt{2})\sqrt{\frac{\ln(2/\delta)}{N(s,a)}}$. This intersects the Weissman and Good-Turing intervals. Note that since we are intersecting two confidence intervals, we must compute both (2.5) and (2.9) using $\delta/2$ so that they will simultaneously hold with probability $1-\delta$.

We can incorporate the bound from (2.9) into UPPERP by adding the lines prefixed by "GT:" in Algorithm 2. These limit the amount of probability that can be shifted to unobserved states according to (2.9). The modified algorithm still only requires time proportional to the number of states s' where $N(s,a,s') > 0$.

2.5.1.1 Experimental Evaluation of the Improved Confidence Bound

To test the effectiveness of this Good-Turing improvement, we ran MBIE-reset and compared its performance with and without the improved confidence interval.

We experimented with four MDPs. The first is a Combination Lock MDP with 500 states. In each state i, there are two possible actions. The first action makes a deterministic transition to state $i+1$ with reward 0 except for state 500, which is a terminal state with a reward of 1. The second action makes a transition (uniformly) to one of the states $1, \ldots, i-1$ with reward 0. The optimal policy is to choose the first action in every state, even though it doesn't provide a reward until the final state.

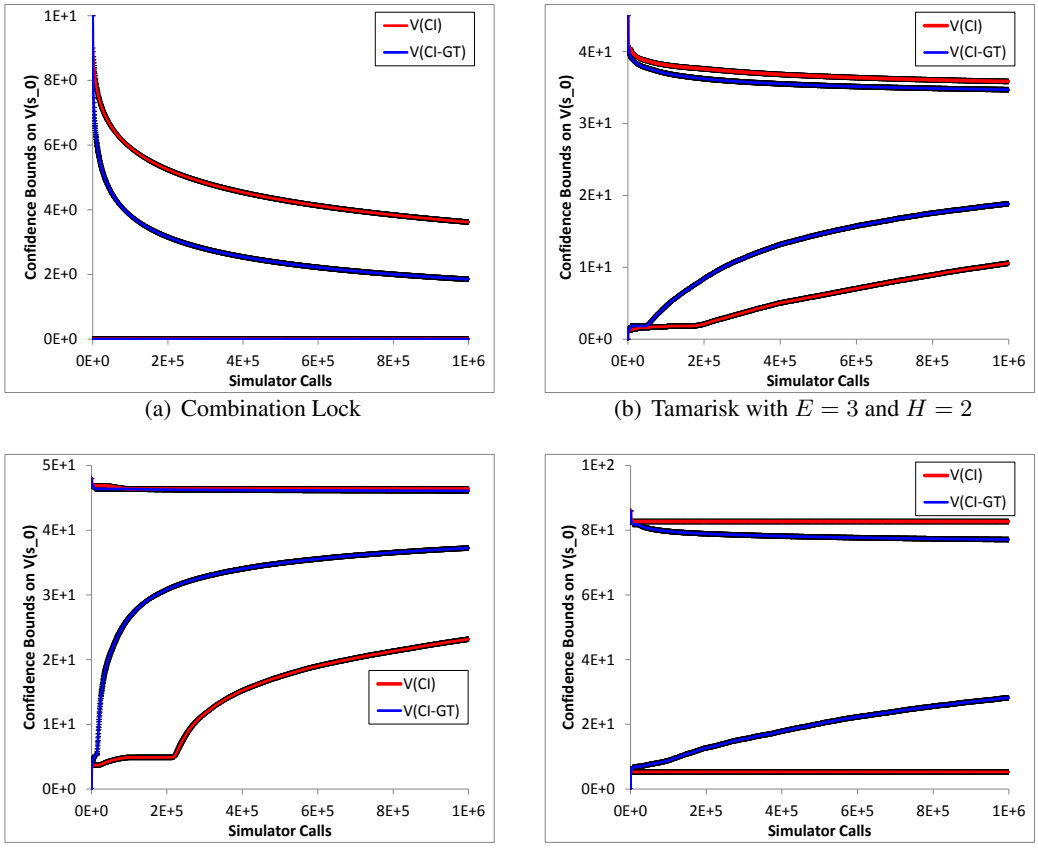
The remaining three MDPs are different versions of the tamarisk management MDP. The specific network configurations that we employed in this experiment were the following:

- $E = 3, H = 2$ with the budget constraint that in each time step we can only choose one edge in which to perform a non-"do nothing" action. This gives a total of 7 actions.
- $E = 3, H = 3$ with the same constraints as for $E = 3, H = 2$.
- $E = 7, H = 1$ with the budget constraint that in each time step we can only choose one edge in which to perform a non-"do nothing" action. The only such action is "restore". This gives a total of 8 actions.

2.5.1.2 Results

Figure 2.2 shows the upper and lower confidence bounds, $V_{upper}(s_0)$ and $V_{lower}(s_0)$, on the value of the starting state s_0 as a function of the number of simulator calls. The confidence

bounds for the Weissman et al. interval are labeled "V(CI)", whereas the bounds for this interval combined with the Good-Turing interval are labeled "V(CI-GT)".



(a) Combination Lock

(b) Tamarisk with $E = 3$ and $H = 2$

(c) Tamarisk with $E = 3$ and $H = 3$

(d) Tamarisk with $E = 7$ and $H = 1$

Figure 2.2: Plots of $V_{upper}(s_0)$ and $V_{lower}(s_0)$ for MBIE-reset on $V(s_0)$ with and without incorporating Good-Turing confidence intervals. Values are the mean of 15 independent trials. Error bars (which are barely visible) show 95% confidence intervals computed from the 15 trials.

2.5.1.3 Discussion

The results show that the Good-Turing interval provides a substantial reduction in the number of required simulator calls. On the combination lock problem, the CI-GT interval after 2×10^5

calls is already better than the CI interval after 10^6 calls, for a more than five-fold speedup. On the $E = 3, H = 2$ tamarisk problem, the speedup is more than a factor of three. On the $E = 3, H = 3$ version, the speedup is more than five-fold. And on the $E = 7, H = 1$ problem, the CI interval does not show any progress toward convergence, whereas the CI-GT interval has begun to make progress.

2.5.2 Improved Exploration Heuristics for MDP Planning

The second contribution of this chapter is to define two new exploration heuristics for MDP planning and compare them to existing algorithms. As with previous work, we wish to exploit reachability and discounting to avoid exploring unnecessarily. However, we want to take advantage of the fact that our simulators are "strong" in the sense that we can explore any desired state-action pair in any order.

As discussed above, our termination condition is to stop when the width of the confidence interval $\Delta V(s_0) = V_{upper}(s_0) - V_{lower}(s_0)$ is less than ϵ. Our heuristics are based on computing the state-action pair (s, a) that will lead to the largest (one step) reduction in $\Delta V(s_0)$. Formally, let $\Delta\Delta V(s_0|s, a) = \mathbb{E}[\Delta V(s_0) - \Delta V'(s_0)|(s, a)]$ be the expected change in $\Delta V(s_0)$ if we draw one more sample from (s, a). Here the prime in $\Delta V'(s_0)$ denotes the value of $\Delta V(s_0)$ after exploring (s, a). The expectation is taken with respect to two sources of uncertainty: uncertainty about the reward $R(s, a)$ and uncertainty about the resulting state $s' \sim P(s'|s, a)$.

Suppose we are considering exploring (s, a). We approximate $\Delta\Delta V(s_0|s, a)$ in two steps. First, we consider the reduction in our uncertainty about $Q(s, a)$ if we explore (s, a). Let $\Delta Q(s, a) = Q_{upper}(s, a) - Q_{lower}(s, a)$ and $\Delta\Delta Q(s, a) = \mathbb{E}[\Delta Q(s, a) - \Delta Q'(s, a)|(s, a)]$. Second, we consider the impact that reducing $\Delta Q(s, a)$ will have on $\Delta V(s_0)$.

We compute $\Delta\Delta Q(s, a)$ as follows.

Case 1: $N(s, a) = 0$. In this case, our current bounds are $Q_{lower}(s, a) = 0$ and $Q_{upper}(s, a) = V_{max}$. After we sample $(r, s') \sim F(s, a)$, we will observe the actual reward $R(s, a) = r$ and we will observe one of the possible successor states s'. For purposes of deriving our heuristic, we will assume a uniform[5] prior on $R(s, a)$ so that the expected value of R is $\overline{R} = R_{max}/2$. We will assume that s' will be a "new" state that we have never observed before, and hence

[5] Any symmetric prior centered on $R_{max}/2$ would suffice.

$V_{upper}(s') = V_{max}$ and $V_{lower}(s') = 0$. This gives us

$$Q'_{upper}(s,a) = \overline{R}(s,a) + \gamma R_{max}/(1-\gamma) \tag{2.12a}$$

$$Q'_{lower}(s,a) = \overline{R}(s,a), \tag{2.12b}$$

If a more informed prior is known for $R(s,a)$, then it could be employed to derive a more informed exploration heuristic.

Case 2: $N(s,a) > 0$. In this case, we have already observed $R(s,a)$, so it is no longer a random variable. Hence, the expectation is only over s'. For purposes of deriving our exploration heuristic, we will assume that s' will be drawn according to our current maximum likelihood estimate $\hat{P}(s'|s,a)$ but that $N_1(s,a)$ will not change. Consequently, the Good-Turing estimate will not change. Under this assumption, the expected value of Q will not change, $M_0(s,a)$ will not change, so the only change to Q_{upper} and Q_{lower} will result from replacing $\omega(N(s,a),\delta)$ by $\omega(N(s,a)+1,\delta)$ in the Weissman et al. confidence interval.

Note that DDV may explore a state-action pair (s,a) even if a is not currently the optimal action in s. That is, even if $Q_{upper}(s,a) < Q_{upper}(s,a')$ for some $a' \neq a$. An alternative rule would be to only explore (s,a) if it would reduce the expected value of $\Delta V(s) = V_{upper}(s) - V_{lower}(s)$. However, if there are two actions a and a' such that $Q_{upper}(s,a) = Q_{upper}(s,a')$, then exploring only one of them will not change $\Delta V(s)$. Our heuristic avoids this problem. We have studied another variant in which we defined $V_{upper}(s) = \text{softmax}_a(\tau)\ Q_{upper}(s,a)$ (the softmax with temperature τ). This gave slightly better results, but it requires that we tune τ, which is a nuisance.

The second component of our heuristic is to estimate the impact of $\Delta\Delta Q(s_0|s,a)$ on $\Delta\Delta V(s_0|s,a)$. To do this, we appeal to the concept of an occupancy measure.

Definition 3 *The* occupancy measure $\mu^\pi(s)$ *is the expected discounted number of times that policy* π *visits state* s*,*

$$\mu^\pi(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t \mathbb{I}[s_t = s] \;\middle|\; s_0, \pi\right], \tag{2.13}$$

where $\mathbb{I}[\cdot]$ *is the indicator function and the expectation taken is with respect to the transition distribution.*

This can be computed via dynamic programming on the Bellman flow equation [64]:

$$\mu^{\pi}(s) := P_0(s) + \gamma \sum_{s^-} \mu^{\pi}(s^-) P(s|s^-, \pi(s^-)).$$

This says that the (discounted) probability of visiting state s is equal to the sum of the probability of starting in state s (as specified by the starting state distribution $P_0(s)$) and the probability of reaching s by first visiting state s^- and then executing an action that leads to state s.

The intuition behind using an occupancy measure is that if we knew that the optimal policy would visit s with measure $\mu^*(s)$ and if exploring (s, a) would reduce our uncertainty at state s by approximately $\Delta\Delta Q(s_0|s, a)$, then a reasonable estimate of the impact on $\Delta V(s_0)$ would be $\mu^*(s)\Delta\Delta Q(s_0|s, a)$. Unfortunately, we don't know μ^* because we don't know the optimal policy. We consider two other occupancy measures instead: μ^{OUU} and $\overline{\mu}$.

The first measure, μ^{OUU} is computed based on the principle of optimism under uncertainty. Specifically, define $\pi^{OUU}(s) := \max_a Q_{upper}(s, a)$ to be the policy that chooses the action that maximizes the upper confidence bound on the Q function. This is the policy followed by MBIE and most other upper-confidence bound methods. This gives us the DDV-OUU heuristic.

Definition 4 *The DDV-OUU heuristic explores the state action pair* (s, a) *that maximizes*

$$\mu^{OUU}(s)\Delta\Delta Q(s_0|s, a).$$

The second measure $\overline{\mu}$ is computed based on an upper bound of the occupancy measure over all possible policies. It gives us the DDV-UPPER heuristic.

Definition 5 *The DDV-UPPER heuristic explores the state action pair* (s, a) *that maximizes*

$$\overline{\mu}(s)\Delta\Delta Q(s_0|s, a).$$

The next section defines $\overline{\mu}$ and proves a property that may be of independent interest.

2.5.2.1 An Upper Bound on the Occupancy Measure

The purpose of this section is to introduce $\overline{\mu}$, which is an upper bound on the occupancy measure of any optimal policy for a restricted set of MDPs $\widetilde{\mathcal{M}}$. This section defines this measure and

presents a dynamic programming algorithm to compute it. The attractive aspect of $\overline{\mu}$ is that it can be computed without knowing the optimal policy. In this sense, it is analogous to the value function, which value iteration computes in a policy-independent way.

To define $\overline{\mu}$, we must first define the set $\widetilde{\mathcal{M}}$ of MDPs. At each point during the execution of DDV, the states S of the unknown MDP can be partitioned into three sets: (a) the *unobserved states* s (i.e., $N(s^-, a^-, s) = 0$ for all s^-, a^-); (b) the *observed but unexplored states* s (i.e., $\exists(s^-, a^-)N(s^-, a^-, s) > 0$ but $N(s, a) = 0$ for all a), and (c) *the (partially) explored states* s (i.e., $N(s, a, s') > 0$ for some a). Consider the set $\widetilde{\mathcal{M}} = \langle \tilde{S}, \tilde{A}, \tilde{T}, \tilde{R}, s_0 \rangle$ of MDPs satisfying the following properties:

- $\tilde{S}$ consists of all states s that have been either observed or explored,
- $\tilde{A} = A$, the set of actions in the unknown MDP,
- $\tilde{T}$ consists of any transition function T such that for explored states s and all actions a, $T(s, a, \cdot) \in CI^{GT}(\hat{P}(s, a), \delta)$. For all observed but not explored states s, $T(s, a, s) = 1$ for all a, so they enter self-loops.
- $\tilde{R}$: For explored (s, a) pairs, $\tilde{R}(s, a) = R(s, a)$. For unexplored (s, a) pairs, $\tilde{R}(s, a) \in [0, R_{max}]$.
- s_0 is the artificial start state.

The set $\widetilde{\mathcal{M}}$ contains all MDPs consistent with the observations with the following restrictions. First, the MDPs do not contain any of the unobserved states. Second, the unexplored states contain self-loops and hence do not transition to any other states.

Define $P_{upper}(s'|s, a)$ as follows:

$$P_{upper}(s'|s, a) = \max_{\tilde{P}(s,a) \in CI^{GT}(P,\delta)} \tilde{P}(s'|s, a).$$

Define $\overline{\mu}$ as the solution to the following dynamic program. For all states s,

$$\overline{\mu}(s) = \sum_{s^- \in pred(s)} \max_{a^-} \gamma P_{upper}(s|s^-, a^-)\overline{\mu}(s^-). \tag{2.14}$$

The intuition is that we allow each predecessor s^- of s to choose the action a^- that would send the most probability mass to s and hence give the biggest value of $\overline{\mu}(s)$. These action choices

a^- are not required to be consistent for multiple successors of s^-. We fix $\overline{\mu}(s_0) = \mu(s_0) = 1$. (Recall, that s_0 is an artificial start state. It is not reachable from any other state—including itself—so $\mu(s_0) = 1$ for all policies.)

Lemma 3 *For all MDPs $\widetilde{M} \in \widetilde{\mathcal{M}}$, $\overline{\mu}(s) \geq \mu^{\pi^*(\widetilde{M})}(s)$, where $\pi^*(\widetilde{M})$ is any optimal policy of $\widetilde{M}$.*

Proof 3 *By construction, $P_{upper}(s'|s,a)$ is the maximum over all transition distributions in $\widetilde{\mathcal{M}}$ of the probability of $(s,a) \rightarrow s'$. According to (2.14), the probability flowing to s is the maximum possible over all policies executed in the predecessor states $\{s^-\}$. Finally, all probability reaching a state s must come from its known predecessors $pred(s)$, because all observed but unexplored states only have self-transitions and hence cannot reach s or any of its predecessors.*

In earlier work, Smith and Simmons [60] employed a less general path-specific bound on μ as a heuristic for focusing Real-Time Dynamic Programming (a method that assumes a full model of the MDP is available).

2.5.2.2 Soundness of DDV-OUU and DDV-UPPER

We now show that DDV, using either of these heuristics, produces an ϵ-optimal policy with probability at least $1 - \delta$ after making only polynomially-many simulator calls. The steps in this proof closely follow previous proofs by [62] and [17].

Theorem 1 (DDV is PAC-RL) *There exists a sample size m polynomial in $|S|$, $|A|$, $1/\epsilon$, $1/\delta$, $1/(1-\gamma)$, R_{max}, such that DDV$(s_0, F, \epsilon, \delta/(m|S||A|))$ with either the DDV-OUU or the DDV-UPPER heuristic terminates after no more than $m|S||A|$ calls on the simulator and returns a policy π such that $|V^\pi(s_0) - V^*(s_0)| < \epsilon$ with probability $1 - \delta$.*

Proof 4 *First, note that every sample drawn by DDV will shrink the confidence interval for some $Q(s,a)$. Hence, these intervals will eventually become tight enough to make the termination condition true. To establish a rough bound on sample complexity, let us suppose that each state must be sampled enough so that $\Delta Q(s,a) = Q_{upper}(s,a) - Q_{lower}(s,a) \leq \epsilon$.*

This will cause termination. Consider state s_0 and let $a_{upper} = \text{argmax}_a\, Q_{upper}(s_0, a)$ be the action chosen by the OUU policy. Then the upper bound on s is $V_{upper}(s) = Q_{upper}(s, a_{upper})$,

and the lower bound on S is $V_{lower}(s_0) = \max_a Q_{lower}(s_0, a) \geq Q_{lower}(s_0, a_{upper})$. Hence, the difference $V_{upper}(s_0) - V_{lower}(s_0) \leq \epsilon$.

How many samples are required to ensure that $\Delta Q(s, a) \leq \epsilon$ for all (s, a)? We can bound $Q_{upper}(s, a) - Q_{lower}(s, a)$ as follows.

$$\begin{aligned} Q_{upper}(s,a) - Q_{lower}(s,a) &= R(s,a) + \gamma \max_{\tilde{P} \in CI(\hat{P}(s,a),\delta')} \sum_{s'} \tilde{P}(s'|s,a) V_{upper}(s') \\ &\quad -R(s,a) - \gamma \min_{\tilde{P} \in CI(\hat{P}(s,a),\delta')} \sum_{s'} \tilde{P}(s'|s,a) V_{lower}(s') \end{aligned}$$

Let P_{upper} be the $\tilde{P}$ chosen in the max and P_{lower} be the $\tilde{P}$ chosen in the min. At termination, we know that in every state $V_{upper} \leq V_{lower} + \epsilon$. Substituting these and simplifying gives

$$Q_{upper}(s,a) - Q_{lower}(s,a) \leq \gamma \sum_{s'} [P_{upper}(s'|s,a) - P_{lower}(s'|s,a)] V_{lower}(s') + \gamma\epsilon.$$

We make two approximations: $P_{upper}(s'|s,a) - P_{lower}(s'|s,a) \leq |P_{upper}(s'|s,a) - P_{lower}(s'|s,a)|$ and $V_{lower}(s') \leq \frac{R_{max}}{1-\gamma}$. This yields

$$Q_{upper}(s,a) - Q_{lower}(s,a) \leq \gamma \frac{R_{max}}{1-\gamma} \sum_{s'} |P_{upper}(s'|s,a) - P_{lower}(s'|s,a)| + \gamma\epsilon.$$

We know that $\|P_{upper}(\cdot|s,a) - P_{lower}(\cdot|s,a)\|_1 \leq 2\omega$, because both distributions belong to the L_1 ball of radius ω around the maximum likelihood estimate $\hat{P}$.

$$Q_{upper}(s,a) - Q_{lower}(s,a) \leq \gamma \frac{R_{max}}{1-\gamma} 2\omega + \gamma\epsilon.$$

Setting this less than or equal to ϵ and solving for ω gives

$$\omega \leq \frac{\epsilon(1-\gamma)^2}{2\gamma R_{max}}.$$

We know that

$$\omega = \sqrt{\frac{2[\ln(2^{|S|} - 2) - \ln \delta']}{N}}.$$

To set δ', we must divide δ by the maximum number of confidence intervals computed by the algorithm. This will be $2|S||A|N$, because we compute two intervals (upper and lower) for ever

(s, a). *Plugging the value for* δ' *in and simplifying gives the following equation:*

$$N \geq \frac{\gamma^2 8 R_{max}^2 [\ln(2^{|S|} - 2) - \ln \delta + \ln 2|S||A| + \ln N]}{\epsilon^2 (1-\gamma)^4}.$$

This has no closed form solution. However, as Strehl and Littman note, there exists a constant C *such that if* $N \geq 2C \ln C$ *then* $N \geq C \ln N$. *Hence, the* $\ln N$ *term on the right-hand side will only require a small increase in* N. *Hence*

$$N = O\left(\frac{\gamma^2 R_{max}^2 |S| + \ln |S||A|/\delta}{\epsilon^2 (1-\gamma)^4}\right).$$

In the worst case, we must draw N *samples for every state-action pair, so*

$$m = O\left(|S|^2 |A| \frac{\gamma^2 R_{max}^2 + \ln |S||A|/\delta}{\epsilon^2 (1-\gamma)^4}\right),$$

which is polynomial in all of the relevant parameters.

To prove that the policy output by DDV is within ϵ *of optimal with probability* $1 - \delta$, *note that the following relationships hold:*

$$V_{upper}(s_0) \geq V^*(s_0) \geq V^{\pi_{lower}}(s_0) \geq V_{lower}(s_0).$$

The inequalities $V_{upper}(s_0) \geq V^*(s_0) \geq V_{lower}(s_0)$ *hold (with probability* $1 - \delta$*) by the admissibility of the confidence intervals. The inequality* $V^*(s_0) \geq V^{\pi_{lower}}(s_0)$ *holds, because the true value of any policy is no larger than the value of the optimal policy. The last inequality,* $V^{\pi_{lower}}(s_0) \geq V_{lower}(s_0)$, *holds because extended value iteration estimates the value of* π_{lower} *by backing up the values* V_{lower} *of the successor states. At termination,* $V_{upper}(s_0) - V_{lower}(s_0) \leq \epsilon$. *Hence,* $V^*(s_0) - V^{\pi_{lower}}(s_0) \leq \epsilon$.

2.5.3 Experimental Evaluation on Exploration Heuristics

We conducted an experimental study to assess the effectiveness of DDV-OUU and DDV-UPPER and compare them to the exploration heuristics of MBIE (with reset) and Fiechter's algorithm.

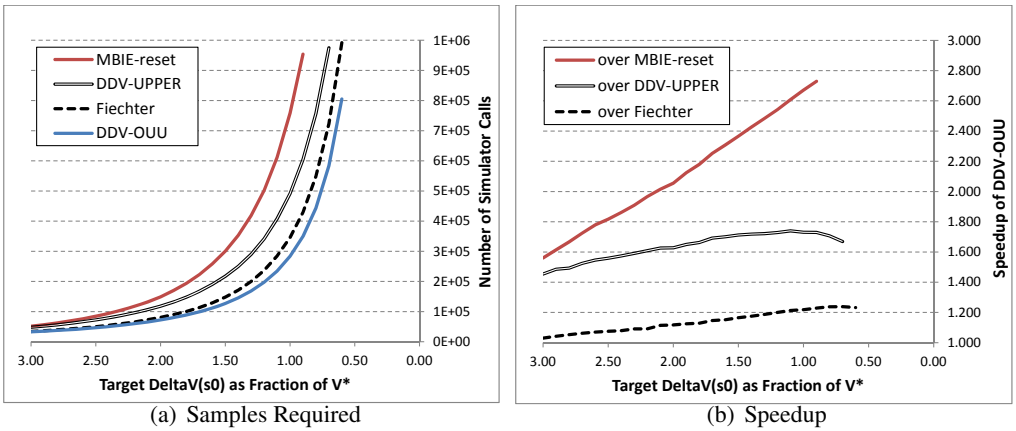
2.5.3.1 Methods

We conducted two experiments. The goal of both experiments was to compare the number of simulator calls required by each algorithm to achieve a target value ϵ for the width of the confidence interval, $\Delta V(s_0)$, on the value of the optimal policy in the starting state s_0. For problems where the value $V^*(s_0)$ of the optimal policy is known, we define $\epsilon = \alpha V^*(s_0)$ and plot the required sample size as a function of α. For the tamarisk problems, where $V^*(s_0)$ is not known, we define $\epsilon = \alpha R_{max}$ and again plot the required sample size as a function of α. This is a natural way for the user to define the required accuracy ϵ.

In the first experiment, we employed four MDPs: the RiverSwim and SixArms benchmarks, which have been studied by Strehl and Littman [63, 62], and two instances of our tamarisk management MDPs ($E = 3$, $H = 1$) and ($E = 3$, $H = 2$). Each of the tamarisk MDPs implemented a budget constraint that permits a non-"do nothing" action in only one edge in each time step. In the $E = 3, H = 2$ MDP, we included exogenous arrivals using the parameters described in Section 2.3 (up to 10 seeds per species per edge; Bernoulli parameters are 0.1 for tamarisk and 0.4 for native plants). The $E = 3, H = 1$ tamarisk MDP has 7 actions and 27 states, and the $E = 3, H = 2$ MDP has 7 actions and 216 states. The discount factor was set to 0.9 in all four MDPs.

Each algorithm was executed for one million simulator calls. Instead of performing dynamic programming updates (for extended value iteration and occupancy measure computation) after every simulator call, we computed them on the following schedule. For MBIE-reset, we performed dynamic programming after each complete trajectory. For DDV-OUU and DDV-UPPER, we performed dynamic programming after every 10 simulator calls. Extended value iteration gives us the confidence limits $V_{lower}(s_0)$ and $V_{upper}(s_0)$ for the starting state from which we also computed $\Delta V(s_0) = V_{upper}(s_0) - V_{lower}(s_0)$. The experiment was repeated 15 times, and the average value of $\Delta V(s_0)$ was computed. For each MDP, we defined a range of target values for $\Delta V(s_0)$ and computed the average number of samples m required by each algorithm to achieve each target value. By plotting these values, we can see how the sample size increases as we seek smaller target values for $\Delta V(s_0)$. We also computed the speedup of DDV-OUU over each of the other algorithms, according to the formula $m_{alg}/m_{\text{DDV-OUU}}$, and plotted the result for each MDP.

We also measured the total amount of CPU time required by each algorithm to complete the one million simulator calls. Because the simulators in these four MDPs are very efficient, the



(a) Samples Required (b) Speedup

Figure 2.3: RiverSwim results: (a) Number of samples required by MBIE-reset, Fiechter, DDV-UPPER, and DDV-OUU to achieve various target confidence interval widths $\Delta V(s_0)$. (b) Speedup of DDV-OUU over the algorithms.

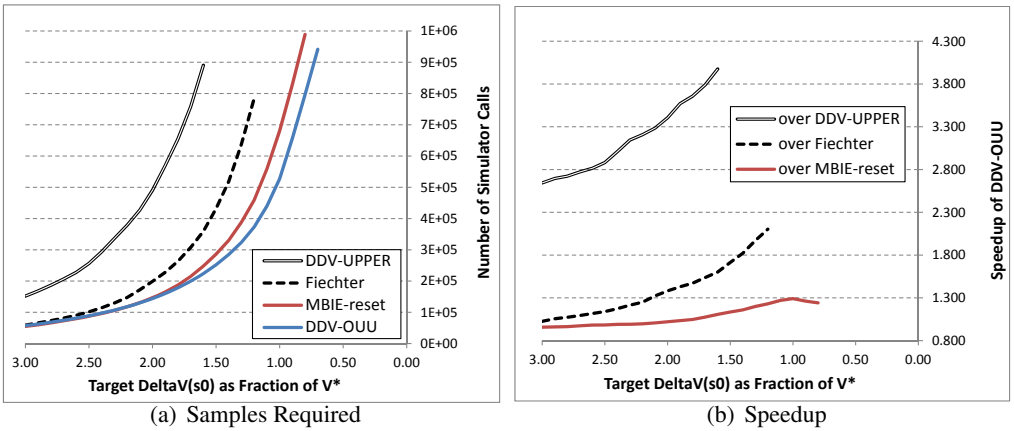
CPU time primarily measures the cost of the various dynamic programming computations. For Fiechter, these involve setting up and solving the exploration MDP. For MBIE-reset, the primary cost is performing extended value iteration to update V_{upper} and π^{OUU}. For the DDV methods, the cost involves extended value iteration for both V_{upper} and V_{lower} as well as the dynamic program for μ.

In the second experiment, we ran all four algorithms on the RiverSwim and SixArms problems until either 40 million calls had been made to the simulator or until $\Delta V(s_0) \leq \alpha R_{max}$, where $\alpha = 0.1$ and $R_{max} = 10000$ (for RiverSwim) and $R_{max} = 6000$ (for SixArms).

2.5.3.2 Results

Figures 2.3, 2.4, 2.5, and 2.6 show the results for the first experiment. In each figure, the left plot shows how the required sample size increases as the target width for $\Delta V(s_0)$ is made smaller. In each figure, the right plot shows the corresponding speedup of DDV-OUU over each of the other algorithms. In all cases, DDV-OUU generally requires the fewest number of samples to reach the target width, and DDV-UPPER generally requires the most. The poor behavior of DDV-UPPER suggests that the policy-free occupancy measure $\overline{\mu}$ is too loose to provide a competitive heuristic.

The relative performance of MBIE-reset and Fiechter's algorithm varies dramatically across



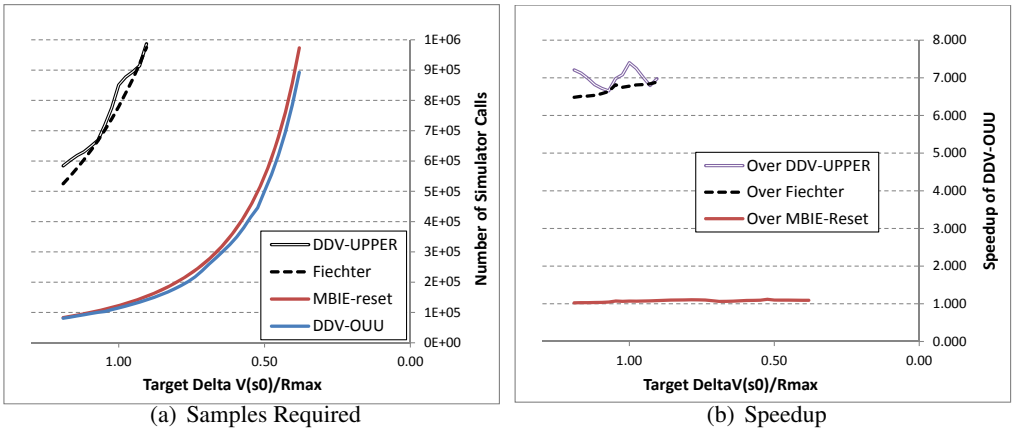
(a) Samples Required (b) Speedup

Figure 2.4: SixArms results: (a) Number of samples required by MBIE-reset, Fiechter, DDV-UPPER, and DDV-OUU to achieve various target confidence interval widths $\Delta V(s_0)$. (b) Speedup of DDV-OUU over the other algorithms.

the four MDPs. On RiverSwim, Fiechter's method is almost as good as DDV-OUU: DDV-OUU shows a speedup of at most 1.23 (23%) over Fiechter. In contrast, MBIE-reset performs much worse. But on SixArms, it is MBIE-reset that is the closest competitor to DDV-OUU. In fact, MBIE-reset is actually better than DDV-OUU for target values larger than 2.1, but as the target width for $\Delta V(s_0)$ is made smaller, DDV-OUU scales much better. On the tamarisk $R = 3$ $H = 1$ problem, MBIE-reset is again almost as good as DDV-OUU. The maximum speedup produced by DDV-OUU is 1.11. Finally, on the tamarisk $R = 3$ $H = 2$ problem, DDV-OUU is definitely superior to MBIE-reset and achieves speedups in the 1.9 to 2.3 range. Surprisingly, on

MDP	Algorithm			
	DDV-UPPER (ms/call)	DDV-OUU (ms/call)	MBIE-reset (ms/call)	Fiechter (ms/call)
RiverSwim	9.59	9.92	3.73	3.29
SixArms	15.54	48.97	10.53	4.87
Tamarisk (E=3 and H=1)	11.93	8.13	4.81	4.68
Tamarisk (E=3 and H=2)	187.30	166.79	12.63	18.79

Table 2.1: RiverSwim clock time per simulator call.



(a) Samples Required (b) Speedup

Figure 2.5: Tamarisk with $E = 3$ and $H = 1$ results: (a) Number of samples required by MBIE-reset, Fiechter, DDV-UPPER, and DDV-OUU to achieve various target confidence interval widths $\Delta V(s_0)$. (b) Speedup of DDV-OUU over the other algorithms.

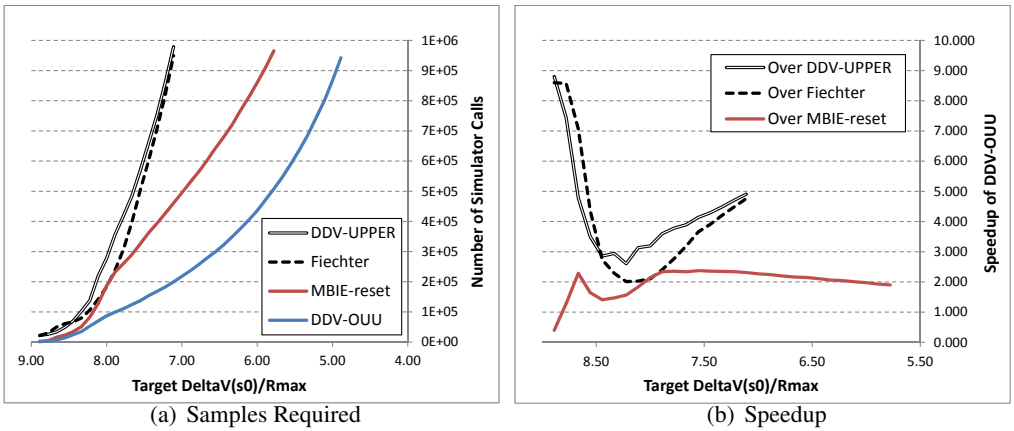
this problem, Fiechter's method is sometimes worse than DDV-UPPER.

The CPU time consumed per simulator call by each algorithm on each problem is reported in Table 2.1. Not surprisingly, MBIE-reset and Fiechter have much lower cost than the DDV methods. All of these methods are designed for problems where the simulator is extremely expensive. For example, in the work of [34] on wildfire management, one call to the simulator can take several minutes. In such problems, the overhead of complex algorithms such as DDV more than pays for itself by reducing the number of simulator calls.

Tables 2.2 and 2.3 report the results of the second experiment. The results are consistent with

Quantity	Algorithm				
	DDV-UPPER	DDV-OUU	MBIE-reset	Fiechter	Optimal
$V_{upper}(s_0)$	2967.2	2936.6	3001.5	2952.6	2203
$V_{lower}(s_0)$	1967.2	1936.6	2001.5	1952.6	2203
$\Delta V(s_0)$	1000	1000	1000	1000	
Simulator Calls ($\times 10^6$)	2.31	**1.44**	4.05	1.76	

Table 2.2: RiverSwim confidence intervals and required sample size to achieve target $\Delta V(s_0) = 1000$.



(a) Samples Required (b) Speedup

Figure 2.6: Tamarisk with $E = 3$ and $H = 2$ results: (a) Number of samples required by MBIE-reset, Fiechter, DDV-UPPER, and DDV-OUU to achieve various target confidence interval widths $\Delta V(s_0)$. (b) Speedup of DDV-OUU over the other algorithms.

those of the first experiment. DDV-OUU reaches the target $\Delta V(s_0)$ with the smallest number of simulator calls on both problems. On RiverSwim, Fiechter's method is second best, whereas on SixArms, MBIE-reset is second best. On SixArms, DDV-UPPER and Fiechter did not reach the target accuracy within the limit of 40 million simulator calls.

Quantity	Algorithm				
	DDV-UPPER	DDV-OUU	MBIE-reset	Fiechter	Optimal
$V_{upper}(s_0)$	5576.7	5203.9	5242.4	5672.8	4954
$V_{lower}(s_0)$	4140.4	4603.9	4642.4	3997.7	4954
$\Delta V(s_0)$	1436.3	600	600	1675.1	
Simulator Calls ($\times 10^6$)	40.0	**14.5**	19.3	40.0	

Table 2.3: SixArms confidence intervals and required sample size to achieve the target $\Delta V(s_0) = 600$.

2.5.3.3 Discussion

The experiments show that DDV-OUU is the most effective of the four algorithms and that it achieves substantial speedups over the other three algorithms (maximum speedups of 2.73 to 7.42 across the four problems).

These results contrast with our previous work [12] in which we showed that DDV-UPPER is better than MBIE. The key difference is that in the present chapter, we are comparing against MBIE with reset, whereas in the previous work, we compared against MBIE without reset. Without resetting, MBIE can spend most of its time in regions of the MDP that are far from the start state, so it can fail to find a good policy for s_0. This behavior also explains the poor performance of Q-learning reported in [12].

2.6 Summary and Conclusions

This chapter has addressed the problem of MDP planning when the MDP is defined by an expensive simulator. In this setting, the planning phase is separate from the execution phase, so there is no tradeoff between exploration and exploitation. Instead, the goal is to compute a PAC-optimal policy while minimizing the number of calls to the simulator. The policy is designed to optimize the cumulative discounted reward starting in the current real-world state s_0. Unlike in most published RL papers, which typically assume that the MDP is ergodic, the starting state of our ecosystem management problems is typically a transient state.

The chapter makes two contributions. First, it shows how to combine the Good-Turing estimate with the L_1-confidence region of Weissman et al. [73] to obtain tighter confidence intervals (and hence, earlier termination) in sparse MDPs. Second, it shows how to use occupancy measures to create better exploration heuristics. The chapter introduced a new policy-independent upper bound $\overline{\mu}$ on the occupancy measure of the optimal policy and applied this to define the DDV-UPPER algorithm. The chapter also employed an occupancy measure μ^{OUU} based on the "optimism under uncertainty" principle to define the DDV-OUU algorithm.

The $\overline{\mu}$ measure is potentially of independent interest. Like the value function computed during value iteration, it does not quantify the behavior of any particular policy. This means that it can be computed without needing to have a specific policy to evaluate. However, the DDV-UPPER exploration heuristic did not perform very well. We have two possible explanations for this. First, $\overline{\mu}$ can be a very loose upper bound on the optimal occupancy measure μ^*. Perhaps

this leads DDV-UPPER to place too much weight on unfruitful state-action pairs. Second, it is possible that while DDV-UPPER is optimizing the one-step gain in $\Delta\Delta V(s_0)$ (as it is designed to do), DDV-OUU does a better job of optimizing gains over the longer term. Further experimentation is needed to determine which of these explanations is correct.

Our DDV-OUU method gave the best performance in all of our experiments. This is yet another confirmation of the power of the "Optimism Principle" [6] in exploration. Hence, we recommend it for solving simulator-defined MDP planning problems. We are applying it to solve moderate-sized instances of our tamarisk MDPs. However, additional algorithm innovations will be required to solve much larger tamarisk instances.

Three promising directions for future research are (a) exploiting tighter confidence interval methods, such as the Empirical Bernstein Bound [2, 65] or improvements on the Good-Turing estimate [50, 70], (b) explicitly formulating the MDP planning problem in terms of sequential inference [71], which would remove the independence assumption in the union bound for partitioning δ, and (c) studying exploration methods based on posterior sampling [69].

Combining Global and Local Confidence Intervals for More Efficient MDP Planning

Majid Alkaee Taleghan and Thomas G. Dietterich
Ready For Submission

Chapter 3: Combining Global and Local Confidence Intervals for More Efficient MDP Planning

Given a Markov Decision Process (MDP) defined by a simulator and a designated starting state s_0, the goal of MDP planning is to find a policy π that with probability $1-\delta$ achieves a value $V^\pi(s_0)$ that is within ϵ of the value of the optimal policy, $V^*(s_0)$, while economizing on the number of calls to the simulator. Existing approaches to this problem have computed "local" confidence intervals for each state-action pair and then combined these via "extended" value iteration to obtain a confidence interval on $|V^\pi(s_0) - V^*(s_0)|$. However, global (trajectory-wise) confidence intervals are usually much tighter than these local intervals. A drawback of global methods is that they require samples to be collected along trajectories, whereas at each point during MDP planning, the samples have been collected piecemeal according to an exploration algorithm. This chapter makes two contributions. First, this chapter develops optimal sampling algorithms for global and local confidence intervals based on the Hoeffding, empirical Bernstein, and multinomial bounds. It then presents experimental evidence that sampling along trajectories and computing global confidence intervals is preferred except in a few extreme cases. Second, this chapter introduces an algorithm called Equivalent Trajectory Policy Evaluation (ETPE) that re-uses the accumulated set of samples to generate a new set of trajectories and compute a global lower confidence bound on the value of the optimal policy. This global lower bound is combined with a local upper bound to obtain a tighter bound on $|V^\pi(s_0) - V^*(s_0)|$. Finally, the chapter introduces a new PAC-optimal MDP planning algorithm, Local-Local-Global Confidence Value (LLGCV) that explores in a series of minibatches. In each minibatch, it chooses either to try to reduce the upper bound or to increase the lower bound. Experiments on benchmark problems show that LLGCV can provide substantial benefits in some cases.

3.1 Introduction

Many important problems in natural resource management can be solved via Markov Decision Process (MDP) planning. For example, Sheldon et al. [59] describe a problem involving the recovery of an endangered species, and Dietterich, Alkaee Taleghan, and Crowley [12] present a

problem of invasive species management. For modest-sized instances of these problems, model-based MDP planning algorithms can be applied. If the transition probabilities of the MDP are available in matrix form, then standard dynamic programming methods can solve these MDPs. However, the transition dynamics of these MDPs are typically available only via an expensive simulator. This leads us to consider adapting methods from model-based reinforcement learning to solve these problems. Specifically, we seek MDP-planning algorithms that draw a minimal number of samples from the simulator and then output a policy that with high probability is near-optimal when executed in a designated start state.

To create a good MDP-planning algorithm, there are two critical design questions: (a) what confidence interval should be employed to bound the optimality of the policy? and (b) how should samples be drawn to shrink this confidence interval as quickly as possible? Most existing work [22, 17, 62, 35] answers the first question by computing a "local" confidence interval for each state-action pair and then combining these intervals via value iteration. Specifically, Even-Dar, Mannor, and Mansour [17] apply the Hoeffding bound at each state. Strehl and Littman [62] apply a multinomial confidence interval developed by Weissman et al. [73] at each state. One can easily apply the empirical Bernstein bound [2] at each state as well.

To answer the second question, most existing work [62, 35] uses the upper confidence bound $\overline{Q}(s, a)$ for each state-action pair to compute the upper confidence bound policy π^{UCB} and then draws samples by executing that policy along trajectories. This "optimism under uncertainty" has been shown both theoretically and experimentally to give good exploration. However, there is no particular reason to think that exploration should take place along trajectories. If we have a "strong" simulator—that is, a simulator that can provide samples from arbitrary state-action pairs in any order—then there could be advantages to focusing sampling on particular parts of the state-action space. In a recent chapter [12], we and our collaborators developed a greedy heuristic sampling strategy, called DDV, that tries to sample the state-action pair that will most rapidly shrink the confidence interval on the value of the optimal policy in the starting state. This was shown experimentally to reduce the number of simulator calls required to obtain a tight confidence interval compared to trajectory-based sampling.

An alternative to local confidence intervals are "global" (or "trajectory-wise") confidence intervals. Given a set of trajectories generated according to a fixed policy π, we can compute the return from each trajectory and then apply either the Hoeffding bound [32] or the empirical Bernstein bound [2] to compute a confidence interval on the expected return of the fixed policy π. Such intervals can be much tighter, because, unlike the local intervals, they do not need to

propagate the error by dynamic programming nor do they need to invoke a union bound to ensure that each of the individual local intervals holds simultaneously. However, these global intervals do not provide an upper bound on the value of the optimal policy, and, hence, they cannot be used to compute π^{UCB}. Is there any way global bounds can still be useful?

The key observation underlying this chapter is that for purposes of computing a *lower* bound on the value of the optimal policy, we can use a global lower confidence bound computed using trajectories from *any* policy. Hence, we introduce a combined local-global algorithm. It uses a local confidence interval to compute an upper bound on V^* (and to compute π^{UCB}), and it uses a global confidence interval (also based on π^{UCB}) to compute a lower bound on V^*.

A naive implementation of the global lower bound would require drawing a new set of trajectories every time π^{UCB} changed. Instead, we introduce an algorithm that re-uses previously-collected samples to compute a set of "equivalent trajectories" for π^{UCB}.

Once we have effective ways of computing the local and global confidence intervals, the remaining task is to develop an exploration algorithm for deciding which state-action pairs to sample at each point. Instead of the DDV greedy heuristic, we develop a new exploration heuristic that is based on optimal sample allocation for policy evaluation. For each of the confidence interval methods (local Hoeffding, local empirical Bernstein, local Weissman, global Hoeffding, and global empirical Bernstein), we derive an optimal or near-optimal algorithm for allocating a fixed sampling budget. We then apply this to the problem of policy optimization by performing exploration in a series of mini-batches. In each mini-batch, we hold π^{UCB} constant and allocate simulator calls according to the optimal policy evaluation method. We call the resulting algorithm LGCV (for Local-Global Confidence Value).

Initial experiments showed that while LGCV provides very substantial reductions in simulator calls on some problems, it can also perform poorly on other problems. This motivated us to develop a second algorithm, LLGCV (for Local-Local-Global Confidence Value), which computes the lower confidence bound by taking the maximum of the global and local confidence intervals.

To assess the effectiveness of LGCV and LLGCV, we perform experiments on several benchmark domains and on multiple configurations of an invasive species management problem. The experiments show that LLGCV provides robust overall performance including in cases where LGCV does not perform well. Hence, we conclude that the local-global strategy is generally preferred over all previous methods.

The chapter is organized as follows. Section 3.2 introduces our notation and the various

confidence interval methods. Section 3.3 considers the question of how to optimally allocate a fixed sampling budget for policy evaluation for each of the methods. It includes an experimental comparison of the different confidence interval methods that may be of independent interest. Section 3.4 introduces LGCV and LLGCV, our new algorithms that combine local confidence bounds with a global lower confidence bound during MDP planning. Section 3.5 presents an experimental evaluation of LGCV, LLGCV, and a comparison with existing methods. Section 3.6 concludes the chapter.

3.2 Problem Definition, Notation, and Confidence Interval Methods

Let a finite simulator-defined MDP consist of a start state s_0, a finite set of possible states S, a finite set of possible actions A, a discount factor $\gamma \in (0, 1]$, and a stochastic function F that maps from an input state-action pair (s, a) to a resulting state s' and reward r, where $s' \sim P(s'|s, a)$ is sampled according to the (unknown) transition function, $r \sim R(r|s, a)$ is sampled according to the unknown reward function, and $0 \leq r \leq R_{max}$. In this chapter, we will assume that the reward is deterministic; our methods can be easily extended to handle stochastic rewards. A (deterministic) policy π is a function mapping from states s to actions $a = \pi(s)$. The value of the policy in the start state, $V^\pi(s_0)$, is the expected discounted cumulative reward:

$$V^\pi(s_0) = E\left[\sum_{t=0}^{\infty} \gamma^t r_t \;\middle|\; s = s_0\right].$$

Let $V_{max} = \frac{R_{max}}{1-\gamma}$ denote the maximum possible value of any state under any policy. The corresponding minimum possible value is zero.

An optimal policy π^* maximizes $V^\pi(s_0)$, and the corresponding value is denoted by $V^*(s_0)$. The action-value of state s and action a under policy π is defined as $Q^\pi(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a)V^\pi(s')$. The optimal action-value is denoted $Q^*(s, a)$.

As a learning algorithm explores the MDP, it collects the following statistics. Let $N(s, a)$ be the number of times state-action pair (s, a) is simulated during learning. Let $N(s, a, s')$ be the number of times that s' has been observed as the resulting state. Let $R(s, a)$ be the observed reward. Let $\hat{P}(s'|s, a) = N(s, a, s')/N(s, a)$ be the maximum likelihood estimate for $P(s'|s, a)$.

Definition 6 *When executing a fixed policy π, the MDP becomes a Markov Reward Process*

(MRP) with transition function $P(s'|s) = P(s'|s, \pi(s))$, reward function $R(s) = R(s, \pi(s))$, and value function $V(s)$. Similarly, for each state s, $N(s)$ samples have been drawn of which $N(s, s')$ samples resulted in a transition to state s'.

The goal of our MDP planning algorithms is to draw samples from the simulator and then output a policy π that is approximately optimal with high probability. This is captured by the following definition due to Fiechter.

Definition 7 *[22]. A learning algorithm is PAC-RL if for any discounted MDP $(S, A, P, R, \gamma, s_0)$, $\epsilon > 0$, $1 > \delta > 0$, and $0 \leq \gamma < 1$, the algorithm halts and outputs a policy π such that*

$$\mathbb{P}\left[|V^*(s_0) - V^\pi(s_0) \mid \leq \epsilon\right] \geq 1 - \delta,$$

in time polynomial in $|S|$, $|A|$, $1/\epsilon$, $1/\delta$, $1/(1-\gamma)$, and R_{max}.

Note that this goal is different from the goal of exhibiting approximately optimal behavior over an infinite horizon, which is formalized by the notion of a PAC-MDP algorithm [36]. Unlike a PAC-MDP algorithm, a PAC-RL algorithm does not face an exploration-exploitation tradeoff. Instead, it performs pure exploration, but with the goal of minimizing the number of calls to the simulator.

A confidence interval is a pair of random variables $\underline{V}(s_0), \overline{V}(s_0)$ such that with probability $1 - \delta$, $\underline{V}(s_0) \leq V^\pi(s_0) \leq \overline{V}(s_0)$. Similarly, $\underline{Q}(s, a)$ and $\overline{Q}(s, a)$ denote the confidence bounds over the action-value functions. We follow the "Optimism Under Uncertainty" principle, and denote by π^{UCB} the policy based on an upper confidence bound on the action-value function:

$$\pi^{UCB}(s) = \operatorname*{argmax}_a \overline{Q}(s, a).$$

Let us now consider the methods available for computing such confidence intervals.

3.2.1 Global (Trajectory-wise) Confidence Intervals

A trajectory of length H provides a truncated sample of the infinite-horizon return:

$$v = \sum_{t=0}^{H} \gamma^t r_t \leq \sum_{t=0}^{\infty} \gamma^t r_t,$$

where $r_t = R(s_t|s_{t-1}, a_{t-1})$. The error introduced by truncating the trajectory lies in the interval $[0, \gamma^H V_{max}]$. Given the returns $v_1, \ldots, v_N$ from N trajectories of length H, let

$$\hat{V}(s_0) = \frac{1}{N} \sum_{n=1}^{N} v_n$$

be the mean of these returns.

Global methods compute confidence intervals on the expected value of the returns at s_0 and obtain a confidence interval for s_0. We refer to this family of confidence intervals as Global Confidence Value (GCV) methods.

The Hoeffding Method: Because the value of s_0, $V^\pi(s_0)$, lies in the interval $[0, V_{max}]$, we can apply the Hoeffding bound [32] to obtain a confidence interval on $V^\pi(s_0)$. We will call this method GCV(H).

Lemma 4 *Let $\hat{V}^\pi(s_0)$ be the mean return obtained from N trajectories of length H obtained by executing policy π starting in s_0. Then with probability $1 - \delta$,*

$$\hat{V}^\pi(s_0) - V_{max}\sqrt{\frac{\ln 2/\delta}{2N}} \leq V^\pi(s_0) \leq \hat{V}^\pi(s_0) + V_{max}\sqrt{\frac{\ln 2/\delta}{2N}} + \gamma^H V_{max}. \tag{3.1}$$

We will denote the width of the confidence interval by $\Delta V(s_0)$. Hence,

$$\Delta V^\pi(s_0) = 2V_{max}\sqrt{\frac{\ln 2/\delta}{2N}} + \gamma^H V_{max}. \tag{3.2}$$

The Empirical Bernstein Method: Let $\widehat{Var}^\pi(s_0) = \frac{1}{N}\sum_{n=1}^{N}[v_n - \hat{V}^\pi(s_0)]^2$ be the estimated variance of the returns along the trajectories generated by π. Then we can apply the empirical Bernstein bound [2] to obtain the following confidence interval. We will call this method GCV(B).

Lemma 5 *Let $\hat{V}^\pi(s_0)$ and $\widehat{Var}^\pi(s_0)$ be the sample mean and sample variance of the return obtained from N trajectories of length H by executing π starting in s_0. Then with probability*

$1-\delta$,

$$\begin{aligned}\hat{V}^{\pi}(s_0) - \sqrt{\frac{2\widehat{Var}^{\pi}(s_0)\ln 3/\delta}{N}} - \frac{3V_{max}\ln 3/\delta}{N} \\ \leq V^{\pi}(s_0) \leq \\ \hat{V}^{\pi}(s_0) + \sqrt{\frac{2\widehat{Var}^{\pi}(s_0)\ln 3/\delta}{N}} + \frac{3V_{max}\ln 3/\delta}{N} + \gamma^{H}V_{max}.\end{aligned} \tag{3.3}$$

Hence,

$$\Delta V^{\pi}(s_0) = 2\sqrt{\frac{2\widehat{Var}^{\pi}(s_0)\ln 3/\delta}{N}} + \frac{6V_{max}\ln 3/\delta}{N} + \gamma^{H}V_{max}.$$

3.2.2 Local Confidence Intervals

Local methods compute confidence intervals on the value function at each state and then combine these via dynamic programming to obtain a confidence interval for s_0. We refer to this family of confidence intervals as Local Confidence Value Iteration (LCVI) methods. We present these intervals in their general form for computing optimal policies. They can be converted to the fixed policy form for Markov Reward Processes by replacing the choice of actions a with $\pi(s)$ for fixed policy π.

The Hoeffding Method: Even-Dar et al. [15, 17] introduced a method for computing a confidence interval at the start state s_0 based on applying dynamic programming to confidence intervals computed in each state. This method is based on the following two Bellman-like equations that incorporate the Hoeffding bound:

$$\overline{V}(s) = \max_a R(s,a) + \gamma \sum_{s'} \hat{P}(s'|s,a)\overline{V}(s') + \gamma V_{max}\sqrt{\frac{\ln 2/\delta_0}{2N(s,a)}} \tag{3.4}$$

$$\underline{V}(s) = \max_a R(s,a) + \gamma \sum_{s'} \hat{P}(s'|s,a)\underline{V}(s') - \gamma V_{max}\sqrt{\frac{\ln 2/\delta_0}{2N(s,a)}} \tag{3.5}$$

Here, δ_0 is derived from δ to ensure that all of the confidence intervals computed hold simultaneously with probability $1-\delta$. For example, if we need to compute a confidence interval for all

$|A||S|$ state-action pairs, then $\delta_0 = \delta/|A||S|$. This is because the confidence bounds (but not the center of the confidence interval) are computed only once for each (s, a) pair, and these bounds depend only on $N(s, a)$.

These equations can be iterated to convergence. At convergence, with probability $1 - \delta$, $\underline{V}(s_0) \leq V^*(s_0) \leq \overline{V}(s_0)$. We will call this method LCVI(H).

The Empirical Bernstein Method: This approach can be extended by replacing the Hoeffding bound with the empirical Bernstein bound to obtain LCVI(B). Let $M(s, a)$ denote the sample mean of the discounted backed-up values from the successor states by taking action a in state s, and $Var(s, a)$ denote the sample variance of these values by taking action a in state s. We denote the upper and lower bounds on these values as $\overline{M}(s, a)$, $\underline{M}(s, a)$,$\overline{Var}(s)$, and $\underline{Var}(s)$.

$$\begin{aligned} \overline{M}(s,a) &= \sum_{s'} \hat{P}(s'|s,a)\gamma\overline{V}(s') \\ \overline{Var}(s,a) &= \sum_{s'} \hat{P}(s'|s,a)[\gamma\overline{V}(s') - \overline{M}(s,a)]^2 \end{aligned}$$

$$\overline{V}(s) = \max_a R(s,a) + \overline{M}(s,a) \; + \; \sqrt{\frac{2\overline{Var}(s,a)\ln(3/\delta_0)}{N(s,a)}} + \frac{3\gamma V_{max}\ln(3/\delta_0)}{N(s,a)} \tag{3.6}$$

$$\begin{aligned} \underline{M}(s,a) &= \sum_{s'} \hat{P}(s'|s,a)\gamma\underline{V}(s') \\ \underline{Var}(s) &= \sum_{s'} \hat{P}(s'|s,a)[\gamma\underline{V}(s') - \underline{M}(s,a)]^2 \end{aligned}$$

$$\underline{V}(s) = \max_a R(s,a) + \underline{M}(s,a) \; - \; \sqrt{\frac{2\underline{Var}(s,a)\ln(3/\delta_0)}{N(s,a)}} - \frac{3\gamma V_{max}\ln(3/\delta_0)}{N(s,a)} \tag{3.7}$$

These equations can be iterated to convergence. At convergence, for an appropriate choice of δ_0, with probability $1 - \delta$, $\underline{V}(s_0) \leq V(s_0) \leq \overline{V}(s_0)$.

The Weissman Method: Strehl and Littman [63, 62] compute a different local confidence

interval based on a multinomial confidence interval proved by Weissman et al. [73]:

$$CI(\hat{P}|N(s,a),\delta_0) = \left\{ \tilde{P} \;\middle|\; \|\tilde{P} - \hat{P}\|_1 \leq \omega(N(s,a),\delta_0) \right\}, \tag{3.8}$$

where $\|\cdot\|_1$ is the L_1 norm and $\omega(N(s,a),\delta_0) = \sqrt{\frac{2[\ln(2^{|S|}-2)-\ln\delta_0]}{N(s,a)}}$. With probability $1-\delta_0$ the true multinomial distribution $P(s'|s,a)$ is contained in this set of distributions. Strehl and Littman then define Equation (3.9) and Dietterich et al. [12] added the corresponding lower bound.

$$\overline{V}(s) = \max_a R(s,a) + \gamma \max_{\tilde{P} \in CI(\hat{P},N(s,a))} \sum_{s'} \tilde{P}(s'|s,a)\overline{V}(s') \tag{3.9}$$

$$\underline{V}(s) = \max_a R(s,a) + \gamma \min_{\tilde{P} \in CI(\hat{P},N(s,a))} \sum_{s'} \tilde{P}(s'|s,a)\underline{V}(s') \tag{3.10}$$

We will refer to this method as LCVI(W). We have included it here for completeness.

3.2.3 The Occupancy Measure

Our derivations employ the occupancy measure μ^π of the MDP for policy π, which is defined as

$$\mu^\pi(s) = \mathbb{E}_P \left[\sum_{t=0}^{\infty} \gamma^t I[s_t = s] \;\middle|\; s_0, \pi \right],$$

where $I[\cdot]$ is the indicator function and the expectation is taken with respect to the transition distribution. This is the cumulative discounted probability that the MDP will occupy state s under policy π for discount factor γ. It can be computed via dynamic programming on the Bellman flow equation [64]:

$$\mu^\pi(s) = I[s = s_0] + \gamma \sum_{s^-} \mu^\pi(s^-) P(s|s^-, \pi(s^-)). \tag{3.11}$$

This says that the discounted probability of visiting state s is equal to the sum of the probability that s is the starting state and the probability of reaching s by first visiting state s^- and then executing an action that leads to state s.

It is easy to show that

$$V^\pi(s_0) = \sum_s \mu^\pi(s) R(s, \pi(s)). \tag{3.12}$$

We adopt the notation μ^{UCB} for the occupancy measure computed based on the principle of optimism under uncertainty (or Upper Confidence Bound).

3.3 Monte Carlo Policy Evaluation

We now consider the problem of PAC policy evaluation. Given a fixed policy π and a sampling budget B, what is the tightest $1-\delta$ confidence interval that we can compute on $V^{\pi}(s_0)$ after B calls to the simulator.

The standard approach to Monte Carlo policy evaluation is to simulate a series of trajectories, compute the return of each trajectory, and take the average of these values. Appropriate confidence intervals on these average values can be used to form local and global methods using the fixed-policy versions of the methods described above. For trajectory-based methods, this makes sense. However, for local methods, it might be better to allocate more simulator calls to some states than to others.

In this section, we study this issue. We give exact answers for the global methods and for the local Hoeffding and empirical Bernstein methods. For the Weissman, et al., bound, we provide a heuristic solution. To assess the relative merits of the different confidence intervals, we perform a series of experiments comparing the methods on four benchmark problems and five configurations of an invasive species management task. The results show that the global method based on the empirical Bernstein bound gives the tightest intervals in all cases, except for two small MRPs. In these two small MRPs, the local method based on empirical Bernstein bound gives the tightest bound.

3.3.1 Optimal Allocation of Sampling

Given an overall sampling budget B, we must decide how to draw the samples. In this section, we develop sampling algorithms for each confidence interval method.

3.3.1.1 Optimal Sampling for Trajectory-wise Methods

For trajectory-wise methods, we must determine the length H of each trajectory. If H is too short, then the truncation error $\gamma^H V_{max}$ will dominate the confidence interval. But if H is too large, then the number of trajectories $N = \lfloor B/H \rfloor$ will be small, so the confidence interval will

be very wide.

Theorem 2 *For trajectory-wise sampling with the Hoeffding bound method with sampling budget B, discount factor γ, and confidence parameter δ, the optimal trajectory length H is the solution to the equation*

$$H = \frac{\frac{1}{2}\ln\ln 2/\delta - \frac{1}{2}\ln 2B - \ln\ln 1/\gamma}{\ln\gamma} - \frac{\ln H}{2\ln\gamma}. \tag{3.13}$$

Proof 5 *We seek to minimize the width of the confidence interval $\Delta V(s_0)$. From Equation 3.2, we can write this as*

$$\Delta V(s_0) = V_{max}\left[2\sqrt{\frac{\ln 2/\delta}{2\frac{B}{H}}} + \gamma^H\right].$$

The derivative of the quantity in square brackets is

$$\frac{1}{H}\sqrt{\frac{H\ln 2/\delta}{2B}} + \gamma^H \ln\gamma.$$

Setting this to zero and rearranging terms, we obtain the result.

This equation lacks a closed-form solution, but it is easy to solve numerically by iterating Equation (3.13) to convergence.

Following the same line of reasoning, we can obtain a similar solution for the empirical Bernstein bound method.

Theorem 3 *For trajectory-wise sampling with the empirical Bernstein method with sampling budget B, maximum value V_{max}, variance Var, discount factor γ, and confidence parameter δ, the optimal trajectory length H is the solution to the equation*

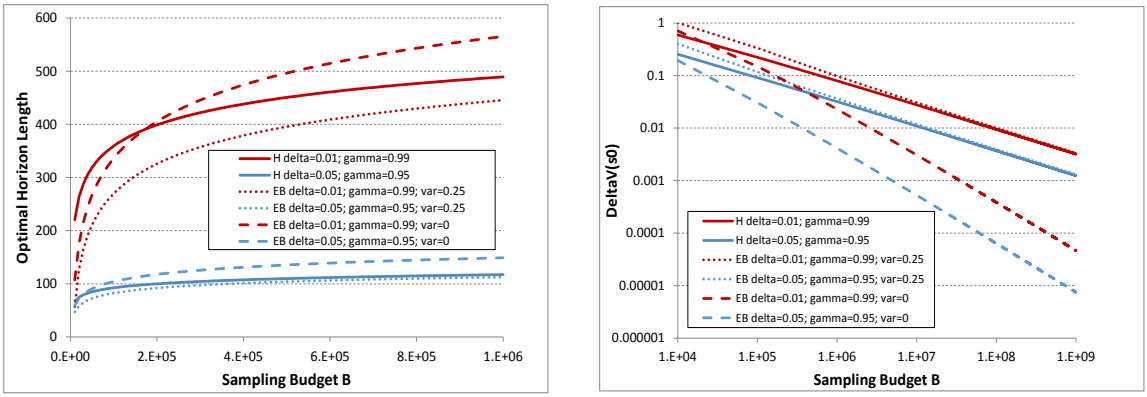
$$H = \frac{\ln\left[\sqrt{\frac{2Var\ln 3/\delta}{BH}} + \frac{6V_{max}\ln 3/\delta}{B}\right] - \ln V_{max} - \ln\ln 1/\gamma}{\ln\gamma}. \tag{3.14}$$

This can also be rapidly solved by simple iteration.

To apply this formula, we need an estimate of the variance of the cumulative discounted reward. It is easy to show that the maximum possible variance is $V_{max}^2/4$, and hence, we can employ this if a tighter value is not known for the particular problem.

Figure 3.1(a) shows the optimal values of H for the Hoeffding and empirical Bernstein bounds. We show the results for various budgets B, two settings of δ and γ, and, for the Bernstein bound, with two settings of Var. Note that the optimal trajectory length is independent of the size of the state space.

We can also compute the width of the global confidence interval as a function of B, as it only depends on the sampling budget, δ, γ, and Var and not on the actual rewards received. Figure 3.1(b) shows that the Hoeffding bound can be tighter for small budgets but that as B grows larger, the empirical Bernstein bound becomes tighter. How large a sampling budget do we need to drive $\Delta V(s_0) \leq 0.1 \times V_{max}$? For $\delta = 0.05$ and $\gamma = 0.95$, we need about 2.4×10^4 and 1.4×10^5 samples for the empirical Bernstein bound when $Var = 1/4$ and $Var = 0$, and 8×10^4 samples for the Hoeffding bound. When $\delta = 0.01$ and $\gamma = 0.99$, we need a much larger budget of about 9.6×10^5 and 1.5×10^5 samples for the empirical Bernstein bound when $Var = 1/4$ and $Var = 0$, and 6×10^5 for the Hoeffding bound.



(a) Optimal horizon length H as a function of the sampling budget B.

(b) $\Delta V(s_0)$ as a function of the sampling budget B.

Figure 3.1: Optimal horizon length H and the gap in the starting state $\Delta V(s_0)$ as a function of sampling budget B for computing the trajectory-wise confidence interval via the Hoeffding and empirical Bernstein bounds. In these plots, we set $V_{max} = 1$.

3.3.1.2 Optimal Sampling for Local Methods

We now present sampling algorithms for three methods based on local confidence intervals: Hoeffding, empirical Bernstein, and Weissman.

Our strategy for minimizing $\Delta V(s_0)$ is to obtain an expression for it in terms of the local confidence intervals. We will do this by computing a dynamic program for the width of the local confidence intervals and then noticing that this dynamic program is identical to value iteration in a MRP with a particular reward function. We can then apply Equation (3.12) to map that reward function into a non-recursive expression for $\Delta V(s_0)$.

Theorem 4 *For the local Hoeffding bound method, $N(s)$ samples should be allocated to state s to minimize*

$$\Delta V(s_0) = \sum_s \mu(s) 2\gamma V_{max} \sqrt{\frac{\ln 2/\delta_0}{2N(s)}}.$$

Proof 6 *We begin by subtracting Equation (3.5) from (3.4) to obtain*

$$\overline{V}(s) - \underline{V}(s) \;=\; \gamma \sum_{s'} \hat{P}(s'|s)[\overline{V}(s') - \underline{V}(s')] + \; 2\gamma V_{max} \sqrt{\frac{\ln 2/\delta_0}{2N(s)}}$$

We rewrite this as

$$\Delta V(s) \;=\; \gamma \sum_{s'} \hat{P}(s'|s) \Delta V(s') + \; 2\gamma V_{max} \sqrt{\frac{\ln 2/\delta_0}{2N(s)}}.$$

We can recognize this as the Bellman equation for a Markov Reward Process for which $\Delta V(s)$ is the value function and $2\gamma V_{max}\sqrt{\frac{\ln 2/\delta_0}{2N(s)}}$ is the reward function. The result is obtained by applying Equation (3.12).

Algorithm 5 presents the HOEFFDINGSAMPLER. It repeatedly samples from the state that will most decrease $\Delta V(s_0)$ and then recomputes the occupancy measure. This is an instance of coordinate descent in the objective. Because the objective is convex, this is guaranteed to find the global optimum. In practice, the sampling could be performed in minibatches and the occupancy measure updated after each minibatch. The algorithm could be initialized by first drawing a small number of trajectories with H computed from Theorem 2. This will provide a useful initial estimate of μ.

Algorithm 5 HOEFFDINGSAMPLER$(\delta, \gamma, R_{max}, F)$

parameters:

B sampling budget
δ confidence parameter
γ discount factor
R_{max} maximum possible reward
$F : state \mapsto (state, reward)$ simulator

$N(s) := N(s, s') := 0$ for all s, s'
$N(s, s') := 0$ for all s, s'
$R_{sum}(s) := 0$ for all s
$K := \{s_0\}$ // the set of known states
Define $priority(s) = \mu(s)[1/\sqrt{N(s)} - 1/\sqrt{N(s)+1}]$
$\mu(s) := 0$ for all s
$\mu(s_0) := 1$
while $B > 0$ **do**
  Let s be the state in K that maximizes $priority(s)$
  $(s', r) := F(s)$ // draw a sample at s
  $N(s) := N(s) + 1$
  $N(s, s') := N(s, s') + 1$
  $R_{sum}(s) := R_{sum}(s) + r$
  $K := K \cup \{s'\}$
  $B := B - 1$
  Compute μ by solving Eq.(3.11)
end while
Return $N(s)$, $N(s, s')$, $R_{sum}(s)$ for all s, s'

Theorem 4 has such a simple form that it is possible to solve for the optimal allocation of samples.

Theorem 5 *The optimal sample allocation for the local Hoeffding bound is to assign samples $N(s)$ in proportion to $\mu(s)^{2/3}$.*

Proof 7 *Our goal is to find values $\vec{N} = (N(s_0), \ldots)$ to minimize*

$$J(\vec{N}) = \sum_s \mu(s) c / \sqrt{2N(s)}$$

subject to the constraint that $\sum_s N(s) = B$, the sampling budget. Here $c = 2\gamma V_{max}\sqrt{\ln 2/\delta_0}$.

We apply the Lagrange multiplier method:

$$\begin{aligned} L(N(s),\lambda) &= \sum_s \mu(s)\frac{c}{\sqrt{2N(s)}} + \lambda\left[\sum_s N(s) - B\right] \\ \frac{\partial L}{\partial N(s_j)} &= -\frac{1}{2}\mu(s_j)cN(s_j)^{-3/2} + \lambda \\ \frac{\partial L}{\partial \lambda} &= \sum_s N(s) - B \end{aligned}$$

Setting the partial derivatives equal to zero we obtain

$$\begin{aligned} N(s) &= \left(\frac{\mu(s)c}{2\lambda}\right)^{2/3} \\ \sum_s N(s) &= B \\ \sum_s \left(\frac{\mu(s)c}{2\lambda}\right)^{2/3} &= B \\ \lambda &= \frac{1}{B^{3/2}}\left[\sum_s \left(\frac{\mu(s)c}{2}\right)^{2/3}\right]^{3/2} = \frac{c}{2B^{3/2}}\left[\sum_s \mu(s)^{2/3}\right]^{3/2} \end{aligned}$$

Therefore,

$$N(s) = \left(\frac{\mu(s)c}{2\frac{c}{2B^{3/2}}\left[\sum_s \mu(s)^{2/3}\right]^{3/2}}\right)^{2/3} = \frac{\mu(s)^{2/3}}{\sum_s \mu(s)^{2/3}}B$$

The effect of this sampling strategy compared to sampling on trajectories is to shift samples somewhat towards lower-occupancy states. In particular, in a loop-free MDP, this will shift samples to states that are deeper in the MDP. This makes sense, because uncertainty deeper in the MDP will be magnified as it is propagated backwards toward the start state, so narrowing those deep confidence intervals gives a better overall result at the start state.

We can obtain similar results for the empirical Bernstein bound.

Theorem 6 *For the local empirical Bernstein bound method, $N(s)$ samples should be allocated*

to state s to minimize

$$\Delta V(s_0) = \sum_s \mu(s) \left[\frac{\sqrt{c_1 \overline{Var}(s)} + \sqrt{c_1 \underline{Var}(s)}}{\sqrt{N(s)}} + \frac{2c_2}{N(s)} \right],$$

where $c_1 = 2 \ln 3|S|/\delta$ and $c_2 = 3\gamma V_{max} \ln 3|S|/\delta$.

Proof 8 *Subtract Equation (3.7) from Equation (3.6) and simplify to obtain*

$$\Delta V(s) \;=\; \gamma \sum_{s'} P(s'|s) \Delta V(s') \;+\; \left[\frac{\sqrt{c_1 \overline{Var}(s)} + \sqrt{c_1 \underline{Var}(s)}}{\sqrt{N(s)}} + \frac{2c_2}{N(s)} \right].$$

We can again recognize this as an instance of the Bellman equation for which the term in brackets is the reward. The result follows by applying Equation (3.12).

To implement this sampling method, we can change the priority function in Algorithm 5 to be

$$\mu(s) \left\{ c_1 \left[\sqrt{\overline{Var}(s)} + \sqrt{\underline{Var}(s)} \right] \times \left(\frac{1}{\sqrt{N(s)}} - \frac{1}{\sqrt{N(s)+1}} \right) + 2c_2 \left(\frac{1}{N(s)} - \frac{1}{N(s)+1} \right) \right\}. \quad (3.15)$$

To obtain a sampling method for local confidence intervals computed with the Weissman method, we must introduce some approximations. Let $\Delta V_{FR}(s)$ be defined as follows:

$$\Delta V_{FR}(s) = \max_{\tilde{P} \in CI} \tilde{P}(s'|s) \hat{V}(s') - \min_{\tilde{P} \in CI} \tilde{P}(s'|s) \hat{V}(s').$$

The subscript FR denotes "fixed result". This is the (undiscounted) width of the Weissman confidence interval on $V(s)$ if we replace $\overline{V}(s')$ and $\underline{V}(s')$ in Equations 3.9 and 3.10 with $\hat{V}(s')$, the value function estimate that would be obtained by performing value iteration using the maximum likelihood estimates of the transition probabilities.

Heuristic 1 *To approximately minimize $\Delta V(s_0)$ for local multinomial confidence intervals, samples should be allocated to minimize*

$$\sum_s \mu(s) \gamma \Delta V_{FR}(s).$$

Derivation. Rewrite Equations (3.9) and (3.10) as

$$\overline{V}(s) = R(s) + \gamma \max_{\tilde{P} \in CI} \sum_{s'} \tilde{P}(s'|s)[\overline{V}(s') - \hat{V}(s') + \hat{V}(s')]$$

$$\underline{V}(s) = R(s) + \gamma \min_{\tilde{P} \in CI} \sum_{s'} \tilde{P}(s'|s)[\underline{V}(s') - \hat{V}(s') + \hat{V}(s')]$$

The first approximation is to split the maximizations and minizations

$$\overline{V}(s) \approx R(s) \ + \ \gamma \max_{\tilde{P} \in CI} \sum_{s'} \tilde{P}(s'|s)[\overline{V}(s') - \hat{V}(s')] + \gamma \max_{\tilde{P} \in CI} \sum_{s'} \tilde{P}(s'|s)\hat{V}(s')$$

$$\underline{V}(s) \approx R(s) \ + \ \gamma \min_{\tilde{P} \in CI} \sum_{s'} \tilde{P}(s'|s)[\underline{V}(s') - \hat{V}(s')] + \gamma \min_{\tilde{P} \in CI} \sum_{s'} \tilde{P}(s'|s)\hat{V}(s')]$$

This approximation will widen the gap between the lower and upper bounds. The second approximation is to replace the maximization and minimization over $\tilde{P}$ in the first terms with the maximum likelihood estimate $\hat{P}$ of the transition probabilities. This will shrink the gap between the lower and upper bounds.

$$\overline{V}(s) \approx R(s) \ + \ \gamma \sum_{s'} \hat{P}(s'|s)[\overline{V}(s') - \hat{V}(s')] + \gamma \max_{\tilde{P} \in CI} \sum_{s'} \tilde{P}(s'|s)\hat{V}(s')$$

$$\underline{V}(s) \approx R(s) \ + \ \gamma \sum_{s'} \hat{P}(s'|s)[\underline{V}(s') - \hat{V}(s')] + \gamma \min_{\tilde{P} \in CI} \sum_{s'} \tilde{P}(s'|s)\hat{V}(s')]$$

Now subtract the lower bound from the upper bound and simplify to obtain

$$\Delta V(s) \approx \gamma \sum_{s'} \hat{P}(s'|s)\Delta V(s') + \gamma \Delta V_{FR}(s).$$

We recognize this as an approximation of the Bellman equation for a MRP with reward function $\gamma \Delta V_{FR}(s)$. Applying Equation (3.12) completes the derivation. □

This method can be implemented as a change to the priority function in Algorithm 5. Let $\Delta V_{FR}^{N}(s)$ denote ΔV_{FR} computed using N samples. Then we can set the priority function to $\mu(s)\gamma[\Delta_{FR}^{N(s)}(s) - \Delta_{FR}^{N(s)+1}(s)]$. For the Weissman interval, we can estimate $\Delta_{FR}^{N(s)+1}(s)$ by replacing N by $N+1$ in Equation (3.8) when computing the confidence intervals.

3.3.2 Experimental Comparison of Global and Local Confidence Intervals for Policy Evaluation

Now that we have determined the optimal way to allocate samples for both the local and global methods, we turn our attention to the question of which methods give the tightest confidence intervals. We present the computed $\Delta V(s_0)$ for five confidence interval methods: the global empirical Bernstein bound ("GCV(B)"), global Hoeffding bound ("GCV(H)"), local Hoeffding bound ("LCVI(H)"), local Bernstein bound ("LCVI(B)"), and the local Weissman bound ("LCVI(W)"). We first describe the benchmark MDPs and corresponding policies. We then provide the numerical results in tabular format.

3.3.2.1 Benchmarks and Evaluated Policies

We employed the following benchmark MDPs with modifications as indicated. We have modified all of the benchmarks to use rewards of the form $R(s, a)$ rather than $R(s, a, s')$ where necessary. We also normalized the rewards in these benchmarks so that the maximum possible reward in any state, R_{max} is 1.

The RiverSwim benchmark is studied by Strehl and Littman [62]. We evaluate the optimal policy, which is $\pi(s) = 1, \forall s \in S$ and corresponds to moving to the rightmost state.

The SixArms benchmark is studied by Strehl and Littman [62]. The policy evaluated on this benchmark is $\pi(0) = 4$ and $\pi(5) = 4$. The optimal policy is $\pi^*(0) = 5$ and $\pi^*(6) = 6$. Under either policy, only three states are visited.

The Casino Land benchmark is studied by Strehl and Littman [63]. We modified Casino Land to introduce additional stochasticity. We modified the transitions of type $(a = 1, p = 1, r = 0)$ for $s \in \{0, 1, 2, 3\}$, where a is an action, p the probability of the corresponding transition, and r is the reward for taking the transition. For each state $s \in \{0, 1, 2\}$, the transition is modified so that it transits to s and $s + 1$ with probabilities 0.5 and 0.5. For state $s = 3$, the transition is modified so that it transits to s and $s - 1$ with probabilities 0.5 and 0.5. The policy that was evaluated is $\pi(s) = 1, \forall s \in S$.

The Combination Lock benchmark is studied by Gheshlaghi Azar et al. [29]. We modified it to provide some intermediate rewards with the goal of increasing the variance of the value function in some of the intermediate states. We thought that this might reveal interesting behavior for the local empirical Bernstein method.

We employed a Combination Lock MDP with 50 states, where state 50 is the terminal state with reward 1000. There are two actions $\{0, 1\}$. The first action's transitions are specified in Equation 3.16. The second action makes a deterministic transition from s to $s + 1$. Odd numbered states get reward 10, and even numbered states get reward 30 for action 0. These rewards are 0 in the original MDP. The policy that we evaluate is $\pi(s) = 1, \forall s \in S \setminus \{2\}$ and $\pi(s) = 0$ for $s = 2$, where the starting state is $s = 0$.

$$n(x_k, x_l) = \begin{cases} \frac{1}{k+1} & \text{for } l = k + 1 \\ \frac{1}{k-l} & \text{for } l < k \\ 0 & \text{otherwise} \end{cases} \quad , \quad P(x_l|x_k, 0) = \frac{n(k, l)}{\sum_{s_m \in S} n(k, m)} \tag{3.16}$$

Finally, we evaluate the methods on the Tamarisk invasive species management problem developed by [31][1]. The state of the MDP consists of a tree-structured river network as shown in Figure 3.2. The network contains E edges. Each edge in turn has H slots at which a plant can grow.

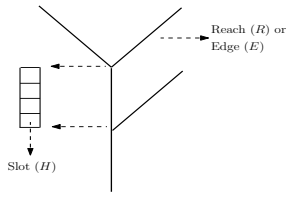


Figure 3.2: Tamarisk structure

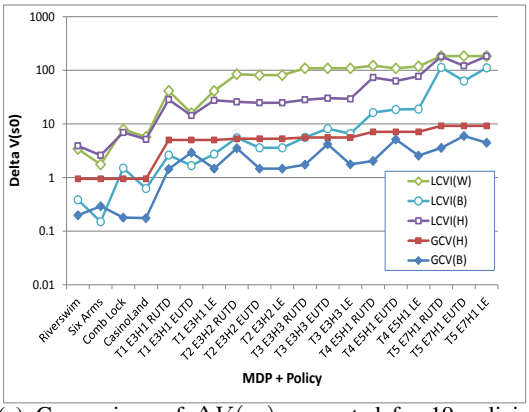
The problem instances vary according to the number of river segments and the number of slots: $(E = 3, H = 1)$, $(E = 3, H = 2)$, $(E = 3, H = 3)$, $(E = 5, H = 1)$, and $(E = 7, H = 1)$. For each of these MDPs, we evaluated three policies [7, 33, 47]: Eradicate upstream first (EUTD), restore upstream first (RUTD), and eradicate the leading edge (LE). We configured the system dynamics and cost function variables for the Tamarisk management benchmark as follows. The values for the system dynamics are: eradication rate (0.85), restoration rate (0.65), downstream spread rate (0.5), upstream spread rates (0.1), death rate (0.2), and production rate for native and tamarisk plants (100, 2000).

The cost function has two kinds of variables: variables that describe the cost of states and variables that describe the cost of actions. For the "cost of states variables", we set the cost per invaded edge to 1), the cost per tree to 0.1, and the cost per empty slot to 0.05. The values for the "cost of actions" variables consist of fixed and variable costs. The values of the fixed cost variables are eradication cost (0.2) and restoration cost (0.2). The values of the variable cost are varied based on the number of slots. The variable cost values for $H = 1$ are variable eradication

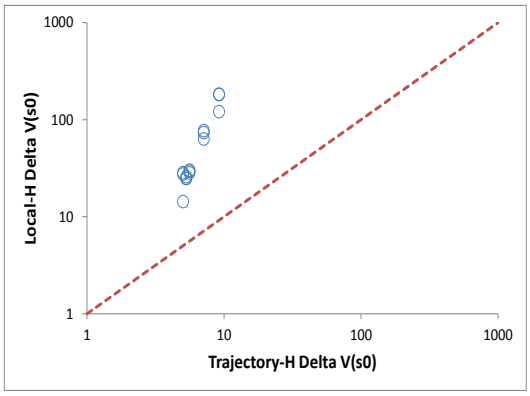
[1]The Tamarisk domain description can be found at `http://2013.rl-competition.org/domains/invasive-species`

cost (0.8), variable restoration cost for empty slot (0.8), and variable restoration cost for invaded slot (1.6). For $H = 2$, the values are the same as $H = 1$, except that the variable restoration cost for invaded slot (0.8). For $H = 3$, the variables are variable eradication cost (0.6), variable restoration cost for empty slot (0.6), and variable restoration cost for invaded slot (0.6). The budget value, which limits the number of possible actions, is 2.

We employed the following starting states. We denote each state by ordering the edges from left-to-right and top-to-bottom and then listing the contents of each slot using the notation T (Tamarisk), N (Native), and E (Empty). The starting states that we employed were TTT ($E = 3, H = 1$), NTNEEE ($E = 3, H = 2$), NTENEEEEE ($E = 3, H = 3$), TNTTTT ($E = 5, H = 1$), and TTTNNEE ($E = 7, H = 1$).



(a) Comparison of $\Delta V(s_0)$ computed for 19 policies (using 9 MDPs). The notation Tn EeHh denotes Tamarisk MDP n, with $E = e$ edges and $H = h$ slots. Solid markers correspond to trajectory-based methods; open markers correspond to local methods.



(b) Comparison of $\Delta V(s_0)$ computed by the local and trajectory-wise Hoeffding bound methods. Each circle corresponds to one of the 19 MDP-policy combinations. All points are above the line, which corresponds to cases where the trajectory-wise interval is smaller.

Figure 3.3: Comparison of $\Delta V(s_0)$ for different five different algorithms [left]. Comparison of $\Delta V(s_0)$ computed by the local and trajectory-wise Hoeffding bound methods [right].

For each MDP and each sampling algorithm, we specified a budget of 500,000 samples and set $\delta = 0.01$, $\gamma = 0.95$. We performed 30 replications of each combination of MDP and algorithm and then computed the average width of the confidence interval in the starting state $\Delta V(s_0)$.

The results are summarized in Figure 3.3(a), where the computed value of $\Delta V(s_0)$ is plotted on a log scale. Figure 3.3(b) plots a direct comparison of the global and local Hoeffding methods.

Table 3.1 show $\Delta V(s_0)$ for the benchmark problems and Tables 3.2, 3.3, 3.4, 3.5, and 3.6 show $\Delta V(s_0)$ for five configurations of the Tamarisk management problem.

The width of the confidence intervals computed by the five confidence interval methods varies widely, often by factors of more than 50. With two exceptions, the global methods are always better, and the Bernstein method, GCV(B), is always better than the Hoeffding method, GCV(H). GCV(H) is always better than LCVI(H), as predicted by our analysis.

Similarly, among the local methods, the Bernstein method, LCVI(B) is the best. In two cases, SixArms and T1E3H1 EUTD, it is even better than the global Hoeffding method, GCV(H). We note that these two cases involve very small MRPs that only visit 2 and 4 states, respectively. When a policy makes repeated visits to a state, the local confidence intervals pool the statistics from those visits and so can become very tight. In addition, the uncertainty does not need to propagate very far in the MRP to reach the start state. This allows the local methods to outperform the global methods.

	GCV(B)	GCV(H)	LCVI(W)	LCVI(H)	LCVI(B)
RiverSwim	$\mathbf{0.198 \pm 0.037}$	0.948 ± 0.174	3.408 ± 0.623	3.907 ± 0.714	0.385 ± 0.071
SixArms	0.292 ± 0.054	0.948 ± 0.174	1.750 ± 0.320	2.598 ± 0.475	$\mathbf{0.150 \pm 0.028}$
CasinoLand	$\mathbf{0.176 \pm 0.032}$	0.948 ± 0.174	5.729 ± 1.046	5.169 ± 0.944	0.623 ± 0.114
CombinationLock	$\mathbf{0.180 \pm 0.033}$	0.948 ± 0.174	7.912 ± 1.445	6.937 ± 1.267	1.503 ± 0.275

Table 3.1: $\Delta V(s_0)$ for Four Benchmarks. The best performance is indicated by bold face.

	RUTD	EUTD	LE
GCV(B)	$\mathbf{1.440 \pm 0.263}$	2.918 ± 0.533	$\mathbf{1.471 \pm 0.269}$
GCV(H)	5.025 ± 0.918	5.025 ± 0.918	5.025 ± 0.918
LCVI(W)	41.350 ± 7.550	15.836 ± 2.892	41.286 ± 7.538
LCVI(H)	28.379 ± 5.182	14.329 ± 2.617	27.490 ± 5.019
LCVI(B)	2.628 ± 0.480	$\mathbf{1.674 \pm 0.306}$	2.742 ± 0.501

Table 3.2: $\Delta V(s_0)$ for Tamarisk $E = 3$ and $H = 1$

The fact that global, trajectory-wise confidence intervals are generally tighter than local intervals makes sense. Every confidence interval is somewhat conservative, because it is designed for extreme distributions of samples. Hence, when we combine many slightly conservative intervals via value iteration, we end up with an interval that is significantly wider (in most cases) than the global interval.

	RUTD	EUTD	LE
GCV(B)	$\mathbf{1.498 \pm 0.274}$	$\mathbf{3.52 \pm 0.643}$	$\mathbf{1.464 \pm 0.268}$
GCV(H)	5.309 ± 0.97	5.309 ± 0.97	5.309 ± 0.97
LCVI(W)	81.17 ± 0.005	84.561 ± 15.439	81.369 ± 14.856
LCVI(H)	24.556 ± 4.484	25.764 ± 4.704	24.77 ± 4.523
LCVI(B)	3.450 ± 0.630	5.502 ± 1.005	3.593 ± 0.656

Table 3.3: $\Delta V(s_0)$ for Tamarisk $E = 3$ and $H = 2$

	RUTD	EUTD	LE
GCV(B)	$\mathbf{1.739 \pm 0.318}$	$\mathbf{4.188 \pm 0.765}$	$\mathbf{1.761 \pm 0.322}$
GCV(H)	5.593 ± 1.022	5.593 ± 1.022	5.593 ± 1.022
LCVI(W)	108.74 ± 19.853	109.23 ± 19.943	108.73 ± 19.851
LCVI(H)	28.268 ± 5.161	30.182 ± 5.511	29.374 ± 5.363
LCVI(B)	5.61 ± 1.025	8.138 ± 1.486	6.597 ± 1.205

Table 3.4: $\Delta V(s_0)$ for Tamarisk $E = 3$ and $H = 3$

3.4 Policy Optimization

Now that we have determined that confidence intervals (global and local) based on the empirical Bernstein bounds are the tightest, we present our MDP planning algorithms, LGCV and LLGCV, that use those bounds. We describe LLGCV first, which is shown in Algorithm 6. In line 15, ExtendedValueIteration is invoked to compute upper and lower confidence bounds on the action values in every (reachable) state including the start state. Then in line 16 the upper confidence bound policy π^{UCB} is computed. The next line, 17 computes the global lower confidence bound using the Equivalent Trajectories Policy Evaluation algorithm that will be described below. Line 18 combines the global and local lower bounds on $V^*(s_0)$.

Exploration is performed in a series of minibatches of size MB, where we allocate one minibatch to 'upper' and another one to 'lower' iteratively. In each minibatch, LLGCV draws samples alternately aiming to reduce the upper confidence bound (ExMethod == upper) and to increase the lower confidence bound (ExMethod == lower) until the gap between them is less than ϵ. When sampling to reduce the upper bound, we use the result from Theorem 5 for the Hoeffding bound and allocate our minibatch samples in proportion to $(\mu^{UCB})^{2/3}$ (line 29). When sampling to increase the lower bound, we allocate our minibatch samples in proportion to

Algorithm 6 LLGCV $(s_0, F, \epsilon, \delta, \gamma, R_{max})$

parameters:

s_0 starting state
$F : (state, action) \mapsto (state, reward)$ simulator
ϵ accuracy parameter
δ confidence parameter
γ discount factor
R_{max} maximum possible reward {Global Data Structures}
$H := 1/(1-\gamma)\log(2R_{max}/\epsilon(1-\gamma))$ {$\epsilon/2$ horizon}
$N(s,a)$ number of times action a was executed in state s
$N(s,a,s')$ number of times the transition $(s,a) \mapsto s'$ was observed
$L(s,a)$ sequence data structure of the observed (s', r) transitions
$N(s,a) := 0, N(s,a,s') := 0, L(s,a) := \emptyset$

ExMethod := upper
loop
  $(\overline{V}(s_0), \underline{V}_{local}(s_0))$:= ExtendedValueIteration()
  $\forall s\ \pi^{UCB}(s) := \text{argmax}_a\ \overline{Q}(s,a)$
  $\underline{V}^{UCB}(s_0)$:= ETPE(π^{UCB})
  $\underline{V}(s_0) := \max\{\underline{V}(s_0), \underline{V}^{UCB}(s_0), \underline{V}_{local}(s_0)\}$

  if $\overline{V}(s_0) - \underline{V}(s_0) \leq \epsilon$ **then**
    Perform value iteration using $\hat{P}$ to obtain $\hat{\pi}$
    return $\hat{\pi}$
  end if

  MB := $4 \times H$
  if (ExMethod == upper) **then**
    Compute μ^{UCB}
    Compute $B_{upper} := \sum_s I[\mu^{UCB}(s) > 0] N(s, \pi^{UCB}(s))$
    N_{new} := AllocateSamples($(\mu^{UCB})^{2/3}, \pi^{UCB}, B_{upper} + MB$)
    DrawSamples(N_{new}, π^{UCB})
    ExMethod := lower
  else if (ExMethod == lower) **then**
    ρ^{UCB} :=UndiscountedOccupancy(π^{UCB}, H)
    Compute $B_{lower} := \sum_s I[\rho^{UCB}(s) > 0] N(s, \pi^{UCB}(s))$
    N_{new} := AllocateSamples($\rho^{UCB}(H), \pi^{UCB}, B_{lower} + MB$)
    DrawSamples(N_{new}, π^{UCB})
    ExMethod := upper
  end if
end loop

	RUTD	EUTD	LE
GCV(B)	**2.036 ± 0.372**	**5.135 ± 0.938**	**2.562 ± 0.468**
GCV(H)	7.11 ± 1.298	7.11 ± 1.298	7.11 ± 1.298
LCVI(W)	122.52 ± 22.368	108.91 ± 19.885	118.57 ± 21.647
LCVI(H)	73.339 ± 13.39	63.099 ± 11.52	76.969 ± 14.052
LCVI(B)	16.27 ± 2.971	18.515 ± 3.381	18.88 ± 3.448

Table 3.5: $\Delta V(s_0)$ for Tamarisk $E = 5$ and $H = 1$

	RUTD	EUTD	LE
GCV(B)	**3.573 ± 0.653**	**5.956 ± 1.088**	**4.463 ± 0.815**
GCV(H)	9.195 ± 1.679	9.195 ± 1.679	9.195 ± 1.679
LCVI(W)	184.3 ± 33.648	184.3 ± 33.648	184.3 ± 33.648
LCVI(H)	181.07 ± 33.059	121.01 ± 22.093	182.86 ± 33.385
LCVI(B)	113.41 ± 20.705	63.366 ± 11.569	110.98 ± 20.261

Table 3.6: $\Delta V(s_0)$ for Tamarisk $E = 7$ and $H = 1$

an undiscounted occupancy measure ρ that is computed by UNDISCOUNTEDOCCUPANCY (line 33). This is an undiscounted, fixed-horizon occupancy measure that corresponds to sampling along trajectories of length H.

The pseudo-code for UNDISCOUNTEDOCCUPANCY is shown in Algorithm 7. It works by converting the infinite-horizon MDP into an "unrolled" finite-horizon MDP as follows. Let $h \in \{0, \ldots, H\}$ index the levels of the unrolled MDP. Then the states of this MDP are pairs (s, h), the transition function is $P((s', h+1)|(s, h), a) = P(s'|s, a)$, and the reward function is $R((s, h), a) = R(s, a)$. Hence, each transition moves from one level to the next deeper level. Trajectories terminate when they reach level H. The undiscounted occupancy measure $\rho(s, h)$ is the probability that π will visit state s at level h given that it starts in state s_0 at level 0. This is computed in a single top-down pass (line 5). Note that $\sum_s \rho(s, h) = 1$ because the policy must visit *some* state at each level h during a trajectory.

We now describe the ETPE algorithm for computing the global trajectory-wise lower bound. Our approach is inspired by the work of [49], where they re-use a sequence of random values to evaluate different policies. Every time we invoke the simulator, we record the observed state transitions. Then, when asked to evaluate a new policy π, we replay the recorded transitions in the same order they were observed.

Algorithm 7 UNDISCOUNTEDOCCUPANCY(π, h_{max})

$\hat{P}(s'|s,a) = N(s,a,s')/N(s,a)$ for all (s,a) where $N(s,a) > 0$
$\rho(s,h) := 0$ for all $s, 0 \leq h < h_{max}$
$\rho(s_0, 0) := 1$
for $h = 1$ **to** h_{max} **do**
    $\rho(s',h) := \rho(s',h) + \rho(s,h-1)\hat{P}(s'|s,\pi(s))$ for all states s and s'
end for
return ρ

More precisely, for each state-action pair (s,a) in the MDP, we define a sequence data structure $SEQ(s,a)$ that supports four operations:

$SEQ(s,a).push(s',r)$ adds state s' and reward r to the end of the sequence.

$SEQ(s,a).reset()$ resets an internal index $SEQ(s,a).index$ so that it points to the first element in the sequence.

$SEQ(s,a).end()$ returns TRUE if the index points beyond the end of the sequence

$SEQ(s,a).next()$ returns the (s',r) sequence element indexed by $SEQ(s,a).index$ and increments the index. It raises an exception if $SEQ(s,a).end()$ is true.

Every time our exploration algorithm invokes the simulator F in (s,a) and observes the result (s',r), we invoke $SEQ(s,a).push(s',r)$ to add (s',r) to the end of the sequence. This is summarized in Algorithm 8. Conceptually, we can imagine that the simulator has access to a separate stream of random values for each (s,a) and uses that stream of randomness to generate its answer each time it is invoked.

Algorithm 8 DRAWSAMPLES(N_{new}, π)

for all s **do**
    for $n = 1$ **to** $N_{new}(s)$ **do**
        $a := \pi(s)$
        $(s',r) := F(s,a)$
        $R(s,a) := r$
        $SEQ(s,a).push(s',r)$
    end for
end for

Now given a policy π and the contents of these sequences, we generate a collection of trajectories, each of length H as shown in Algorithm 9. First, for all s and a, we invoke $SEQ(s,a).reset()$ to reset their indexes. Then we generate trajectory $n = 1, \ldots$ by starting in state s_0, choosing action $\pi(s_0)$, and then making a transition to the state $(s', r) := SEQ(s,a).next()$. We repeat this until we reach horizon depth H. At that point, we compute the cumulative discounted return v_n from the observed r values. The algorithm terminates when it attempts to execute the $next$ action on a sequence that has been fully consumed. At that point, let N be the number of trajectories that we have generated. We can obtain a confidence interval on $V^\pi(s_0)$ by computing the mean and variance of the returns $v_1, \ldots, v_N$ and evaluating the one-sided empirical Bernstein bound.

Theorem 7 *Given an MDP, a policy π, a horizon H, a confidence parameter $0 < \delta < 1/2$, and a collection of observed (s, a, s', r) transitions that have been recorded in the SEQ data structures, Algorithm 9 terminates and returns a lower confidence bound $\underline{V}(s_0)$ such that with probability $1 - \delta$*

$$\underline{V}(s_0) \leq V^\pi(s_0).$$

Proof 9 *Because in each (s, a) we observe the same state transitions in the same order as they were originally collected, we obtain the same confidence interval that we would have obtained if we had been executing policy π from the beginning.*

The final part of LLGCV that we need to describe is the ALLOCATESAMPLES method (see Algorithm 10). This populates the N_{new} array with the number of samples that should be drawn from each state. The goal is to allocate a minibatch of new samples to states so that *after* drawing the new samples, a total budget of B samples will have been drawn in proportion to the occupancy measure μ that is provided. Because some states may already have more samples than this, we can treat those "extra" samples as excess and add them to B.

As LLGCV runs, it computes many confidence intervals. To ensure that all of those intervals are simultaneously valid with probability $1 - \delta$, we need to partition δ into smaller individual δ' values when computing each interval. To do this, we use the fact that

$$\sum_{t=1}^{\infty} \frac{1}{t(t+1)} = 1$$

to create a sequence of δ' values that sum to δ. The variable t indexes the iterations of the

Algorithm 9 ETPE (π, δ, H)

parameters:

1: δ confidence parameter
2: π policy to evaluate
3: M MDP with starting state s_0, discount factor γ, and maximum value V_{max}.
4: H horizon depth (trajectory length)
5: SEQ collection of sequence data structures one for each (s, a)
6: For all s and a in M, $SEQ(s, a).reset()$
7: $J := \emptyset$ is the set of generated trajectories
8: **loop**
9: $s := s_0, h := 0, v := 0$
10: **while** $h < H$ **do**
11: **if** $SEQ(s, \pi(s)).end()$ **then**
12: goto FINISH
13: **end if**
14: $(s', r) := SEQ(s, \pi(s)).next()$
15: $s := s'; \quad v := v + \gamma^h r$
16: $h := h + 1$
17: **end while**
18: $J := J \cup \{v\}$
19: **end loop**
20: FINISH: $N := |J|$
21: $\hat{V} := \frac{1}{N} \sum_{j=1}^{N} v_j$
22: $\widehat{Var} := \frac{1}{J} \sum_{j=1}^{J} (v_j - \hat{V})^2$
23: halfwidth := $\sqrt{\frac{2\widehat{Var} \ln 3/\delta}{N}} + \frac{3V_{max} \ln 3/\delta}{N}$
24: **return** $\hat{V}$ − halfwidth

main loop of LLGCV (line 14). Hence, in each main loop iteration, we have a total of $\frac{\delta}{t(t+1)}$ confidence probability to consume.

We divide δ equally between computing local and global bounds. For computing the lower global bound in iteration t, we only need to compute one confidence bound, so we set

$$\delta' = \frac{\delta}{2t(t+1)}.$$

For computing the upper and lower local bounds in iteration t, we need to compute $2K_t|A|$ confidence bounds, where K_t is the number of states that participate in the extended value iteration

Algorithm 10 ALLOCATESAMPLES (μ, π, B)

1: $N_{new}(s) := 0$ for all s
2: muTotal := $\sum_s \mu(s)$
3: toAllocate := B // Total number of samples to allocate
4: $S' = \{s | s \in S,\ \mu(s) > 0\}$
5: $S_{excess} = \emptyset$
6: **repeat**
7: continue := false
8: **for** s in S' **do**
9: share := Round(toAllocate $\times \mu(s)$/muTotal)
10: **if** $(N(s, \pi(s)) >$ share) **then**
11: toAllocate $-= N(s, \pi(s))$
12: $S_{excess} := S_{excess} \cup \{s\}$
13: continue := true
14: **end if**
15: **end for**
16: $S' := S' \setminus S_{excess}$
17: muTotal := $\sum_{s \in S'} \mu(s)$
18: **until** (continue = false)
{Allocate the samples}
19: **for** s in S' **do**
20: share := Round(toAllocate $\times \mu(s)$/muTotal)
21: $N_{new}(s) :=$ share $- N(s, \pi(s))$
22: **end for**

for minibatch t (line 15 of LLGCV). Hence, when computing each bound, we set

$$\delta' = \frac{\delta}{4t(t+1)K_t|A|}.$$

Note that although extended value iteration computes multiple confidence intervals for each state-action pair, the confidence intervals computed prior to convergence do not need to be accounted for. The key is that the final confidence intervals that solve the Bellman equation and hence support π^{UCB} must be valid.

The following theorem formalizes the correctness of LLGCV. The proof is in Appendix A.

Theorem 8 (LLGCV is PAC-RL) *There exists a sample size m polynomial in $|S|$, $|A|$, $1/\epsilon$, $1/\delta$, $1/(1-\gamma)$, R_{max}, such that LLGCV $(s_0, \pi_0, F, \epsilon, \delta, \gamma, R_{max})$ terminates after no more than*

$m|S||A|$ calls on the simulator and returns a policy π such that $|V^{\pi}(s_0) - V^{*}(s_0)| < \epsilon$ with probability $1 - \delta$.

We now describe how LGCV differs from LLGCV. The only change is that LGCV does not compute the local lower bounds in line 15 of Algorithm LLGCV and it does not include those bounds in the maximization in line 18. This also means that during each call to ExtendedValueIteration, we can set δ' according to

$$\delta' = \frac{\delta}{2t(t+1)K_t|A|},$$

because we are computing half as many confidence bounds.

3.5 Experimental Evaluation

To evaluate the effectiveness of LGCV and LLGCV, we compared their performance to two other algorithms. The first is the DDV algorithm [12, 66] but improved to use the empirical Bernstein bound and Theorem 6 instead of the multinomial bound. The second is Fiechter's original MDP planning algorithm [22]. We compare the performance on the same MDP benchmarks as in Section 3.3. In these experiments, $\gamma = 0.9$, $\delta = 0.01$, and $\epsilon = 0.1$. The algorithms terminate either if the width of the confidence interval falls below ϵR_{max} or if 10 million samples are drawn. We report the number of samples drawn at termination. The results are averaged over 15 independent runs and plotted in Figure 3.4(a).

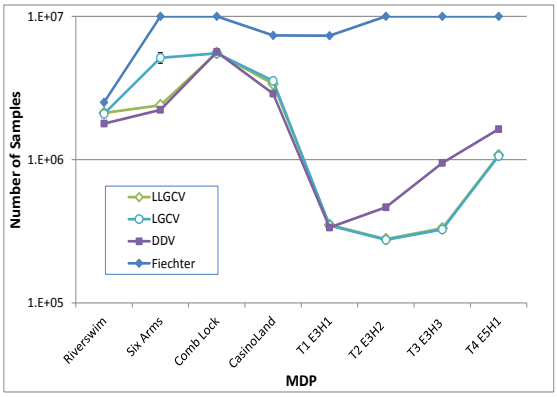
Figure 3.4(a) shows that Fiechter's algorithm gives the worst performance on all problems and LLGCV matches or exceeds the performance of the other algorithms. The performance of LGCV is often identical to LLGCV, except on Six Arms, where it is much worse. DDV gives the best performance, by slight margins, on RiverSwim, Six Arms, and CasinoLand. It is particularly notable that both LGCV and LLGCV give major improvements over DDV on three Tamarisk problem instances.

Figure 3.4(b) presents a more detailed comparison of LLGCV and DDV. Points above the diagonal line correspond to MDPs where LLGCV requires fewer samples than DDV. This shows again that when LLGCV is better than DDV, the improvement is huge, whereas when DDV is better than LLGCV, the difference is small.

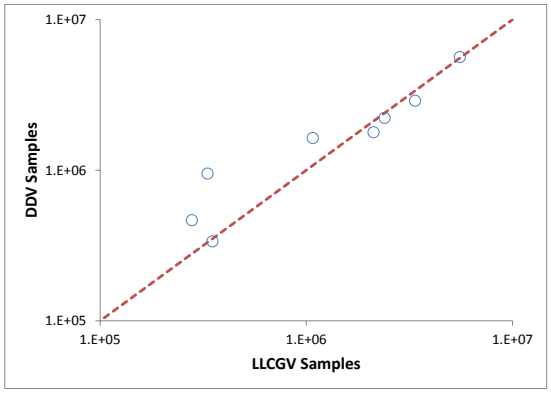
Table 3.7 reports the upper and lower confidence bounds computed by each of the algorithms on the four benchmark MDPs. The table also includes the value of the optimal policy for these

problems. In all cases, the confidence intervals include the true value of $V^*(s_0)$.

Table 3.8 reports the upper and lower confidence bounds for the four Tamarisk instances. Notice that the lower bounds computed by LGCV and LLGCV are almost always higher (tighter) than the lower bounds computed by DDV and Fiechter. This reflects the "two-sided" exploration strategy of LLGCV. Higher lower bounds allows higher upper bounds to achieve the same final confidence level width. This allows LGCV and LLGCV to terminate sooner.



(a) Comparison of the number of samples to termination (using 8 MDPs). The notation Tn EeHh denotes Tamarisk MDP n, with $E = e$ edges and $H = h$ slots. Solid markers correspond to local methods methods; open markers correspond to LLGCV methods.



(b) Comparison of number of samples computed by the best local and the best LLGCV heuristics. Each circle corresponds to one of the 8 MDPs. All points are above the line, which corresponds to cases where the LLGCV method is better.

Figure 3.4: Comparison of number of samples taken by each algorithm to reach to the termination point. Each circle corresponds to one of the 8 MDPs.

3.6 Concluding Remarks

This chapter has made three main contributions. First, it has developed optimal sampling strategies for Monte Carlo policy evaluation using both local and trajectory-wise forms of the Hoeffding and empirical Bernstein bounds. Second, it provided experimental evidence that for policy evaluation the trajectory-wise bounds generally out-perform the local bounds except in a few special cases. Third, it introduced two new MDP planning algorithms, LGCV and LLGCV, for simulator-defined MDPs. These algorithms combine a trajectory-wise confidence intervals with local confidence intervals to reduce the number of samples needed to achieve target levels

	LLGCV	LGCV	DDV	Fiechter	$V^*(s_0)$
RiverSwim$\overline{V}(s_0)$	0.271 ± 0.001	0.271 ± 0.001	0.272 ± 0.002	0.271 ± 0.001	0.221
RiverSwim$\underline{V}(s_0)$	0.171 ± 0.001	0.171 ± 0.001	0.172 ± 0.002	0.171 ± 0.001	
SixArms$\overline{V}(s_0)$	0.876 ± 0.007	0.859 ± 0.004	0.874 ± 0.006	0.910 ± 0.011	0.826
SixArms$\underline{V}(s_0)$	0.776 ± 0.007	0.760 ± 0.004	0.774 ± 0.005	0.732 ± 0.011	
CasinoLand$\overline{V}(s_0)$	0.576 ± 0.002	0.575 ± 0.002	0.566 ± 0.001	0.586 ± 0.001	0.516
CasinoLand$\underline{V}(s_0)$	0.476 ± 0.002	0.475 ± 0.002	0.466 ± 0.001	0.447 ± 0.001	
CombinationLock$\overline{V}(s_0)$	0.200 ± 0.001	0.193 ± 0.001	0.200 ± 0.001	0.200 ± 0.001	0.150
CombinationLock$\underline{V}(s_0)$	0.100 ± 0.001	0.093 ± 0.001	0.100 ± 0.001	0.100 ± 0.001	

Table 3.7: Upper ($\overline{V}(s_0)$) and lower ($\underline{V}(s_0)$) confidence bounds at termination for four RL benchmarks

	LLGCV	LGCV	DDV	Fiechter
E=3 and H=1 $\overline{V}(s_0)$	27.865 ± 0.055	27.842 ± 0.044	27.334 ± 0.030	27.737 ± 0.015
E=3 and H=1 $\underline{V}(s_0)$	24.371 ± 0.054	24.345 ± 0.045	23.835 ± 0.031	24.238 ± 0.015
E=3 and H=2 $\overline{V}(s_0)$	37.476 ± 0.021	37.464 ± 0.028	36.825 ± 0.006	38.000 ± 0.001
E=3 and H=2 $\underline{V}(s_0)$	33.711 ± 0.040	33.687 ± 0.030	33.027 ± 0.006	30.393 ± 0.095
E=3 and H=3 $\overline{V}(s_0)$	40.669 ± 0.027	40.659 ± 0.025	39.721 ± 0.008	41.000 ± 0.001
E=3 and H=3 $\underline{V}(s_0)$	36.605 ± 0.041	36.632 ± 0.063	35.622 ± 0.008	26.842 ± 0.151
E=5 and H=1 $\overline{V}(s_0)$	49.198 ± 0.075	49.159 ± 0.056	47.319 ± 0.018	50.799 ± 0.055
E=5 and H=1 $\underline{V}(s_0)$	43.500 ± 0.073	43.467 ± 0.049	41.620 ± 0.018	34.299 ± 0.077

Table 3.8: Upper ($\overline{V}(s_0)$) and lower ($\underline{V}(s_0)$) confidence bounds at termination for four configurations of the tamarisk domain

	LLGCV	LGCV	DDV	Fiechter
RiverSwim	2.120 ± 0.022	2.090 ± 0.026	1.780 ± 0.003	2.510 ± 0.004
SixArms	2.400 ± 0.019	5.140 ± 0.433	2.220 ± 0.004	10.000 ± 0.000
CasinoLand	5.570 ± 0.089	5.520 ± 0.073	5.640 ± 0.002	10.000 ± 0.000
CombinationLock	3.370 ± 0.005	3.550 ± 0.005	2.890 ± 0.001	7.360 ± 0.009
E=3 and H=1	0.351 ± 0.007	0.347 ± 0.005	0.336 ± 0.005	7.350 ± 0.042
E=3 and H=2	0.279 ± 0.005	0.276 ± 0.006	0.465 ± 0.005	10.000 ± 0.000
E=3 and H=3	0.332 ± 0.010	0.326 ± 0.007	0.949 ± 0.010	10.000 ± 0.000
E=5 and H=1	1.080 ± 0.022	1.060 ± 0.017	1.630 ± 0.005	10.000 ± 0.000

Table 3.9: Number of samples for 8 benchmark MDPs ($\times 10^6$)

of accuracy. The LGCV algorithm combines a local upper bound with a global lower bound. Although this performs well on some problems, it exhibits poor performance on others. The LLGCV algorithm extends LGCV by intersecting both the local and global lower bounds. Although this requires computing additional confidence intervals, it eliminates LGCV's failure cases. Hence, LLGCV provides a robust method that can provide significant improvements over previous methods.

Efficient Exploration for Constrained MDPs

Chapter 4: Efficient Exploration for Constrained MDPs

Given a Markov Decision Process (MDP) defined by a simulator, a designated starting state s_0, and a downside risk constraint defined as the probability of reaching catastrophic states, our goal is to find a stationary deterministic policy π that with probability $1 - \delta$ achieves a value $V^{\pi}(s_0)$ that is within ϵ of the value of the optimal stationary deterministic ν-feasible policy, $V^{*}(s_0)$, while economizing on the number of calls to the simulator. This chapter presents the first **PAC-Safe-RL** algorithm for this purpose. The algorithm extends PAC-RL algorithms for efficient exploration while providing guarantees that the downside constraint is satisfied. Experiments comparing our CONSTRAINEDDDV algorithm to baselines show substantial reductions in the number of simulator calls required to find a feasible policy.

4.1 Introduction

This work is inspired by problems in natural resource management centered on the challenge of invasive species [12, 66]. Computing optimal management policies for ecosystems is challenging because they exhibit complex spatio-temporal interactions at multiple scales. Many ecosystem management problems can be formulated as MDP (Markov Decision Process) planning problems [59]. In a simulator-defined MDP, the Markovian dynamics and rewards are provided by a simulator from which samples can be drawn. Simulators in natural resource management can be very expensive to execute, so that the time required to solve such MDPs is dominated by the number of calls to the simulator.

Efficient MDP planning algorithms attempt to minimize the number of simulator calls before terminating and outputting a policy that is approximately optimal with high probability. For unconstrained MDPs, the standard formulation of this is the notion of PAC-RL [22, 12]. This is in contrast to the PAC-MDP formalization, which minimizes various measures of infinite-horizon regret [62]. A common component of PAC-RL algorithms is to compute confidence intervals and explore using the optimism principle.

In many practical scenarios, such as natural resource management, a desirable policy needs to satisfy certain constraints imposed by decision makers. In these scenarios, maximizing the

expected reward does not necessarily avoid rare catastrophic or dangerous situations. For example, in conservation problems, catastrophic outcomes include species extinction, long-term establishment of an invasive species, and severe wildfires. A standard approach to finding policies that avoid catastrophic states is to assign a large negative reward to those states [26, 27]. This is equivalent to a so-called Big M method for establishing a lexicographic preference for policies that do not enter catastrophic states. However, this approach does not quantify the risk (probability) of entering a catastrophic state, nor does it determine whether there are policies that control this risk. A better approach is to adopt the Constrained MDP (C-MDP) formalism [1], which seeks to maximize one objective (e.g., economic value) while satisfying one or more constraints probabilistically. For example, in invasive species management, we can define a C-MDP to minimize the economic cost of invasive species management while ensuring that the probability of native species extinction is less than a specified threshold.

Recently, Geibel and Wysotzki [27] developed a model-free Q-learning algorithm for C-MDPs. Their formulation is applicable to episodic tasks with a combination of absorbing catastrophic and goal states. As Geramifard [28] pointed out, the Geibel, et al., work does not provide a performance guarantee on the result.

An alternative to constrained MDPs is to consider risk-sensitive objectives such as variance penalties, value at risk (VaR), and conditional value at risk (CVaR) [26, 1]. Var and CVar optimize the α-quantile of the expected return, and CVaR has favorable mathematical properties. While these are all very interesting approaches, we find the constrained MDP formulation easier to understand and explain to stakeholders, and for this reason, we focus our efforts on C-MDPs.

A drawback of C-MDPs is that the optimal policy can be stochastic in some cases. Specifically, if there are c constraints, then the optimal policy may be stochastic in up to c states. From the perspective of our stakeholders, this stochastic behavior is confusing and undesirable. Hence, in this chapter, we aim to find a stationary deterministic policy that satisfies a downside risk constraint as well as maximizing the discounted reward. We seek to do this while economizing on the number of calls to the simulator and while providing PAC guarantees both that the constraints are satisfied and that the resulting policy is within a fixed bound of optimality. This provides the first PAC-RL algorithm for deterministic policies in C-MDPs.

The chapter is organized as follows. Section 2 introduces our notation for MDPs, C-MDPs, and confidence intervals. Section 3 introduces our new planning algorithm CONSTRAINED-DDV. Section 4 presents an experimental evaluation of CONSTRAINEDDDV and a comparison with other methods. Section 5 concludes the chapter. We evaluate our algorithms on an invasive

species problem as well as on standard reinforcement learning benchmarks.

4.2 Problem Definition and Notation

Let a simulator-defined MDP consist of a start state s_0, a set of possible states S, a set of possible actions A, a discount factor $\gamma \in (0, 1]$ and a stochastic function F that maps from an input state-action pair (s, a) to a resulting state s' and reward r, where $s' \sim P(s'|s, a)$ is sampled according to the (unknown) transition function, $r \sim R(r|s, a)$ is sampled according to the unknown reward function, and $0 \leq r \leq R_{max}$. In this chapter, we will assume that the reward is deterministic; our methods can be easily extended to handle stochastic rewards. A (deterministic) policy π is a function mapping from states s to actions $a = \pi(s)$. The value of the policy in the start state, $V^\pi(s_0)$, is the expected discounted cumulative reward:

$$V^\pi(s_0) = E\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s = s_0\right].$$

Let $V_{max} = \frac{R_{max}}{1-\gamma}$ be the maximum possible value of any state under any policy. The corresponding minimum possible value is zero.

An optimal policy π^* maximizes $V^\pi(s_0)$, and the corresponding value is denoted by $V^*(s_0)$. The action-value of state s and action a under policy π is defined as $Q^\pi(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a)V^\pi(s')$. The optimal action-value is denoted $Q^*(s, a)$. Later, we indicate these functions with subscript R to distinguish them from the catastrophe value function.

Definition 8 *The occupancy measure μ of an MDP under policy π is defined as*

$$\mu^\pi(s) = \mathbb{E}_P\left[\sum_{t=0}^{\infty} \gamma^t I[s_t = s]|s_0, \pi\right],$$

where $I[\cdot]$ is the indicator function and the expectation is taken with respect to the transition distribution.

This is the cumulative discounted probability that the MDP will occupy state s under policy π for discount factor γ. It can be computed via dynamic programming on the Bellman flow equation [64]:

$$\mu^\pi(s) = I[s = s_0] + \gamma \sum_{s^-} \mu(s^-)P(s|s^-, \pi(s^-)). \tag{4.1}$$

This says that the discounted probability of visiting state s is equal to the sum of the probability that s is the starting state and the probability of reaching s by first visiting state s^- and then executing an action that leads to state s.

It is easy to show that

$$V^\pi(s_0) = \sum_s \mu^\pi(s) R(s, \pi(s)). \tag{4.2}$$

We adopt $\mu^{\pi^{UCB}}$ (also written as μ^{UCB}) as the occupancy measure computed based on the principle of optimism under uncertainty and maximum likelihood estimates of transition probabilities.

Let a subset of states $S_C \subset S$ be "catastrophic" states in the sense that we want to limit the probability of entering those states. Let us assume that all states in S_C are absorbing.

Definition 9 *For a policy π, the risk in state s is defined as*

$$\xi^\pi(s) = \sum_t \gamma_C^t P(s_t \in S_C | s, \pi), \tag{4.3}$$

which is the (discounted) probability of entering a catastrophic state when following π. γ_C denotes the catastrophe discount factor.

As a learning algorithm explores the MDP, it collects the following statistics. Let $N(s, a)$ be the number of times state-action pair (s, a) is simulated during learning and $N(s) = \sum_a N(s, a)$. Let $N(s, a, s')$ be the corresponding number of times that s' has been observed as the resulting state. Let $R(s, a)$ be the observed reward. Let $\hat{P}(s'|s, a) = N(s, a, s')/N(s, a)$ be the maximum likelihood estimate for $P(s'|s, a)$.

A $1 - \delta$ confidence interval is a pair of random variables $\underline{V}(s_0), \overline{V}(s_0)$ such that with probability $1 - \delta$, $\underline{V}(s_0) \leq V^\pi(s_0) \leq \overline{V}(s_0)$. Similarly, $\underline{Q}(s, a)$ and $\overline{Q}(s, a)$ denote the confidence bounds over the action-value functions. We follow the "Optimism Under Uncertainty" principle, and denote by π^{UCB} the policy based on an upper confidence bound on the action-value function, $\pi^{UCB}(s) = \mathrm{argmax}_a \, \overline{Q}(s, a)$.

Definition 10 *[22]. A learning algorithm is PAC-RL if for any discounted MDP $(S, A, P, R, \gamma, P_0)$, $\epsilon > 0$, $1 > \delta > 0$, and $0 \leq \gamma < 1$, the algorithm halts and outputs a policy π such that*

$$\mathbb{P}[|V^*(s_0) - V^\pi(s_0)| \leq \epsilon] \geq 1 - \delta,$$

in time polynomial in $|S|$, $|A|$, $1/\epsilon$, $1/\delta$, $1/(1-\gamma)$, *and* R_{max}.

4.2.1 Extended Value Iteration

Classical value iteration computes an optimal policy for a fixed MDP. Extended value iteration can compute optimal policy for finite-sampled optimistic/pessimistic MDPs by defining confidence intervals on the value function at each state of the MDP based on samples from that MDP. Different confidence interval methods (e.g., Hoeffding bound [32], empirical Bernstein bound [2], multinomial confidence interval [73], etc.) at each state lead to different confidence intervals throughout the MDP. One can obtain robust policies from pessimistic MDPs [67]. Based on the experiments in Chapter 3, the empirical Bernstein bound is the tightest bound compared to the other bounds.

The Empirical Bernstein Method: This approach uses the empirical Bernstein bound. Let $M(s,a)$ denote the sample mean of the discounted backed-up values from the successor states that result from taking action a in state s, and $Var(s,a)$ denote the sample variance of these values. We denote the upper and lower bounds on these values as $\overline{M}(s,a)$, $\underline{M}(s,a)$,$\overline{Var}(s)$, and $\underline{Var}(s)$.

$$\begin{aligned}\overline{M}(s,a) &= \sum_{s'} \hat{P}(s'|s,a)\gamma\overline{V}(s') \\ \overline{Var}(s,a) &= \sum_{s'} \hat{P}(s'|s,a)[\gamma\overline{V}(s') - \overline{M}(s,a)]^2\end{aligned}$$

$$\overline{V}(s) = \max_a R(s,a) + \overline{M}(s,a) + \sqrt{\frac{2\overline{Var}(s)\ln(3/\delta_0)}{N(s,a)}} + \frac{3\gamma V_{max}\ln(3/\delta_0)}{N(s,a)} \tag{4.4}$$

$$\begin{aligned}\underline{M}(s,a) &= \sum_{s'} \hat{P}(s'|s,a)\gamma\underline{V}(s') \\ \underline{Var}(s) &= \sum_{s'} \hat{P}(s'|s,a)[\gamma\underline{V}(s') - \underline{M}(s,a)]^2\end{aligned}$$

$$\underline{V}(s) = \max_a R(s,a) + \underline{M}(s,a) \; - \; \sqrt{\frac{2\underline{Var}(s,a)\ln(3/\delta_0)}{N(s,a)}} - \frac{3\gamma V_{max}\ln(3/\delta_0)}{N(s,a)} \tag{4.5}$$

We need to define δ_0 so that the confidence intervals hold simultaneously with probability $1-\delta$. These equations can be iterated to convergence. At convergence, with probability $1-\delta$, $\underline{V}(s_0) \leq V^*(s_0) \leq \overline{V}(s_0)$.

4.2.2 Optimal Policies for C-MDPs

Before delving into additional definitions for C-MDPs, let's clarify the class of optimal policies for C-MDPs. It has been shown that, unlike unconstrained MDPs, the optimal policies in C-MDPs are not necessarily stationary and deterministic and may depend on the starting state [21, 74]. In standard discounted unconstrained MDPs, one can find optimal policies that are stationary and deterministic from any state in $O\left(|S|^2|A|\right)$. In a C-MDP with two objectives (the standard value function and the risk of catastrophe), if the two objectives have unequal discount factors, then finding deterministic and stationary policies is NP-complete [14, 18, 8]. Optimal policies in C-MDPs with equal discount factors are randomized and stationary for a fixed starting state. The solution can be found by solving a linear program, where the dual variables represent the state occupancy measure, if the model is known. In our case where we only have one constraint, the optimal randomized policy is called a "1-randomized" policy [74]. This means the difference between deterministic and the 1-randomized policy will arise in at most one state, where the randomized policy may choose probabilistically between two actions [20].

In this chapter, we focus on finding a best policy in the class of stationary and deterministic policies with performance guarantees, even when a randomized policy is the optimal policy. It is a challenge to present a randomized policy to stakeholders. Feinberg [19] points out that implementation of randomized policies is not natural in many applications, and the use of randomization procedures could increase the variance of the expected return. Boutilier and Lu [5] also give an example of how randomized policy could be undesirable.

4.2.3 Additional Definitions for C-MDPs

Let Π be the space of deterministic polices over the constrained MDP $\mathcal{M}(\tau) = \langle S, A, P, R_R, R_C, \tau, \gamma, s_0 \rangle$. Every policy π induces two value functions V_R^π and V_C^π. We will say two policies π_1 and π_2 are equivalent if $V_R^{\pi_1} = V_R^{\pi_2}$ and $V_C^{\pi_1} = V_C^{\pi_2}$ over all states $s \in S$. Let $\overline{\pi}$ denote the set of policies equivalent to π. Let $\overline{\pi}_1$ and $\overline{\pi}_2$ be two distinct equivalence classes of policies. We will say that $\overline{\pi}_1$ dominates $\overline{\pi}_2$ if $V_R^{\overline{\pi}_1}(s_0) \geq V_R^{\overline{\pi}_2}(s_0)$ and $V_C^{\overline{\pi}_1} \leq V_C^{\overline{\pi}_2}$. That is, $\overline{\pi}_1$ is superior in either R_R or R_C or both. An equivalence class is non-dominated if there does not exist an equivalence class that dominates it.

Let $\Pi(\tau)$ be the space of deterministic policies such that $\forall \pi \in \Pi(\tau), V_C^\pi(s_0) \leq \tau$. These are the feasible deterministic policies. An optimal feasible deterministic policy $\pi_\tau^* \in \Pi(\tau)$ satisfies

$$V_R^{\pi_\tau^*}(s_0) \geq V_R^\pi(s_0) \; \forall \pi \in \Pi(\tau).$$

Values are defined in the usual way as the expected cumulative discounted return:

$$V_C(s_0) = \mathbb{E}[R_C(s_0, \pi(s_0)) + \gamma R_C(s_1, \pi(s_1)) + \ldots + \gamma^t R_C(s_t, \pi(s_t)) + \ldots],$$

and

$$V_R(s_0) = \mathbb{E}[R_R(s_0, \pi(s_0)) + \gamma R_R(s_1, \pi(s_1)) + \ldots + \gamma^t R_R(s_t, \pi(s_t)) + \ldots].$$

An optimal feasible policy π_τ^* is not necessarily non-dominated. There might be another policy π' that achieves the same $V_R(s_0)$ but has larger $V_C^{\pi'}(s_0) > V_C^{\pi_\tau^*}(s_0)$ that is still feasible.

Define the Lagrangian MDP $\mathcal{L}(\lambda) = \langle S, A, P, \lambda R_R - (1-\lambda) R_C, \gamma, s_0 \rangle$ whose reward function is a linear combination of R_R and R_C.

Claim 1 *For any fixed λ, the optimal policy π_λ^* for $\mathcal{L}(\lambda)$ is a non-dominated policy.*

Proof 10 *The argument is by contradiction. If there were a way to increase V_R or reduce V_C, then π_λ^* would not be optimal for the Lagrangian MDP.*

Definition 11 *Let Π_L be the set of all stationary deterministic policies that are solutions to the Lagrangian MDP for some value of λ.*

Claim 2 *Let λ_1 and λ_2 be a pair of values such that $\lambda_2 = \lambda_1 - \delta$ for some positive δ. Let π_1 be a policy that optimizes the Lagrangian for $\lambda = \lambda_1$ and π_2 be the policy that optimizes the Lagrangian for $\lambda = \lambda_2$. Then one of two cases holds:*

Case 1: $V_C^{\pi_2}(s_0) = V_C^{\pi_1}(s_0)$, *and* $V_R^{\pi_2}(s_0) = V_R^{\pi_1}(s_0)$ *or*

Case 2: $\pi_1 \neq \pi_2$, $V_C^{\pi_2}(s_0) < V_C^{\pi_1}(s_0)$, *and* $V_R^{\pi_2}(s_0) < V_R^{\pi_1}(s_0)$.

Proof 11 *Each solution is non-dominated, which means that both V_C and V_R must change because otherwise, the non-dominated condition would be violated. Because λ has decreased, less weight is placed on V_R and more weight on V_C. Hence V_C must decrease, which means that V_R must also decrease.*

Claim 3 *There exists a value λ^* such that $\forall \lambda \leq \lambda^*$, the optimal policy, π^*_λ, of the Lagrangian MDP $\mathcal{L}(\lambda)$ is feasible for $\mathcal{M}(\tau)$; that is $V_C^{\pi^*_\lambda}(s_0) \leq \tau$.*

Proof 12 *As λ decreases, V_C decreases. Hence, at some point, $\lambda = \lambda^*$, $V_C(s_0) \leq \tau$, and hence the optimal policies for the Lagrangian become feasible for the constrained MDP. Let λ^* be the largest value of λ for which $V_C^{\pi^*_\lambda}(s_0) \leq \tau$.*

For computational efficiency, we will not consider all possible values of λ. Instead, we discretize the space by introducing a precision parameter η. Define $\Pi_{\mathcal{L},\eta}$ to be the class of all policies in $\Pi_{\mathcal{L}}$ where $\lambda = k\eta$, for $k \in \{0, 1, \ldots, 1/\eta\}$. We will restrict our attention to only these policies.

4.3 PAC-RL for Constrained MDPs

We now consider the problem of finding an approximately optimal policy by sampling from a simulator-defined Constrained MDP. We introduce the following parameters:

- τ defines the feasibility constraint. A policy π is feasible if $V_C^\pi(s_0) \leq \tau$.
- ϵ defines a tolerance on the optimality of $V_R^\pi(s_0)$.
- ν defines a tolerance on feasibility. We will accept any policy for which $|V_C^\pi(s_0) - V_C^*(s_0)| \leq \nu$, which means that in the worst case, $V_C^\pi(s_0) = \tau + \nu$.

- η controls the numerical precision of the λ values.
- δ is the confidence parameter.

Definition 12 *[8]. A deterministic policy π is called ν-feasible if $V_C^\pi(s_0) \leq \tau + \nu$ for $\nu \geq 0$.*

To obtain a polynomial time sampling algorithm, we need to relax our goal (based on ideas from Chang [8]). Let $\Pi_{\mathcal{L},\eta}(\tau)$ be the set of all policies $\pi \in \Pi_{\mathcal{L},\eta}$ such that $V_C^\pi(s_0) \leq \tau$. These are the τ-feasible policies. We will be interested in two other policy classes: $\Pi_{\mathcal{L},\eta}(\tau - \nu)$ and $\Pi_{\mathcal{L},\eta}(\tau + \nu)$.

Let $\pi^{*(-\nu)} \in \Pi_{\mathcal{L},\eta}(\tau - \nu)$ be a policy that is feasible with respect to the threshold $\tau - \nu$ and that among all such policies maximizes $V_R(s_0)$. More precisely, $\pi^{*(-\nu)} = \text{argmax}_{\pi \in \Pi_{\mathcal{L},\eta}(\tau-\nu)} V_R^\pi(s_0)$.

Denote the value of $\pi^{*(-\nu)}$ by $V_R^{*(-\nu)}(s_0)$. Our goal will be to output a policy $\pi \in \Pi_{\mathcal{L},\eta}(\tau + \nu)$ such that $V_R^{*(-\nu)}(s_0) - V_R^\pi(s_0) \leq \epsilon$ and to do so in polynomial time.

Definition 13 *An algorithm is Lagrangian-PAC-SAFE-RL if, for any C-MDP $M(\tau) = \langle S, A, P, R_R, R_C, \tau, \gamma, s_0 \rangle$ and any parameters $\epsilon > 0, \delta \in (0,1), \tau \in (0,1], \eta > 0$, and $\nu > 0$ the algorithm halts in time polynomial in $|S|, |A|, 1/(1-\gamma), 1/\epsilon, 1/\nu, 1/\delta$, and $1/\eta$ and does one of the following two things:*

1. *Outputs a policy $\pi \in \Pi_{\mathcal{L},\eta}$ such that with probability $1-\delta$ the following are simultaneously true:*
   - *(a) $V_C^\pi(s_0) < \tau + \nu$ (π is $\tau + \nu$ feasible)*
   - *(b) $V_R^{*(-\nu)}(s_0) - V_R^\pi(s_0) \leq \epsilon$ (the value of π is never less than ϵ below the value of the optimal $\tau - \nu$ feasible policy, and it may be significantly higher)*
2. *Outputs the message Fail, in which case with probability $1 - \delta$ there does not exist any policy $\pi \in \Pi_{\mathcal{L},\eta}$ such that $V_C^\pi(s_0) \leq \tau + \nu$.*

This definition gives us control over how close to feasible the policy is (via ν) and how close to the optimal feasible policy its V_R return is (via ϵ).

4.3.1 Confidence intervals for V_R and V_C for policy evaluation

Suppose we have drawn a set of samples for various states and actions. For any fixed policy π, we can perform extended policy evaluation (i.e., extended value iteration with a fixed policy) to obtain lower and upper confidence bounds on $V_C(s_0)$ and $V_R(s_0)$. We will denote these as $\underline{V}_C^\pi(s_0)$, $\overline{V}_C^\pi(s_0)$, $\underline{V}_R^\pi(s_0)$, and $\overline{V}_R^\pi(s_0)$. Suppose our goal is to determine whether π is feasible and if it is, then to determine confidence intervals on $V_R^\pi(s_0)$. The policy π will be feasible with probability $1-\delta$ if $\overline{V}_C^\pi(s_0) \leq \tau$. Conversely, π is not feasible with probability $1-\delta$ if $\underline{V}_C^\pi(s_0) > \tau$.

4.3.2 Confidence intervals for V_R and V_C for policy optimization

Instead of using a fixed policy, we can set a value of λ and perform extended value iteration based on the upper confidence bound of the Lagrangian objective. This will define the $\pi^{UCB(\lambda)}$ policy. More generally, we can perform binary search on λ to find three values:

- λ_{lower} is the largest value of $\lambda \in \Lambda$ such that $\overline{V}_C^{UCB(\lambda)}(s_0) \leq \tau$. This means that given our current sample, $\pi^{UCB(\lambda)}$ is the "best" policy (in the sense of having the largest λ) for which we can guarantee with probability $1-\delta$ that it is feasible.

- λ_{upper} is the largest value of $\lambda \in \Lambda$ such that $\underline{V}_C^{UCB(\lambda)}(s_0) \leq \tau$. This means that given our current sample, this is the largest value of λ that we cannot prove is not feasible. Graphically, the situation looks like the plot in figure 4.1, which shows V_R (blue) and V_C (green) as a function of λ.

The solid lines denote the true values of V_C and V_R. The dashed lines denote the corresponding upper and lower confidence bounds. For purposes of this section, let π^* be the policy in $\Pi(\tau, \eta)$ that maximizes $V_R^\pi(s_0)$. That is, π^* is τ-feasible and among all such policies it maximizes the V_R return.

Claim 4 *The optimal value $\lambda^* \in [\lambda_{lower}, \lambda_{upper}]$ with probability $1-\delta$.*

Proof 13 *The optimal value of $\lambda^* \geq \lambda_{lower}$ because we know all policies π^λ for $\lambda < \lambda_{lower}$ are τ-feasible and $V_R^\lambda(s_0)$ increases monotonically with λ. Therefore, there is no reason to consider $\lambda < \lambda_{lower}$. The optimal value of $\lambda^* \leq \lambda_{upper}$ because we know that λ_{upper} is the largest value of λ that could possibly make π^λ τ-feasible.*

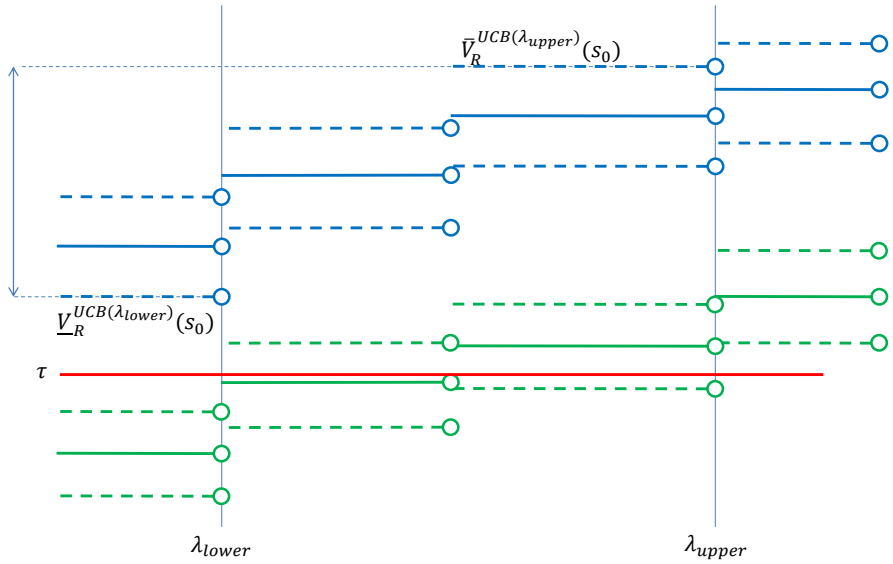


Figure 4.1: Graphical representation of confidence intervals for different value of λ. The horizontal axis plots the value of $0 \leq \lambda \leq 1$. The solid lines denote the true values of V_C and V_R. The dashed lines denote the corresponding upper and lower confidence bounds.

Claim 5 $\underline{V}_R^{UCB(\lambda_{lower})}(s_0) \leq V_R^*(s_0) \leq \overline{V}_R^{UCB(\lambda_{upper})}(s_0)$ *with probability* $1-\delta$

Proof 14 *From Claim 4, we know that the optimal value of* λ, λ^* *is in* $[\lambda_{lower}, \lambda_{upper}]$. *Therefore,* $V_R^{\lambda_{lower}}(s_0) \leq V_R^*(s_0) \leq V_R^{\lambda_{upper}}(s_0)$. *The value* $\overline{V}_R^{UCB(\lambda_{upper})}(s_0) \geq V_R^{\lambda_{upper}}(s_0)$ *because at each state s, it overestimates* $V_R^{\lambda_{upper}}(s)$. *The value* $\underline{V}_R^{UCB(\lambda_{lower})}(s_0) \leq V_R^{\lambda_{lower}}(s_0)$, *because the value of any policy, such as* $\pi^{UCB(\lambda_{lower})}$ *is a lower bound on the value of the optimal policy, and* $V_R^{\lambda_{lower}}(s_0)$ *is the optimal policy for* λ_{lower}.

Note that the gap between $\overline{V}_R^{UCB(\lambda_{upper})}(s_0)$ and $\underline{V}_R^{UCB(\lambda_{lower})}(s_0)$ is composed of three parts. First, there is the width of the upper confidence interval $\overline{V}_R^{UCB(\lambda_{upper})}(s_0) - V_R^{UCB(\lambda_{upper})}(s_0)$. Second, there is the difference in the values of the policies $\pi^{UCB(\lambda_{upper})}$ and $\pi^{UCB(\lambda_{lower})}$, which we can write as $V_R^{UCB(\lambda_{upper})}(s_0) - V_R^{UCB(\lambda_{lower})}(s_0)$. Finally, there is the width of the lower confidence interval $V_R^{UCB(\lambda_{lower})}(s_0) - \underline{V}_R^{UCB(\lambda_{lower})}(s_0)$.

4.4 Algorithm

We start by defining extended value iteration for the Lagrangian objective as shown in Algorithm 11. As a side effect, this also computes upper and lower bounds on V_R and V_C in all states and on $Q_R(s,a)$ and $Q_C(s,a)$ in all state-action pairs.

Now we define a binary search algorithm on λ. Given a set of samples, our goal is to find λ_{lower} and λ_{upper} to within tolerance η. The pseudo-code for BINARYSEARCH is shown in Algorithm 12.

We will apply BINARYSEARCH to find λ_{lower} and λ_{upper}. Algorithm 13 defines three functions; FINDLOWER, FINDUPPER, and NEXTLARGERLAMBDA. For λ_{lower}, we are looking for the point λ where $\overline{V}_C^{\lambda}(s_0)$ crosses τ, which is exactly what BINARYSEARCH does. For λ_{upper}, we need to find the point where $\underline{V}_C^{\lambda}(s_0)$ crosses τ, determine the value on the larger side, and then find the largest value of λ_{upper} that achieves that value. The function NEXTLARGERLAMBDA finds the next larger value of λ that will cause the UCB policy to change by calling LAGRANGIANEVI.

The main algorithm works by maintaining an upper bound $\overline{V}_R^{UCB(\lambda_{upper}^{-\nu})}(s_0)$ on the value of the best $(\tau-\nu)$-feasible policy and a lower bound $\underline{V}_R^{UCB(\lambda_{lower}^{+\nu})}(s_0)$ on the value of the best $(\tau+\nu)$-feasible policy. Here the notation $\lambda^{-\nu}$ refers the $(\tau-\nu)$ feasibility and $\lambda^{+\nu}$ refers to $(\tau+\nu)$ feasibility. Sampling proceeds in a series of minibatches that cause these bounds to shrink toward one another. Execution terminates when $\overline{V}_R^{UCB(\lambda_{upper}^{-\nu})}(s_0) - \underline{V}_R^{UCB(\lambda_{lower}^{+\nu})}(s_0) \leq \epsilon$. This is summarized in Algorithm14).

The rationale is the following. The largest value that $V_R^{*(-\nu)}(s_0)$ could have is $\overline{V}_R^{UCB(\lambda_{upper}^{-\nu})}(s_0)$. The smallest value that $\pi^{UCB(\lambda_{lower}^{+\nu})}$ could have is $\underline{V}_R^{UCB(\lambda_{lower}^{+\nu})}(s_0)$. We want the value of $\pi^{UCB\lambda_{lower}^{+\nu})}$ to be no less than ϵ below the value of $V_R^{*(-\nu)}(s_0)$. We attain this by ensuring that $\overline{V}_R^{UCB(\lambda_{lower}^{+\nu})}(s_0) - \underline{V}_R^{UCB(\lambda_{upper}^{-\nu})}(s_0) < \epsilon$. Figure 4.2 shows the above rationale graphically.

4.5 Correctness and Polynomial Running Time

To prove correctness, we must show that, under appropriate conditions, the CONSTRAINEDDDV algorithm will terminate at line 18. Specifically, we will prove the following claim:

Claim 6 *If $\Pi_{\mathcal{L},\eta}(\tau-\nu)$ and $\Pi_{\mathcal{L},\eta}(\tau+\nu)$ are non-empty and $0<\lambda^*<1$, then with probability $1-\delta$,* CONSTRAINEDDDV *will terminate at line 18.*

Algorithm 11 LAGRANGIANEVI(λ, η, δ)

repeat
  for each state s where $N(s) > 0$ **do**
    for each action a **do**
      if $N(s,a) = 0$ **then**
        $\overline{Q}_R^\lambda(s,a) := V_{max}$; $\underline{Q}_R^\lambda(s,a) := 0$
        $\overline{Q}_C^\lambda(s,a) := 1$; $\underline{Q}_C^\lambda(s,a) := 0$
      else
        {Empirical Bernstein Bound backups for V_R}
        {V_R upper bound}
        $\overline{M}_R(s,a) := \sum_{s'} \hat{P}(s'|s,a)\gamma\overline{V}_R^\lambda(s')$
        $\overline{Var}_R(s,a) := \sum_{s'} \hat{P}(s'|s,a)[\gamma\overline{V}_R^\lambda(s') - \overline{M}_R(s,a)]^2$
        $UpperCI_R(s,a) := \sqrt{\frac{2\overline{Var}_R(s)\ln(3/\delta)}{N(s,a)}} + \frac{3\gamma V_{max}\ln(3/\delta)}{N(s,a)}$
        $\overline{Q}_R^\lambda(s,a) := \min\{V_{max}, R(s,a) + \overline{M}(s,a) + UpperCI_R(s,a)\}$

        {V_R lower bound}
        $\underline{M}_R(s,a) := \sum_{s'} \hat{P}(s'|s,a)\gamma\underline{V}_R^\lambda(s')$
        $\underline{Var}_R(s) := \sum_{s'} \hat{P}(s'|s,a)[\gamma\underline{V}_R^\lambda(s') - \underline{M}_R(s,a)]^2$
        $LowerCI_R(s,a) := \sqrt{\frac{2\underline{Var}_R(s)\ln(3/\delta)}{N(s,a)}} + \frac{3\gamma V_{max}\ln(3/\delta)}{N(s,a)}$
        $\underline{Q}_R^\lambda(s,a) := \max\{0, R(s,a) + \underline{M}_R(s,a) - LowerCI_R(s,a)\}$
        {Empirical Bernstein Bound backups for V_C}
        {V_C upper bound [not shown; exactly analogous to the above]}
        {V_C lower bound [not shown; exactly analogous to the above]}
      end if
    end for
    {Now update the UCB policy}
    $\pi^{UCB(\lambda)}(s) := a^* := \text{argmax}_a\, \lambda\overline{Q}_R^\lambda(s,a) - (1-\lambda)\underline{Q}_C^\lambda(s,a)$
    $\overline{V}_R^\lambda(s) := \overline{Q}_R^\lambda(s,a^*)$
    $\underline{V}_R^\lambda(s) := \underline{Q}_R^\lambda(s,a^*)$
    $\overline{V}_C^\lambda(s) := \overline{Q}_C^\lambda(s,a^*)$
    $\underline{V}_C^\lambda(s) := \underline{Q}_C^\lambda(s,a^*)$
  end for
until no Q value changes by more than η

Algorithm 12 BINARYSEARCH($\lambda_{left}, \lambda_{right}, \tau, \eta,$ function $Bound$)

1: {Goal: Find $Bound(\lambda_{left}) \leq \tau \leq Bound(\lambda_{right})$ and $\lambda_{right} - \lambda_{left} \leq \eta$}
2: {The Bound function can be either $\overline{V}_C^{\lambda}(s_0)$ or $\underline{V}_C^{\lambda}(s_0)$ }
3: $\lambda := 1$
4: **repeat**
5: $\quad \lambda_{last} := \lambda$
6: $\quad \lambda = \eta\lceil \frac{1}{2\eta}(\lambda_{left} + \lambda_{right})\rceil$
7: $\quad$ **if** $\lambda = \lambda_{left}$ **or** $\lambda = \lambda_{right}$ **then**
8: $\quad\quad$ **if** $\lambda_{last} = \lambda_{left}$ **then**
9: $\quad\quad\quad \lambda := \lambda_{right}$
10: $\quad\quad$ **else**
11: $\quad\quad\quad \lambda := \lambda_{left}$
12: $\quad\quad$ **end if**
13: $\quad$ **end if**
14: $\quad$ LAGRANGIANEVI(λ, η, δ)
15: $\quad B = Bound(\lambda)$
16: $\quad$ **if** $B \leq \tau$ **then**
17: $\quad\quad \lambda_{left} := \lambda$
18: $\quad$ **end if**
19: $\quad$ **if** $\tau \leq B$ **then**
20: $\quad\quad \lambda_{right} := \lambda$
21: $\quad$ **end if**
22: **until** $\lambda_{right} - \lambda_{left} \leq \eta$
23: **return** $(\lambda_{left}, \lambda_{right})$

Proof 15 *We must show that the conditions of line 17 will be satisfied. Let us start with the first condition that $\lambda_{lower}^{+\nu} = \lambda_{upper}^{-\nu}$. Suppose exploration proceeds until the confidence intervals on $V_C(s_0)$ have width 2ν (so that $V_C^{\lambda}(s_0) - \nu < V_C^{\lambda}(s_0) < V_C^{\lambda}(s_0) + \nu$). We will show that by this point—if not sooner—the first condition will be satisfied. Given that $\Pi_{\mathcal{L},\eta}(\tau - \nu)$ is non-empty and $\lambda^* \in (0, 1)$, the value of $\lambda_{upper}^{-\nu}$ will be less than 1. At $\lambda = \lambda_{upper}^{-\nu}$, by the definition of $\lambda_{upper}^{-\nu}$ we must have*

$$\underline{V}_C^{\lambda}(s_0) < \tau - \nu < \underline{V}_C^{\lambda+\eta}(s_0).$$

Add 2ν to all sides to obtain

$$\underline{V}_C^{\lambda}(s_0) + 2\nu < \tau + \nu < \underline{V}_C^{\lambda+\eta}(s_0) + 2\nu.$$

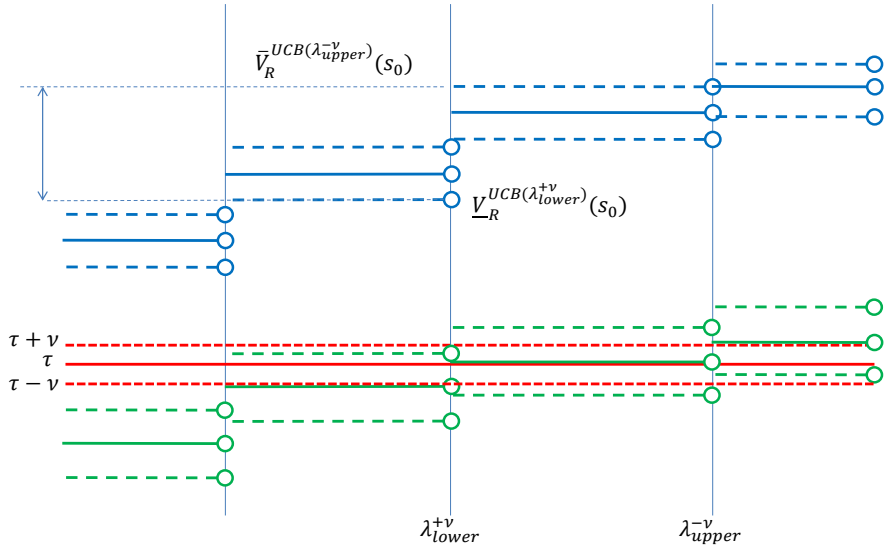


Figure 4.2: Graphical representation of confidence intervals for $\lambda_{lower}^{+\nu}$ and $\lambda_{upper}^{-\nu}$. The horizontal axis plots the value of $0 \leq \lambda \leq 1$. The solid lines denote the true values of V_C and V_R. The dashed lines denote the corresponding upper and lower confidence bounds.

Algorithm 13 FINDLOWER & FINDUPPER

Function FINDLOWER $(\lambda_{left}, \lambda_{right}, \eta, \tau)$
  $(\lambda_{left}, \lambda_{right}) :=$ BINARYSEARCH$(\lambda_{left}, \lambda_{right}, \eta, \tau, \overline{V}_C^{\lambda}(s_0))$
  return λ_{left}
End
Function FINDUPPER$(\lambda_{left}, \lambda_{right}, \eta, \tau)$
  $(\lambda_{left}, \lambda_{right}) :=$ BINARYSEARCH$(\lambda_{left}, \lambda_{right}, \eta, \tau, \underline{V}_C^{\lambda}(s_0))$
  $\lambda_{left} :=$ NEXTLARGERLAMBDA(λ_{right})
  return λ_{left}
End
Function NEXTLARGERLAMBDA (λ)
  for $k = \lambda/\eta + 1$ *to* $1/\eta$ **do**
    LAGRANGIANEVI$(k\eta, \eta, \delta)$
  if $\underline{V}_C^{UCB(k\eta)}(s_0) > \underline{V}_C^{UCB(\lambda)}(s_0)$ **then**
    return $k\eta$
  return 1
End

Algorithm 14 CONSTRAINEDDDV$(s_0, \tau, \nu, F, \epsilon, \delta, \gamma, R_{max})$

1: $\lambda^{+\nu}_{lower} := 0$
2: $\lambda^{-\nu}_{upper} := 1$
3: CheckFeasibility:=true
4: **loop**
5: $\quad \lambda^{-\nu}_{upper} = \text{FINDUPPER}(0, 1, \max(0, \tau - \nu), \eta)$
6: $\quad \lambda^{+\nu}_{lower} = \text{FINDLOWER}(0, 1, \min(1, \tau + \nu), \eta)$
7: $\quad$ **if** CheckFeasibility **then**
8: $\quad\quad$ LAGRANGIANEVI$(0, \eta, \delta)$
9: $\quad\quad$ **if** $\underline{V}_C^{UCB(0)}(s_0) \geq \tau - \nu$ **then**
10: $\quad\quad\quad$ {we have a proof there is no $(\tau - \nu)$-feasible policy}
11: $\quad\quad\quad$ **return** No feasible policy
12: $\quad\quad$ **else if** $\overline{V}_C^{UCB(0)}(s_0) < \tau - \nu$ **then**
13: $\quad\quad\quad$ {we have a proof that there is a $(\tau - \nu)$-feasible policy}
14: $\quad\quad\quad$ CheckFeasibility:=false
15: $\quad\quad$ **end if**
16: $\quad$ **end if**
17: $\quad$ **if** $\left(\lambda^{-\nu}_{upper} = \lambda^{+\nu}_{lower}\right)$ **and** $\left(\overline{V}_R^{UCB(\lambda^{-\nu}_{upper})}(s_0) - \underline{V}_R^{UCB(\lambda^{+\nu}_{lower})}(s_0) \leq \epsilon\right)$ **then**
18: $\quad\quad$ **return** $\left(Success, \pi^{UCB(\lambda^{+\nu}_{lower})}\right)$
19: $\quad$ **end if**
20: $\quad$ Explore for a minibatch of B samples using DDV on $\pi^{UCB(\lambda^{-\nu}_{upper})}$
21: **end loop**

The left hand side is equal to $\overline{V}_C^{\lambda}(s_0)$ because the confidence intervals have width 2ν. Similarly, the right hand side is equal to $\overline{V}_C^{\lambda+\eta}(s_0)$. Substituting these in gives

$$\overline{V}_C^{\lambda}(s_0) < \tau + \nu < \overline{V}_C^{\lambda+\eta}(s_0).$$

This satisfies the definition of $\lambda^{+\nu}_{lower}$. Therefore, $\lambda = \lambda^{+\nu}_{lower}$.

Now let us consider the second condition. Once $\lambda^{+\nu}_{lower} = \lambda^{-\nu}_{upper} = \lambda$, the second condition becomes $\overline{V}_R^{UCB(\lambda^{-\nu}_{upper})}(s_0) - \underline{V}_R^{UCB(\lambda^{+\nu}_{lower})}(s_0) = \overline{V}_R^{UCB(\lambda)}(s_0) - \underline{V}_R^{UCB(\lambda)}(s_0) < \epsilon$. This is a simple constraint on the width of the confidence interval for $V_R^{UCB(\lambda)}(s_0)$. Hence, if the condition is not already satisfied, then additional sampling will shrink the confidence interval to the point where it is satisfied. Hence, the algorithm will terminate at line 18 in CONSTRAINEDDDV.

We can also show the following.

Claim 7 *If there is no $(\tau - \nu)$-feasible policy, then the* CONSTRAINEDDDV *algorithm will terminate at line 11.*

Proof 16 *During the exploration we either observe $\overline{V}_C^{UCB(0)}(s_0) \leq \tau - \nu$ or $\underline{V}_C^{UCB(0)}(s_0) > \tau - \nu$. In the former case, we have a proof that there exists a $(\tau - \nu)$-feasible policy, and the algorithm explores until it meets the termination condition. In the latter case, we have a proof that there is no feasible policy, so the* CONSTRAINEDDDV *terminates at line 11.*

Theorem 9 CONSTRAINEDDDV *requires polynomial sample size and terminates in polynomial computation time.*

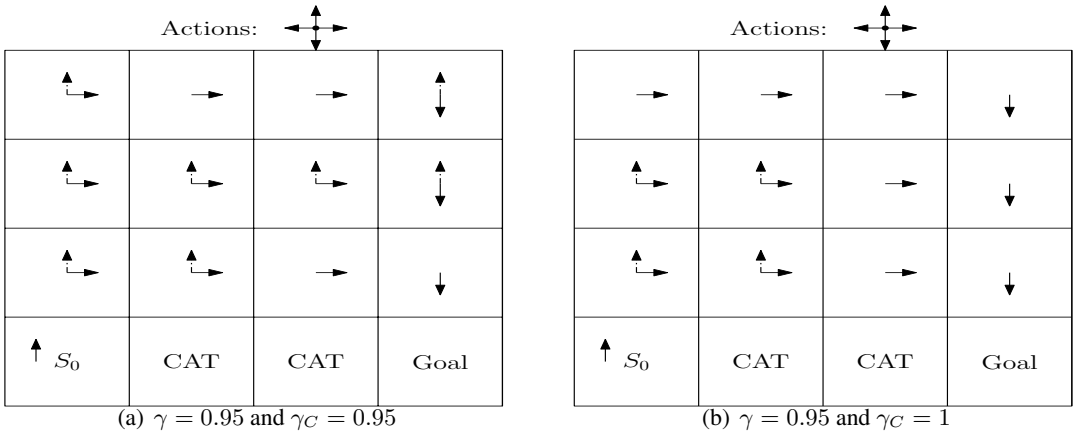
Proof 17 *Collecting total number of samples $N = \tilde{O}\left((|S|^2|A|V_{max}^3)/(\epsilon^3(1-\gamma)^3)\right)$ ensures that every state-action pair is sufficiently sampled [42]. We can fix the total number of minibatches to T and then select the size of each minibatch so that the number of samples in each batch grows quadratically. The result is that the confidence intervals shrink by (approximately) a constant amount $1/T$ in each iteration (similar to doubling trick in [35]).*

After each minibatch, we need to perform two binary searches. The number of times L through the main loop of BINARYSEARCH *algorithm will be the solution to $\eta \geq 1/2^L$, which is $L = \log_2 1/\eta$. Each iteration of* BINARYSEARCH *algorithm requires a call to* LAGRANGIANEVI. *Each such call has worst-case time $O(|S|^2|A|)$. Each call to* FINDUPPER *also requires a call to* NEXTLARGERLAMBDA, *which requires $O(1/\eta)$. Putting this all together, we obtain $\tilde{O}\left((T|S|^2|A|V_{max}^3)/(\eta\epsilon^3(1-\gamma)^3)\right)$, which is polynomial.*

4.6 Experiments

We report four experiments. First, we study the GridWord domain shown in Figure 4.3(a) (there is one starting state, one goal state, and two catastrophic states). Our goal is to gain some intuition about the C-MDP formulation. Specifically, we look at the policies for $\lambda = 0$ and $\lambda = 1$.

In Figure 4.3, we assume the model is known. The solid lines show the optimal policy for $\lambda = 1$ (maximizing the reward), and the dotted actions show the optimal policy for $\lambda = 0$ (minimizing the risk). Notice that even for unequal discount factors, we are able to find a desirable policy, which may not be optimal. The main difference between the policies for discounted and



(a) $\gamma = 0.95$ and $\gamma_C = 0.95$ (b) $\gamma = 0.95$ and $\gamma_C = 1$

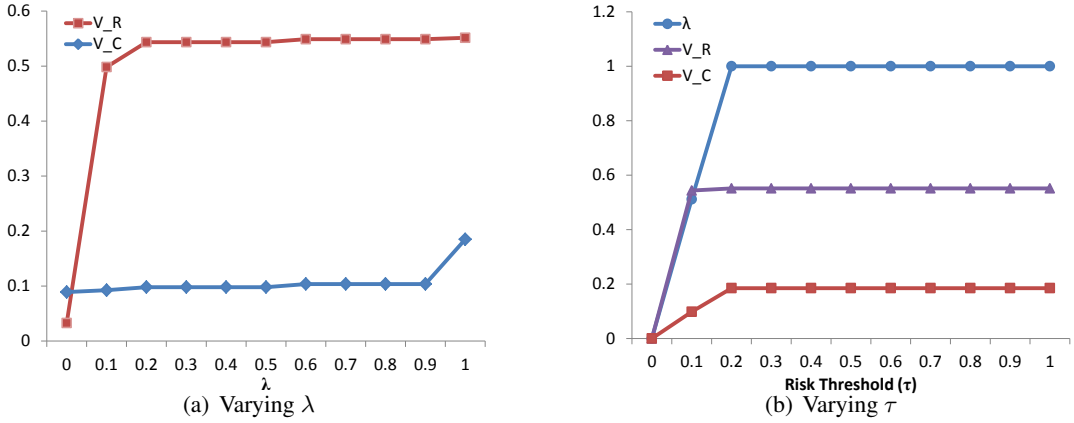
Figure 4.3: Derived policies for the GridWorld domain; solid arrows are when $\lambda = 1$ and dotted arrows are when $\lambda = 0$. When both policies agree on an action in a cell, only one is shown.

undiscounted risk is that for discounted risk the best stationary deterministic policy that minimizes the risk takes the discount into account and moves toward the goal more slowly than the undiscounted risk policy.

In the second experiment, we solve for the optimal policy when the MDP is known while varying λ and the constraint threshold τ. Our goal is to determine the right answer and see the impact of τ and λ. Figure 4.4 shows the value of reward (V_R) and value of risk (V_C) in the starting state for the GridWorld domain while varying the value of λ (4.4(a)) and while varying the value of τ (4.4(b)). There is no feasible policy when $\tau = 0$.

In Figure 4.4(a), we see that when λ is close to 1, we can easily reduce V_C without any impact on V_R. As λ shrinks, V_C and V_R both shrink gradually, so that for values of τ in the range (0.185 to 0.1), there continues to be little impact on V_R. However, when λ goes from 0.1 to 0.0, we see a huge drop in V_R for very little gain in V_C. This kind of sudden drop causes difficulty for obtaining PAC results. The problem is that in this region, the confidence intervals on V_R will be very wide, and it can require a huge number of training samples to shrink them enough to achieve a width of ϵ.

In the third experiment, we compare the sample complexity of CONSTRAINEDDDV against three benchmark algorithms: GW-MLE, ϵ_g-greedy GW-MLE, ϵ_g-greedy UCB. GW-MLE is the improved version of the algorithm of [27], which basically maximizes the Lagrangian defined as

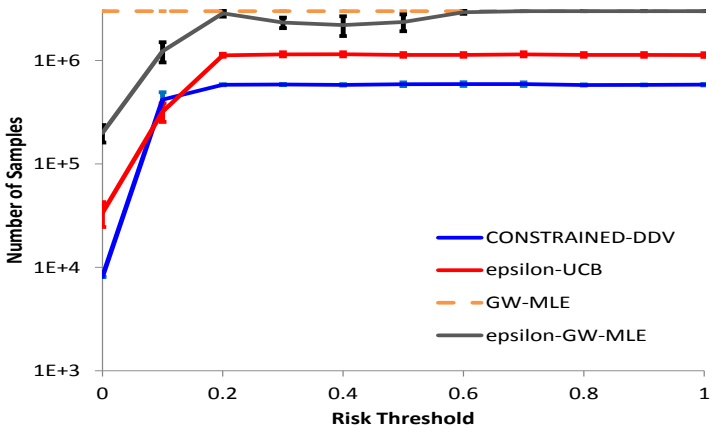


(a) Varying λ (b) Varying τ

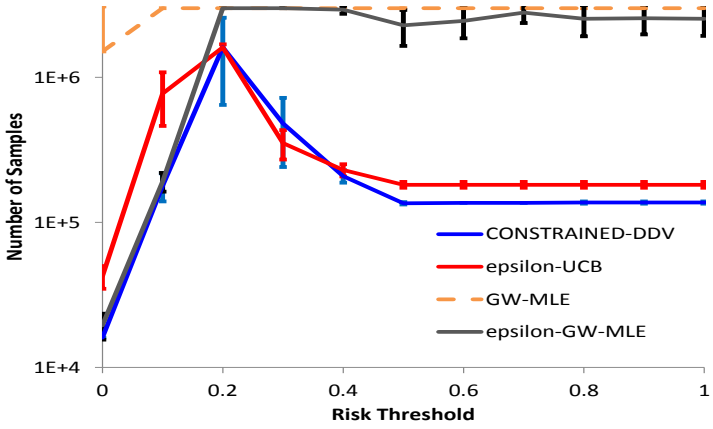
Figure 4.4: Value of reward and risk while varying λ and risk threshold (τ) for the GridWorld domain.

$\mathcal{L}(\hat{\lambda}) = \langle S, A, \hat{P}, \hat{\lambda}R_R - (1-\hat{\lambda})R_C, \gamma, s_0 \rangle$, where $\hat{\lambda}$ is the maximum likelihood estimate of λ calculated over the MDP with transition probability $\hat{P}$ and reward functions R_R and R_C. The GW-MLE algorithm samples along the induced $\pi^{\hat{\lambda}}$ policy at each mini-batch. UCB algorithm calculates $\pi^{UCB} = \text{argmax}_a \overline{Q}_R(s,a)$ and samples along the π^{UCB} policy. Since the UCB algorithm ignores the risk in its default operation, we have added an adjustable ϵ_g parameter for better exploration. The algorithms are modified to have stopping condition similar to the lines 9 and 17 in Algorithm 14 .

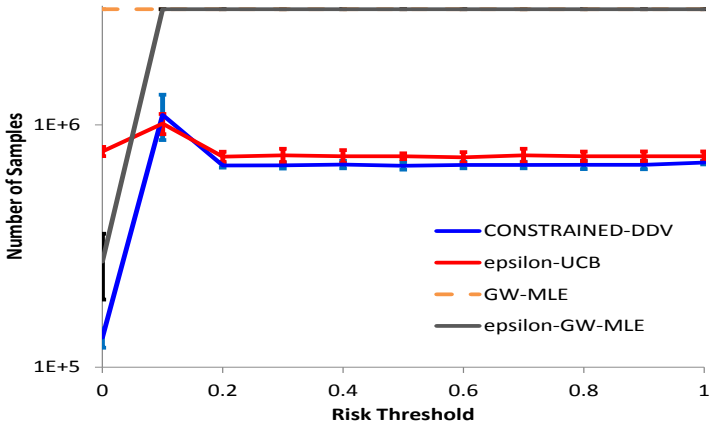
We compared these algorithms on the GridWorld MDP and two instances of the tamarisk domain. In these experiments, we learn the model by sampling from the simulator. Tamarisk problem instances are configured with the number of river segments $(E = 3)$ and the number of slots $(H = 1)$ and $(H = 2)$ (for more detail see [66]). For the $(E = 3, H = 1)$ problem, the starting state was NTE (one site contains a native species, one is invaded by tamarisk, and one site at the bottom of river is empty). For the $(E = 3, H = 2)$ instance, the starting state is NTEEEE (one site contains a native species and an invasive species and the rest of the sites in the river are empty). A catastrophic state is any state in which there are no natives (species extinction). The goal state is that all sites are fully occupied by native species. We optimized the value of ϵ_g for ϵ_g-greedy GW-MLE and ϵ_g-greedy UCB algorithms among the candidate values $\epsilon_g \in \{0.01, 0.1, 0.25\}$. After sampling a minibatch of size $B = 1000$ we update the model



(a) GridWorld



(b) Tamarisk $R = 3$ and $H = 1$



(c) Tamarisk $R = 3$ and $H = 2$

Figure 4.5: Comparison of number of samples taken by each algorithm to reach to the termination point.

and calculate the corresponding confidence bounds. We calculate λ_{upper} and λ_{lower} every 8000 samples.

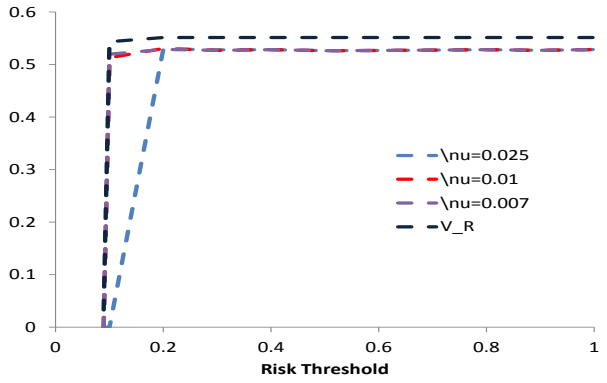
In these experiments, $\gamma = \gamma_C = 0.95$, $\delta = 0.01$, $\eta = 0.01$, and $\nu = 0.025$. For the GridWorld domain, $\epsilon = 0.2$, and for the Tamarisk problems $\epsilon = 1$. The algorithms terminate either if the width of the confidence interval falls below ϵR_{max} or if 3 million samples are drawn.

We report the number of samples drawn at termination in Figure 4.5. The results are averaged over 10 independent runs, and the vertical axis is plotted on a log scale. Error bars indicate one standard deviation. The GW-MLE and ϵ_g-GW-MLE algorithms perform very poorly; much worse than CONSTRAINEDDDV. In many cases, they hit the 3 million maximum sampling budget without achieving the desired confidence interval width. CONSTRAINEDDDV and ϵ_g-UCB give much more similar performance, if ϵ_g is properly tuned. CONSTRAINEDDDV almost always requires smaller sample sizes, particularly for small values of τ (which would be the values normally encountered in a real application). This is presumably because CONSTRAINEDDDV focuses its exploration on the parts of the MDP visited by policies with low V_C, whereas ϵ_g-UCB is only maximizing V_R. However, these MDPs exhibit a "critical point", where the required sample size grows quite large. At couple of those critical points, CONSTRAINEDDDV and ϵ_g-UCB require the same sample size, but the CONSTRAINEDDDV sample size exhibits more variance.

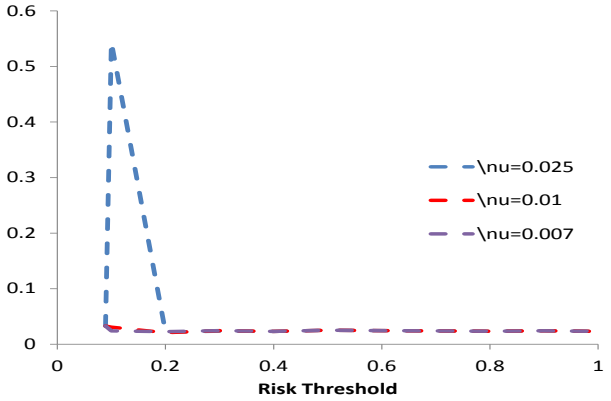
In the final experiment, the average values of the lower bounds for reward and catastrophe and the value of $V_R - \underline{V}_R^{UCB(\lambda^{\nu}_{lower})}$ at the termination point are reported. Figure 4.6 shows the mean $\underline{V}_R^{UCB(\lambda^{\nu}_{lower})}$, $V_R - \underline{V}_R^{UCB(\lambda^{\nu}_{lower})}$, and $\underline{V}_C^{UCB(\lambda^{\nu}_{lower})}$ values obtained by running the CONSTRAINEDDDV algorithm on the GridWorld domain for five independent runs. In these experiments, $\epsilon = 0.05$ and $\nu \in \{0.025, 0.01, 0.007\}$. Figures 4.6(a) and 4.6(c) show both $\underline{V}_R^{UCB(\lambda^{\nu}_{lower})}$ and $\underline{V}_C^{UCB(\lambda^{\nu}_{lower})}$ for different ν are the lower bounds on the true values. Figure 4.6(b) shows $\underline{V}_R^{UCB(\lambda^{\nu}_{lower})}$ is $\epsilon/2$ close to the true V_R value as the PAC-Safe-RL definition claims. Note that when $\nu = 0.025$ and $\tau = 0.1$ there is no $(\tau - \nu)$-feasible policy.

4.7 Conclusion

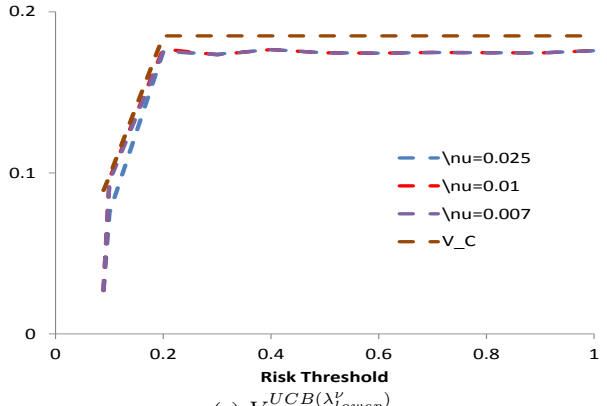
Many computational sustainability problems involving MDPs must be concerned with catastrophic outcomes such as species extinction. One approach to this is to limit the probability of catastrophic outcomes by imposing a constraint on the MDP policy, which converts the MDP into a Constrained MDP (C-MDP). Previous work on simulation-based MDP planning for constrained MDPs has not provided formal guarantees. This chapter is the first to provide an algo-



(a) $\underline{V}_R^{UCB(\lambda^{\nu}_{lower})}$



(b) $V_R - \underline{V}_R^{UCB(\lambda^{\nu}_{lower})}$; when $\nu = 0.025$ and $\tau = 0.1$ there is no $(\tau - \nu)$-feasible policy.



(c) $\underline{V}_C^{UCB(\lambda^{\nu}_{lower})}$

Figure 4.6: Plots of $\underline{V}_R^{UCB(\lambda^{\nu}_{lower})}$, $V_R - \underline{V}_R^{UCB(\lambda^{\nu}_{lower})}$, and $\underline{V}_C^{UCB(\lambda^{\nu}_{lower})}$ on the vertical axis for three values of ν in the GridWorld domain.

rithm with formal guarantees by extending the notion of PAC-RL algorithms to PAC-Safe-RL algorithms. We proved that this new algorithm, CONSTRAINEDDDV, is PAC-Safe-RL. Our experiments demonstrated that CONSTRAINEDDDV is also able to match or beat the sample complexity of very competitive baseline algorithms that lack formal performance guarantees.

Chapter 5: Conclusion

5.1 Conclusion

This thesis has addressed the problem of unconstrained and constrained MDP planning in natural resource management when the MDP is defined by an expensive simulator. In this setting, the planning phase is separate from the execution phase, so there is no tradeoff between exploration and exploitation. Instead, the goal is to compute a PAC-optimal policy while minimizing the number of calls to the simulator. The policy is designed to optimize the cumulative discounted reward starting in the current real-world state s_0. Unlike in most published RL papers, which typically assume that the MDP is ergodic, the starting state of our ecosystem management problems is typically a transient state.

Chapter 2 has made two contributions. First, it showed how to combine the Good-Turing estimate with the L_1-confidence region of Weissman et al. [73] to obtain tighter confidence intervals (and hence, earlier termination) in sparse MDPs. Second, it showed how to use occupancy measures to create better exploration heuristics. It introduced a new policy-independent upper bound $\overline{\mu}$ on the occupancy measure of the optimal policy and applied this to define the DDV-UPPER algorithm. It also employed an occupancy measure μ^{OUU} based on the "optimism under uncertainty" principle to define the DDV-OUU algorithm.

The $\overline{\mu}$ measure is potentially of independent interest. Like the value function computed during value iteration, it does not quantify the behavior of any particular policy. This means that it can be computed without needing to have a specific policy to evaluate. However, the DDV-UPPER exploration heuristic did not perform very well. We have two possible explanations for this. First, $\overline{\mu}$ can be a very loose upper bound on the optimal occupancy measure μ^*. Perhaps this leads DDV-UPPER to place too much weight on unfruitful state-action pairs. Second, it is possible that while DDV-UPPER is optimizing the one-step gain in $\Delta\Delta V(s_0)$ (as it is designed to do), DDV-OUU does a better job of optimizing gains over the longer term. Further experimentation is needed to determine which of these explanations is correct.

Our DDV-OUU method gave the best performance in all of our experiments in Chapter 2. This is yet another confirmation of the power of the "Optimism Principle" [6] in exploration.

Hence, we recommend it for solving simulator-defined MDP planning problems. We are applying it to solve moderate-sized instances of our tamarisk MDPs. However, additional algorithm innovations will be required to solve much larger tamarisk instances.

Chapter 3 has made three main contributions. First, it has developed optimal sampling strategies for Monte Carlo policy evaluation using both local and trajectory-wise forms of the Hoeffding and empirical Bernstein bounds. Second, it provided experimental evidence that for policy evaluation the trajectory-wise bounds generally out-perform the local bounds except in a few special cases. Third, it introduced two new MDP planning algorithms, LGCV and LLGCV, for simulator-defined MDPs. These algorithms combine trajectory-wise confidence intervals with local confidence intervals to reduce the number of samples needed to achieve target levels of accuracy. The LGCV algorithm combines a local upper bound with a global lower bound. Although this performs well on some problems, it exhibits poor performance on others. The LLGCV algorithm extends LGCV by intersecting both the local and global lower bounds. Although this requires computing additional confidence intervals, it eliminates LGCV's failure cases. Hence, LLGCV provides a robust method that can obtain significant improvements over previous methods.

Chapter 4 has studied limiting the probability of catastrophic outcomes by imposing a constraint on the MDP policy, which converts the MDP into a C-MDP. Many computational sustainability problems must deal with catastrophic outcomes such as species extinction. Previous published work on MDP planning for constrained MDPs has not provided formal guarantees. This chapter is the first to provide an algorithm with formal guarantees by extending the notion of PAC-RL algorithms to PAC-Safe-RL algorithms. We proved that this new algorithm, CONSTRAINEDDDV, is PAC-Safe-RL. Our experiments demonstrated that CONSTRAINEDDDV is also able to match or beat the sample complexity of very competitive baseline algorithms that lack formal performance guarantees.

5.2 Future Work

Promising directions for future research are summarized as follows.

One natural direction is to explore the benefits of other confidence interval methods, such as Kullback-Leibler divergence [24], extensions of empirical Bernstein bound using Anderson's inequality [68], Bernstein-like derivation bound for the missing mass [40], and improvements on the Good-Turing estimate [50, 70]. Moreover, as shown in this thesis, Monte-Carlo policy

evaluation based on empirical Bernstein bound outperformed the other studied bounds. However, performing the policy improvement step wasn't efficient. Another research direction is to apply global confidence intervals for Monte-Carlo policy improvement in an efficient way.

The other direction is incorporating other exploration methods in RL and multi-armed bandit literature. For example, in RL literature, posterior sampling [69] algorithm has been developed to minimize regret (PSRL [52, 51]). In such settings, it has been shown experimentally to be better than UCB methods. The open research question is whether posterior sampling might be useful for PAC algorithms that must halt and output a policy. In multi-armed bandit literature, LUCB [37] has been proposed for finding the best PAC m-arms, where m could be one. How can one extend the similar algorithms from multi-armed bandits to MDPs?

In this thesis we studied algorithms to compute optimal solutions exactly for discrete MDPs. One can expand the proposed exploration algorithms to continuous MDPs with linear function approximation [54, 43]. Moreover, recent advances in deep reinforcement learning for (non-linear) value function approximation and policy gradient suggest applying the proposed methods in this thesis to find approximate optimal solutions using deep architectures. One approach is to estimate the occupancy measure using a parameterized representation as a deep architecture [41] and use the heuristic in this thesis for exploration. One trend in deep reinforcement learning is using ensemble methods to support robustness and safety [56]. Osband et al. [53] proposed bootstrapped DQN to adapt posterior sampling for deep exploration. The algorithm uses (bootstrap) data to approximate a distribution over Q values, and samples accordingly. Could we apply our algorithms to bootstrap confidence intervals using this architecture?

The constrained MDP algorithm presented in this thesis is an instance of multi-objective sequential decision making, where a decision maker faces multiple conflicting objectives [58, 57]. The biggest challenge with CONSTRAINEDDDV is that it only considers policies that can be produced by the Lagrangian. These are a subset of the complete Pareto set of non-dominated policies. There are methods in the Operations Research (OR) literature for finding all points in the Pareto set, and extending CONSTRAINEDDDV to do this is an important next step.

Bibliography

[1] Eitan Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

[2] Jean Yves Audibert, Remi Munos, and Csaba Szepesvári. Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410 (19):1876–1902, 2009.

[3] Mohammad Gheshlaghi Azar, Remi Munos, and Hilbert J Kappen. On the sample complexity of reinforcement learning with a generative model. In *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*, 2012.

[4] Richard Bellman. *Dynamic Programming*. Princeton University Press, New Jersey, 1957.

[5] Craig Boutilier and Tyler Lu. Budget allocation using weakly coupled, constrained Markov decision processes. `https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45291.pdf`, 2016.

[6] Lucian Buşoniu and Remi Munos. Optimistic planning for Markov decision processes. In *15th International Conference on Artificial Intelligence and Statistics (AI-STATS-12)*, 2012.

[7] Iadine Chadès, Tara G. Martin, Samuel Nicol, Mark A. Burgman, Hugh P. Possingham, and Yvonne M. Buckley. General rules for managing and surveying networks of pests, diseases, and endangered species. *Proceedings of the National Academy of Sciences of the United States of America*, 108(20):8323–8, 2011.

[8] Hyeong Soo Chang. Sleeping experts and bandits approach to constrained Markov decision processes. *Automatica*, 63:182 – 186, 2016. ISSN 0005-1098.

[9] Hyeong Soo Chang, Jiaqiao Hu, Michael C Fu, and Steven I Marcus. *Simulation-based algorithms for Markov decision processes*. Springer Science & Business Media, 2013.

[10] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[11] Mark Crowley. Using equilibrium policy gradients for spatiotemporal planning in forest ecosystem management. *IEEE Transactions on Computers*, 63(1):142–154, 2014.

[12] Thomas G Dietterich, Majid Alkaee Taleghan, and Mark Crowley. PAC optimal planning for invasive species management: improved exploration for reinforcement learning from simulator-defined MDPs. In *Association for the Advancement of Artificial Intelligence AAAI 2013 Conference (AAAI-2013)*, 2013.

[13] Joseph M DiTomaso and Carl E Bell. *Proceedings of the Saltcedar Management Workshop*. www.invasivespeciesinfo.gov/docs/news/workshopJun96/index.html, Rancho Mirage, CA, 1996.

[14] Dmitri A. Dolgov and Edmund H. Durfee. Stationary deterministic policies for constrained MDPs with multiple rewards, costs, and discount factors. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1326–1332, Edinburgh, Scotland, 2005.

[15] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. PAC bounds for multi-armed bandit and Markov decision processes. In *Proceedings of the 15th Annual Conference on Computational Learning Theory*, pages 255–270, London, 2002. Springer-Verlag.

[16] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for reinforcement learning. In Tom Fawcett and Nina Mishra, editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 162–169. AAAI Press, 2003.

[17] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research*, 7:1079–1105, 2006.

[18] Eugene A Feinberg. Constrained discounted Markov decision processes and Hamiltonian cycles. *Mathematics of Operations Research*, 25(1):130–140, 2000.

[19] Eugene A Feinberg. Optimality of deterministic policies for certain stochastic control problems with multiple criteria and constraints. In *Mathematical Control Theory and Finance*, pages 137–148. Springer, 2008.

[20] Eugene A Feinberg and Uriel G Rothblum. Splitting randomized stationary policies in total-reward Markov decision processes. *Mathematics of Operations Research*, 37(1):129–153, 2012.

[21] Eugene A Feinberg and Adam Shwartz. Constrained discounted dynamic programming. *Mathematics of Operations Research*, 21(4):922–945, 1996.

[22] Claude-Nicolas Fiechter. Efficient reinforcement learning. In *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pages 88–97. ACM Press, 1994.

[23] Claude-Nicolas Fiechter. *Design and Analysis of Efficient Reinforcement Learning Algorithms*. PhD thesis, University of Pittsburgh, Pittsburgh, PA, USA, 1997.

[24] Sarah Filippi, Olivier Cappé, and Aurélien Garivier. Optimism in reinforcement learning and Kullback-Leibler divergence. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pages 115–122. IEEE, 2010.

[25] Mark Arnold Finney et al. *FARSITE: Fire area simulator: model development and evaluation*. US Department of Agriculture, Forest Service, Rocky Mountain Research Station Ogden, UT, 2004.

[26] Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16:1437–1480, 2015.

[27] Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *J. Artif. Intell. Res.(JAIR)*, 24:81–108, 2005.

[28] Alborz Geramifard. *Practical Reinforcement Learning Using Representation Learning And Safe Exploration For Large Scale Markov Decision Processes*. PhD thesis, Massachusetts Institute of Technology, 2012.

[29] Mohammad Gheshlaghi Azar, Rémi Munos, Mohammad Ghavamzadeh, and Hilbert Kappen. Reinforcement learning with a near optimal rate of convergence. Technical report, Radboud University Nijmegen, 2011.

[30] Irving John Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3):237–264, 1953.

[31] Kim M. Hall. *Optimal Spatial-Dynamic Resource Allocation Facing Uncertainty: Integrating Economics and Ecology for Invasive Species Policy*. Doctoral dissertation, Oregon State University, 2014. URL `http://hdl.handle.net/1957/52661`.

[32] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[33] J. Hof. Optimizing spatial and dynamic population-based control strategies for invading forest pests. *Natural Resource Modeling*, 11:197–216, 1998.

[34] Rachel M. Houtman, Claire A Montgomery, Aaron R Gagnon, David E. Calkin, Thomas G. Dietterich, Sean McGregor, and Mark Crowley. Allowing a wildfire to burn: estimating the effect on future fire suppression costs. *International Journal of Wildland Fire*, 22:871–882, 2013.

[35] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.

[36] Sham M. Kakade. *On the Sample Complexity of Reinforcement Learning*. Doctoral dissertation, University College London, 2003.

[37] Shivaram Kalyanakrishnan, Ambuj Tewari, Peter Auer, and Peter Stone. PAC subset selection in stochastic multi-armed bandits. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 655–662, 2012.

[38] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49:260–268, 1998.

[39] Michael J Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. In *IJCAI*, pages 1231–1324, 1999.

[40] Bahman Yari Saeed Khanloo and Gholamreza Haffari. Novel Bernstein-like concentration inequalities for the missing mass. *arXiv preprint arXiv:1503.02768*, 2015.

[41] Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016.

[42] Lihong Li. Sample complexity bounds of exploration. In *Reinforcement Learning*, pages 175–204. Springer, 2012.

[43] Lihong Li, Michael L Littman, and Christopher R Mansley. Online exploration in least-squares policy iteration. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 733–739. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[44] Shie Mannor, O Mebel, and H Xu. Lightning does not strike twice: robust MDPs with coupled uncertainty. In *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*, 2012.

[45] David McAllester and Luis Ortiz. Concentration inequalities for the missing mass and for histogram rule error. *Journal of Machine Learning Research*, 4:895–911, 2003.

[46] David McAllester and Robert E Schapire. On the convergence rate of Good-Turing estimators. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 1–6, 2000.

[47] Michael E. Moody and Richard N. Mack. Controlling the spread of plant invasions: the importance of Nascent Foci. *Journal of Applied Ecology*, 25(3):1009–1021, 1988.

[48] Rachata Muneepeerakul, Simon A Levin, Andrea Rinaldo, and Ignacio Rodriguez-Iturbe. On biodiversity in river networks: a trade-off metapopulation model and comparative analysis. *Water Resources Research*, 43(7):1–11, 2007.

[49] Andrew Y. Ng and Michael I. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *UAI 2000*, pages 406–415. Morgan Kaufmann Publishers Inc., 2000.

[50] Alon Orlitsky, Narayana P. Santhanam, and Junan Zhang. Always Good Turing: asymptotically optimal probability estimation. *Science*, 302(5644):427–431, 2003.

[51] Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning. *arXiv preprint arXiv:1607.00215*, 2016.

[52] Ian Osband, Dan Russo, and Benjamin Van Roy. (More) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, pages 3003–3011, 2013.

[53] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. *arXiv preprint arXiv:1602.04621*, 2016.

[54] Jason Pazis and Ronald Parr. PAC optimal exploration in continuous space Markov decision processes. In *AAAI*. Citeseer, 2013.

[55] Martin Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Mathematical Statistics. Wiley, 1994.

[56] Aravind Rajeswaran, Sarvjeet Ghotra, Sergey Levine, and Balaraman Ravindran. EPOpt: learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

[57] Diederik Marijn Roijers. *Multi-Objective Decision-Theoretic Planning*. PhD thesis, University of Amsterdam, 2016.

[58] Diederik Marijn Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 2013.

[59] Daniel Sheldon, Bistra Dilkina, Adam Elmachtoub, Ryan Finseth, Ashish Sabharwal, Jon Conrad, Carla Gomes, David Shmoys, W. Allen, and O. Amundsen. Maximizing the spread of cascades using network design. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 517–526, 2010.

[60] Trey Smith and Reid Simmons. Focused real-time dynamic programming for MDPs: squeezing more out of a heuristic. In *AAAI 2006*, pages 1227–1232, 2006.

[61] Scott M Stenquist. *Saltcedar Management and Riparian Restoration Workshop*. www.invasivespeciesinfo.gov/docs/news/workshopSep96/index.html, Las Vegas, NV, 1996.

[62] Alexander Strehl and Michael Littman. An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

[63] Alexander L Strehl and Michael L Littman. An empirical evaluation of interval estimation for Markov decision processes. In *16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004)*, pages 128–135, 2004.

[64] Umar Syed, Michael Bowling, and Robert Schapire. Apprenticeship learning using linear programming. In *International Conference on Machine Learning*, Helsinki, Finland, 2008.

[65] Istvan Szita and Csaba Szepesvári. Model-based reinforcement learning with nearly tight exploration complexity bounds. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 1031–1038, 2010.

[66] Majid Alkaee Taleghan, Thomas G. Dietterich, Mark Crowley, Kim Hall, and H. Jo Albers. PAC optimal MDP planning with application to invasive species management. *Journal of Machine Learning Research*, 16:3877–3903, 2015.

[67] Aviv Tamar, Shie Mannor, and Huan Xu. Scaling up robust MDPs using function approximation. In *ICML 2014*, volume 32, 2014.

[68] Philip S Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High-confidence off-policy evaluation. In *AAAI*, pages 3000–3006, 2015.

[69] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3):285–294, 1933.

[70] Gregory Valiant and Paul Valiant. Estimating the unseen: improved estimators for entropy and other properties. In *Neural Information Processing Systems 2013*, pages 1–9, 2013.

[71] Abraham Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945.

[72] Thomas J Walsh, Sergiu Goschin, and Michael L Littman. Integrating sample-based planning and model-based reinforcement learning. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, Atlanta, GA, 2010.

[73] Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marcelo J Weinberger. Inequalities for the L1 deviation of the empirical distribution. Technical report, HP Labs, 2003.

[74] Alexander Zadorojniy, Guy Even, and Adam Shwartz. A strongly polynomial algorithm for controlled queues. *Mathematics of Operations Research*, 34(4):992–1007, 2009.

APPENDICES

Appendix A: Proofs of the Main Result

A.1 Proof of Theorem 8

Suppose the current optimistic policy at time t is $\pi^t = \pi^{UCB}$ and the current maximum likelihood transition estimates at time t are $\hat{P}^t$. Following the same argument as [16] and the bounds on $\overline{V}^{\pi^t}(s_0)$, we can bound $Q^*(s_0,a) - \underline{Q}(s_0,a) \leq \overline{Q}(s_0,a) - \underline{Q}(s_0,a)$. This means that by bounding $\overline{Q}(s_0,a) - \underline{Q}(s_0,a) \leq \epsilon(1-\gamma)/2$ for $a = \pi_t(s_0)$ we can guarantee $|V^{\pi_t}(s_0) - V^*(s_0)| < \epsilon$. We therefore compute the sample size required to get $\Delta Q(s,a) \leq \epsilon(1-\gamma)/2$ for all (s,a).

We use the same style of proof as Proposition 1 in [62]. Let $(s^*,a^*) = \text{argmax}_{(s,a)} |\overline{Q}(s,a) - \underline{Q}(s,a)|$. For a given optimistic policy π, we use the largest value of variance as $\overline{Var}^{\pi}(s_0,\pi(s_0)) = \gamma^2 V_{max}^2/4$. Therefore,

$$
\begin{aligned}
&\max_{(s,a)} |\overline{Q}(s,a) - \underline{Q}(s,a)| \\
&= \overline{Q}(s^*,a^*) - \underline{Q}(s^*,a^*) \\
&= R(s^*,a^*) + \sum_{s'} \hat{P}(s'|s^*,a^*)\gamma\overline{V}(s') \; + \; \sqrt{\frac{2\overline{Var}(s^*,a^*)\ln(3/\delta_0)}{N(s^*,a^*)}} + \frac{3\gamma V_{max}\ln(3/\delta_0)}{N(s^*,a^*)} \\
&- R(s^*,a^*) - \sum_{s'} \hat{P}(s'|s^*,a^*)\gamma\underline{V}(s') \; + \; \sqrt{\frac{2\underline{Var}(s^*,a^*)\ln(3/\delta_0)}{N(s^*,a^*)}} + \frac{3\gamma V_{max}\ln(3/\delta_0)}{N(s^*,a^*)} \\
&\leq \gamma \sum_{s'} \hat{P}(s'|s^*,a^*)\left(\overline{V}(s') - \underline{V}(s')\right) \; + \; \gamma\sqrt{\frac{2V_{max}^2\ln(3/\delta_0)}{N(s^*,a^*)}} + \frac{6\gamma V_{max}\ln(3/\delta_0)}{N(s^*,a^*)} \\
&\leq \gamma \max_s |\overline{V}(s) - \underline{V}(s)| \; + \; \gamma\sqrt{\frac{2V_{max}^2\ln(3/\delta_0)}{N(s^*,a^*)}} + \frac{6\gamma V_{max}\ln(3/\delta_0)}{N(s^*,a^*)} \\
&\leq \gamma \max_{(s,a)} |\overline{Q}(s,a) - \underline{Q}(s,a)| \; + \; \gamma\sqrt{\frac{2V_{max}^2\ln(3/\delta_0)}{N(s^*,a^*)}} + \frac{6\gamma V_{max}\ln(3/\delta_0)}{N(s^*,a^*)}.
\end{aligned}
$$

Now, we have

$$\max_{(s,a)} |\overline{Q}(s,a) - \underline{Q}(s,a)| \quad \leq \quad \frac{1}{(1-\gamma)} \left(\gamma V_{max} \sqrt{\frac{2\ln(3/\delta_0)}{N(s,a)}} + \frac{6\gamma V_{max} \ln(3/\delta_0)}{N(s,a)} \right).$$

To make $\Delta Q(s,a) \leq \epsilon(1-\gamma)/2$ for all (s,a), we need to bound the right-hand side of the above equations as follows:

$$\gamma V_{max} \sqrt{\frac{2\ln(3/\delta_0)}{N(s,a)}} + \frac{6\gamma V_{max} \ln(3/\delta_0)}{N(s,a)} \leq \frac{\epsilon(1-\gamma)}{2}$$
$$\sqrt{\frac{2\ln(3/\delta_0)}{N(s,a)}} + \frac{6\ln(3/\delta_0)}{N(s,a)} \leq \frac{\epsilon(1-\gamma)}{2V_{max}\gamma}.$$

We use $\delta_0 = \frac{6\delta}{2\pi^2|S||A|t^2}$ in the following derivation instead of the value $\delta_0 = \frac{\delta}{2|S||A|t(t+1)}$ to make the derivation simpler:

$$\sqrt{\frac{2\ln(\frac{\pi^2|S||A|m^2}{\delta})}{m}} + \frac{6\ln(\frac{\pi^2|S||A|m^2}{\delta})}{m} = \frac{\epsilon(1-\gamma)}{2V_{max}\gamma}$$
$$6\left(\sqrt{\frac{\ln(\frac{\pi^2|S||A|m^2}{\delta})}{m}} + \frac{\sqrt{2}}{12}\right)^2 = \frac{1}{12} + \frac{\epsilon(1-\gamma)}{2\gamma V_{max}}$$
$$\frac{\ln(\frac{\pi^2|S||A|m^2}{\delta})}{m} = \left[\sqrt{\frac{1}{6}\left(\frac{1}{12} + \frac{\epsilon(1-\gamma)}{2\gamma V_{max}}\right)} - \frac{\sqrt{2}}{12}\right]^2$$
$$\frac{\ln(\pi^2|S||A|)+\ln(\frac{1}{\delta})+2\ln(m)}{m} = \left[\sqrt{\frac{1}{6}\left(\frac{1}{12} + \frac{\epsilon(1-\gamma)}{2\gamma V_{max}}\right)} - \frac{\sqrt{2}}{12}\right]^2$$

One can derive the above using simple algebra. We use the following substitutions to make the equations easier to follow. Let $c_1 = \ln(\pi^2|S||A|) + \ln(\frac{1}{\delta})$, $c_2 = \left[\sqrt{\frac{1}{6}\left(\frac{1}{12} + \frac{\epsilon(1-\gamma)}{2\gamma V_{max}}\right)} - \frac{\sqrt{2}}{12}\right]^2$, and $u = \ln(m)$. Then, the last equation becomes:

$$(c_1 + 2u)e^{-u} = c_2.$$

Define $x = 0.5c_1 + u$ and substitute:

$$xe^{-x} = 0.5c_2e^{-0.5c_1}.$$

Replacing $x = -y$ gives

$$ye^{y} = -0.5c_2e^{-0.5c_1} \Rightarrow y = W_{-1}(-0.5c_2e^{-0.5c_1}),$$

where W is Lambert W function. By tracing back the substitutions, we end up with a sample complexity of

$$m = e^{-0.5(\ln(\pi^2|S||A|)+\ln(\frac{1}{\delta}))-W_{-1}\left(-0.5*\left[\sqrt{\frac{1}{6}\left(\frac{1}{12}+\frac{\epsilon(1-\gamma)}{2\gamma V_{max}}\right)}-\frac{\sqrt{2}}{12}\right]^2 e^{-0.5*(\ln(\pi^2|S||A|)+\ln(\frac{1}{\delta}))}\right)}.$$

We use an upper bound on Lambert W function to eliminate it:

$$m = e^{-0.5(\ln(\pi^2|S||A|)+\ln(\frac{1}{\delta}))-2\ln(0.5*\left[\sqrt{\frac{1}{6}\left(\frac{1}{12}+\frac{\epsilon(1-\gamma)}{2\gamma V_{max}}\right)}-\frac{\sqrt{2}}{12}\right]^2 e^{-0.5*(\ln(\pi^2|S||A|)+\ln(\frac{1}{\delta}))})}.$$

Therefore, the sample complexity for the maximum variance is

$$m = \sqrt{\frac{\pi^2|S||A|}{\delta}}\left(\frac{1}{\sqrt{\frac{1}{6}\left(\frac{1}{12}+\frac{\epsilon(1-\gamma)}{2\gamma V_{max}}\right)}-\frac{\sqrt{2}}{12}}\right)^4.$$

Repeating the above steps with $\overline{Var} = 0$, we obtain

$$\frac{6\gamma V_{max}\ln(\frac{\pi^2|S||A|m^2}{\delta})}{m} = \frac{\epsilon}{2(1-\gamma)}$$
$$\frac{\ln(\frac{\pi^2|S||A|m^2}{\delta})}{m} = \frac{\epsilon(1-\gamma)}{12\gamma V_{max}}$$
$$\frac{\ln(\pi^2|S||A|)+\ln(\frac{1}{\delta})+2\ln(m)}{m} = \frac{\epsilon(1-\gamma)}{12\gamma V_{max}}$$
$$m = e^{-0.5(\ln(\pi^2|S||A|)+\ln(\frac{1}{\delta}))-W_{-1}\left(-0.5*\frac{\epsilon(1-\gamma)}{12\gamma V_{max}}e^{-0.5*(\ln(\pi^2|S||A|)+\ln(\frac{1}{\delta}))}\right)}$$
$$m = e^{-0.5(\ln(\pi^2|S||A|)+\ln(\frac{1}{\delta}))-2\ln\left(0.5*\frac{\epsilon(1-\gamma)}{12\gamma V_{max}}e^{-0.5*(\ln(\pi^2|S||A|)+\ln(\frac{1}{\delta}))}\right)}.$$

Therefore, the sample complexity for this case is

$$m = \sqrt{\frac{\pi^2|S||A|}{\delta}}\left(\frac{1}{\frac{\epsilon(1-\gamma)}{24\gamma V_{max}}}\right)^2.$$

Therefore, asymptotic sample complexity for both cases is

$$O\left(|S||A|\sqrt{\frac{|S||A|}{\delta}}\frac{R_{max}^2\gamma^2}{\epsilon^2(1-\gamma)^4}\right).$$