PLANNING IN HYBRID STRUCTURED STOCHASTIC DOMAINS

by

Branislav Kveton

M.S., Comenius University, Slovak Republic, 2001

Submitted to the Graduate Faculty of

the Intelligent Systems Program in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2006

UNIVERSITY OF PITTSBURGH

INTELLIGENT SYSTEMS PROGRAM

This dissertation was presented

by

Branislav Kveton

It was defended on

September 7, 2006

and approved by

Milos Hauskrecht

Department of Computer Science, University of Pittsburgh

Gregory Cooper

Department of Biomedical Informatics, University of Pittsburgh

Marek Druzdzel

School of Information Sciences, University of Pittsburgh

Carlos Guestrin

Machine Learning Department *and* Computer Science Department

Carnegie Mellon University

Dissertation Director: **Milos Hauskrecht**

Department of Computer Science, University of Pittsburgh

PLANNING IN HYBRID STRUCTURED STOCHASTIC DOMAINS

Branislav Kveton, PhD

University of Pittsburgh, 2006

Efficient representations and solutions for large structured decision problems with continuous and discrete variables are among the important challenges faced by the designers of automated decision support systems. In this work, we describe a novel hybrid factored Markov decision process (MDP) model that allows for a compact representation of these problems, and a hybrid approximate linear programming (HALP) framework that permits their efficient solutions. The central idea of HALP is to approximate the optimal value function of an MDP by a linear combination of basis functions and optimize its weights by linear programming. We study both theoretical and practical aspects of this approach, and demonstrate its scale-up potential on several hybrid optimization problems.

TABLE OF CONTENTS

LIST OF FIGURES

PREFACE

First and foremost, I would like to thank my graduate advisor, Milos Hauskrecht, for encouraging me to join the graduate program at the University of Pittsburgh and taking me as a student. During my first year, Milos helped me to understand the fundamentals of Bayesian networks and decision theory, which proved invaluable for my research and thesis. His machine learning classes expanded my horizon and helped me to connect seemingly unrelated ideas. This thesis benefited greatly from our frequent discussions, which were often heated but always constructive. Over the course of four years, Milos turned out to be not only a good supervisor but a friend. He always took an interest in our personal lives. His high professional standards and constant drive for improvement helped me to obtain an FAS Fellowship for the academic year 2003–04, and two Andrew Mellon Predoctoral Fellowships for the academic years 2004–06. These scholarships from the University of Pittsburgh gave me the freedom to pursue independent research ideas early in my career. Thank you, Milos!

Many thanks to my committee members for their valuable time and helpful suggestions. During my first year, I took Gregory Cooper's class on Bayesian networks and he sparked my interest in this topic. My subsequent meetings with Greg typically showed how little I knew and how much harder I needed to study. I am thankful for meeting Carlos Guestrin at a conference in winter 2003. Our early collaboration laid a foundation for this thesis. For me, his hard work and justified success served as a prime example of how good academic research should be conducted. Finally, I engaged a lot in discussions with Marek Druzdzel's research group. A part of this thesis is motivated by an earlier work of one of his former students, Changhe Yuan.

Many thanks to the professors and researchers who helped to shape my research identity in the past four years: Roger Day gave me a solid background in empirical methods for solving Bayesian problems; Kirk Pruhs' algorithms course was one of the most intensive undertakings in my career; and Rebecca Hwa, Diane Litman, and Janyce Wiebe introduced me to the computational problems

of natural language processing. My industrial experience at Intel in the past three summers would not have been as enjoyable had I not met Denver Dash, John Mark Agosta, Georgios Theocharous, and Eve Schooler. During these internships, I learned the painful truth of how complex real-world problems are usually solved efficiently by simple machine learning techniques. On the other hand, complex machine learning approaches often fail to solve even simple problems. Finally, my life in graduate school would be much more difficult without our departmental secretary, Keena Walker. She has mastered the art of satisfying University requirements, which allowed me to focus entirely on my research.

Last but not least, I want to thank my parents, brother, and friends for their consistent support. My family, and lifelong friends in Slovakia and Czech Republic, are the main reason for my regular visits to Bratislava and Prague. In the past four years, I have also made a lot of good friends in the Pittsburgh area. I thank them for many unforgettable memories.

Parts of this thesis have been presented at the following conferences: NIPS 2003 [43], ICAPS 2004 [56], UAI 2004 [35], IJCAI 2005 [57], ICAPS 2006 [59], and AAAI 2006 [58]. A significant portion of Chapter 3 has been previously published in the Journal of Artificial Intelligence Research [60]. The work was partially supported by two National Science Foundation grants: CMS-0416754 and ANI-0325353.

1.0 INTRODUCTION

Humans have been always fascinated by the question whether our intelligent behavior can be imitated by a machine. Artificial intelligence (AI) is a branch of computer science that studies building of these intelligent systems. This thesis is a contribution to this field. At the highest level, we study a traditional framework for sequential decision making and generalize it to allow building of rationally behaving systems under uncertainty in real-world domains.

A rationally behaving system and its stochastic behavior can be usually formulated as a Markov decision process (MDP). An MDP is a controlled stochastic process whose dynamics is represented by state transitions. Objectives of the control are modeled by rewards (or costs), which are assigned to state-action configurations. In the simplest form, the states and actions of an MDP are assumed to be discrete and unstructured. These models can be solved efficiently by dynamic programming methods [6, 82, 10].

Unfortunately, textbook models rarely meet the practice and its needs. First, real-world decision problems are naturally described in a factored form and may involve a combination of discrete and continuous variables. Second, there are no guarantees that compact forms of the optimal value function or policy for these problems exist. Therefore, hybrid optimization problems are typically discretized and solved approximately by the methods for discrete-state MDPs. The contribution of this work is a principled, sound, and efficient approach to solving large-scale factored MDPs that avoids this discretization step.

The presented framework is based on approximate linear programming (ALP) [86], which has been already applied to solve decision problems with discrete state and action variables efficiently [85, 23, 39]. These applications include context-specific planning [40], multiagent planning [38], relational MDPs [36], and first-order MDPs [84]. In this work, we show how to adapt ALP to solve large-scale factored MDPs in hybrid state and action spaces.

Our approach combines factored MDP representations (Section 2.3 and Chapter 3) and optimization techniques for solving large-scale structured linear programs (Section 3.3). This leads to various benefits. First, the quality and complexity of value function approximations is controlled by using basis functions (Section 2.3.2). Therefore, we can prevent an exponential blowup in the complexity of computations when other techniques cannot. Second, we always guarantee that HALP returns some solution. Its quality naturally depends on the choice of basis function. As analyzed in Section 3.2.1, if these are selected appropriately, we achieve a close approximation to the optimal value function V^*. Third, a well-chosen class of basis functions yields closed-form solutions to the backprojections of our value functions (Section 3.2.2). This step seems critical for solving hybrid optimization problems more efficiently. Finally, solving hybrid factored MDPs reduces to building and satisfying relaxed formulations of the original problem (Section 3.3). These formulations can be solved efficiently by the cutting plane method, which has been studied extensively in operations research and applied mathematics.

This thesis is organized as follows. Chapter 2 introduces the fundamentals of Markov decision processes and reviews state-of-the-art methods for solving large-scale factored MDPs with discrete and continuous variables. In Chapter 3, we present a novel hybrid factored MDP model that allows for a compact representation of these problems, and a new hybrid approximate linear programming (HALP) framework that permits their efficient solutions. In Chapter 4, we extend the concepts from Chapter 3 and show how to solve a very general class of hybrid factored MDPs efficiently. A novel approach to learning basis functions in hybrid state and action spaces is demonstrated in Chapter 5. Finally, the results of the thesis are summarized in Chapter 6. For better readability, our proofs are deferred to the appendix.

2.0 BACKGROUND AND RELATED WORK

This chapter introduces background concepts and related work. First, we review the fundamentals of Markov decision processes (Section 2.1). Second, we discuss dynamic programming techniques for solving these problems efficiently (Section 2.2). Unfortunately, neither the flat MDP representation nor its efficient solutions scale up to real-world problems. To provide a sufficient background for our work, we review state-of-the-art methods for solving large-scale MDPs (Section 2.3). Special attention is paid to the scalability of these methods.

The following notation is used throughout the work. Sets and their members are represented by capital and small italic letters as $\mathcal{S}$ and s, respectively. Sets of variables, their subsets, and members of these sets are denoted by capital letters as $\mathbf{X}$, $\mathbf{X}_i$, and X_i. In general, corresponding small letters represent value assignments to these objects. The subscripted indices D and C denote the discrete and continuous variables in a variable set and its value assignment. The function $\mathrm{Dom}(\cdot)$ computes the domain of a variable or the domain of a function. The function $\mathsf{Par}(\cdot)$ returns the parent set of a variable in a graphical model [47, 25].

2.1 MARKOV DECISION PROCESSES

Markov decision processes [6] provide an elegant mathematical framework for modeling and solving sequential decision problems under uncertainty. Formally, a *finite-state Markov decision process (MDP)* is defined as a 4-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R)$, where $\mathcal{S} = \{s_1, \ldots, s_n\}$ is a set of states, $\mathcal{A} = \{a_1, \ldots, a_m\}$ is a set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a stochastic transition function of state dynamics conditioned on the preceding state and action, and $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a reward function assigning immediate payoffs to state-action configurations. Without loss of generality, the

reward function is assumed to be nonnegative and bounded from above by a constant $R_{\max}$ [82]. Moreover, we assume that the transition and reward models are stationary and known a priori.

Once a decision problem is formulated as an MDP, the goal is to find a policy that maximizes some objective function. Formally, a *policy* π is a mapping $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ from the states s to a distribution over the actions a that determines the probability of taking actions. The policy is *deterministic* if this mapping is a deterministic function $\pi : \mathcal{S} \to \mathcal{A}$.

In this work, we assume that Markov decision processes have infinite horizons and their future rewards are discounted exponentially. Therefore, the quality of policies is measured by the *infinite horizon discounted reward*:

$$\mathrm{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s^{(t)}, \pi(s^{(t)})) \middle| s^{(0)} \sim \varphi \right], \tag{2.1}$$

where $\gamma \in [0, 1)$ is a *discount factor*, $s^{(t)}$ is the state at the time step t, and the expectation is taken with respect to state-action trajectories that start in the states $s^{(0)}$ and follow the policy π thereafter. The initial states $s^{(0)}$ are chosen according to a distribution φ. This optimality criterion guarantees that there always exists an *optimal policy* π^* which is stationary and deterministic [82].

Value functions express the preference for occupying states s and their computation is often an intermediate step towards deriving a policy. In infinite-horizon MDPs, the value of being in a state s under a policy π:

$$V^\pi(s) = \mathrm{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s^{(t)}, \pi(s^{(t)})) \middle| s^{(0)} = s \right] \tag{2.2}$$

equals to the expected long-term return accrued by starting in this state and following the policy π thereafter. The value function V^π satisfies the recursive relationship:

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) V^\pi(s'). \tag{2.3}$$

It follows that V^π is a solution to a system of $|\mathcal{S}|$ linear equations (Equation 2.3), one for each state s. The *optimal value function* V^* yields the highest long-term return (Equation 2.2) in every state

s. In other words, the inequality $V^* \geq V^\pi$ is satisfied for any value function V^π. The optimal value function is also a fixed point of the Bellman equation [6]:

$$V^*(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^*(s') \right]. \tag{2.4}$$

The *optimal policy* π^* can be defined greedily with respect to V^* as:

$$\pi^*(s) = \arg\max_a \left[R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^*(s') \right]. \tag{2.5}$$

In addition to the value function, we define a *Q-function*:

$$Q^\pi(s,a) = R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^\pi(s'), \tag{2.6}$$

which represents the expected long-term return accrued by starting in the state s, taking an action a, and following the policy π thereafter. This definition permits the rewriting of Equations 2.4 and 2.5 as:

$$V^*(s) = \max_a Q^*(s,a) \tag{2.7}$$

$$\pi^*(s) = \arg\max_a Q^*(s,a) \tag{2.8}$$

Finally, we introduce two backup operators:

$$\mathcal{T}^\pi V(s) = R(s,\pi(s)) + \gamma \sum_{s'} P(s' \mid s,\pi(s)) V(s') \tag{2.9}$$

$$\mathcal{T}^* V(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V(s') \right]. \tag{2.10}$$

These allow for a compact formulation of Equations 2.3 and 2.4 as:

$$V^\pi = \mathcal{T}^\pi V^\pi \tag{2.11}$$

$$V^* = \mathcal{T}^* V^*. \tag{2.12}$$

The operator $\mathcal{T}^*$ is known as the *Bellman operator* and the function V^* is its fixed point. The value function V^π is a fixed point of the backup operator $\mathcal{T}^\pi$.

2.2 SOLVING MARKOV DECISION PROCESSES

Markov decision processes can be solved by various exact and approximate methods [82, 10, 91]. In this section, we introduce the most fundamental dynamic programming (DP) techniques, which establish a starting point for later discussed approximations. These approaches are mathematically well developed. However, they require a complete model of the environment, which is often viewed as their limitation. Nevertheless, there exist numerous large real-world problems where the dynamics is known but the long-term optimal control is hard to determine. For a comprehensive overview of other approaches to solving MDPs, refer to the book by Sutton and Barto [91].

2.2.1 Value Iteration

Value iteration (Figure 1) builds an approximation to the optimal value function V^* by the iterative application of the Bellman operator $\mathcal{T}^*$. The backups $V^{(0)}, V^{(1)} = \mathcal{T}^* V^{(0)}, V^{(2)} = \mathcal{T}^* V^{(1)}, \dots$ are guaranteed to converge to V^* independently of their starting point $V^{(0)}$. The convergence is due to the properties of the Bellman operator $\mathcal{T}^*$, which is a contraction mapping with a contraction factor $\gamma \in [0, 1)$. Based on this fact, Littman [63] proved that value iteration converges to a greedy policy:

$$\pi^{(t)}(s) = \arg\max_a \left[R(s,a) + \gamma \sum_{s'} P(s' \mid s, a) V^{\pi^{(t)}}(s') \right] \tag{2.13}$$

that yields $V^{\pi^{(t)}} = V^*$ in a polynomial number of steps in the number of states $|\mathcal{S}|$, the number of actions $|\mathcal{A}|$, $\log \max_{s,a} |R(s,a)|$, and $1/(1-\gamma)$. The computational complexity of performing the backup $V^{(t+1)} = \mathcal{T}^* V^{(t)}$ is $O(|\mathcal{S}|^2 |\mathcal{A}|)$.

Instead of applying this bound, value iteration (Figure 1) is often terminated when the *Bellman error* $\|V - \mathcal{T}^* V\|_\infty = \max_s |V(s) - \mathcal{T}^* V(s)|$ is small. The loss of acting greedily with respect to an imperfect value function $V^{(t)}$ can be bounded as follows.

Proposition 1 (Williams and Baird III [100]) *The loss of acting greedily with respect to a value function V:*

$$u(s) = \arg\max_a \left[R(s,a) + \gamma \sum_{s'} P(s' \mid s, a) V(s') \right]$$

Inputs:
a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R)$
an initial value function $V^{(0)}$
Bellman error ε

Algorithm:
$t = -1$
repeat
$t = t + 1$
for every state $s \in \mathcal{S}$
$V^{(t+1)}(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P(s' \mid s, a) V^{(t)}(s')\right]$
until $\left(\left\|V^{(t)} - V^{(t+1)}\right\|_\infty < \varepsilon\right)$

Outputs:
a value function $V^{(t)}$

Figure 1: Pseudo-code implementation of value iteration.

can be bounded as:

$$\left\|V^* - V^u\right\|_\infty \leq \frac{\gamma \left\|V - \mathcal{T}^* V\right\|_\infty}{1 - \gamma},$$

where V^u is a value function generated by following the greedy policy u.

2.2.2 Policy Iteration

Policy iteration (Figure 2) builds the optimal policy π^* in a sequence of alternating policy evaluation and policy improvement steps [6, 46]. The *policy evaluation* step computes the value function $V^{\pi^{(t)}}$ for a fixed policy $\pi^{(t)}$. This step can be easily implemented by solving a system of $|\mathcal{S}|$ linear equations:

$$V^{\pi^{(t)}}(s) = R(s, \pi^{(t)}(s)) + \gamma \sum_{s'} P(s' \mid s, \pi^{(t)}(s)) V^{\pi^{(t)}}(s') \quad \forall\, s \in \mathcal{S}, \tag{2.14}$$

where $V^{\pi^{(t)}}$ denotes the variables in the LP. Subsequently, the *policy improvement* step generates a greedy policy $\pi^{(t+1)}$ with respect to the value function $V^{\pi^{(t)}}$, which is no worse than the existing

Inputs:

a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R)$

an initial policy $\pi^{(0)}$

Algorithm:

$t = -1$

repeat

$t = t + 1$

compute $V^{\pi^{(t)}}$ by solving the system of $|\mathcal{S}|$ linear equations:

$V^{\pi^{(t)}}(s) = R(s, \pi^{(t)}(s)) + \gamma \sum_{s'} P(s' \mid s, \pi^{(t)}(s)) V^{\pi^{(t)}}(s') \quad \forall\, s \in \mathcal{S}$

for every state $s \in \mathcal{S}$

$\pi^{(t+1)}(s) = \arg\max_a \left[R(s, a) + \gamma \sum_{s'} P(s' \mid s, a) V^{\pi^{(t)}}(s') \right]$

until $\pi^{(t)}$ equals to $\pi^{(t+1)}$

Outputs:

a policy $\pi^{(t)}$

Figure 2: Pseudo-code implementation of policy iteration.

solution $\pi^{(t)}$. These two steps are alternated until the two successive policies $\pi^{(t)}$ and $\pi^{(t+1)}$ are the same. This stopping criterion guarantees that $\pi^{(t)} = \pi^*$.

Every MDP induces $|\mathcal{A}|^{|\mathcal{S}|}$ different deterministic policies ($|\mathcal{A}|$ choices for each state s). Therefore, the length of the sequence $V^{\pi^{(0)}}, V^{\pi^{(1)}}, V^{\pi^{(2)}}, \ldots$ is exponential in the number of states $|\mathcal{S}|$ in the worst case. A tighter bound, which shows that policy iteration always converges to the optimal value function V^* in no more steps than value iteration, is discussed in the work of Puterman [82]. In practice, policy iteration is typically faster than value iteration or linear programming. Its faster convergence rate is partially offset by a higher computational cost of $O(|\mathcal{S}|^3 + |\mathcal{S}|^2 |\mathcal{A}|)$ for a single iteration.

2.2.3 Linear Programming

The optimal value function V^* is known to be a fixed point of the Bellman equation (Equation 2.4). Interestingly, a similar system of $|\mathcal{S}|$ equations:

$$V(s) \geq \max_a \left[R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V(s') \right] \quad \forall\, s \in \mathcal{S}, \tag{2.15}$$

which are nonlinear in V, can be rewritten as $|\mathcal{S}|\,|\mathcal{A}|$ linear equations:

$$V(s) \geq R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V(s') \quad \forall\, s \in \mathcal{S}, a \in \mathcal{A}. \tag{2.16}$$

Since the Bellman operator $\mathcal{T}^*$ is a monotonic contraction mapping, we conclude that for any value function V, $V \geq \mathcal{T}^* V$ implies $V \geq \mathcal{T}^* V \geq \cdots \geq V^*$. Therefore, if a value function V satisfies the constraints imposed by Equation 2.16, it constitutes an upper bound on the optimal value function V^*. Furthermore, V^* is a pointwise minimum of these upper bounds. As a result, the optimal value function V^* must be a solution to the *linear programming (LP)* formulation [67]:

$$\begin{aligned} \text{minimize} \quad & \sum_s \psi(s) V(s) \\ \text{subject to:} \quad & V(s) \geq R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V(s') \quad \forall\, s \in \mathcal{S}, a \in \mathcal{A}; \end{aligned} \tag{2.17}$$

where $V(s)$ denotes the variables in the LP, one for each state s, and $\psi(s) > 0$ is a strictly positive weighting on the state space $\mathcal{S}$. The number of constraints is equal to the cardinality of the cross product of the state and action spaces $|\mathcal{S} \times \mathcal{A}|$.

Based on the duality theorem [11], the optimal value function V^* is also a solution to the *dual linear programming* formulation:

$$\begin{aligned} \text{maximize} \quad & \sum_{s,a} R(s,a) \phi(s,a) \\ \text{subject to:} \quad & \sum_a \phi(s',a) = 1 + \gamma \sum_{s,a} P(s' \mid s,a) \phi(s,a) \quad \forall\, s' \in \mathcal{S}; \end{aligned} \tag{2.18}$$

where the initial roles of the variables and constraints are reversed. This formulation is interpreted as maximizing the total reward $\sum_{s,a} R(s,a)\phi(s,a)$ of the policy flow $\phi(s,a)$. The consistency of the flow is assured by the constraints in the dual LP. For every state s', the flow that exists the state

$\sum_a \phi(s', a)$ is equal to its discounted inflow $\sum_{s,a} P(s' \mid s, a)\phi(s, a)$ increased by a constant flow of 1. Note that both the primal and dual formulations (2.17 and 2.18) can be solved by any linear programming solver.

Linear programming and its efficient solutions have been studied extensively in applied mathematics and operations research [11]. The simplex algorithm is a common way of solving LPs. Its worst-case time complexity is exponential in the number of variables. The ellipsoid method [52] offers polynomial time guarantees but it is impractical for solving LPs of even moderate size.

The primal LP formulation (2.17) can be solved compactly by the *cutting plane method* [11] if its objective function and constraint space are structured. Briefly, this method searches for violated constraints in relaxed formulations of the original LP. In every step, we start with a relaxed solution $V^{(t)}$, find a violated constraint given $V^{(t)}$, add it to the LP, and resolve for a new vector $V^{(t+1)}$. The method is iterated until no violated constraint is found, so that $V^{(t)}$ is an optimal solution to the LP. This technique has a potential to solve large structured linear programs if we can identify violated constraints efficiently [11]. The violated constraint and the method that found it are often referred to as a *separating hyperplane* and a *separation oracle*, respectively.

Delayed column generation [11] is based on a similar idea as the cutting plane method, which is applied to the column space of variables instead of the row space of constraints. Dantzig-Wolfe and Bender's decompositions reflect the structure in the constraint space and are typically used for solving large structured linear programs.

2.3 SOLVING LARGE-SCALE MARKOV DECISION PROCESSES

In Sections 2.1 and 2.2, we introduced the standard textbook representation for MDPs and the most fundamental dynamic programming methods for solving these problems efficiently. Unfortunately, neither this representation nor the DP methods scale up to real-world problems. First, the standard MDP formulation does not allow for representing the factored nature of the state space and related structure in the transition function, which is typical for many real-world problems. Second, even if this structure can be described compactly by state variables and their dynamics (Section 2.3.1), the time complexity of the exact DP methods remains polynomial in the size of the state space $\mathbf{X}$ [82].

Since the state space $\mathbf{X}$ is exponential in the number of state variables if the variables are discrete, the DP methods are not suitable for solving large factored MDPs. Finally, the DP methods assume a finite support for the optimal value function or policy, which may not exist if continuous variables are present. As a result, any feasible approach to solving large-scale real-world MDPs is likely to be approximate.

The goal of this section is to discuss these approaches and provide sufficient background for our work. First, we review discrete-state factored MDPs [14], which allow for a compact representation of the structure in discrete-state spaces (Section 2.3.1). This representation is extended to problems with discrete and continuous state and action variables in Sections 3.1 and 4.1. Second, we discuss standard methods for solving factored MDPs with continuous variables: parametric value function (Section 2.3.2), parametric policy (Section 2.3.3), and grid (Section 2.3.4) approximations. Special attention is paid to the approximate linear programming (Section 2.3.2.3), which has a potential to solve large-scale problems [85, 23, 39] but has not been applied to solve continuous-state or hybrid factored MDPs yet. This extension is presented in Sections 3.2 and 4.2.

2.3.1 Factored Transition and Reward Models

Real-world decision problems are often described in a factored form. Discrete-state factored MDPs [14] allow for a compact representation of this structure.

A *discrete-state factored MDP* [14] is a 4-tuple $\mathcal{M} = (\mathbf{X}, \mathcal{A}, P, R)$, where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a state space described by state variables, $\mathcal{A} = \{a_1, \ldots, a_m\}$ is a set of actions[1], $P(\mathbf{X}' \mid \mathbf{X}, \mathcal{A})$ is a stochastic transition model of state dynamics conditioned on the preceding state and action, and R is a reward function assigning immediate payoffs to state-action configurations. The state of the system is completely observed and represented by a vector of value assignments $\mathbf{x} = (x_1, \ldots, x_n)$. We assume that the values of every state variable X_i are restricted to a finite domain $\mathrm{Dom}(X_i)$.

Transition model: The transition model is characterized by the conditional probability distribution $P(\mathbf{X}' \mid \mathbf{X}, \mathcal{A})$, where $\mathbf{X}$ and $\mathbf{X}'$ denote the state variables at two successive time steps t and $t+1$.

[1] For simplicity of exposition, we discuss a simpler MDP model, which assumes a single action variable $\mathcal{A}$ instead of the factored action space $\mathbf{A} = \{A_1, \ldots, A_m\}$. Our conclusions in Sections 2.3.1 and 2.3.2.3 extend to MDPs with factored action spaces [38].

Since the tabular representation of $P(\mathbf{X}' \mid \mathbf{X}, \mathcal{A})$ is infeasible, we assume that the transition model factors along $\mathbf{X}'$ as:

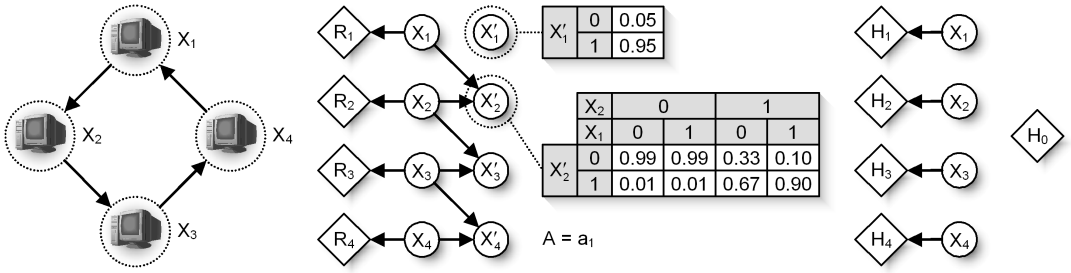
$$P(\mathbf{X}' \mid \mathbf{X}, a) = \prod_{i=1}^{n} P(X_i' \mid \mathsf{Par}(X_i'), a) \tag{2.19}$$

and can be represented compactly by a *dynamic Bayesian network (DBN)* [25]. This representation captures independencies among the state variables $\mathbf{X}$ and $\mathbf{X}'$ given an action a. One-step dynamics of every state variable X_i is modeled by its conditional probability distribution $P(X_i' \mid \mathsf{Par}(X_i'), a)$, where $\mathsf{Par}(X_i') \subseteq \mathbf{X}$ is the parent set of X_i'. Typically, this parent set is a subset of state variables which simplifies the parameterization of the model.

Reward model: The reward model is factored similarly to the transition model. In particular, the reward function $R(\mathbf{x}, a) = \sum_j R_j(\mathbf{x}_j, a)$ is an additive function of local reward functions defined on the subsets $\mathbf{X}_j$ and $\mathcal{A}$. In graphical models, the local functions can be described compactly by reward nodes R_j, which are conditioned on their parent sets $\mathsf{Par}(R_j) = \mathbf{X}_j \cup \mathcal{A}$. To allow for this representation, we formally extend our DBN into an influence diagram [47].

Example 1 (Guestrin *et al.* [37]) *To illustrate the concept of a factored MDP, we consider a network administration problem, in which the computers are unreliable and fail. The failures of these computers propagate through network connections to the whole network. For instance, if the server X_1 (Figure 3a) is down, the chance that the neighboring computer X_2 crashes increases. The administrator can prevent the propagation of the failures by rebooting computers that have already crashed.*

This network administration problem can be formulated as a factored MDP. The state of the network is completely observed and represented by n binary variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, where the variable X_i denotes the state of the i-th computer: 0 (being down) or 1 (running). At each time step, the administrator selects an action from the set $\mathcal{A} = \{a_1, \ldots, a_{n+1}\}$. The action a_i ($i \leq n$) corresponds to rebooting the i-th computer. The last action a_{n+1} is dummy. The transition function reflects the propagation of failures in the network and can be encoded locally by conditioning on



(a) **(b)** **(c)**

Figure 3: **a.** Four computers in a ring topology. The direction of propagating failures is represented by arrows. **b.** A graphical representation of factored transition and reward models after taking an action a_1 in the 4-ring topology. The future state of the server X_1' is independent of the rest of the network because the server is rebooted. Reward nodes R_1 and R_j ($j \geq 2$) denote the components $2x_1$ and x_j ($j \geq 2$) of the reward model. **c.** A graphical representation of the linear value function approximation $V^{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{i=1}^{4} w_i x_i$ in the 4-ring topology. Reward nodes H_0 and H_i ($i \geq 1$) denote the value function components w_0 and $w_i x_i$ ($i \geq 1$).

the parent set of every computer. A natural metric for evaluating the performance of an administrator is the total number of running computers. This metric factors along the computer states x_i and can be represented compactly by an additive reward function:

$$R(\mathbf{x}, a) = 2x_1 + \sum_{j=2}^{n} x_j.$$

The weighting of states establishes our preferences for maintaining the server X_1 and workstations $X_2, \ldots, X_n$. An example of transition and reward functions after taking an action a_1 in the 4-ring topology (Figure 3a) is given in Figure 3b.

Discrete-state factored MDPs with finite horizons can be solved exactly by structured DP methods [12]. These approaches [14, 13] rely on the structured representations of value functions, which are updated iteratively by structured Bellman backups. The value function is represented by a decision tree [14, 13] or by an algebraic decision diagram [45]. Unfortunately, the structure in the function is often lost after few backups. Therefore, structured dynamic programming is unsuitable for solving infinite-horizon MDPs. To address this problem, Boutilier and Dearden [13] proposed approximate versions of the structured DP techniques.

Unfortunately, the cost of solving discrete-state factored MDPs optimally is exponential in the number of state variables in the worst case. Therefore, to solve these problems efficiently, we have to trade the quality of the solutions for their computational cost. A common approach is to refrain from computing the optimal value function or policy, and search for their approximations in some parametric class (Sections 2.3.2 and 2.3.3). These ideas can be applied to large problems with both discrete and continuous variables. Moreover, continuous-state factored MDPs can be approximated by discrete models (Section 2.3.4), which can be solved by the standard DP methods (Section 2.2).

2.3.2 Parametric Value Function Approximations

A popular way of solving large factored MDPs is the approximation of the optimal value function V^* by some parametric function V^λ [10, 91, 32]. The parameters λ are fit to minimize some notion of an error between the value functions V^* and V^λ. In general, this optimization step can be viewed as approximate value iteration (Section 2.3.2.1), approximate policy iteration (Section 2.3.2.2), or approximate linear programming (Section 2.3.2.3). These three methods generalize the exact DP

techniques discussed in Section 2.2. After the optimization is completed, the policy can be derived greedily with respect to the value function V^{λ}.

The form of the function V^{λ} is often domain specific and chosen in advance. The most popular approximators are neural networks [10] and *linear value function approximation* [7, 98]:

$$V^{\mathbf{w}}(\mathbf{x}) = \sum_{i} w_i f_i(\mathbf{x}). \tag{2.20}$$

This commonly employed approximation restricts the value function $V^{\mathbf{w}}$ to the linear combination of $|\mathbf{w}|$ basis functions $f_i(\mathbf{x})$, where $\mathbf{w}$ is a vector of optimized weights.[2] Each basis function can be defined over the complete state space $\mathbf{X}$, but often is limited to a small subset of state variables $\mathbf{X}_i$ [7, 54]. The role of basis functions is very similar to features in machine learning. They are usually provided by domain experts, although there is a growing body of work on learning basis functions automatically (Section 2.3.2.4).

Example 2 *To demonstrate the concept of the linear value function model, we consider the network administration problem (Example 1) and assume a low chance of a single computer failing. Then the value function in Figure 3c is sufficient to derive a close-to-optimal policy on the 4-ring topology (Figure 3a) because the indicator functions $f_i(\mathbf{x}) = x_i$ can capture changes in the states of individual computers. For instance, if the computer X_i fails, the policy:*

$$u(\mathbf{x}) = \arg\max_{a} \left[R(\mathbf{x}, a) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, a) V^{\mathbf{w}}(\mathbf{x}') \right]$$

immediately leads to rebooting it. If the failure has already propagated to the computer X_{i+1}, the policy recovers it in the next step. This procedure is repeated until the spread of the initial failure is stopped.

Inputs:

a hybrid factored MDP $\mathcal{M} = (\mathbf{X}, \mathbf{A}, P, R)$
basis functions $f_0(\mathbf{x}), f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots$
initial basis function weights $\mathbf{w}^{(0)}$
a set of states $G = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\right\}$

Algorithm:

$t = 0$
while a stopping criterion is not met
  for every state $\mathbf{x}^{(j)}$
    for every basis function $f_i(\mathbf{x})$
      $\mathbf{X}_{ji} = f_i(\mathbf{x}^{(j)})$
    $\mathbf{y}_j = \max_{\mathbf{a}} \left[R(\mathbf{x}^{(j)}, \mathbf{a}) + \gamma \mathrm{E}_{P(\mathbf{x}'|\mathbf{x}^{(j)}, \mathbf{a})} \left[V^{\mathbf{w}^{(t)}}(\mathbf{x}')\right]\right]$
  $\mathbf{w}^{(t+1)} = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathbf{y}$
  $t = t + 1$

Outputs:

basis function weights $\mathbf{w}^{(t)}$

Figure 4: Pseudo-code implementation of the least-squares value iteration ($\mathcal{L}_2$ VI) with the linear value function approximation (Equation 2.20). The stopping criterion is often based on the number of steps or the $\mathcal{L}_2$-norm error $\left\|V^{\mathbf{w}^{(t)}} - \mathcal{T}^* V^{\mathbf{w}^{(t)}}\right\|_2$ measured on the set of states G. Our discussion in Sections 3.2.2 and 3.3 provides a recipe for an efficient implementation of the backup operation $\mathcal{T}^* V^{\mathbf{w}^{(t)}}(\mathbf{x}^{(j)})$ in hybrid state and action spaces.

2.3.2.1 Approximate Value Iteration

Approximate value iteration is a natural generalization of value iteration (Section 2.2.1), where the tabular value function $V(\mathbf{x})$ is substituted for some parametric function $V^{\lambda}(\mathbf{x})$ [10]. This approach can be traced back to Bellman *et al.* [7]. Comparing to value iteration, note that the full DP backup $V^{\lambda^{(t+1)}} = \mathcal{T}^* V^{\lambda^{(t)}}$ cannot be typically performed. Therefore, approximate value iteration methods usually substitute this step by optimizing the parameters $\lambda^{(t+1)}$ to minimize some distance between the new parametric value function $V^{\lambda^{(t+1)}}$ and the backup $\mathcal{T}^* V^{\lambda^{(t)}}$ of the latest approximation $V^{\lambda^{(t)}}$. The minimization of the $\mathcal{L}_2$-norm distance with the linear value function approximation $V^{\mathbf{w}}$:

$$\sum_{\mathbf{x} \in G} \psi(\mathbf{x}) \left[V^{\mathbf{w}^{(t+1)}} - \mathcal{T}^* V^{\mathbf{w}^{(t)}} \right]^2 \tag{2.21}$$

is computationally attractive because it is identical to optimizing the least-squares error (Figure 4). The optimization is done on a set of representative states $G = \left\{ \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)} \right\}$ that are reweighted according to a state relevance function $\psi(\mathbf{x})$. Although this approach is popular [10], it can diverge in embarrassingly simple situations [16]. The reason for the divergence is that the least-squares fit can be an expansion under max-norm [32]. Alternatively, the optimization of the $\mathcal{L}_2$-norm distance can be rewritten in the form of a gradient update and it is known as *incremental value iteration* [10]. Unfortunately, this approach suffers from the same drawbacks as the iterative minimization of the least-squares error (Equation 2.21).

Parametric approximations often assume fixed value function models. However, in some cases, it is possible to derive adaptive forms of V^{λ} that combine well with the Bellman operator $\mathcal{T}^*$. For instance, Sondik [88] demonstrated that convex piecewise linear functions are sufficient to represent value functions and their DP backups in partially-observable MDPs (POMDPs) [2, 42]. Based on this idea, Feng *et al.* [28] proposed a method for solving MDPs with continuous variables. To allow for full DP backups, the value function approximation is restricted to *rectangular piecewise linear and convex (RPWLC)* functions. Further restrictions are placed on the transition and reward models of MDPs. The main advantage of this approach is its adaptivity. The major disadvantages are restrictions on solved MDPs and the complexity of RPWLC value functions, which may grow

[2] In the rest of the work, linear value function approximations are always referred to by the symbol $V^{\mathbf{w}}$. In contrast, arbitrary parametric value function approximations are denoted by the symbol V^{λ}.

exponentially in the number of backups. As a result, without further modifications, this approach is less likely to succeed in solving high-dimensional and distributed decision problems.

The work of Feng *et al.* [28] was also inspired by the experiments of Boyan and Littman [15] with a single continuous state variable. Li and Littman [62] extended these approaches by allowing piecewise constant transition functions. The final *lazy* approximation is a piecewise constant value function. By applying similar ideas, Marecki *et al.* [69] showed that phase-type transition functions over continuous state variables generate piecewise gamma value function approximations for every discrete state of an MDP. Unfortunately, similarly to the work of Feng *et al.* [28], these methods do not scale up beyond few continuous variables.

2.3.2.2 Approximate Policy Iteration

Approximate policy iteration is a common generalization of policy iteration (Section 2.2.2), where the tabular value function $V(\mathbf{x})$ is substituted for the parametric function $V^{\lambda}(\mathbf{x})$ [10]. Similarly to policy iteration, this method alternates policy evaluation and policy improvement steps. The policy evaluation step fits the parametric approximation $V^{\lambda^{(t)}}$ to the value function $V^{\pi^{(t)}}$ for a fixed policy $\pi^{(t)}$. Once the policy evaluation is completed, the policy improvement step generates a new greedy policy:

$$\pi^{(t+1)}(\mathbf{x}) = \max_{\mathbf{a}} \left[R(\mathbf{x}, \mathbf{a}) + \gamma \mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})} \left[V^{\lambda^{(t)}}(\mathbf{x}') \right] \right]. \tag{2.22}$$

The two steps are repeated until a stopping criterion is met. This iterative approach yields a close approximation to the optimal value function V^* if all policy evaluation and all policy improvement errors are close to zero [10]. Note that a close-to-zero reduction of the evaluation errors is difficult in practice since the errors depend on our ability to approximate the often unknown value function V^*. In agreement with this observation [10], approximate policy iteration tends to make rapid and fairly monotonic progress initially, but eventually gets trapped in an oscillatory pattern.

Existing approximate policy iteration techniques [10, 91, 61] differ in the way they implement the policy evaluation and policy improvement step. In the simplest case, the value function V^{π} can be estimated by a Monte Carlo simulation of the policy π from a set of representative states G, and its approximation V^{λ} is fit to minimize the $\mathcal{L}_2$-norm error $\left\| V^{\pi} - V^{\lambda} \right\|_2$ on the set G. Unfortunately,

the efficiency of this approach heavily depends on the choice of the representative states as well as the quality of our Monte Carlo estimates [10].

Temporal-difference (TD) learning [89] provides a fully incremental solution to the minimization of this $\mathcal{L}_2$-norm error. In short, when the value function is tabular, the TD(0) update for a fixed policy π is given by [91]:

$$V^{\pi}(\mathbf{x}^{(t)}) = V^{\pi}(\mathbf{x}^{(t)}) + \alpha \left[R(\mathbf{x}^{(t)}, \pi(\mathbf{x}^{(t)})) + \gamma V^{\pi}(\mathbf{x}^{(t+1)}) - V^{\pi}(\mathbf{x}^{(t)}) \right], \tag{2.23}$$

where $\alpha \in [0, 1)$ is a *learning step*, $\mathbf{x}^{(t)}$ is the state at the time step t, and the state $\mathbf{x}^{(t+1)}$ is a result of taking an action $\pi(\mathbf{x}^{(t)})$ in the state $\mathbf{x}^{(t)}$. The convergence of this update to the value function V^{π} was proved by Dayan and Sejnowski [22]. When the value function has a parametric form V^{λ}, the general gradient-descent update is equal to [91]:

$$\lambda^{(t+1)} = \lambda^{(t)} + \alpha \left[v^{(t)} - V^{\lambda^{(t)}}(\mathbf{x}^{(t)}) \right] \nabla_{\lambda^{(t)}} V^{\lambda^{(t)}}, \tag{2.24}$$

where $\lambda^{(t+1)}$ is a vector of updated parameters, $\nabla_{\lambda^{(t)}} V^{\lambda^{(t)}}$ is the vector of partial derivatives for the value function $V^{\lambda^{(t)}}$, and $v^{(t)}$ is an unbiased estimate of the value $V^{\pi}(\mathbf{x}^{(t)})$. The convergence of this update for the linear value function approximation $V^{\mathbf{w}}$ was proved by Tsitsiklis and Van Roy [97]. The $\mathcal{L}_2$-norm error $\left\| V^{\pi} - V^{\mathbf{w}^{(\infty)}} \right\|_2$ of the resulting value function $V^{\mathbf{w}^{(\infty)}}$ is bounded by a function of the $\mathcal{L}_2$-norm error $\min_{\mathbf{w}} \|V^{\pi} - V^{\mathbf{w}}\|_2$. The potential divergence of TD learning with non-linear value function approximations was pointed out by Bertsekas [8].

Successful applications of TD learning include backgammon [93, 94, 95], job-shop scheduling [103, 104], elevator dispatching [21], and acrobot [90]. The main advantage of the methods is that they are model free. In other words, interaction with the environment is necessary but its complete model in the form of an MDP is not needed. On the other hand, when the model is available, these methods fit approximate value functions much slower than the approximate dynamic programming techniques [91].

2.3.2.3 Approximate Linear Programming

A variety of methods for optimizing the linear value function approximation have been studied and analyzed (Sections 2.3.2.1 and 2.3.2.2). The *approximate linear programming (ALP)* [86] recasts this problem as a linear program:

$$\begin{aligned} \text{minimize}_{\mathbf{w}} \quad & \sum_{\mathbf{x}} \psi(\mathbf{x}) \sum_i w_i f_i(\mathbf{x}) \\ \text{subject to:} \quad & \sum_i w_i f_i(\mathbf{x}) \geq R(\mathbf{x}, a) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, a) \sum_i w_i f_i(\mathbf{x}') \quad \forall\, \mathbf{x} \in \mathbf{X}, a \in \mathcal{A}; \end{aligned} \tag{2.25}$$

where $\mathbf{w}$ represents the optimized variables in the LP, $\psi(\mathbf{x}) \geq 0$ are *state relevance weights* weighting the quality of the approximation, and $\gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, a) \sum_i w_i f_i(\mathbf{x}')$ is a discounted *backprojection* of the value function $V^{\mathbf{w}}$ (Equation 2.20). The ALP formulation can be easily derived from the standard LP formulation (2.17) by substituting $V^{\mathbf{w}}(\mathbf{x})$ for $V(\mathbf{x})$. The formulation is feasible if the set of basis functions contains a constant function $f_0(\mathbf{x}) \equiv 1$. We assume that such a function is always present. Note that the state relevance weights are no longer enforced to be strictly positive (Section 2.2.3). Comparing to the standard LP formulation (2.17), which is solved by the optimal value function V^* for arbitrary weights $\psi(s) > 0$, a solution $\tilde{\mathbf{w}}$ to the ALP formulation depends on the weights $\psi(\mathbf{x})$. In short, the higher the weights, the higher the quality of the approximation $V^{\tilde{\mathbf{w}}}$ in a corresponding state.

Since our basis functions are often restricted to subsets of state variables (Section 2.3.2), summation terms in the ALP formulation can be computed compactly [37, 85]. For example, the order of summation in the backprojection term can be changed as $\gamma \sum_i w_i \sum_{\mathbf{x}'_i} P(\mathbf{x}'_i \mid \mathbf{x}, a) f_i(\mathbf{x}'_i)$, which allows for its aggregation in the space of $\mathbf{X}_i$ instead of $\mathbf{X}$. Similarly, a factored form of $\psi(\mathbf{x})$ yields an efficiently computable objective function [34].

The ALP formulation involves exponentially many constraints in the number of state variables $\mathbf{X}$. Fortunately, the constraints are structured. This results from combining factored transition and reward models (Section 2.3.1) with the linear value function approximation (Equation 2.20). As a consequence, the constraints can be satisfied without enumerating them exhaustively.

Example 3 *The notion of a* factored constraint space *is important for the compact satisfaction of exponentially many constraints. To illustrate this concept, let us consider the linear value function*

(Example 2) on the 4-ring network administration problem (Example 1). Intuitively, by combining the graphical representations of $P(\mathbf{x}' \mid \mathbf{x}, a_1)$, $R(\mathbf{x}, a_1)$ *(Figure 3b), and* $V^{\mathbf{w}}(\mathbf{x})$ *(Figure 3c), we obtain a factored model of constraint violations:*

$$\begin{aligned}
\tau^{\mathbf{w}}(\mathbf{x}, a_1) &= V^{\mathbf{w}}(\mathbf{x}) - \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, a_1) V^{\mathbf{w}}(\mathbf{x}') - R(\mathbf{x}, a_1) \\
&= \sum_i w_i f_i(\mathbf{x}) - \gamma \sum_i w_i \sum_{\mathbf{x}'_i} P(\mathbf{x}'_i \mid \mathbf{x}, a_1) f_i(\mathbf{x}'_i) - R(\mathbf{x}, a_1) \\
&= w_0 + \sum_{i=1}^{4} w_i x_i - \gamma w_0 - \gamma w_1 P(x'_1 = 1 \mid a_1) - \\
&\quad \gamma \sum_{i=2}^{4} w_i P(x'_i = 1 \mid x_i, x_{i-1}, a_1) - 2x_1 - \sum_{j=2}^{4} x_j .
\end{aligned}$$

for an arbitrary solution $\mathbf{w}$ *(Figure 5a). Note that this cost function:*

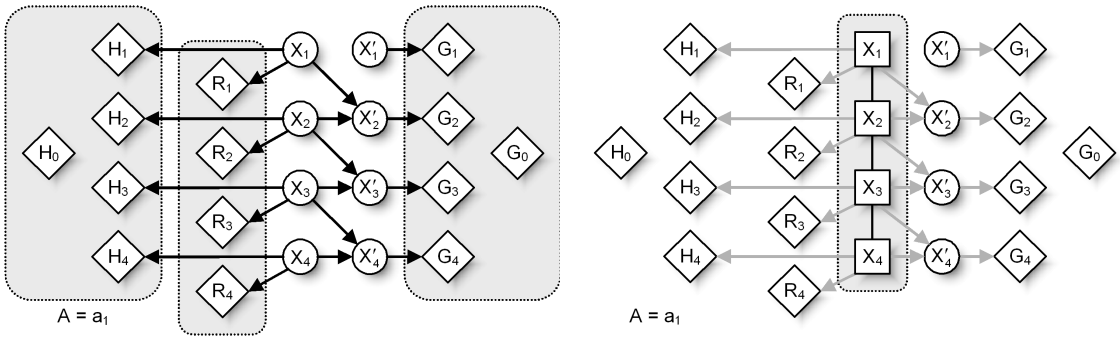
$$\tau^{\mathbf{w}}(\mathbf{x}, a_1) = \phi^{\mathbf{w}} + \sum_{i=1}^{4} \phi^{\mathbf{w}}(x_i) + \sum_{i=2}^{4} \phi^{\mathbf{w}}(x_i, x_{i-1})$$

is a linear combination of a constant $\phi^{\mathbf{w}}$ *in* $\mathbf{x}$*, and univariate and bivariate functions* $\phi^{\mathbf{w}}(x_i)$ *and* $\phi^{\mathbf{w}}(x_i, x_{i-1})$*. It can be described compactly by a* cost network *[37], which is an undirected graph over a set of variables* $\mathbf{X}$*. Two nodes in the graph are connected if any of the cost terms depends on both variables. Therefore, the cost network corresponding to the function* $\tau^{\mathbf{w}}(\mathbf{x}, a_1)$ *must contain edges* $X_1 - X_2$*,* $X_2 - X_3$*, and* $X_3 - X_4$ *(Figure 5b).*

Savings achieved by the compact representation of constraints are related to the efficiency of computing $\arg\min_{\mathbf{x}} \tau^{\mathbf{w}}(\mathbf{x}, a_1)$ *[34]. This computation can be done by variable elimination and its complexity increases exponentially in the width of the* tree decomposition *of the cost network. The smallest width of all tree decompositions is referred to as* treewidth.

Inspired by the factorization, Guestrin *et al.* [37] proposed a variable-elimination method [26] that rewrites the constraint space in ALP compactly. Schuurmans and Patrascu [85] applied the cutting plane method to build this compact representation incrementally. This method iteratively searches for the most violated constraint:

$$\arg\min_{\mathbf{x}, a} \left[\sum_i w_i^{(t)} \left[f_i(\mathbf{x}_i) - \gamma \sum_{\mathbf{x}'_i} P(\mathbf{x}'_i \mid \mathbf{x}, a) f_i(\mathbf{x}'_i) \right] - R(\mathbf{x}, a) \right] \tag{2.26}$$



(a) **(b)**

Figure 5: **a.** A graphical representation of combining the linear value function $V^{\mathbf{w}}$ (Figure 3c) with factored transition and reward models (Figure 3b). Reward nodes G_0 and G_i ($i \geq 1$) represent the discounted backprojection terms $-\gamma w_0$ and $-\gamma w_i x'_i$ ($i \geq 1$). Gray regions are the cost components of the constraint space. **b.** A cost network corresponding to our constraint space (Figure 5a). The network captures pairwise dependencies $X_1 - X_2$, $X_2 - X_3$, and $X_3 - X_4$. The treewidth of the cost network is 1.

with respect to the solution $\mathbf{w}^{(t)}$ of a relaxed ALP. The constraint is added to the LP, which is then resolved for a new solution $\mathbf{w}^{(t+1)}$. This procedure is iterated until no violated constraint is found, so that $\mathbf{w}^{(t)}$ is an optimal solution to the ALP.

The quality of the ALP formulation has been studied by de Farias and Van Roy [23]. Based on this work, we conclude that ALP yields a close approximation $V^{\tilde{\mathbf{w}}}$ to the optimal value function V^* if the weighted max-norm error $\|V^* - V^{\mathbf{w}}\|_{\infty,1/L}$ can be minimized. We return to this theoretical result in Section 3.2.1.

Theorem 1 (de Farias and Van Roy [23]) *Let $\tilde{\mathbf{w}}$ be an optimal solution to the ALP formulation (2.25). Then the expected error of the value function $V^{\tilde{\mathbf{w}}}$ can be bounded as:*

$$\left\|V^* - V^{\tilde{\mathbf{w}}}\right\|_{1,\psi} \leq \frac{2\psi^\intercal L}{1-\kappa} \min_{\mathbf{w}} \|V^* - V^{\mathbf{w}}\|_{\infty,1/L},$$

where $\|\cdot\|_{1,\psi}$ denotes an $\mathcal{L}_1$-norm weighted by the state relevance weights ψ, $L(\mathbf{x}) = \sum_i w_i^L f_i(\mathbf{x})$ is a Lyapunov function *such that the inequality $\kappa L(\mathbf{x}) \geq \gamma \sup_{\mathbf{a}} \mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})}[L(\mathbf{x}')]$ holds, $\kappa \in [0,1)$ is its* contraction factor, *and $\|\cdot\|_{\infty,1/L}$ is a max-norm reweighted by the reciprocal $1/L$.*

Note that the $\mathcal{L}_1$-norm distance $\left\|V^* - V^{\tilde{\mathbf{w}}}\right\|_{1,\psi}$ is equal to the expectation $\mathrm{E}_\psi\left|V^* - V^{\tilde{\mathbf{w}}}\right|$ over the state space $\mathbf{X}$ with respect to the state relevance weights ψ. Similarly to Theorem 1, we utilize the $\mathcal{L}_1$ and $\mathcal{L}_\infty$ norms in the rest of this thesis to measure the expected and worst-case errors of value functions. These norms are defined as follows.

Definition 1 *The $\mathcal{L}_1$* (Manhattan) *and $\mathcal{L}_\infty$* (infinity) norms *are defined as $\|f\|_1 = \sum_{\mathbf{x}} |f(\mathbf{x})|$ and $\|f\|_\infty = \max_{\mathbf{x}} |f(\mathbf{x})|$. If the state space $\mathbf{X}$ is represented by both discrete and continuous variables $\mathbf{X}_D$ and $\mathbf{X}_C$, the definition of the norms changes accordingly:*

$$\|f\|_1 = \sum_{\mathbf{x}_D} \int_{\mathbf{x}_C} |f(\mathbf{x})| \, \mathrm{d}\mathbf{x}_C \quad \text{and} \quad \|f\|_\infty = \sup_{\mathbf{x}} |f(\mathbf{x})| . \tag{2.27}$$

The following definitions:

$$\|f\|_{1,\psi} = \sum_{\mathbf{x}_D} \int_{\mathbf{x}_C} \psi(\mathbf{x}) \, |f(\mathbf{x})| \, \mathrm{d}\mathbf{x}_C \quad \text{and} \quad \|f\|_{\infty,\psi} = \sup_{\mathbf{x}} \psi(\mathbf{x}) \, |f(\mathbf{x})| \tag{2.28}$$

correspond to the $\mathcal{L}_1$ and $\mathcal{L}_\infty$ norms reweighted by a function $\psi(\mathbf{x})$.

2.3.2.4 Learning Basis Functions

The linear value function approximation $V^{\mathbf{w}}$ (Equation 2.20) is very flexible and popular. However, good basis functions are rarely known in advance. Therefore, automatic learning of basis functions seems critical for the practical application of this approximation to new domains. In the context of discrete-state ALP, Patrascu *et al.* [81] proposed a greedy method for learning basis functions. The method is based on the dual ALP formulation and optimizing its objective function. In Chapter 5, we generalize this work to hybrid state and action spaces.

Parallel to our work, Mahadevan *et al.* [64, 65, 66] have recently proposed a different approach to learning basis functions. Briefly, basis functions are obtained by the spectral analysis of the state space irrespective of the reward model. Prior to this analysis, the connectivity of the state space is described by a graph. In turn, the eigenvectors of the graph Laplacian establish a set of orthogonal basis functions. Since these basis functions are independent of the reward model, they can be used for solving any decision problem on the state space. We believe that the approach can be combined with our ideas (Chapter 5) and provide an informative prior for our basis functions.

2.3.3 Parametric Policy Approximations

Methods based on parametric value function approximations (Section 2.3.2) are often criticized for optimizing policies through their value functions. This may lead to suboptimal results because the distance between the value function V^* and its approximation V^λ is not always correlated with the quality of a greedy policy for the value function V^λ. In contrast, parametric policy approximations [99, 33, 55, 92, 4, 5, 68] parameterize the policy π^λ by a small set of continuous parameters λ and directly maximize its long-term return:

$$\mathrm{E}_\varphi\left[V^{\pi^\lambda}\right] = \mathrm{E}_{\pi^\lambda}\left[\sum_{t=0}^{\infty} \gamma^t R(\mathbf{x}^{(t)}, \pi^\lambda(\mathbf{x}^{(t)})) \middle| \mathbf{x}^{(0)} \sim \varphi\right], \tag{2.29}$$

where $\mathbf{x}^{(t)}$ is the state at the time step t, and the expectation is taken with respect to all state-action trajectories that start in the states $\mathbf{x}^{(0)}$ and follow the policy π^λ thereafter. The initial states $\mathbf{x}^{(0)}$ are chosen according to a distribution φ.

The most common approach to optimizing the policy π^λ is by modifying its parameters λ in the direction of the gradient $\nabla_\lambda \mathrm{E}_\varphi\left[V^{\pi^\lambda}\right]$. The evaluation of the gradient is a difficult problem because

the expectation is taken with respect to a stochastic process. Existing gradient methods either rely on auxiliary policy value models [99, 55, 92] or estimate the performance of the policy π^λ directly by Monte Carlo sampling [31, 68]. The variance of the estimates can be reduced by importance or adaptive sampling [71, 87, 1]. To avoid the stochasticity, Ng and Jordan [76] demonstrated how an efficient policy search can be always performed on a deterministic version of the original problem. The resulting PEGASUS algorithm [76] has been successfully applied in a variety of domains [75].

2.3.4 Grid-Based Approximations

Grid-based techniques [19, 83] approximate continuous-state MDPs by discrete-state MDPs. First, the initial state space $\mathbf{X}$ is transformed into a set of grid points $G = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\right\}$. These points are used to estimate the optimal value function V_G^* on the grid, which in turn approximates V^*. The backup operator on the grid is defined as [83]:

$$\mathcal{T}_G^* V(\mathbf{x}^{(i)}) = \max_{\mathbf{a}} \left[R(\mathbf{x}^{(i)}, \mathbf{a}) + \gamma \sum_{j=1}^{N} P_G(\mathbf{x}^{(j)} \mid \mathbf{x}^{(i)}, \mathbf{a}) V(\mathbf{x}^{(j)}) \right], \tag{2.30}$$

where $P_G(\mathbf{x}^{(j)} \mid \mathbf{x}^{(i)}, \mathbf{a}) = \Psi_{\mathbf{a}}^{-1}(\mathbf{x}^{(i)}) P(\mathbf{x}^{(j)} \mid \mathbf{x}^{(i)}, \mathbf{a})$ is a transition function normalized by the term $\Psi_{\mathbf{a}}(\mathbf{x}^{(i)}) = \sum_{j=1}^{N} P(\mathbf{x}^{(j)} \mid \mathbf{x}^{(i)}, a)$. Note that the operator $\mathcal{T}_G^*$ permits the computation of the value function V_G^* by standard dynamic programming techniques (Section 2.2).

Rust [83] analyzed the convergence of these methods for random and pseudo-random samples. In short, a uniform discretization of increasing precision guarantees the convergence of V_G^* to V^* but results in an exponential blowup in the state space [19]. To overcome this concern, Munos and Moore [74] proposed an algorithm for non-uniform discretization based on the Kuhn triangulation. Moreover, Munos and Moore [73] showed the superior performance of this approach on a variety of control problems, such as car on the hill [72], cart-pole balancing [3], and acrobot [90]. Ferns *et al.* [29] studied metrics for state aggregation in the context of continuous-state MDPs based on the notion of bisimulation. Trick and Zin [96] employed linear programming to solve low-dimensional growth models with continuous variables. These continuous variables were discretized on uniform and adaptively constructed grids.

3.0 SOLVING HYBRID FACTORED MDPS BY LINEAR PROGRAMMING

Discrete-state factored MDPs (Section 2.3) permit a compact representation of decision problems with discrete states. Unfortunately, real-world domains usually involve continuous quantities, such as temperature or pressure. A sufficient discretization of these quantities may require hundreds of points in a single dimension, which renders the representation of our discrete-state transition model (Equation 2.19) infeasible. In addition, rough and uninformative discretization impacts the quality of policies. Therefore, we want to avoid discretization or defer it until necessary. As a step in this direction, we discuss a formalism for representing decision problems with discrete and continuous state and action variables.

3.1 FACTORED TRANSITION AND REWARD MODELS

A *hybrid factored MDP (HMDP)* is a 4-tuple $\mathcal{M} = (\mathbf{X}, \mathbf{A}, P, R)$, where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a state space described by state variables, $\mathbf{A} = \{A_1, \ldots, A_m\}$ is an action space described by action variables, $P(\mathbf{X}' \mid \mathbf{X}, \mathbf{A})$ is a stochastic transition function of state dynamics, which is conditioned on the preceding state and action, and R is a reward function assigning immediate payoffs to state-action configurations.[1]

State variables: State variables are either discrete or continuous. Every discrete variable X_i takes on values from a finite domain $\mathrm{Dom}(X_i)$. Following Hauskrecht and Kveton [43], we assume that every continuous variable is bounded to the $[0, 1]$ subspace. Generally, this assumption is very mild

[1] *General state and action space MDP* is an alternative term for a hybrid MDP. The term *hybrid* does not refer to the dynamics of the model, which is discrete-time.

and permits modeling of any closed interval on $\mathbb{R}$. The state of the system is completely observed and defined by a vector of value assignments $\mathbf{x} = (\mathbf{x}_D, \mathbf{x}_C)$ which partitions along its discrete and continuous components $\mathbf{x}_D$ and $\mathbf{x}_C$.

Action variables: The action space is fully distributed and represented by action variables $\mathbf{A}$. The composite action is defined by a vector of individual action choices $\mathbf{a} = (\mathbf{a}_D, \mathbf{a}_C)$ which partitions along its discrete and continuous components $\mathbf{a}_D$ and $\mathbf{a}_C$.

Transition model: The transition model is characterized by the conditional probability distribution $P(\mathbf{X}' \mid \mathbf{X}, \mathbf{A})$, where $\mathbf{X}$ and $\mathbf{X}'$ denote the state variables at two successive time steps t and $t+1$. We assume that this distribution factors along $\mathbf{X}'$ as $P(\mathbf{X}' \mid \mathbf{X}, \mathbf{A}) = \prod_{i=1}^{n} P(X_i' \mid \mathsf{Par}(X_i'))$ and can be represented compactly by a DBN [25]. Typically, the parent set $\mathsf{Par}(X_i') \subseteq \mathbf{X} \cup \mathbf{A}$ is a small subset of state and action variables which allows for a local parameterization of the model.

Parameterization of transition model: One-step dynamics of every state variable is described by the conditional probability $P(X_i' \mid \mathsf{Par}(X_i'))$. If X_i' is a continuous variable, its transition function is represented by a mixture of beta distributions [43]:

$$\begin{aligned} P(X_i' = x \mid \mathsf{Par}(X_i')) &= \sum_j \pi_{ij} P_{\text{beta}}(x \mid \alpha_j, \beta_j) \\ P_{\text{beta}}(x \mid \alpha, \beta) &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \end{aligned} \tag{3.1}$$

where π_{ij} is the weight assigned to the j-th component of the mixture, and $\alpha_j = \phi_{ij}^{\alpha}(\mathsf{Par}(X_i'))$ and $\beta_j = \phi_{ij}^{\beta}(\mathsf{Par}(X_i'))$ are arbitrary positive functions of the parent set. The mixture of beta distributions provides a very general class of transition functions and yet allows closed-form solutions[2] to the expectation terms in HALP (Section 3.2). Note that if every $\beta_j = 1$, Equation 3.1 turns into a polynomial in X_i'. Due to the Weierstrass approximation theorem [48], the polynomial is sufficient

[2]The term *closed-form* refers to a generally accepted set of closed-form operations and functions extended by the gamma and incomplete beta functions.

to approximate any transition density over X'_i with any precision. If X'_i is a discrete state variable, its transition model is parameterized by $|\mathrm{Dom}(X'_i)|$ discriminant functions $\theta_j = \phi^{\theta}_{ij}(\mathsf{Par}(X'_i))$ [35]:

$$P(X'_i = j \mid \mathsf{Par}(X'_i)) = \frac{\theta_j}{\sum_{j=1}^{|\mathrm{Dom}(X'_i)|} \theta_j}. \tag{3.2}$$

Note that the parameters α_j, β_j, and θ_j (Equations 3.1 and 3.2) are functions instantiated by value assignments to the variables $\mathsf{Par}(X'_i) \subseteq \mathbf{X} \cup \mathbf{A}$. We keep separate parameters for every variable X'_i although our indexing does not reflect it explicitly. The only restriction on the functions is that they must return valid parameters for all state-action pairs $(\mathbf{x}, \mathbf{a})$. Hence, we assume that $\alpha_j(\mathbf{x}, \mathbf{a}) \geq 0$, $\beta_j(\mathbf{x}, \mathbf{a}) \geq 0$, $\theta_j(\mathbf{x}, \mathbf{a}) \geq 0$, and $\sum_{j=1}^{|\mathrm{Dom}(X'_i)|} \theta_j(\mathbf{x}, \mathbf{a}) > 0$.

Reward model: The reward model is factored similarly to the transition model. In particular, the reward function $R(\mathbf{x}, \mathbf{a}) = \sum_j R_j(\mathbf{x}_j, \mathbf{a}_j)$ is an additive function of local reward functions defined on the subsets $\mathbf{X}_j$ and $\mathbf{A}_j$. In graphical models, these local functions can be described compactly by reward nodes R_j, which are conditioned on their parent sets $\mathsf{Par}(R_j) = \mathbf{X}_j \cup \mathbf{A}_j$. To allow this representation, we formally extend our DBN into an influence diagram [47]. Note that the form of the reward functions $R_j(\mathbf{x}_j, \mathbf{a}_j)$ is unrestricted.

Optimal value function and policy: The *optimal policy* π^* can be defined greedily with respect to the *optimal value function* V^*, which is a fixed point of the Bellman equation:

$$\begin{aligned} V^*(\mathbf{x}) &= \sup_{\mathbf{a}} \left[R(\mathbf{x}, \mathbf{a}) + \gamma \mathrm{E}_{P(\mathbf{x}' | \mathbf{x}, \mathbf{a})}[V^*(\mathbf{x}')] \right] \\ &= \sup_{\mathbf{a}} \left[R(\mathbf{x}, \mathbf{a}) + \gamma \sum_{\mathbf{x}'_D} \int_{\mathbf{x}'_C} P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a}) V^*(\mathbf{x}') \, \mathrm{d}\mathbf{x}'_C \right]. \end{aligned} \tag{3.3}$$

Accordingly, the *hybrid Bellman operator* $\mathcal{T}^*$ is given by:

$$\mathcal{T}^* V(\mathbf{x}) = \sup_{\mathbf{a}} \left[R(\mathbf{x}, \mathbf{a}) + \gamma \mathrm{E}_{P(\mathbf{x}' | \mathbf{x}, \mathbf{a})}[V(\mathbf{x}')] \right]. \tag{3.4}$$

In the rest of the chapter, we denote expectation terms over discrete and continuous variables in a unified form:

$$\mathrm{E}_{P(\mathbf{x})}[f(\mathbf{x})] = \sum_{\mathbf{x}_D} \int_{\mathbf{x}_C} P(\mathbf{x}) f(\mathbf{x}) \, \mathrm{d}\mathbf{x}_C. \tag{3.5}$$

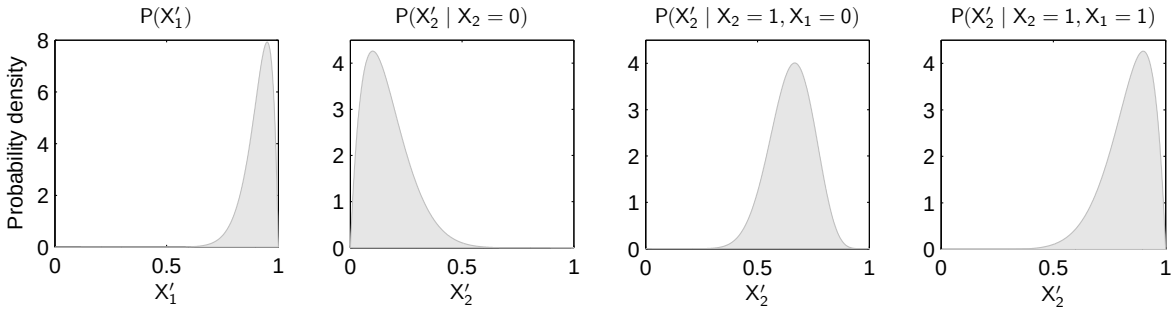


Figure 6: Transition functions for continuous variables X_1' and X_2' after taking an action a_1 in the 4-ring topology (Example 4). The densities are shown for extreme values of their parent variables X_1 and X_2.

Example 4 (Hauskrecht and Kveton [43]) *Continuous-state network administration is a variation on Example 1, where the computer states are described by continuous variables on the interval between 0 (being down) and 1 (running). At each time step, the administrator selects an action from the set $\mathcal{A} = \{a_1, \ldots, a_{n+1}\}$. The action a_i ($i \leq n$) corresponds to rebooting the i-th computer. The last action a_{n+1} is dummy. The transition function captures the propagation of failures in the network and is encoded locally by beta distributions:*

$$P(X_i' = x \mid \mathsf{Par}(X_i')) = P_{\text{beta}}(x \mid \alpha, \beta) \quad \left| \begin{array}{ll} \alpha = 20 & a = i \\ \beta = 2 & \\ \alpha = 2 + 13x_i - 5x_i \mathrm{E}[\mathsf{Par}(X_i')] & a \neq i \\ \beta = 10 - 2x_i - 6x_i \mathrm{E}[\mathsf{Par}(X_i')] & \end{array} \right.$$

where the variables x_i and $\mathrm{E}[\mathsf{Par}(X_i')]$ denote the state of the i-th computer and the expected state of its parents. Note that this transition function is similar to Example 1. For instance, in the 4-ring topology, the modes of transition densities for continuous variables X_1' and X_2' after taking an action a_1 (Figure 6):

$$\widehat{P}(X_1' \mid a = a_1) = 0.95 \quad \widehat{P}(X_2' \mid X_2 = 1, X_1 = 0, a = a_1) \approx 0.67$$
$$\widehat{P}(X_2' \mid X_2 = 0, a = a_1) = 0.10 \quad \widehat{P}(X_2' \mid X_2 = 1, X_1 = 1, a = a_1) = 0.90$$

are equal to the expected values of their discrete counterparts (Figure 3b). The reward function is additive:

$$R(\mathbf{x}, a) = 2x_1^2 + \sum_{j=2}^{n} x_j^2$$

and establishes our preferences for maintaining the server X_1 and workstations $X_2, \ldots, X_n$.

3.2 HYBRID APPROXIMATE LINEAR PROGRAMMING

In Section 2.2, we introduced the most fundamental dynamic programming techniques for solving MDPs. However, these methods are unsuitable for solving hybrid factored MDPs (Section 2.3). To allow for an efficient solution to these problems, we present a novel approach based on approximate linear programming (Section 2.3.2.3). Comparing to the existing work (Section 2.3), this approach has several nice properties. First, unlike other parametric value function approximation techniques (Section 2.3.2), it cannot diverge in principle and its feasibility can be easily guaranteed. Second, in contrast to parametric policy approximations (Section 2.3.3), ALP policies can be derived without relying on gradient optimization methods and a potentially suboptimal parametric class of policies (Section 5). Finally, comparing to the nearest neighbor methods (Section 2.3.4), approximate linear programming has a potential to scale up to large distributed decision problems.

Hybrid approximate linear programming (HALP) extends ALP (Section 2.3.2.3) to hybrid state and action domains. Note that it optimizes the linear value function approximation (Equation 2.20). Therefore, it transforms an initially intractable problem of computing V^* in the hybrid state space $\mathbf{X}$ into a lower dimensional space of $\mathbf{w}$. The HALP formulation is given by a linear program[3]:

$$\begin{aligned} &\text{minimize}_{\mathbf{w}} \quad \sum_i w_i \alpha_i \\ &\text{subject to:} \quad \sum_i w_i F_i(\mathbf{x}, \mathbf{a}) - R(\mathbf{x}, \mathbf{a}) \geq 0 \quad \forall\, \mathbf{x} \in \mathbf{X}, \mathbf{a} \in \mathbf{A}; \end{aligned} \tag{3.6}$$

[3]More precisely, the HALP formulation (3.6) is a *linear semi-infinite optimization* problem with an infinite number of constraints. Note that the number of basis functions remains finite. For brevity, we refer to this optimization problem as linear programming.

where $\mathbf{w}$ represents the variables in the LP, α_i denotes *basis function relevance weight*:

$$\begin{aligned} \alpha_i &= \mathrm{E}_{\psi(\mathbf{x})}[f_i(\mathbf{x})] \\ &= \sum_{\mathbf{x}_D} \int_{\mathbf{x}_C} \psi(\mathbf{x}) f_i(\mathbf{x}) \, \mathrm{d}\mathbf{x}_C, \end{aligned} \tag{3.7}$$

$\psi(\mathbf{x}) \geq 0$ is a *state relevance density function* that weights the quality of the approximation, and $F_i(\mathbf{x}, \mathbf{a}) = f_i(\mathbf{x}) - \gamma g_i(\mathbf{x}, \mathbf{a})$ is the difference between the basis function $f_i(\mathbf{x})$ and its discounted *backprojection*:

$$\begin{aligned} g_i(\mathbf{x}, \mathbf{a}) &= \mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})}[f_i(\mathbf{x}')] \\ &= \sum_{\mathbf{x}'_D} \int_{\mathbf{x}'_C} P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a}) f_i(\mathbf{x}') \, \mathrm{d}\mathbf{x}'_C. \end{aligned} \tag{3.8}$$

Vectors $\mathbf{x}_D$ ($\mathbf{x}'_D$) and $\mathbf{x}_C$ ($\mathbf{x}'_C$) are the discrete and continuous components of value assignments $\mathbf{x}$ ($\mathbf{x}'$) to all state variables $\mathbf{X}$ ($\mathbf{X}'$). The linear program can be rewritten compactly:

$$\begin{aligned} \text{minimize}_{\mathbf{w}} \quad & \mathrm{E}_{\psi}[V^{\mathbf{w}}] \\ \text{subject to:} \quad & V^{\mathbf{w}} - \mathcal{T}^* V^{\mathbf{w}} \geq 0 \end{aligned} \tag{3.9}$$

by using the Bellman operator $\mathcal{T}^*$.

The HALP formulation reduces to the discrete-state ALP (Section 2.3.2.3) if the state and action variables are discrete, and to the continuous-state ALP [43] if the state variables are continuous. The formulation is feasible if the set of basis functions contains a constant function $f_0(\mathbf{x}) \equiv 1$. We assume that such a basis function is present.

In the rest of the chapter, we address several concerns related to the HALP formulation. First, we analyze the quality of the approximation and relate it to the minimization of the max-norm error $\|V^* - V^{\mathbf{w}}\|_\infty$, which is a commonly-used metric (Section 3.2.1). Second, we present rich classes of basis functions that lead to closed-form solutions to the expectation terms in the objective function and constraints (Equations 3.7 and 3.8). The terms involve sums and integrals over the complete state space $\mathbf{X}$ (Section 3.2.2), and therefore are hard to evaluate. Finally, we discuss approximations to the constraint space in HALP and introduce a framework for solving HALP formulations in a unified way (Section 3.3). Note that complete satisfaction of this constraint space may not be possible since every state-action pair $(\mathbf{x}, \mathbf{a})$ induces a constraint.

3.2.1 Error Bounds

The quality of the ALP approximation (Section 2.3.2.3) was studied by de Farias and Van Roy [23]. We follow up on their work and generalize it to structured state and action spaces with continuous variables. Before we proceed, we show that a solution to the HALP formulation (3.6) constitutes an upper bound on the optimal value function V^*.

Proposition 2 *Let $\widetilde{\mathbf{w}}$ be a solution to the HALP formulation (3.6). Then $V^{\widetilde{\mathbf{w}}} \geq V^*$.*

This result allows us to restate the objective $\mathrm{E}_\psi[V^{\mathbf{w}}]$ in HALP.

Proposition 3 *Vector $\widetilde{\mathbf{w}}$ is a solution to the HALP formulation (3.6):*

$$\begin{aligned} \text{minimize}_{\mathbf{w}} \quad & \mathrm{E}_\psi[V^{\mathbf{w}}] \\ \text{subject to:} \quad & V^{\mathbf{w}} - \mathcal{T}^* V^{\mathbf{w}} \geq 0 \end{aligned}$$

if and only if it solves:

$$\begin{aligned} \text{minimize}_{\mathbf{w}} \quad & \left\| V^* - V^{\mathbf{w}} \right\|_{1,\psi} \\ \text{subject to:} \quad & V^{\mathbf{w}} - \mathcal{T}^* V^{\mathbf{w}} \geq 0; \end{aligned}$$

where $\|\cdot\|_{1,\psi}$ is an $\mathcal{L}_1$-norm weighted by the state relevance density function ψ and $\mathcal{T}^*$ is the hybrid Bellman operator.

Based on Proposition 3, we conclude that HALP optimizes the linear value function approximation with respect to the weighted $\mathcal{L}_1$-norm error $\left\| V^* - V^{\mathbf{w}} \right\|_{1,\psi}$. The following theorem draws a parallel between minimizing this objective and max-norm error $\left\| V^* - V^{\mathbf{w}} \right\|_\infty$. More precisely, the theorem says that HALP yields a close approximation $V^{\widetilde{\mathbf{w}}}$ to the optimal value function V^* if V^* is close to the span of basis functions $f_i(\mathbf{x})$.

Theorem 2 *Let $\widetilde{\mathbf{w}}$ be an optimal solution to the HALP formulation (3.6). Then the expected error of the value function $V^{\widetilde{\mathbf{w}}}$ can be bounded as:*

$$\left\| V^* - V^{\widetilde{\mathbf{w}}} \right\|_{1,\psi} \leq \frac{2}{1-\gamma} \min_{\mathbf{w}} \left\| V^* - V^{\mathbf{w}} \right\|_\infty ,$$

where $\|\cdot\|_{1,\psi}$ is an $\mathcal{L}_1$-norm weighted by the state relevance density function ψ and $\|\cdot\|_\infty$ is a max-norm.

Unfortunately, Theorem 2 rarely yields a tight bound on $\left\|V^* - V^{\tilde{\mathbf{w}}}\right\|_{1,\psi}$. First, it is hard to guarantee a uniformly low max-norm error $\left\|V^* - V^{\mathbf{w}}\right\|_\infty$ if the dimensionality of a problem grows but the basis functions $f_i(\mathbf{x})$ are local. Second, this bound ignores the state relevance density function $\psi(\mathbf{x})$ even if this one impacts the quality of HALP solutions. To address these issues, we introduce non-uniform weighting of the max-norm error in Theorem 3.

Theorem 3 *Let $\widetilde{\mathbf{w}}$ be an optimal solution to the HALP formulation (3.6). Then the expected error of the value function $V^{\tilde{\mathbf{w}}}$ can be bounded as:*

$$\left\|V^* - V^{\tilde{\mathbf{w}}}\right\|_{1,\psi} \leq \frac{2\mathrm{E}_\psi[L]}{1-\kappa} \min_{\mathbf{w}} \left\|V^* - V^{\mathbf{w}}\right\|_{\infty,1/L},$$

where $\|\cdot\|_{1,\psi}$ denotes an $\mathcal{L}_1$-norm weighted by the state relevance density ψ, $L(\mathbf{x}) = \sum_i w_i^L f_i(\mathbf{x})$ is a Lyapunov function *such that the inequality $\kappa L(\mathbf{x}) \geq \gamma \sup_{\mathbf{a}} \mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})}[L(\mathbf{x}')]$ holds, $\kappa \in [0,1)$ is its* contraction factor, *and $\|\cdot\|_{\infty,1/L}$ is a max-norm reweighted by the reciprocal $1/L$.*

Note that Theorem 2 is a special form of Theorem 3 when $L(\mathbf{x}) \equiv 1$ and $\kappa = \gamma$. As a result, the Lyapunov function $L(\mathbf{x})$ permits at least as good bounds as Theorem 2. To make the bounds tight, the function $L(\mathbf{x})$ should return large values in the regions of the state space, which are unimportant for modeling. In turn, the reciprocal $1/L(\mathbf{x})$ is approaching zero in the unimportant regions, which makes their impact on the max-norm error $\left\|V^* - V^{\mathbf{w}}\right\|_{\infty,1/L}$ less likely. Since the state relevance density function $\psi(\mathbf{x})$ reflects the importance of states, the term $\mathrm{E}_\psi[L]$ should remain small. These two factors contribute to tighter bounds than those by Theorem 2.

Since the Lyapunov function $L(\mathbf{x}) = \sum_i w_i^L f_i(\mathbf{x})$ lies in the span of basis functions $f_i(\mathbf{x})$, Theorem 3 provides a recipe for high-quality approximations. Intuitively, a good set of basis functions always involves two types of functions. The first type guarantees small errors $|V^*(\mathbf{x}) - V^{\mathbf{w}}(\mathbf{x})|$ in the important regions of the state space, where the state relevance density $\psi(\mathbf{x})$ is high. The second type returns high values where the state relevance density $\psi(\mathbf{x})$ is low, and vice versa. The latter functions allow the satisfaction of the constraint space $V^{\mathbf{w}} \geq \mathcal{T}^* V^{\mathbf{w}}$ in the unimportant regions of the state space without impacting the optimized objective $\left\|V^* - V^{\mathbf{w}}\right\|_{1,\psi}$. Note that a trivial value function $V^{\mathbf{w}}(\mathbf{x}) = (1-\gamma)^{-1} R_{\max}$ satisfies all constraints in any HALP but unlikely leads to good policies. For a comprehensive discussion on selecting appropriate $\psi(\mathbf{x})$ and $L(\mathbf{x})$, refer to the case studies of de Farias and Van Roy [23].

Our discussion is concluded by clarifying the notion of the state relevance density $\psi(\mathbf{x})$. As shown by Theorem 4, its choice is related to the quality of a greedy policy for the value function $V^{\tilde{\mathbf{w}}}$ [23].

Theorem 4 *Let $\tilde{\mathbf{w}}$ be an optimal solution to the HALP formulation (3.6). Then the expected error of a greedy policy:*

$$u(\mathbf{x}) = \arg\sup_{\mathbf{a}} \left[R(\mathbf{x}, \mathbf{a}) + \gamma \mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})} \left[V^{\tilde{\mathbf{w}}}(\mathbf{x}') \right] \right]$$

can be bounded as:

$$\left\| V^* - V^u \right\|_{1,\nu} \leq \frac{1}{1-\gamma} \left\| V^* - V^{\tilde{\mathbf{w}}} \right\|_{1,\mu_{u,\nu}},$$

where $\|\cdot\|_{1,\nu}$ and $\|\cdot\|_{1,\mu_{u,\nu}}$ are weighted $\mathcal{L}_1$-norms, V^u is a value function for the policy u, and $\mu_{u,\nu}$ is the expected frequency of state visits generated by following the greedy policy u given the initial state distribution ν.

Based on Theorem 4, we may conclude that the expected error of greedy policies for HALP approximations is bounded when $\psi = \mu_{u,\nu}$. Note that the distribution $\mu_{u,\nu}$ is not known when optimizing $V^{\tilde{\mathbf{w}}}$ because it is a function of the optimized quantity itself. To break this cycle, de Farias and Van Roy [23] suggested an iterative procedure that solves several LPs and adapts $\mu_{u,\nu}$ accordingly. In addition, real-world control problems exhibit a lot of structure, which permits the guessing of $\mu_{u,\nu}$.

Finally, it is important to realize that although our bounds (Theorems 3 and 4) build a foundation for better HALP approximations, they can be rarely used in practice because the optimal value function V^* is generally unknown. After all, if V^* was known, there is no need to approximate it. Moreover, note that the optimization of $\|V^* - V^{\mathbf{w}}\|_{\infty,1/L}$ (Theorem 3) is a hard problem and there are no methods that would minimize this error directly [81]. Despite these facts, both bounds provide a loose guidance for empirical choices of basis functions. In Section 3.4, we use this intuition and propose basis functions that should closely approximate unknown optimal value functions V^*.

3.2.2 Expectation Terms

Since our basis functions are often restricted to small subsets of state variables, expectation terms (Equations 3.7 and 3.8) in the HALP formulation (3.6) should be efficiently computable. To unify the analysis of these expectation terms, $\mathrm{E}_{\psi(\mathbf{x})}[f_i(\mathbf{x})]$ and $\mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})}[f_i(\mathbf{x}')]$, we show that their evaluation constitutes the same computational problem $\mathrm{E}_{P(\mathbf{x})}[f_i(\mathbf{x})]$, where $P(\mathbf{x})$ denotes some factored distribution.

Before we discuss expectation terms in the constraints, note that the transition function $P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a})$ is factored and its parameterization is determined by the state-action pair $(\mathbf{x}, \mathbf{a})$. We keep the pair $(\mathbf{x}, \mathbf{a})$ fixed in the rest of the section, which corresponds to choosing a single constraint $(\mathbf{x}, \mathbf{a})$. Based on this selection, we can rewrite the expectation terms $\mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})}[f_i(\mathbf{x}')]$ in a simpler notation $\mathrm{E}_{P(\mathbf{x}')}[f_i(\mathbf{x}')]$, where $P(\mathbf{x}') = P(\mathbf{x}' \,|\, \mathbf{x}, \mathbf{a})$ denotes a factored distribution with fixed parameters.

We also assume that the state relevance density function $\psi(\mathbf{x})$ factors along $\mathbf{X}$ as:

$$\psi(\mathbf{x}) = \prod_{i=1}^{n} \psi_i(x_i), \tag{3.10}$$

where $\psi_i(x_i)$ is a distribution over the random state variable X_i. Based on this assumption, we can rewrite the expectation terms $\mathrm{E}_{\psi(\mathbf{x})}[f_i(\mathbf{x})]$ in the objective function in a new notation $\mathrm{E}_{P(\mathbf{x})}[f_i(\mathbf{x})]$, where $P(\mathbf{x}) = \psi(\mathbf{x})$ is a factored distribution. In line with our discussion in the last two paragraphs, efficient solutions to the expectation terms in HALP are obtained by solving the generalized term $\mathrm{E}_{P(\mathbf{x})}[f_i(\mathbf{x})]$ efficiently. We address this problem in the rest of the section.

Before computing the expectation term $\mathrm{E}_{P(\mathbf{x})}[f_i(\mathbf{x})]$ over the complete state space $\mathbf{X}$, we recall that the basis function $f_i(\mathbf{x})$ is defined on a subset of state variables $\mathbf{X}_i$. Therefore, we immediately conclude that $\mathrm{E}_{P(\mathbf{x})}[f_i(\mathbf{x})] = \mathrm{E}_{P(\mathbf{x}_i)}[f_i(\mathbf{x}_i)]$, where $P(\mathbf{x}_i)$ denotes a factored distribution on a lower dimensional space $\mathbf{X}_i$. If no further assumptions are made, the local expectation term $\mathrm{E}_{P(\mathbf{x}_i)}[f_i(\mathbf{x}_i)]$ may be still hard to compute. Although it can be estimated by a variety of numerical methods, for instance Monte Carlo [1], these techniques are typically imprecise if the sample size is small, and quite computationally expensive if a high precision is required. Consequently, we try to avoid such an approximation step. Instead, we introduce an appropriate form of basis functions that guarantees closed-form solutions to the expectation term $\mathrm{E}_{P(\mathbf{x}_i)}[f_i(\mathbf{x}_i)]$.

In particular, let us assume that every basis function $f_i(\mathbf{x}_i)$ factors as:

$$f_i(\mathbf{x}_i) = f_{i_D}(\mathbf{x}_{i_D}) f_{i_C}(\mathbf{x}_{i_C}) \tag{3.11}$$

along its discrete and continuous components $f_{i_D}(\mathbf{x}_{i_D})$ and $f_{i_C}(\mathbf{x}_{i_C})$, where the continuous component further decouples as a product:

$$f_{i_C}(\mathbf{x}_{i_C}) = \prod_{X_j \in \mathbf{X}_{i_C}} f_{ij}(x_j) \tag{3.12}$$

of univariate basis function factors $f_{ij}(x_j)$. Note that the basis functions are multivariate despite the two independence assumptions. We make these presumptions for computational purposes and they are relaxed later in the section.

Based on Equation 3.11, we conclude that the expectation term:

$$\begin{aligned} \mathrm{E}_{P(\mathbf{x}_i)}[f_i(\mathbf{x}_i)] &= \mathrm{E}_{P(\mathbf{x}_i)}[f_{i_D}(\mathbf{x}_{i_D}) f_{i_C}(\mathbf{x}_{i_C})] \\ &= \mathrm{E}_{P(\mathbf{x}_{i_D})}[f_{i_D}(\mathbf{x}_{i_D})]\, \mathrm{E}_{P(\mathbf{x}_{i_C})}[f_{i_C}(\mathbf{x}_{i_C})] \end{aligned} \tag{3.13}$$

decomposes along the discrete and continuous variables $\mathbf{X}_{i_D}$ and $\mathbf{X}_{i_C}$, where $\mathbf{x}_i = (\mathbf{x}_{i_D}, \mathbf{x}_{i_C})$ and $P(\mathbf{x}_i) = P(\mathbf{x}_{i_D}) P(\mathbf{x}_{i_C})$. The evaluation of the discrete part $\mathrm{E}_{P(\mathbf{x}_{i_D})}[f_{i_D}(\mathbf{x}_{i_D})]$ requires aggregation in the subspace $\mathbf{X}_{i_D}$:

$$\mathrm{E}_{P(\mathbf{x}_{i_D})}[f_{i_D}(\mathbf{x}_{i_D})] = \sum_{\mathbf{x}_{i_D}} P(\mathbf{x}_{i_D}) f_{i_D}(\mathbf{x}_{i_D}), \tag{3.14}$$

which can be carried out efficiently in $O(\prod_{X_j \in \mathbf{X}_{i_D}} |\mathrm{Dom}(X_j)|)$ time (Section 2.3.2.3). Following Equation 3.12, the continuous term $\mathrm{E}_{P(\mathbf{x}_{i_C})}[f_{i_C}(\mathbf{x}_{i_C})]$ decouples as a product:

$$\begin{aligned} \mathrm{E}_{P(\mathbf{x}_{i_C})}[f_{i_C}(\mathbf{x}_{i_C})] &= \mathrm{E}_{P(\mathbf{x}_{i_C})}\left[\prod_{X_j \in \mathbf{X}_{i_C}} f_{ij}(x_j)\right] \\ &= \prod_{X_j \in \mathbf{X}_{i_C}} \mathrm{E}_{P(x_j)}[f_{ij}(x_j)]\,, \end{aligned} \tag{3.15}$$

where $\mathrm{E}_{P(x_j)}[f_{ij}(x_j)]$ represents the expectation terms over individual random variables X_j. Consequently, an efficient solution to the local expectation term $\mathrm{E}_{P(\mathbf{x}_i)}[f_i(\mathbf{x}_i)]$ is guaranteed by efficient solutions to its univariate components $\mathrm{E}_{P(x_j)}[f_{ij}(x_j)]$.

In this work, we consider three univariate basis function factors $f_{ij}(x_j)$: piecewise linear functions, polynomials, and beta distributions. These factors support a very general class of basis functions and yet allow closed-form solutions to the expectation terms $\mathrm{E}_{P(x_j)}[f_{ij}(x_j)]$. These solutions are provided in the following propositions and demonstrated in Example 5.

Proposition 4 (Polynomial basis functions) *Let:*

$$P(x) = P_{\text{beta}}(x \mid \alpha, \beta)$$

be a beta distribution over X and:

$$f(x) = x^n(1-x)^m$$

be a polynomial in x and $(1-x)$. Then $\mathrm{E}_{P(x)}[f(x)]$ has a closed-form solution:

$$\mathrm{E}_{P(x)}[f(x)] = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \frac{\Gamma(\alpha+n)\Gamma(\beta+m)}{\Gamma(\alpha+\beta+n+m)}.$$

Corollary 1 (Beta basis functions) *Let:*

$$\begin{aligned} P(x) &= P_{\text{beta}}(x \mid \alpha, \beta) \\ f(x) &= P_{\text{beta}}(x \mid \alpha_f, \beta_f) \end{aligned}$$

be beta distributions over X. Then $\mathrm{E}_{P(x)}[f(x)]$ has a closed-form solution:

$$\mathrm{E}_{P(x)}[f(x)] = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \frac{\Gamma(\alpha_f+\beta_f)}{\Gamma(\alpha_f)\Gamma(\beta_f)} \frac{\Gamma(\alpha+\alpha_f-1)\Gamma(\beta+\beta_f-1)}{\Gamma(\alpha+\alpha_f+\beta+\beta_f-2)}.$$

Proof: The claim directly follows from Proposition 4. Since integration is a distributive operation, it straightforwardly generalizes to the mixture of beta distributions $P(x)$. ■

Proposition 5 (Piecewise linear basis functions) *Let:*

$$P(x) = P_{\text{beta}}(x \mid \alpha, \beta)$$

be a beta distribution over X and:

$$f(x) = \sum_i \mathbf{1}_{[l_i, r_i]}(x)(a_i x + b_i)$$

be a piecewise linear (PWL) function in x, where $\mathbf{1}_{[l_i,r_i]}(x)$ represents the indicator function of the interval $[l_i, r_i]$. Then $\mathrm{E}_{P(x)}[f(x)]$ has a closed-form solution:

$$\mathrm{E}_{P(x)}[f(x)] = \sum_i \left[a_i \frac{\alpha}{\alpha + \beta}(F^+(r_i) - F^+(l_i)) + b_i(F(r_i) - F(l_i)) \right],$$

where $F(u) = F_{\text{beta}}(u \mid \alpha, \beta)$ and $F^+(u) = F_{\text{beta}}(u \mid \alpha + 1, \beta)$ denote the cumulative density functions of beta distributions.

Example 5 *Efficient closed-form solutions to the expectation terms in HALP are illustrated on the 4-ring network administration problem (Example 4) with three univariate basis functions:*

$$\begin{aligned} f_{\text{poly}}(x_2') &= x_2'^4 \\ f_{\text{beta}}(x_2') &= P_{\text{beta}}(x_2' \mid 2, 6) \\ f_{\text{pwl}}(x_2') &= \mathbf{1}_{[0.3,0.5]}(x_2')(5x_2' - 1.5) + \mathbf{1}_{[0.5,0.7]}(x_2')(-5x_2' + 3.5) \end{aligned}$$

Suppose that our goal is to evaluate expectation terms in a single constraint that corresponds to the network state $\mathbf{x} = (0, 1, 0, 0)$ and the administrator rebooting the server. Based on the assumptions, the expectation terms in the constraint $(\mathbf{x}, a_1)$ simplify as:

$$\mathrm{E}_{P(\mathbf{x}' \mid \mathbf{x}, a_1)}[f(x_2')] = \mathrm{E}_{P(x_2' \mid \mathbf{x}, a_1)}[f(x_2')],$$

where the transition function $P(x_2' \mid \mathbf{x}, a_1)$ is given by:

$$\begin{aligned} P(x_2' \mid \mathbf{x}, a_1) &= P(X_2' = x_2' \mid X_2 = 1, X_1 = 0, a = a_1) \\ &= P_{\text{beta}}(x_2' \mid 15, 8). \end{aligned}$$

Closed-form solutions to the simplified expectation terms $\mathrm{E}_{P(x_2'|\mathbf{x},a_1)}[f(x_2')]$ *are computed as:*

$$
\begin{aligned}
\mathrm{E}_{P(x_2'|\mathbf{x},a_1)}[f_{\text{poly}}(x_2')] &= \int_{x_2'} P_{\text{beta}}(x_2' \mid 15, 8) x_2'^4 \,\mathrm{d}x_2' \\
\text{(Proposition 4)} &= \frac{\Gamma(15+8)}{\Gamma(15)\Gamma(8)} \frac{\Gamma(15+4)\Gamma(8)}{\Gamma(15+8+4)} \\
&\approx 0.20 \\
\mathrm{E}_{P(x_2'|\mathbf{x},a_1)}[f_{\text{beta}}(x_2')] &= \int_{x_2'} P_{\text{beta}}(x_2' \mid 15, 8) P_{\text{beta}}(x_2' \mid 2, 6) \,\mathrm{d}x_2' \\
\text{(Corollary 1)} &= \frac{\Gamma(15+8)}{\Gamma(15)\Gamma(8)} \frac{\Gamma(2+6)}{\Gamma(2)\Gamma(6)} \frac{\Gamma(15+2-1)\Gamma(8+6-1)}{\Gamma(15+2+8+6-2)} \\
&\approx 0.22 \\
\mathrm{E}_{P(x_2'|\mathbf{x},a_1)}[f_{\text{pwl}}(x_2')] &= \int_{x_2'} P_{\text{beta}}(x_2' \mid 15, 8) \mathbf{1}_{[0.3,0.5]}(x_2')(5x_2' - 1.5) \,\mathrm{d}x_2' + \\
&\quad \int_{x_2'} P_{\text{beta}}(x_2' \mid 15, 8) \mathbf{1}_{[0.5,0.7]}(x_2')(-5x_2' + 3.5) \,\mathrm{d}x_2' \\
\text{(Proposition 5)} &= 5\frac{15}{15+8}(F^+(0.5) - F^+(0.3)) - 1.5(F(0.5) - F(0.3)) - \\
&\quad 5\frac{15}{15+8}(F^+(0.7) - F^+(0.5)) + 3.5(F(0.7) - F(0.5)) \\
&\approx 0.30
\end{aligned}
$$

where $F(u) = F_{\text{beta}}(u \mid 15, 8)$ *and* $F^+(u) = F_{\text{beta}}(u \mid 15+1, 8)$ *denote the cumulative density functions of beta distributions. A graphical interpretation of the computations is given in Figure 7. Brief inspection verifies that the term* $\mathrm{E}_{P(x_2'|\mathbf{x},a_1)}[f_{\text{pwl}}(x_2')]$ *is indeed the largest one.*

Up to this point, we obtained efficient closed-form solutions for factored basis functions and state relevance densities. Unfortunately, the factorization assumptions in Equations 3.10, 3.11, and 3.12 are rarely justified in practice. In the rest of the section, we show how to relax them. In Section 3.3, we apply our current results and propose several methods that approximately satisfy the constraint space in HALP.

3.2.2.1 Factored State Relevance Density Functions

Note that the state relevance density function $\psi(\mathbf{x})$ is unlikely to be factored (Section 3.2.1). Therefore, the independence assumption in Equation 3.10 is extremely limiting. To relax the assumption,

we approximate the function $\psi(\mathbf{x})$ by a linear combination $\psi^{\omega}(\mathbf{x}) = \sum_{\ell} \omega_{\ell} \psi_{\ell}(\mathbf{x})$ of factored state relevance densities $\psi_{\ell}(\mathbf{x}) = \prod_{i=1}^{n} \psi_{\ell i}(x_i)$. As a result, the expectation terms in the objective function decompose as:

$$\begin{aligned} \mathrm{E}_{\psi^{\omega}(\mathbf{x})}[f_i(\mathbf{x})] &= \mathrm{E}_{\sum_{\ell} \omega_{\ell} \psi_{\ell}(\mathbf{x})}[f_i(\mathbf{x})] \\ &= \sum_{\ell} \omega_{\ell} \mathrm{E}_{\psi_{\ell}(\mathbf{x})}[f_i(\mathbf{x})] \,, \end{aligned} \tag{3.16}$$

where the factored terms $\mathrm{E}_{\psi_{\ell}(\mathbf{x})}[f_i(\mathbf{x})]$ can be evaluated based on Equation 3.13. In addition, if we assume the factored densities $\psi_{\ell}(\mathbf{x})$ are polynomials, the linear combination $\psi^{\omega}(\mathbf{x})$ is a polynomial. Due to the Weierstrass approximation theorem [48], such a polynomial is sufficient to approximate any state relevance density $\psi(\mathbf{x})$ with any precision. It follows that the linear combinations permit state relevance densities that reflect arbitrary dependencies among the state variables $\mathbf{X}$.

3.2.2.2 Factored Basis Functions

Following the discussion in Section 3.2.2.1, note that the linear value function $V^{\mathbf{w}}(\mathbf{x}) = \sum_i w_i f_i(\mathbf{x})$ with factored basis functions $f_i(\mathbf{x})$ (Equations 3.11 and 3.12) is sufficient to approximate the optimal value function V^* within any max-norm error $\|V^* - V^{\mathbf{w}}\|_{\infty}$. Based on Theorem 2, we know that the same set of basis functions yields a bound on the $\mathcal{L}_1$-norm error $\left\|V^* - V^{\widetilde{\mathbf{w}}}\right\|_{1,\psi}$. Therefore, despite our independence assumptions (Equations 3.11 and 3.12), we have a potential to obtain an arbitrarily close HALP approximation $V^{\widetilde{\mathbf{w}}}$ to V^*.

3.3 CONSTRAINT SPACE APPROXIMATIONS

An optimal solution $\widetilde{\mathbf{w}}$ to the HALP formulation (3.6) is determined by a finite set of *active constraints* at a vertex of the feasible region. Unfortunately, identification of this active set is a hard computational problem. In particular, it requires searching through an exponential number of constraints, if the state and action variables are discrete, and an infinite number of constraints, if any of the variables are continuous. As a result, it is in general infeasible to find the optimal solution $\widetilde{\mathbf{w}}$ to the HALP formulation. Hence, we resort to approximations to the constraint space in HALP whose optimal solution $\widehat{\mathbf{w}}$ is close to $\widetilde{\mathbf{w}}$. This notion of an approximation is formalized as follows.

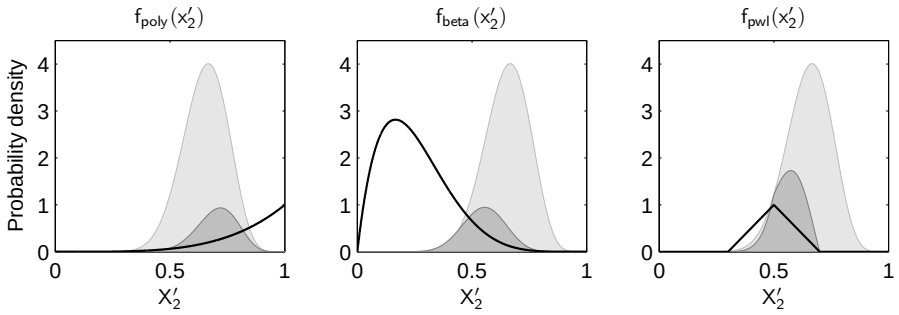


Figure 7: The expectation of three basis functions $f(x_2')$ (Example 5) with respect to the transition function $P(X_2' \mid X_2 = 1, X_1 = 0, a = a_1)$ from Figure 6. Every basis function $f(x_2')$ is depicted by a thick black line. The transition function is shown in a light gray color. Darker gray lines represent the values of the product $P(x_2' \,|\, \mathbf{x}, a_1) f(x_2')$. The area below corresponds to the expectation terms $\mathrm{E}_{P(x_2'|\mathbf{x},a_1)}[f(x_2')]$.

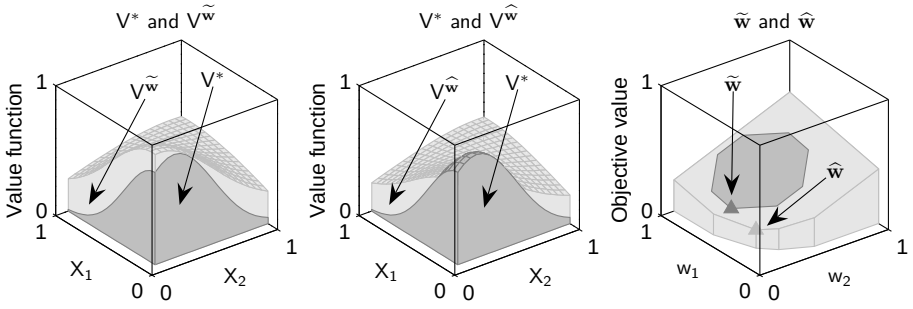


Figure 8: **a.** Graphical relation between the value function V^* and its HALP approximation $V^{\tilde{\mathbf{w}}}$. The function $V^{\tilde{\mathbf{w}}}$ is guaranteed to be an upper bound on V^*. **b.** The relaxed HALP approximation $V^{\hat{\mathbf{w}}}$ may not be an upper bound. **c.** Graphical relation between the optimal and relaxed solutions $\tilde{\mathbf{w}}$ and $\hat{\mathbf{w}}$. The feasible regions of the complete and relaxed formulations are shown in dark and light gray colors. The value function approximations $V^{\tilde{\mathbf{w}}}$ and $V^{\hat{\mathbf{w}}}$ are often nonlinear in the state space $\mathbf{X}$ but always linear in the space of parameters $\mathbf{w}$.

Definition 2 *The HALP formulation is* relaxed*:*

$$\begin{aligned} \text{minimize}_{\mathbf{w}} \quad & \sum_i w_i \alpha_i \\ \text{subject to:} \quad & \sum_i w_i F_i(\mathbf{x}, \mathbf{a}) - R(\mathbf{x}, \mathbf{a}) \geq 0 \quad (\mathbf{x}, \mathbf{a}) \in \mathcal{C}; \end{aligned} \tag{3.17}$$

if only a subset $\mathcal{C}$ of its constraints is satisfied.

HALP formulations (3.6) can be solved approximately by solving their relaxed formulations (3.17). Several methods for building and solving these approximate LPs have been proposed: Monte Carlo sampling of constraints, [43], their ε-grid discretization [35], and an adaptive search for a violated constraint [57]. In the rest of this section, we introduce these methods. From now on, we denote optimal solutions to the complete and relaxed HALP formulations by the symbols $\widetilde{\mathbf{w}}$ and $\widehat{\mathbf{w}}$.

Before we proceed, note that while $V^{\widetilde{\mathbf{w}}}$ is an upper bound on the optimal value function V^* (Figure 8a), the relaxed value function $V^{\widehat{\mathbf{w}}}$ does not have to be (Figure 8b). The main reason is that the relaxed HALP formulation does not guarantee that the constraint $V^{\widehat{\mathbf{w}}} \geq \mathcal{T}^* V^{\widehat{\mathbf{w}}}$ is satisfied for all states $\mathbf{x}$. As a result, we cannot simply use Proposition 2 to prove $V^{\widehat{\mathbf{w}}} \geq V^*$. Furthermore, note that the inequality $\mathrm{E}_\psi\left[V^{\widehat{\mathbf{w}}}\right] \leq \mathrm{E}_\psi\left[V^{\widetilde{\mathbf{w}}}\right]$ always holds because the optimal solution $\widetilde{\mathbf{w}}$ is feasible in the relaxed HALP (Figure 8c). These observations become critical for understanding the rest of the section.

3.3.1 MC-HALP

In the simplest case, the constraint space in HALP can be approximated by its Monte Carlo (MC) sample. In this relaxation, the set of constraints $\mathcal{C}$ is selected with respect to some proposal distribution φ over state-action pairs $(\mathbf{x}, \mathbf{a})$. Since the set $\mathcal{C}$ is finite, it establishes a relaxed formulation (3.17), which can be solved by any LP solver. An algorithm that builds and satisfies relaxed MC-HALP formulations is outlined in Figure 9.

Constraint sampling is easily applied in continuous spaces and its space complexity is proportional to the number of state and action components. Hauskrecht and Kveton [43] used it to solve continuous-state factored MDPs and further refined it by heuristics [56]. In discrete-state domains, the quality of the sampled approximations has been analyzed by de Farias and Van Roy [24]. Their result is summarized by Theorem 5.

Theorem 5 (de Farias and Van Roy [24]) *Let $\widetilde{\mathbf{w}}$ be a solution to the ALP formulation (2.25) and $\widehat{\mathbf{w}}$ be a solution to its relaxed formulation whose constraints are sampled with respect to a proposal distribution φ over state-action pairs $(\mathbf{x}, \mathbf{a})$. Then there exist a distribution φ and sample size:*

$$N \geq O\left(\frac{A\theta}{(1-\gamma)\epsilon}\left(K \ln \frac{A\theta}{(1-\gamma)\epsilon} + \ln \frac{1}{\delta}\right)\right)$$

such that with probability at least $1-\delta$:

$$\left\|V^* - V^{\widehat{\mathbf{w}}}\right\|_{1,\psi} \leq \left\|V^* - V^{\widetilde{\mathbf{w}}}\right\|_{1,\psi} + \epsilon \left\|V^*\right\|_{1,\psi},$$

where $\|\cdot\|_{1,\psi}$ denotes an $\mathcal{L}_1$-norm weighted by the state relevance weights ψ, θ is a problem-specific constant, A and K denote the numbers of actions and basis functions, and ϵ and δ are scalars from the interval $(0, 1)$.

Unfortunately, proposing a sampling distribution φ that guarantees this polynomial bound on the sample size is as hard as knowing the optimal policy π^* [24]. This conclusion is parallel to those in importance sampling. Note that uniform Monte Carlo sampling can guarantee a low probability of constraints being violated but it is not sufficient to bound the magnitude of their violation [24].

3.3.2 ε-HALP

Another way of approximating the constraint space in HALP is by discretizing its continuous state and action variables $\mathbf{X}_C$ and $\mathbf{A}_C$ on a uniform ε-grid. The discretized constraint space preserves its original factored form but spans discrete variables only. Therefore, it can be compactly satisfied by the methods for discrete-state ALP (Section 2.3.2.3). An algorithm that builds and satisfies relaxed ε-HALP formulations is outlined in Figure 10. Note that the discretized constraint space involves exponentially many constraints $O(\lceil 1/\varepsilon + 1 \rceil^{|\mathbf{X}_C|+|\mathbf{A}_C|})$ in the number of state and action variables $\mathbf{X}_C$ and $\mathbf{A}_C$.

Inputs:

a hybrid factored MDP $\mathcal{M} = (\mathbf{X}, \mathbf{A}, P, R)$
basis functions $f_0(\mathbf{x}), f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots$
a proposal distribution φ

Algorithm:

initialize a relaxed HALP formulation with an empty set of constraints
$t = 0$
while a stopping criterion is not met
    sample $(\mathbf{x}, \mathbf{a}) \sim \varphi$
    add the constraint $(\mathbf{x}, \mathbf{a})$ to the relaxed HALP
    $t = t + 1$
solve the relaxed MC-HALP formulation

Outputs:

basis function weights $\mathbf{w}$

Figure 9: Pseudo-code implementation of the MC-HALP solver.

Inputs:

a hybrid factored MDP $\mathcal{M} = (\mathbf{X}, \mathbf{A}, P, R)$
basis functions $f_0(\mathbf{x}), f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots$
grid resolution ε

Algorithm:

discretize continuous variables $\mathbf{X}_C$ and $\mathbf{A}_C$ into $\lceil 1/\varepsilon + 1 \rceil$ equally-spaced values
identify subsets $\mathbf{X}_i$ and $\mathbf{A}_i$ ($\mathbf{X}_j$ and $\mathbf{A}_j$) corresponding to the domains of $F_i(\mathbf{x}, \mathbf{a})$ ($R_j(\mathbf{x}, \mathbf{a})$)
evaluate $F_i(\mathbf{x}_i, \mathbf{a}_i)$ ($R_j(\mathbf{x}_j, \mathbf{a}_j)$) for all configurations $\mathbf{x}_i$ and $\mathbf{a}_i$ ($\mathbf{x}_j$ and $\mathbf{a}_j$) on the ε-grid
calculate basis function relevance weights α_i
solve the relaxed ε-HALP formulation (Section 2.3.2.3)

Outputs:

basis function weights $\mathbf{w}$

Figure 10: Pseudo-code implementation of the ε-HALP solver.

3.3.2.1 Error Bounds

Recall that the ε-HALP formulation approximates the constraint space in HALP by a finite set of equally-spaced grid points. In this section, we analyze the quality of this approximation and bound it in terms violating constraints in the complete HALP. More precisely, we prove that if a relaxed HALP solution $\widehat{\mathbf{w}}$ violates the constraints in the complete HALP by a small amount, the quality of the approximation $V^{\widehat{\mathbf{w}}}$ is close to $V^{\widetilde{\mathbf{w}}}$. In the next section, we generalize this result and relate $V^{\widehat{\mathbf{w}}}$ to the grid resolution ε. Before we proceed, we quantify our notion of constraint violation.

Definition 3 *Let $\widehat{\mathbf{w}}$ be an optimal solution to a relaxed HALP formulation (3.17). The vector $\widehat{\mathbf{w}}$ is* δ-infeasible *if:*

$$V^{\widehat{\mathbf{w}}} - \mathcal{T}^* V^{\widehat{\mathbf{w}}} \geq -\delta, \tag{3.18}$$

where $\mathcal{T}^*$ is the hybrid Bellman operator.

Intuitively, the lower the δ-infeasibility of a relaxed HALP solution $\widehat{\mathbf{w}}$, the closer the quality of the approximation $V^{\widehat{\mathbf{w}}}$ to $V^{\widetilde{\mathbf{w}}}$. Proposition 6 states this intuition formally. In particular, it says that the relaxed HALP formulation leads to a close approximation $V^{\widehat{\mathbf{w}}}$ to the optimal value function V^* if the complete HALP does and the solution $\widehat{\mathbf{w}}$ violates its constraints by a small amount.

Proposition 6 *Let $\widetilde{\mathbf{w}}$ be an optimal solution to the HALP formulation (3.6) and $\widehat{\mathbf{w}}$ be an optimal δ-infeasible solution to its relaxed formulation (3.17). Then the expected error of the value function $V^{\widehat{\mathbf{w}}}$ can be bounded as:*

$$\left\| V^* - V^{\widehat{\mathbf{w}}} \right\|_{1,\psi} \leq \left\| V^* - V^{\widetilde{\mathbf{w}}} \right\|_{1,\psi} + \frac{2\delta}{1-\gamma},$$

where $\|\cdot\|_{1,\psi}$ is an $\mathcal{L}_1$-norm weighted by the state relevance density function ψ.

Based on Proposition 6, we can generalize our conclusions from Section 3.2.1 to relaxed HALP formulations. For instance, we draw a parallel between optimizing the relaxed objective $\mathrm{E}_{\psi}\left[V^{\widehat{\mathbf{w}}}\right]$ and the max-norm error $\|V^* - V^{\mathbf{w}}\|_{\infty,1/L}$.

Theorem 6 *Let $\widehat{\mathbf{w}}$ be an optimal δ-infeasible solution to a relaxed HALP formulation (3.17). Then the expected error of the value function $V^{\widehat{\mathbf{w}}}$ can be bounded as:*

$$\left\|V^* - V^{\widehat{\mathbf{w}}}\right\|_{1,\psi} \leq \frac{2\mathrm{E}_{\psi}[L]}{1-\kappa} \min_{\mathbf{w}} \left\|V^* - V^{\mathbf{w}}\right\|_{\infty,1/L} + \frac{2\delta}{1-\gamma},$$

where $\|\cdot\|_{1,\psi}$ denotes an $\mathcal{L}_1$-norm weighted by the state relevance density ψ, $L(\mathbf{x}) = \sum_i w_i^L f_i(\mathbf{x})$ is a Lyapunov function *such that the inequality $\kappa L(\mathbf{x}) \geq \gamma \sup_{\mathbf{a}} \mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})}[L(\mathbf{x}')]$ holds, $\kappa \in [0,1)$ is its* contraction factor*, and $\|\cdot\|_{\infty,1/L}$ is a max-norm reweighted by the reciprocal $1/L$.*

Proof: Direct combination of Theorem 3 and Proposition 6. ■

3.3.2.2 Grid Resolution

In the previous section, we bounded the error of a relaxed HALP formulation by its δ-infeasibility (Theorem 6), a measure of constraint violation in the complete HALP. However, it is unclear how the grid resolution ε relates to δ-infeasibility. In this section, we analyze the relationship between ε and δ. Moreover, we show how to exploit the factored structure in the constraint space to achieve the δ-infeasibility of a relaxed HALP solution $\widehat{\mathbf{w}}$ efficiently.

First, let us assume that $\widehat{\mathbf{w}}$ denotes an optimal δ-infeasible solution to an ε-HALP formulation and $\mathbf{Z} = \mathbf{X} \cup \mathbf{A}$ is the joint set of state and action variables. To derive an informative bound relating both ε and δ, we assume that the magnitudes of constraint violations $\tau^{\widehat{\mathbf{w}}}(\mathbf{z}) = \sum_i \widehat{w}_i F_i(\mathbf{z}) - R(\mathbf{z})$ are Lipschitz continuous.

Definition 4 *The function $f(\mathbf{x})$ is* Lipschitz continuous *if:*

$$|f(\mathbf{x}) - f(\mathbf{x}')| \leq K \left\|\mathbf{x} - \mathbf{x}'\right\|_{\infty} \quad \forall\, \mathbf{x}, \mathbf{x}' \in \mathbf{X}; \tag{3.19}$$

where K is referred to as a Lipschitz constant.

Based on the ε-grid discretization of the constraint space, we know that the distance of any point $\mathbf{z}$ to its closest grid point $\mathbf{z}_G = \arg\min_{\mathbf{z}'} \left\|\mathbf{z} - \mathbf{z}'\right\|_{\infty}$ is bounded as:

$$\left\|\mathbf{z} - \mathbf{z}_G\right\|_{\infty} < \frac{\varepsilon}{2}. \tag{3.20}$$

From the Lipschitz continuity of $\tau^{\widehat{\mathbf{w}}}(\mathbf{z})$, we conclude:

$$\left|\tau^{\widehat{\mathbf{w}}}(\mathbf{z}_G) - \tau^{\widehat{\mathbf{w}}}(\mathbf{z})\right| \leq K \left\|\mathbf{z}_G - \mathbf{z}\right\|_\infty \leq \frac{K\varepsilon}{2}. \tag{3.21}$$

Since every constraint in the relaxed ε-HALP formulation is satisfied, $\tau^{\widehat{\mathbf{w}}}(\mathbf{z}_G)$ is nonnegative for all grid points $\mathbf{z}_G$. Hence, Equation 3.21 yields $\tau^{\widehat{\mathbf{w}}}(\mathbf{z}) > -K\varepsilon/2$ for every state-action pair $\mathbf{z} = (\mathbf{x}, \mathbf{a})$. Based on Definition 3, the solution $\widehat{\mathbf{w}}$ is δ-infeasible for $\delta \geq K\varepsilon/2$. Conversely, the δ-infeasibility of $\widehat{\mathbf{w}}$ is guaranteed by choosing $\varepsilon \leq 2\delta/K$.

Unfortunately, K often increases rapidly with the dimensionality of a function. To address this concern, we use the structure in the constraint space and demonstrate that this is not our case. First, note that the *global Lipschitz constant* K_{glob} is additive in *local Lipschitz constants* that correspond to the terms $\widehat{w}_i F_i(\mathbf{z})$ and $R_j(\mathbf{z})$. Moreover, $K_{\text{glob}} \leq N K_{\text{loc}}$, where N denotes the total number of the terms and K_{loc} is the maximum over the local constants. Finally, parallel to Equation 3.21, the δ-infeasibility of a relaxed HALP solution $\widehat{\mathbf{w}}$ is achieved by the discretization:

$$\varepsilon \leq \frac{2\delta}{N K_{\text{loc}}} \leq \frac{2\delta}{K_{\text{glob}}}. \tag{3.22}$$

Since the terms $\widehat{w}_i F_i(\mathbf{z})$ and $R_j(\mathbf{z})$ are usually restricted to small subsets of state and action variables, K_{loc} should change a little when the size of a problem increases but its structure is kept fixed. To prove that K_{loc} is bounded, we have to bound the weights $\widehat{w}_i$. If all basis functions are of unit magnitude, the weights $\widehat{w}_i$ are intuitively bounded as $|\widehat{w}_i| \leq (1-\gamma)^{-1} R_{\max}$, where $R_{\max}$ denotes the maximum one-step reward in the HMDP.

Based on Equation 3.22, we conclude that the number of discretization points in a single dimension $\lceil 1/\varepsilon + 1 \rceil$ is bounded by a polynomial in N, K_{loc}, and $1/\delta$. Hence, the constraint space in the relaxed ε-HALP formulation involves $O([N K_{\text{loc}}(1/\delta)]^{|\mathbf{X}|+|\mathbf{A}|})$ constraints, where $|\mathbf{X}|$ and $|\mathbf{A}|$ denote the number of state and action variables. The idea of variable elimination can be applied to write the constraints compactly by $O([N K_{\text{loc}}(1/\delta)]^{T+1}(|\mathbf{X}|+|\mathbf{A}|))$ constraints (Example 3), where T is the treewidth of a corresponding cost network. As a result, satisfying this constraint space is polynomial in N, K_{loc}, $1/\delta$, $|\mathbf{X}|$, and $|\mathbf{A}|$, but still exponential in T.

Inputs:

a hybrid factored MDP $\mathcal{M} = (\mathbf{X}, \mathbf{A}, P, R)$
basis functions $f_0(\mathbf{x}), f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots$
initial basis function weights $\mathbf{w}^{(0)}$
a separation oracle $\mathcal{O}$

Algorithm:

initialize a relaxed HALP formulation with an empty set of constraints
$t = 0$
while a stopping criterion is not met
    query the oracle $\mathcal{O}$ for a violated constraint $(\mathbf{x}_{\mathcal{O}}, \mathbf{a}_{\mathcal{O}})$ with respect to $\mathbf{w}^{(t)}$
    if the constraint $(\mathbf{x}_{\mathcal{O}}, \mathbf{a}_{\mathcal{O}})$ is violated
        add the constraint to the relaxed HALP
    resolve the LP for a new vector $\mathbf{w}^{(t+1)}$
    $t = t + 1$

Outputs:

basis function weights $\mathbf{w}^{(t)}$

Figure 11: Pseudo-code implementation of a HALP solver with the cutting plane method.

3.3.3 Cutting Plane Method

Both MC and ε-HALP formulations (Sections 3.3.1 and 3.3.2) approximate the constraint space in HALP by a finite set of constraints $\mathcal{C}$. Therefore, these formulations can be solved directly by any linear programming solver. However, if the number of constraints is large, formulating and solving LPs with the complete set of constraints is infeasible. In this section, we show how to build relaxed HALP approximations efficiently by the cutting plane method.

The cutting plane method for solving HALP formulations is outlined in Figure 11. Briefly, this approach builds the set of LP constraints incrementally by adding a violated constraint to this set in every step. In the rest of this thesis, we refer to any method that returns a violated constraint for an arbitrary vector $\widehat{\mathbf{w}}$ as a *separation oracle*. Formally, the HALP oracle approaches the optimization problem:

$$\arg\min_{\mathbf{x},\mathbf{a}} \left[V^{\widehat{\mathbf{w}}}(\mathbf{x}) - \gamma \mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})} \left[V^{\widehat{\mathbf{w}}}(\mathbf{x}') \right] - R(\mathbf{x},\mathbf{a}) \right] . \tag{3.23}$$

In turn, the problem of solving hybrid factored MDPs efficiently reduces to the design of efficient separation oracles. The cutting plane method (Figure 11) can be applied to suboptimal solutions to Equation 3.23 if these correspond to violated constraints.

The presented approach can be directly used to satisfy the constraints in relaxed ε-HALP formulations [85]. Briefly, the solver from Figure 11 iterates until no violated constraint is found and the ε-HALP separation oracle $\mathcal{O}_\varepsilon$ (Figure 12) returns the most violated constraint in the discretized cost network given an intermediate solution $\mathbf{w}^{(t)}$. Note that although the search for the most violated constraint is polynomial in $|\mathbf{X}|$ and $|\mathbf{A}|$ (Section 3.3.2.2), the running time of our solver does not have to be [34]. In fact, the number of generated cuts is exponential in $|\mathbf{X}|$ and $|\mathbf{A}|$ in the worst case. However, the same oracle embedded into the ellipsoid method [52] yields a polynomial-time algorithm [11]. Although this technique is impractical for solving large LPs, we may conclude that our approach is indeed polynomial-time if implemented in this particular way.

Finally, note that search for the most violated constraint (Equation 3.23) has application beyond satisfying the constraint space in HALP. For instance, computation of a greedy policy for the value

Inputs:

a hybrid factored MDP $\mathcal{M} = (\mathbf{X}, \mathbf{A}, P, R)$
basis functions $f_0(\mathbf{x}), f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots$
basis function weights $\mathbf{w}$
grid resolution ε

Algorithm:

discretize continuous variables $\mathbf{X}_C$ and $\mathbf{A}_C$ into ($\lceil 1/\varepsilon + 1 \rceil$) equally-spaced values
identify subsets $\mathbf{X}_i$ and $\mathbf{A}_i$ ($\mathbf{X}_j$ and $\mathbf{A}_j$) corresponding to the domains of $F_i(\mathbf{x}, \mathbf{a})$ ($R_j(\mathbf{x}, \mathbf{a})$)
evaluate $F_i(\mathbf{x}_i, \mathbf{a}_i)$ ($R_j(\mathbf{x}_j, \mathbf{a}_j)$) for all configurations $\mathbf{x}_i$ and $\mathbf{a}_i$ ($\mathbf{x}_j$ and $\mathbf{a}_j$) on the ε-grid
build a cost network for the factored cost function:
$\tau^{\mathbf{w}}(\mathbf{x}, \mathbf{a}) = \sum_i w_i F_i(\mathbf{x}, \mathbf{a}) - R(\mathbf{x}, \mathbf{a})$
find the most violated constraint in the cost network:
$(\mathbf{x}_{\mathcal{O}}, \mathbf{a}_{\mathcal{O}}) = \arg\min_{\mathbf{x},\mathbf{a}} \tau^{\mathbf{w}}(\mathbf{x}, \mathbf{a})$

Outputs:

state-action pair $(\mathbf{x}_{\mathcal{O}}, \mathbf{a}_{\mathcal{O}})$

Figure 12: Pseudo-code implementation of the ε-HALP separation oracle $\mathcal{O}_\varepsilon$.

function $V^{\widehat{\mathbf{w}}}$:

$$\begin{aligned} u(\mathbf{x}) &= \arg\max_{\mathbf{a}} \left[R(\mathbf{x}, \mathbf{a}) + \gamma \mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})} \left[V^{\widehat{\mathbf{w}}}(\mathbf{x}') \right] \right] \\ &= \arg\min_{\mathbf{a}} \left[-R(\mathbf{x}, \mathbf{a}) - \gamma \mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})} \left[V^{\widehat{\mathbf{w}}}(\mathbf{x}') \right] \right] \end{aligned} \tag{3.24}$$

is almost an identical optimization problem, where the state variables $\mathbf{X}$ are kept fixed. Moreover, the magnitude of the most violated constraint is equal to the lowest δ for which the relaxed HALP solution $\widehat{\mathbf{w}}$ is δ-infeasible (Equation 3.18):

$$\begin{aligned} \underline{\delta} &= \min_{\mathbf{x}} \left[V^{\widehat{\mathbf{w}}}(\mathbf{x}) - \max_{\mathbf{a}} \left[R(\mathbf{x}, \mathbf{a}) + \gamma \mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})} \left[V^{\widehat{\mathbf{w}}}(\mathbf{x}') \right] \right] \right] \\ &= \min_{\mathbf{x},\mathbf{a}} \left[V^{\widehat{\mathbf{w}}}(\mathbf{x}) - R(\mathbf{x}, \mathbf{a}) - \gamma \mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})} \left[V^{\widehat{\mathbf{w}}}(\mathbf{x}') \right] \right] . \end{aligned} \tag{3.25}$$

3.3.4 MCMC-HALP

In practice, both the MC and ε-HALP formulation (Sections 3.3.1 and 3.3.2) are built on a blindly selected set of constraints $\mathcal{C}$. More specifically, the constraints in the MC-HALP formulation are chosen randomly (with respect to a prior distribution φ) while the ε-HALP formulation is based on a uniform ε-grid. This discretized constraint space preserves its original factored structure, which allows for its compact satisfaction. However, the complexity of solving the ε-HALP formulation grows exponentially in the treewidth of its discretized constraint space. Note that if the discretized constraint space is represented by binary variables only, the treewidth increases by a multiplicative factor $\log_2 \lceil 1/\varepsilon + 1 \rceil$, where $\lceil 1/\varepsilon + 1 \rceil$ is the number of discretization points in a single dimension. As a result, although the treewidth of a problem is relatively small, solving its ε-HALP formulation becomes intractable for small values of ε.

To address the issues of the discussed approximations (Sections 3.3.1 and 3.3.2), we propose a novel Markov chain Monte Carlo (MCMC) method for finding the most violated constraint of a relaxed HALP. The procedure directly operates in the domains of continuous variables, takes into account the structure of factored MDPs, and its space complexity is proportional to the number of variables. This separation oracle can be easily embedded into the ellipsoid or cutting plane method for solving linear programs (Section 3.3.3), and therefore constitutes a crucial step towards solving HALP efficiently. Before we proceed, we represent the constraint space in HALP compactly and state an optimization problem for finding violated constraints in this factored representation.

3.3.4.1 Compact Representation of Constraints

In Section 2.3.2.3, we showed how the factored representation of the constraint space allows for its compact satisfaction. Following this idea, we define *violation magnitude* $\tau^{\mathbf{w}}(\mathbf{x}, \mathbf{a})$:

$$\begin{aligned} \tau^{\mathbf{w}}(\mathbf{x}, \mathbf{a}) &= -\left[V^{\mathbf{w}}(\mathbf{x}) - \gamma \mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})}[V^{\mathbf{w}}(\mathbf{x}')] - R(\mathbf{x}, \mathbf{a})\right] \\ &= -\sum_i w_i [f_i(\mathbf{x}) - \gamma g_i(\mathbf{x}, \mathbf{a})] + R(\mathbf{x}, \mathbf{a}), \end{aligned} \tag{3.26}$$

which measures the amount by which the solution $\mathbf{w}$ violates the constraints in the complete HALP. We represent the violation magnitude $\tau^{\mathbf{w}}(\mathbf{x}, \mathbf{a})$ compactly by an influence diagram (ID), where $\mathbf{X}$ and $\mathbf{A}$ are decision nodes, and $\mathbf{X}'$ are random variables. This representation is built on the transition

model $P(\mathbf{X}' \mid \mathbf{X}, \mathbf{A})$, which is factored and captures independencies among the variables $\mathbf{X}$, $\mathbf{X}'$, and $\mathbf{A}$. We extend the diagram by three types of reward nodes, one for each term in Equation 3.26: $H_i = -w_i f_i(\mathbf{x})$ for every basis function, $G_i = \gamma w_i f_i(\mathbf{x}')$ for its backprojection, and $R_j = R_j(\mathbf{x}_j, \mathbf{a}_j)$ for every reward function. The construction is completed by adding arcs that graphically represent the dependencies of the reward nodes on the variables. Finally, we can verify that:

$$\tau^{\mathbf{w}}(\mathbf{x}, \mathbf{a}) = \mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})}\left[\sum_i (H_i + G_i) + \sum_j R_j\right]. \tag{3.27}$$

As a result, the decision that maximizes the expected utility in the influence diagram corresponds to the most violated constraint. A graphical representation of the violation magnitude $\tau^{\mathbf{w}}(\mathbf{x}, \mathbf{a})$ on the 4-ring network administration problem (Example 4) is given in Figure 5a. The structure of the constraint space is identical to Example 3 if the basis functions are univariate.

We conclude that any method for solving IDs can be used to find the most violated constraint. However, most of these methods [20, 49, 78] are restricted to discrete variables only. Fortunately, special properties of the ID representation allow for its further simplification. If the basis functions are chosen conjugate to the transition model (Section 3.2.2), we obtain closed-form solutions to the expectation term $\mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})}[G_i]$ (Equation 3.8), and the random variables $\mathbf{X}'$ are marginalized out of the diagram. The new representation contains no random variables and is known as a cost network (Section 2.3.2.3).

Note that the problem of finding the most violated constraint in the ID representation is almost identical to finding the maximum a posteriori (MAP) configuration of random variables in Bayesian networks [26, 79, 80, 102]. The latter problem is difficult because of the alternating maximization and summation operators. Since we have marginalized out the random variables $\mathbf{X}'$, we can solve the maximization problem by standard large-scale optimization techniques.

3.3.4.2 Separation Oracle $\mathcal{O}_{\mathrm{MCMC}}$

To find the most violated constraint in the cost network, we utilize the Metropolis-Hastings (MH) algorithm [70, 41] and propose a Markov chain whose invariant distribution converges to the vicinity of $\arg\max_{\mathbf{z}} \tau^{\mathbf{w}}(\mathbf{z})$, where $\mathbf{z} = (\mathbf{x}, \mathbf{a})$ is a value assignment to the joint set of state and action variables $\mathbf{Z} = \mathbf{X} \cup \mathbf{A}$.

In short, the Metropolis-Hastings algorithm defines a Markov chain that transits from an existing state $\mathbf{z}$ to a proposed state $\mathbf{z}^*$ with the *acceptance probability*:

$$A(\mathbf{z}, \mathbf{z}^*) = \min\left\{1, \frac{p(\mathbf{z}^*)q(\mathbf{z} \mid \mathbf{z}^*)}{p(\mathbf{z})q(\mathbf{z}^* \mid \mathbf{z})}\right\}, \tag{3.28}$$

where $q(\mathbf{z}^* \mid \mathbf{z})$ and $p(\mathbf{z})$ are a *proposal distribution* and a *target density*, respectively. Under mild restrictions on both $p(\mathbf{z})$ and $q(\mathbf{z}^* \mid \mathbf{z})$, the frequency of state visits generated by the Markov chain always converges to the target function $p(\mathbf{z})$ [1]. In the rest of this section, we discuss the choices of $p(\mathbf{z})$ and $q(\mathbf{z}^* \mid \mathbf{z})$ to solve our optimization problem.[4]

Target density: The violation magnitude $\tau^{\mathbf{w}}(\mathbf{z})$ is turned into a density by a simple transformation $p(\mathbf{z}) = \exp[\tau^{\mathbf{w}}(\mathbf{z})]$. Due to its monotonic character, $p(\mathbf{z})$ retains the same set of global maxima as $\tau^{\mathbf{w}}(\mathbf{z})$. Therefore, the search for $\arg\max_{\mathbf{z}} \tau^{\mathbf{w}}(\mathbf{z})$ can be done on the new function $p(\mathbf{z})$. To prove that $p(\mathbf{z})$ is a density, we show that $\sum_{\mathbf{z}_D} \int_{\mathbf{z}_C} p(\mathbf{z})\, \mathrm{d}\mathbf{z}_C$ is its normalizing constant, where $\mathbf{z}_D$ and $\mathbf{z}_C$ are the discrete and continuous parts of the value assignment $\mathbf{z}$. First, note that the integrand $\mathbf{z}_C$ is restricted to the space $[0,1]^{|\mathbf{z}_C|}$. As a result, the integral $\int_{\mathbf{z}_C} p(\mathbf{z})\, \mathrm{d}\mathbf{z}_C$ is proper if $p(\mathbf{z})$ is bounded, and thus it is Riemann integrable and finite. To prove that $p(\mathbf{z}) = \exp[\tau^{\mathbf{w}}(\mathbf{z})]$ is bounded, we bound the magnitude of violation $\tau^{\mathbf{w}}(\mathbf{z})$. If all basis functions are of unit magnitude, the weights $\widehat{w}_i$ can be bounded as $|\widehat{w}_i| \leq (1-\gamma)^{-1} R_{\max}$ (Section 3.3.2.2), which in turn leads to the following bound $|\tau^{\mathbf{w}}(\mathbf{z})| \leq (|\mathbf{w}|\,(1-\gamma)^{-1} + 1) R_{\max}$. Therefore, the function $p(\mathbf{z})$ is bounded and can be treated as a density.

To find the mode of $p(\mathbf{z})$, we use simulating annealing [53] and generate a non-homogeneous Markov chain whose invariant distribution is equal to $p^{1/T_t}(\mathbf{z})$, where T_t is a cooling schedule such that $\lim_{t\to\infty} T_t = 0$. Under weak regularity assumptions on $p(\mathbf{z})$, $p^{\infty}(\mathbf{z})$ is a probability distribution that concentrates on the set of the global maxima of $p(\mathbf{z})$ [1]. If our cooling schedule T_t decreases such that $T_t \geq c/\ln(t+1)$, where c is a problem-specific constant, the chain from Equation 3.28 converges to the vicinity of $\arg\max_{\mathbf{z}} \tau^{\mathbf{w}}(\mathbf{z})$ with the probability that converges to 1 [30]. However, this logarithmic cooling schedule is often slow in practice, especially for a high initial temperature c. To overcome this problem, we select a smaller value of c [30] than is required by the convergence

Inputs:

a hybrid factored MDP $\mathcal{M} = (\mathbf{X}, \mathbf{A}, P, R)$
basis functions $f_0(\mathbf{x}), f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots$
basis function weights $\mathbf{w}$

Algorithm:

initialize a state-action pair $\mathbf{z}^{(t)}$
$t = 0$
while a stopping criterion is not met
  for every variable Z_i
    sample $u \sim \mathrm{U}_{[0,1]}$
    sample $z_i^* \sim p(Z_i \mid \mathbf{z}_{-i}^{(t)})$
    if $u < \min\left\{1, \frac{p^{1/T_t - 1}(z_i^* | \mathbf{z}_{-i}^{(t)})}{p^{1/T_t - 1}(z_i^{(t)} | \mathbf{z}_{-i}^{(t)})}\right\}$
      $z_i^{(t+1)} = z_i^*$
    else
      $z_i^{(t+1)} = z_i^{(t)}$
  update T_{t+1} according to the cooling schedule
  $t = t + 1$
$(\mathbf{x}_\mathcal{O}, \mathbf{a}_\mathcal{O}) = \mathbf{z}^{(t)}$

Outputs:

state-action pair $(\mathbf{x}_\mathcal{O}, \mathbf{a}_\mathcal{O})$

Figure 13: Pseudo-code implementation of the MCMC-HALP oracle $\mathcal{O}_{\mathrm{MCMC}}$. The symbol $\mathrm{U}_{[0,1]}$ represents the uniform distribution on the interval $[0, 1]$. Since the testing for violated constraints (Figure 11) is inexpensive, our implementation of the MCMC-HALP solver in Section 3.4 tests all constraints $\mathbf{z}^{(t)}$ generated by the Markov chain and not only the last one. Therefore, the separation oracle $\mathcal{O}_{\mathrm{MCMC}}$ returns more than one constraint per chain.

criterion. Hence, the convergence of our chain to the global optimum $\arg\max_{\mathbf{z}} \tau^{\mathbf{w}}(\mathbf{z})$ is no longer guaranteed.

Proposal distribution: We take advantage of the factored character of $\mathbf{Z}$ and adopt the following proposal distribution [30]:

$$q(\mathbf{z}^* \mid \mathbf{z}) = \begin{cases} p(z_i^* \mid \mathbf{z}_{-i}) & \text{if } \mathbf{z}_{-i}^* = \mathbf{z}_{-i} \\ 0 & \text{otherwise} \end{cases}, \tag{3.29}$$

where $\mathbf{z}_{-i}$ and $\mathbf{z}_{-i}^*$ are value assignments to all variables but Z_i in the original and proposed states. If Z_i is a discrete variable, its conditional:

$$p(z_i^* \mid \mathbf{z}_{-i}) = \frac{p(z_1, \ldots, z_{i-1}, z_i^*, z_{i+1}, \ldots, z_{n+m})}{\sum_{z_i} p(z_1, \ldots, z_{i-1}, z_i, z_{i+1}, \ldots, z_{n+m})} \tag{3.30}$$

has a closed form. If Z_i is a continuous variable, a closed form of its cumulative density function is unlikely to exist. To allow sampling from the conditional, we embed another Metropolis-Hastings step within the original chain. In the experimental section, we apply the Metropolis algorithm with the acceptance probability:

$$A(z_i, z_i^*) = \min\left\{1, \frac{p(z_i^* \mid \mathbf{z}_{-i})}{p(z_i \mid \mathbf{z}_{-i})}\right\}, \tag{3.31}$$

where z_i and z_i^* denote the original and proposed values of the variable Z_i, respectively. Note that sampling from both conditionals can be performed in the space of $\tau^{\mathbf{w}}(\mathbf{z})$ and locally.

Finally, by assuming that $\mathbf{z}_{-i}^* = \mathbf{z}_{-i}$ (Equation 3.29), we derive a non-homogenous Markov chain with the acceptance probability:

$$\begin{aligned} A(\mathbf{z}, \mathbf{z}^*) &= \min\left\{1, \frac{p^{1/T_t}(\mathbf{z}^*) q(\mathbf{z} \mid \mathbf{z}^*)}{p^{1/T_t}(\mathbf{z}) q(\mathbf{z}^* \mid \mathbf{z})}\right\} \\ &= \min\left\{1, \frac{p^{1/T_t}(z_i^* \mid \mathbf{z}_{-i}^*) p^{1/T_t}(\mathbf{z}_{-i}^*) p(z_i \mid \mathbf{z}_{-i}^*)}{p^{1/T_t}(z_i \mid \mathbf{z}_{-i}) p^{1/T_t}(\mathbf{z}_{-i}) p(z_i^* \mid \mathbf{z}_{-i})}\right\} \\ &= \min\left\{1, \frac{p^{1/T_t}(z_i^* \mid \mathbf{z}_{-i}) p^{1/T_t}(\mathbf{z}_{-i}) p(z_i \mid \mathbf{z}_{-i})}{p^{1/T_t}(z_i \mid \mathbf{z}_{-i}) p^{1/T_t}(\mathbf{z}_{-i}) p(z_i^* \mid \mathbf{z}_{-i})}\right\} \\ &= \min\left\{1, \frac{p^{1/T_t - 1}(z_i^* \mid \mathbf{z}_{-i})}{p^{1/T_t - 1}(z_i \mid \mathbf{z}_{-i})}\right\}, \end{aligned} \tag{3.32}$$

which converges to the vicinity of the most violated constraint. Yuan *et al.* [102] proposed a similar chain for finding the MAP configuration of random variables in Bayesian networks.

[4] For an introduction to Markov chain Monte Carlo (MCMC) methods, refer to the work of Andrieu *et al.* [1].

3.3.4.3 Constraint Satisfaction

If the MCMC-HALP separation oracle $\mathcal{O}_{\text{MCMC}}$ (Figure 13) converges to a violated constraint (not necessarily the most violated) in polynomial time, the ellipsoid method [52] is guaranteed to solve HALP formulations in polynomial time [11]. Unfortunately, convergence of our chain within arbitrary precision requires an exponential number of steps [30]. Although the bound is loose to be of practical interest, it suggests that the time complexity of proposing violated constraints dominates the time complexity of solving relaxed HALP formulations. Therefore, the oracle $\mathcal{O}_{\text{MCMC}}$ should search for violated constraints efficiently. Convergence speedups that directly apply to our method include hybrid Monte Carlo (HMC) [27], Rao-Blackwellization [18], and slice sampling [44].

3.4 EXPERIMENTS

Experimental section is divided in three parts. First, we show that HALP can solve a simple hybrid MDP problem at least as efficiently as alternative approaches. Second, we demonstrate the scale-up potential of our framework and compare several approaches to satisfy the constraint space in HALP (Section 3.3). Finally, we argue for solving our constraint satisfaction problem in the domains of continuous variables without discretizing them.

All experiments were performed on a Dell Precision 380 workstation with 3.2GHz Pentium 4 CPU and 2GB RAM. Linear programs are solved by the simplex method in the LP_SOLVE package. The expected return of policies is estimated by the Monte Carlo simulation of 100 trajectories. The results of randomized algorithms are additionally averaged over 10 randomly initialized runs. Whenever necessary, we present errors on the expected values. The discount factor γ is 0.95.

3.4.1 A Simple Example

To illustrate the ability of HALP to solve factored MDPs, we compare it to $\mathcal{L}_2$ (Figure 4) and grid-based value iteration (Section 2.3.4) on the 4-ring topology of the network administration problem (Example 4). Our experiments are conducted on uniform and non-uniform grids of varying sizes. Grid points are kept fixed for all compared methods, which allows for their fair comparison. Both

value iteration methods are iterated for up to 100 steps and terminated earlier if their Bellman error drops below 10^{-6}. Both the $\mathcal{L}_2$ and HALP methods approximate the optimal value function V^* by a linear combination of basis functions, one for each computer X_i ($f_i(\mathbf{x}) = x_i$), and one for every connection $X_i \to X_j$ in the ring topology ($f_{i\to j}(\mathbf{x}) = x_i x_j$). We assume that our basis functions are sufficient to derive a one-step lookahead policy that reboots the least efficient computer. We believe that this policy is close-to-optimal in the ring topology. The constraint space in the complete HALP formulation is approximated by its MC-HALP and ε-HALP relaxations (Sections 3.3.1 and 3.3.2). The state relevance density function $\psi(\mathbf{x})$ is uniform. The results of our experiments are reported in Figure 14.

To verify that our solutions are non-trivial, we compare them to three heuristic policies: dummy, random, and server. The dummy policy $\pi_{\text{dummy}}(\mathbf{x}) = a_5$ always takes the dummy action a_5. Therefore, it establishes a lower bound on the performance of any administrator. The random policy takes random actions. The server policy $\pi_{\text{server}}(\mathbf{x}) = a_1$ protects the server X_1. The performance of our heuristics is shown in Figure 14. Assuming that we can reboot all computers at each time step, a utopian upper bound on the performance of any policy π can be derived as:

$$\begin{aligned}
\mathrm{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R(\mathbf{x}_t, \pi(\mathbf{x}_t))\right] &\leq \frac{1}{1-\gamma} \max_{\mathbf{x},a} \mathrm{E}_{P(\mathbf{x}'|\mathbf{x},a)}\left[\max_{a'} R(\mathbf{x}', a')\right] \\
&= \frac{1}{1-\gamma} \max_{\mathbf{x},a} \int_{\mathbf{x}'} 2P(x_1' \mid \mathbf{x}, a) x_1'^2 + \sum_{j=2}^{4} P(x_j' \mid \mathbf{x}, a) x_j'^2 \, \mathrm{d}\mathbf{x}' \\
&\leq \frac{5}{1-\gamma} \int_{x'} P_{\text{beta}}(x' \mid 20, 2) x'^2 \, \mathrm{d}x' \\
&\approx 83.0. \qquad (3.33)
\end{aligned}$$

We do not analyze the quality of HALP solutions with respect to the optimal value function V^* (Section 3.2.1) because this one is unknown.

Based on our results, we draw the following conclusions. First, grid-based value iteration is not practical for solving hybrid optimization problems of even small size. The main reason is the space complexity of the method, which is quadratic in the number of grid points N. If the state space is discretized uniformly, N grows exponentially in the number of state variables. Second, the quality of the HALP policies is close to the $\mathcal{L}_2$ VI policies. This result is positive since $\mathcal{L}_2$ value iteration is often applied in large-scale approximate dynamic programming (Section 2.3.2). Third, both the

Uniform ε-grid

		ε-HALP		$\mathcal{L}_2$ VI		Grid-based VI	
ε	N	Reward	Time	Reward	Time	Reward	Time
1	8	52.1 ± 0.2	< 1	52.1 ± 0.2	2		
1/2	91	52.1 ± 0.2	< 1	52.1 ± 0.2	7	47.6 ± 0.2	< 1
1/4	625	52.1 ± 0.2	< 1	52.1 ± 0.2	55	51.5 ± 0.2	20
1/8	6 561	52.1 ± 0.2	2	52.1 ± 0.2	577	52.0 ± 0.2	2 216

Heuristics

Policy	Reward
Dummy	25.0 ± 0.3
Random	42.1 ± 0.3
Server	47.6 ± 0.2
Utopian	83.0

Non-uniform grid

	MC-HALP		$\mathcal{L}_2$ VI		Grid-based VI	
N	Reward	Time	Reward	Time	Reward	Time
10	45.2 ± 0.5	< 1	45.9 ± 0.6	1	47.5 ± 0.3	< 1
50	50.2 ± 0.2	< 1	51.8 ± 0.2	4	48.7 ± 0.3	< 1
250	51.5 ± 0.2	< 1	51.9 ± 0.2	22	50.4 ± 0.2	2
1 250	51.8 ± 0.2	< 1	51.9 ± 0.2	110	51.6 ± 0.2	60

Figure 14: Comparison of three approaches to solving hybrid factored MDPs on the 4-ring topology of the network administration problem (Example 4). The approaches are compared on uniform and non-uniform grids of varying size (N) by the expected discounted reward of policies and their computation time (in seconds).

$\mathcal{L}_2$ and HALP approaches yield better policies than grid-based value iteration. This result is caused by the quality of our value function estimator $V^{\mathbf{w}}$. Its good performance for $\varepsilon = 1$ can be explained from the monotonicity of the reward and basis functions. Finally, the computation time of the $\mathcal{L}_2$ VI policies is significantly longer than the computation time of the HALP policies. Since a step of $\mathcal{L}_2$ value iteration (Figure 4) is as hard as formulating a corresponding relaxed HALP, this result comes at no surprise.

3.4.2 Scale-up Potential

To show the scale-up potential of the HALP approach, we apply three relaxed HALP approximations (Section 3.3) to solve two irrigation network problems of varying complexity. These problems are challenging for state-of-the-art MDP solvers due to the factored state and action spaces.

Example 6 (Irrigation network operator) *An irrigation network is a system of irrigation channels connected by regulation devices (Figure 15). The goal of an irrigation network operator is to*

route water between the channels to optimize water levels in the whole system. The optimal levels are given by the type of a planted crop. For simplicity of exposition, we assume that all irrigation channels are oriented and of the same size.

This optimization problem can be formulated as a factored MDP. The state of the network is completely observable and represented by n continuous variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, where the variable X_i denotes the water level in the i-th channel. At each time step, the irrigation network operator regulates m devices A_i that pump water between any pair of their inbound and outbound channels. Their operation modes are represented by discrete action variables $\mathbf{A} = \{A_1, \ldots, A_m\}$. Inflow and outflow devices (no inbound or outbound channels) are not controlled and pump water in and out of the network.

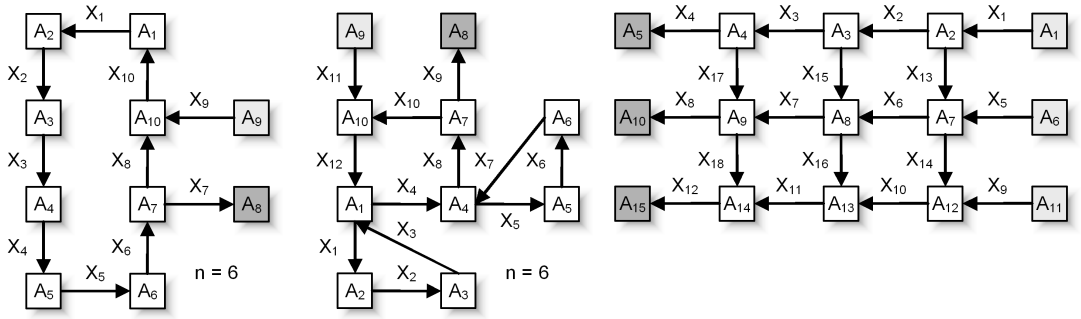
The transition function reflects water flows in the irrigation network and is encoded locally by conditioning on the operation modes $\mathbf{A}$:

$$
\begin{aligned}
P(X'_{i\to j} = x \mid \mathsf{Par}(X'_{i\to j})) \;&\propto\; P_{\text{beta}}(x \mid \alpha, \beta) \quad \Big| \quad \alpha = 46\mu'_{i\to j} + 2 \\
&\qquad\qquad\qquad\qquad\quad\; \beta = 46(1 - \mu'_{i\to j}) + 2 \\
\mu'_{i\to j} \;&=\; \mu_{i\to j} + \sum_h \mathbf{1}_{a_{h\to i\to j}}(A_i)\min(1 - \mu_{i\to j}, \min(x_{h\to i}, \tau_i)) \\
\mu_{i\to j} \;&=\; x_{i\to j} - \sum_k \mathbf{1}_{a_{i\to j\to k}}(A_j)\min(x_{i\to j}, \tau_j)
\end{aligned}
$$

where $X_{i\to j}$ represents the water level between the regulation devices A_i and A_j, $\mathbf{1}_{a_{h\to i\to j}}(A_i)$ and $\mathbf{1}_{a_{i\to j\to k}}(A_j)$ are the indicator functions of water routing actions $a_{h\to i\to j}$ and $a_{i\to j\to k}$ at the devices A_i and A_j, and τ_i and τ_j denote the tolerated flows through these devices. In short, this transition function conserves water mass in the network and adds some variance to the resulting state $X'_{i\to j}$. The introduced indexing of state and action variables is explained on the 6-ring irrigation network in Figure 16a. In the rest of the thesis, we assume an inflow of 0.1 to any inflow device A_i ($\tau_i = 0.1$), an outflow of 1 from any outflow device A_j ($\tau_j = 1$), and the tolerated flow of $1/3$ at the remaining devices A_k ($\tau_k = 1/3$).

The reward model $R(\mathbf{x}, \mathbf{a}) = \sum_j R_j(x_j)$ is factored along individual irrigation channels and described by the univariate function:

$$R_j(x_j) = 2x_j$$



(a) **(b)** **(c)**

Figure 15: Illustrations of three irrigation network topologies: **a.** 6-ring, **b.** 6-ring-of-rings, and **c.** 3×3 grid. Irrigation channels and their regulation devices are represented by arrows and rectangles. Inflow and outflow nodes are colored in light and dark gray. The ring and ring-of-rings networks are parameterized by the total number of regulation devices except for the last four (n).

for each outflow channel (one of its regulation devices must be outflow), and by the function:

$$R_j(x_j) = \frac{\mathcal{N}(x_j \mid 0.4, 0.025)}{25.6} + \frac{\mathcal{N}(x_j \mid 0.55, 0.05)}{32}$$

for the remaining channels (Figure 16b). Therefore, we reward both for maintaining optimal water levels and pumping water out of the irrigation network. Several examples of irrigation network topologies are shown in Figure 15.

Similarly to Equation 3.33, we derive a utopian upper bound on the performance of any policy π in an arbitrary irrigation network as:

$$\mathrm{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R(\mathbf{x}_t, \pi(\mathbf{x}_t))\right] \leq \frac{1}{1-\gamma}\Bigg[0.2n_{\text{in}} + (n - n_{\text{out}}) \max_x \int_{x'} P_{\text{beta}}(x' \mid 46x + 2, 46(1-x) + 2)R(x')\,\mathrm{d}x'\Bigg], \quad (3.34)$$

where n is the total number of irrigation channels, n_{in} and n_{out} denote the number of inflow and outflow channels, respectively, and $R(x') = \mathcal{N}(x' \mid 0.4, 0.025)/25.6 + \mathcal{N}(x' \mid 0.55, 0.05)/32$. We do

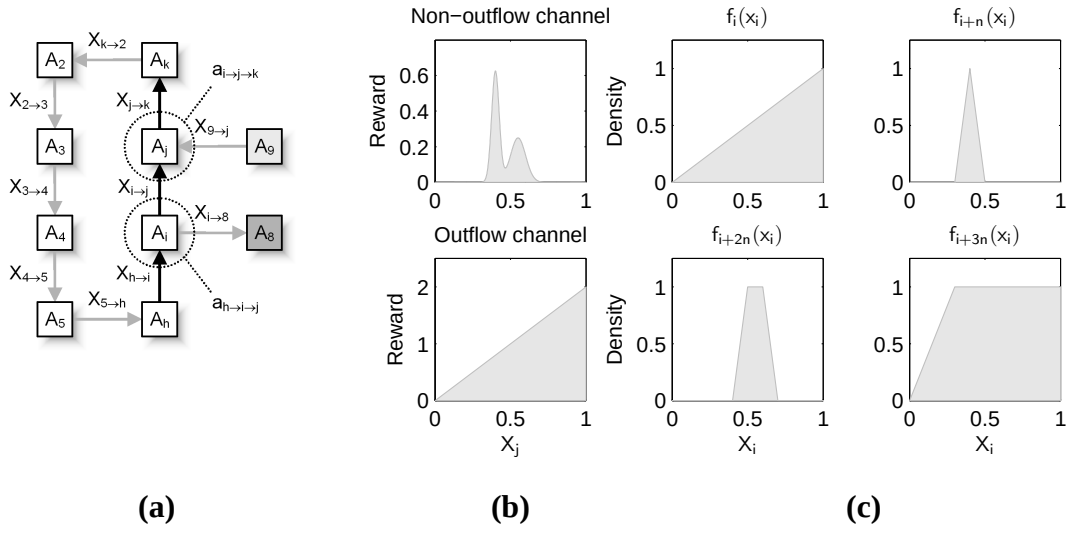


Figure 16: **a.** Indexing used in the description of the transition model in Example 6. The parameters h, i, j, and k are equal to 6, 7, 10, and 1, respectively. **b.** Univariate reward functions over water levels X_j (Example 6). **c.** Univariate basis functions over water levels X_i.

not analyze the quality of HALP solutions with respect to the optimal value function V^* (Section 3.2.1) because this one is unknown.

In the rest of the section, we study the performance of three HALP approximations, MC-HALP, ε-HALP, and MCMC-HALP (Section 3.3), on the ring and ring-of-rings topologies (Figure 15) of the irrigation network problem. The constraints in the MC-HALP relaxation are chosen uniformly at random. This establishes a baseline for the quality of HALP approximations. The ε-HALP and MCMC-HALP formulations are built iteratively by the cutting plane method. The MCMC oracle $\mathcal{O}_{\text{MCMC}}$ is simulated for 500 steps from the initial temperature $c = 0.2$, which yields a decreasing cooling schedule from $T_0 = 0.2$ to $T_{500} \approx 0.02$. These parameters are selected to demonstrate the characteristics of the oracle $\mathcal{O}_{\text{MCMC}}$ rather than to maximize its performance. The value function V^* is approximated by a linear combination of four univariate piecewise linear basis functions for each irrigation channel (Figure 16c). We assume that our basis functions are sufficient to derive a one-step lookahead policy that routes water between the channels if their water levels are too high or too low (Figure 16b). We believe that this policy is close-to-optimal in irrigation networks. The state relevance density function $\psi(\mathbf{x})$ is uniform. Our experimental results are reported in Figures

Ring topology		$n = 6$			$n = 12$			$n = 18$		
		OV	Reward	Time	OV	Reward	Time	OV	Reward	Time
ε-**HALP**	$1/4$	24.3	34.6 ± 0.2	11	36.2	53.9 ± 0.3	44	48.0	74.3 ± 0.3	87
$\varepsilon =$	$1/8$	55.4	39.6 ± 0.3	41	88.1	61.5 ± 0.4	107	118.8	84.3 ± 0.4	178
	$1/16$	59.1	40.3 ± 0.3	281	93.2	62.6 ± 0.3	665	126.1	86.3 ± 0.4	1 119
MCMC	10	60.9	30.3 ± 0.5	38	86.3	47.6 ± 0.6	62	109.5	56.8 ± 0.7	87
$N =$	50	70.1	40.2 ± 0.3	194	110.3	62.4 ± 0.4	328	148.8	85.0 ± 0.4	483
	250	70.7	40.2 ± 0.3	940	112.0	63.0 ± 0.3	1 609	151.7	85.4 ± 0.4	2 280
MC	10^2	16.2	25.0 ± 0.5	< 1	16.9	41.9 ± 0.6	< 1	17.2	51.8 ± 0.9	< 1
$N =$	10^4	40.8	37.9 ± 0.3	10	52.8	58.8 ± 0.4	18	63.8	75.9 ± 0.7	31
	10^6	51.2	39.4 ± 0.3	855	67.1	60.3 ± 0.4	1 415	81.1	82.9 ± 0.4	1 938
Utopian			49.1			79.2			109.2	

Ring-of-rings topology		$n = 6$			$n = 12$			$n = 18$		
		OV	Reward	Time	OV	Reward	Time	OV	Reward	Time
ε-**HALP**	$1/4$	28.4	40.4 ± 0.3	85	44.1	66.5 ± 0.3	382	59.8	93.0 ± 0.4	931
$\varepsilon =$	$1/8$	65.4	47.5 ± 0.3	495	107.9	76.1 ± 0.4	2 379	148.8	105.3 ± 0.4	5 877
	$1/16$	68.9	47.0 ± 0.3	4 417	113.1	77.3 ± 0.4	19 794	156.9	107.8 ± 0.4	53 655
MCMC	10	66.9	35.3 ± 0.6	60	94.6	54.4 ± 0.9	107	110.6	47.8 ± 1.3	157
$N =$	50	80.9	47.1 ± 0.3	309	131.9	76.6 ± 0.4	571	181.4	104.6 ± 0.4	859
	250	81.7	47.2 ± 0.3	1 522	134.1	77.3 ± 0.4	2 800	186.0	106.6 ± 0.4	4 291
MC	10^2	13.7	31.0 ± 0.5	< 1	15.4	46.1 ± 0.6	< 1	16.8	66.6 ± 0.9	1
$N =$	10^4	44.3	43.3 ± 0.3	12	59.0	68.9 ± 0.5	26	71.5	92.2 ± 0.7	49
	10^6	55.8	45.1 ± 0.3	1 026	75.1	74.3 ± 0.4	1 738	92.0	103.1 ± 0.4	2 539
Utopian			59.1			99.2			139.3	

Figure 17: Comparison of HALP solvers on two irrigation network topologies of varying sizes (n). The solvers are compared by the objective value of a relaxed HALP (OV), the expected discounted reward of a corresponding policy, and its computation time (in seconds). The ε-HALP, MC-HALP, and MCMC-HALP solvers are parameterized by the resolution of ε-grid (ε), the number of samples (N), and the number of MCMC chains (N). Note that the quality of policies improves with higher grid resolution ($1/\varepsilon$) and larger sample size (N). Upper bounds on their expected returns are shown in the last rows of the tables.

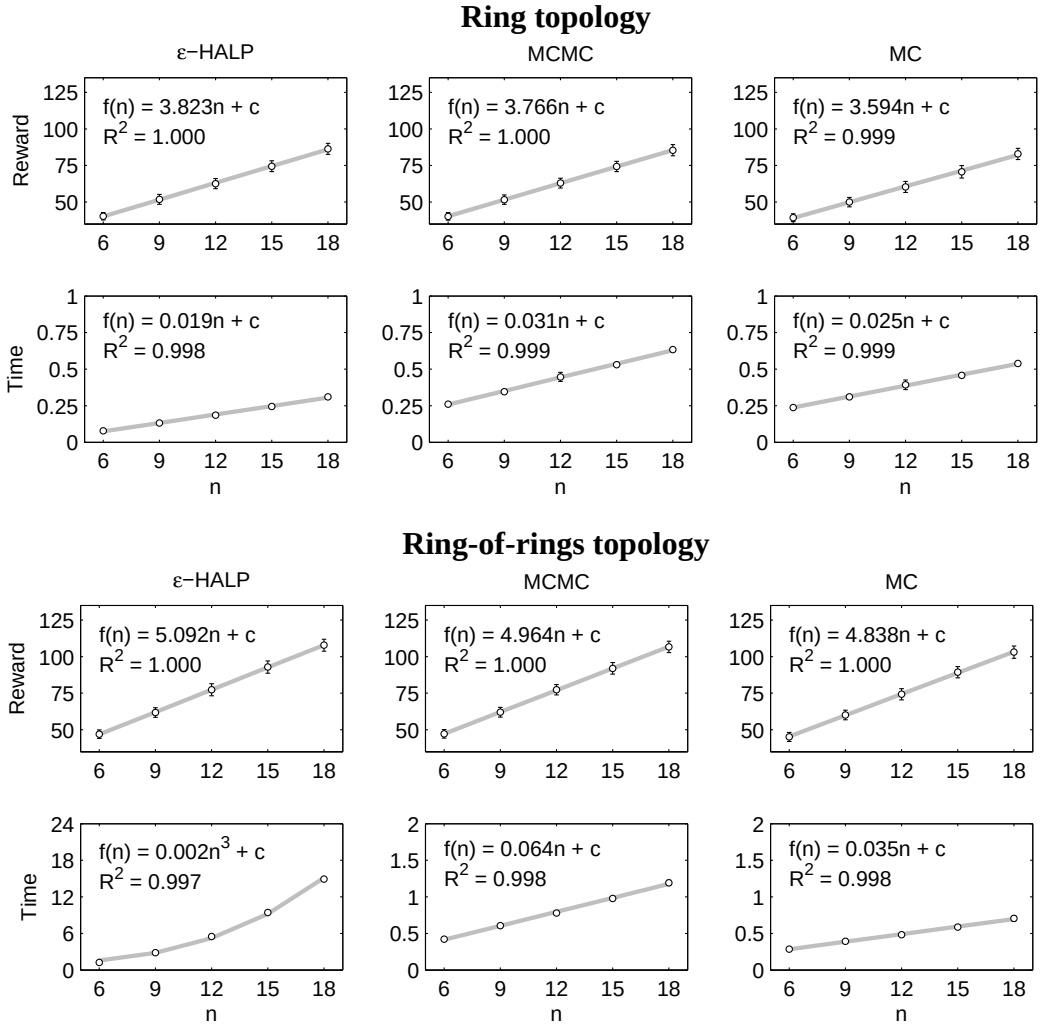


Figure 18: Scale-up potential of the ε-HALP, MCMC-HALP, and MC-HALP solvers on two irrigation network topologies of varying sizes (n). The graphs show the expected discounted reward of policies and their computation time (in hours) as functions of n. The solvers are parameterized by the resolution of ε-grid ($\varepsilon = 1/16$), the number of MCMC chains ($N = 250$), and the number of samples ($N = 10^6$). Note that all trends can be approximated by a polynomial $f(n)$ (gray line) with a high degree of confidence (the coefficient of determination R^2), where c denotes a constant independent of n.

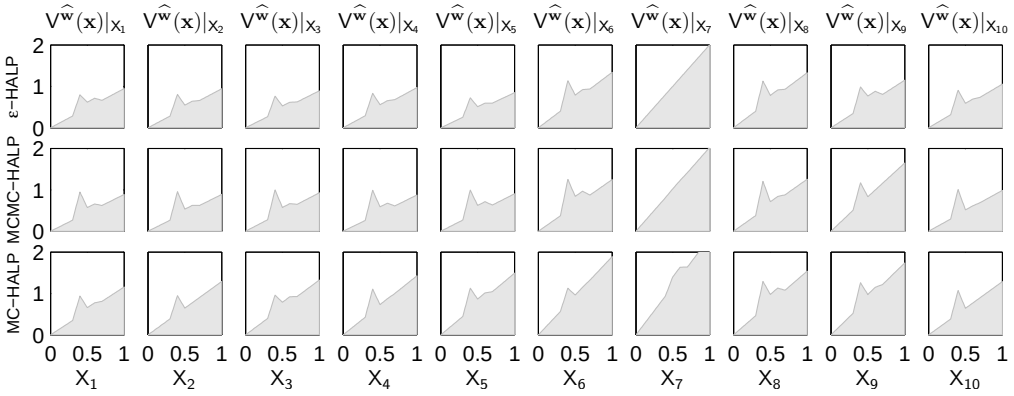


Figure 19: Univariate projections $V^{\widehat{\mathbf{w}}}(\mathbf{x})|_{X_j} = \sum_{i:X_j=X_i} \widehat{w}_i f_i(x_i)$ of approximate value functions $V^{\widehat{\mathbf{w}}}$ on the 6-ring irrigation network problem (Figure 15a). These functions are computed from 40 basis functions (Figure 16c) by the ε-HALP, MCMC-HALP, and MC-HALP solvers. The solvers are parameterized by the resolution of ε-grid ($\varepsilon = 1/16$), the number of MCMC chains ($N = 250$), and the number of samples ($N = 10^6$). Note that the univariate projections $V^{\widehat{\mathbf{w}}}(\mathbf{x})|_{X_j}$ are extremely similar. The proximity of their greedy policies can be explained based on this observation.

17, 18, and 19.

Based on the results, we draw the following conclusions. First, all HALP approximations scale up with the dimensionality of solved problems. As presented in Figure 18, the return of the policies grows linearly in n. Moreover, the time complexity of computing them is increasing polynomially in n. Therefore, if a problem and its approximate solution are structured, we take advantage of this structure to avoid an exponential blowup in the computation time. At the same time, the quality of the policies is not deteriorating with increasing problem size n.

Second, the MCMC-HALP solver ($N = 250$) achieves the highest objective values on all solved problems. Higher objective values are interpreted as closer approximations to the constraint space in HALP since the solvers operate on the relaxed formulations of HALP. Third, the quality of the MCMC-HALP policies ($N = 250$) surpasses the MC-HALP policies ($N = 10^6$) while both solvers consume approximately the same computation time. This result is due to the informative search for violated constraints in the MCMC-HALP solver. Fourth, the quality of the MCMC-HALP policies ($N = 250$) is close to the ε-HALP policies ($\varepsilon = 1/16$) although there is a significant difference

between their objective values. Further analysis shows that the shape of the value functions is very similar (Figure 19) and they differ the most in the weight of the constant basis function $f_0(\mathbf{x}) \equiv 1$. Note that increasing w_0 does not affect the quality of a greedy policy for $V^{\mathbf{w}}$. However, this trick allows the satisfaction of the constraint space in HALP (Section 3.2.1).

Finally, the computation time of the ε-HALP solver is seriously affected by the topologies of the irrigation networks, which can be explained as follows. For a small value of ε and large n, the time complexity of formulating cost networks for the ring and ring-of-rings topologies grows by the rates of $\lceil 1/\varepsilon + 1 \rceil^2$ and $\lceil 1/\varepsilon + 1 \rceil^3$, respectively. Since the ε-HALP method consumes a significant amount of time by constructing cost networks, its quadratic (in $\lceil 1/\varepsilon + 1 \rceil$) time complexity on the ring topology worsens to cubic (in $\lceil 1/\varepsilon + 1 \rceil$) on the ring-of-rings topology. On the other hand, a similar cross-topology comparison of the MCMC-HALP solver shows that its computation times differ only by a multiplicative factor of 2. This difference is mainly due to the increased complexity of sampling $p(z_i^* \mid \mathbf{z}_{-i})$, which results from more complex local dependencies in the ring-of-rings topology and not its treewidth.

Before we proceed, note that our relaxed HALP formulations (Figure 17) involve significantly less constraints than their complete sets (Section 3.3.3). For instance, the MC-HALP formulation ($N = 10^6$) on the 6-ring irrigation network problem is originally defined by 10^6 randomly sampled constraints. Based on our empirical results, these constraints can be satisfied greedily by a subset of 400 constraints on average [56]. Similarly, the MCMC-HALP oracle $\mathcal{O}_{\text{MCMC}}$ ($N = 250$) iterates through $250 \times 500 \times (10+10) = 2,500,000$ state-action pairs (Figure 13). However, corresponding LP formulations involve only 700 constraints on average.

3.4.3 The Curse of Treewidth

In the ring and ring-of-rings topologies, the treewidth of the constraint space (in continuous variables) is 2 and 3, respectively. Consequently, the oracle $\mathcal{O}_\varepsilon$ can perform variable elimination for a small ε, and the ε-HALP solver returns close-to-optimal policies. Unfortunately, small treewidth is atypical in real-world domains. For instance, the treewidth of a more complex 3×3 grid irrigation network (Figure 15c) is 6. To perform variable elimination for $\varepsilon = 1/16$, the separation oracle $\mathcal{O}_\varepsilon$ requires the space of $\lceil 1/\varepsilon + 1 \rceil^7 \approx 2^{28}$, which is at the memory limit of existing PCs. To analyze

ε-HALP

ε	OV	Reward	Time
1	30.4	48.3 ± 0.3	9
$1/2$	42.9	58.7 ± 0.3	342
$1/4$	49.1	61.9 ± 0.3	9 443

MCMC

N	OV	Reward	Time
10	45.3	43.6 ± 0.7	83
50	116.2	72.2 ± 0.4	458
250	118.5	73.2 ± 0.4	2 012

MC

N	OV	Reward	Time
10^2	12.8	56.6 ± 0.5	< 1
10^4	49.9	53.4 ± 0.6	19
10^6	71.7	70.3 ± 0.4	1 400

Figure 20: Comparison of HALP solvers on the 3×3 grid irrigation network problem (Figure 15). The solvers are compared by the objective value of a relaxed HALP (OV), the expected discounted reward of a corresponding policy, and its computation time (in seconds). The ε-HALP, MC-HALP, and MCMC-HALP solvers are parameterized by the resolution of ε-grid (ε), the number of samples (N), and the number of MCMC chains (N). Note that the quality of policies improves with higher grid resolution ($1/\varepsilon$) and larger sample size (N). An upper bound on the expected returns is 87.2.

the behavior of our separation oracles (Section 3.3) in this setting, we repeat our experiments from Section 3.4.2 on the 3×3 grid irrigation network.

Based on the results in Figure 20, we conclude that the time complexity of the ε-HALP solver grows by the rate of $\lceil 1/\varepsilon + 1 \rceil^7$. Therefore, approximate constraint space satisfaction (MC-HALP and MCMC-HALP) generates better results than a combinatorial optimization on an insufficiently discretized ε-grid (ε-HALP). This conclusion is parallel to those in large-scale structured optimization problems with continuous variables. We believe that a combination of exact and approximate steps delivers the best tradeoff between the quality and complexity of our solutions (Section 3.3.4).

3.4.4 Hybrid Approximate Policy Iteration

Approximate policy iteration (API) [37, 81]:

$$\begin{aligned} \text{minimize} \quad & \delta \\ \text{subject to:} \quad & \|V^{\mathbf{w}} - \mathcal{T}^* V^{\mathbf{w}}\|_\infty \leq \delta \end{aligned} \tag{3.35}$$

is an alternative to the ALP (Section 2.3.2.3) for solving structured Markov decision processes. Instead of minimizing the $\mathcal{L}_1$-norm error $\|V^* - V^{\mathbf{w}}\|_{1,\psi}$ (Proposition 3), API indirectly optimizes the max-norm distance $\|V^* - V^{\mathbf{w}}\|_\infty$ through the minimization of the Bellman error $\|V^{\mathbf{w}} - \mathcal{T}^* V^{\mathbf{w}}\|_\infty$.

Inputs:

a hybrid factored MDP $\mathcal{M} = (\mathbf{X}, \mathbf{A}, P, R)$
basis functions $f_0(\mathbf{x}), f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots$
a separation oracle $\mathcal{O}$ (Equation 3.36)
the absolute error ε of the bisection method

Algorithm:

initialize a relaxed HAPI formulation with an empty set of constraints
$\overline{\delta} = 2(1-\gamma)^{-1} R_{\max}$
$\underline{\delta} = 0$
$\delta = (\overline{\delta} + \underline{\delta})/2$
$t = 0$
while $(\overline{\delta} - \underline{\delta} > \varepsilon)$
    initialize basis function weights $\mathbf{w}$
    iteratively add the constraints generated by the oracle $\mathcal{O}$ until:
        $\|V^{\mathbf{w}} - \mathcal{T}^* V^{\mathbf{w}}\|_\infty \leq \delta$
    if the relaxed HAPI formulation is infeasible
        remove all constraints added in the last step
        $\underline{\delta} = \delta$
    else
        $\overline{\delta} = \delta$
    $\delta = (\overline{\delta} + \underline{\delta})/2$
    $t = t + 1$

Outputs:

basis function weights $\mathbf{w}$

Figure 21: Pseudo-code implementation of the HAPI solver. The bisection method is used to find the smallest δ that guarantees the feasibility of the HAPI formulation (3.35).

Ring topology		$n = 6$ Reward	Time
ε-**HALP**	$1/4$	34.6 ± 0.2	2
$\varepsilon =$	$1/8$	39.6 ± 0.3	4
	$1/16$	40.3 ± 0.3	29
ε-**HAPI**	$1/4$	32.9 ± 0.2	16
$\varepsilon =$	$1/8$	37.5 ± 0.3	21
	$1/16$	38.3 ± 0.3	246

Ring-of-rings topology		$n = 6$ Reward	Time
ε-**HALP**	$1/4$	40.4 ± 0.3	9
$\varepsilon =$	$1/8$	47.5 ± 0.3	48
	$1/16$	47.0 ± 0.3	757
ε-**HAPI**	$1/4$	39.1 ± 0.3	58
$\varepsilon =$	$1/8$	45.9 ± 0.3	325
	$1/16$	45.9 ± 0.3	3 596

Figure 22: Comparison of the ε-HALP and ε-HAPI solvers on the 6-ring and 6-ring-of-rings irrigation network problems (Figure 15a and 15b). The solvers are compared by the expected discounted reward of policies and their computation time (in seconds). The ε-HALP and ε-HAPI solvers are parameterized by the resolution of ε-grid (ε). Note that the quality of policies improves with higher grid resolution ($1/\varepsilon$). The computation time of the ε-HALP policies is significantly shorter than in Figure 17. The speedup is due to precomputing cost networks in the ε-HALP and ε-HAPI solvers.

In discrete-state spaces, Patrascu *et al.* [81] and Guestrin [34] showed that API returns better policies than ALP for the same set of basis functions. In this section, we perform a similar analysis for hybrid state and action spaces. Since the constraint space in API (3.35) exhibits the same structure as the constraint space in HALP (3.9):

$$\|V^{\mathbf{w}} - \mathcal{T}^* V^{\mathbf{w}}\|_\infty \leq \delta \quad \begin{cases} \mathcal{T}^* V^{\mathbf{w}}(\mathbf{x}) - V^{\mathbf{w}}(\mathbf{x}) + \delta \geq 0 & \text{if } V^{\mathbf{w}}(\mathbf{x}) - \mathcal{T}^* V^{\mathbf{w}}(\mathbf{x}) \geq 0 \\ V^{\mathbf{w}}(\mathbf{x}) - \mathcal{T}^* V^{\mathbf{w}}(\mathbf{x}) + \delta \geq 0 & \text{otherwise} \end{cases}, \quad (3.36)$$

Sections 3.2.2 and 3.3 provide a recipe for efficient solutions to hybrid API (HAPI) formulations (Figure 21). Our results on the ring and ring-of-rings irrigation network problems (Figures 15a and 15b) are reported in Figure 22.

Based on these results, we draw the following conclusions. First, the computation time of the ε-HAPI policies is significantly longer than the computation time of the ε-HALP policies. Since a single step of ε-HAPI (Figure 21) is roughly as hard as formulating a corresponding relaxed HALP, this result comes at not surprise. Second, in contrast to the previous work in discrete-state domains [37, 81], the quality of the ε-HALP policies always surpasses the ε-HAPI policies. To explain this phenomenon, note that the irrigation network problems can be viewed as average-cost optimization

problems. Therefore, the optimization of the average error $\|V^* - V^{\mathbf{w}}\|_{1,\psi}$ for a given class of basis functions may yield better policies than minimizing the Bellman error $\|V^{\mathbf{w}} - \mathcal{T}^* V^{\mathbf{w}}\|_\infty$.

3.5 CONCLUSIONS

Development of scalable algorithms for solving real-world decision problems is a challenging task. In this work, we presented a new framework that allows for a compact representation (Section 3.1) and efficient solutions (Section 3.2) to hybrid factored MDPs. In addition, we analyzed the quality of the approximation (Section 3.2.1), and introduced the concepts of conjugate basis functions and relaxed HALP formulations (Sections 3.2.2 and 3.3). These concepts are critical for the computational tractability of our value function approximations. In the remainder of this thesis, we extend the presented framework in two new directions.

First, note that the concept of closed-form solutions to the expectations terms in HALP is not limited to the choices in Section 3.2.2. For instance, if both $P(x)$ and $f(x)$ are normal distributions, $\mathrm{E}_{P(x)}[f(x)]$ has a closed-form solution [59]. Therefore, we can easily reason with normal transition functions instead of approximating them by a mixture of beta distributions. Similar conclusions are true for piecewise constant, piecewise linear, and gamma transition and basis functions. This topic is discussed in detail in Chapter 4. Note that our efficient solutions apply to any approach to solving hybrid factored MDPs that approximates the optimal value function V^* by a linear combination of basis functions (Equation 2.20).

Second, automatic learning of basis functions is critical for the practical application of HALP formulations to new real-world domains. Patrascu *et al.* [81] studied this problem in discrete-state spaces and proposed a greedy technique for learning basis functions. Kveton and Hauskrecht [58] generalized these ideas and showed how to learn parametric basis functions in hybrid spaces. This topic is discussed in detail in Chapter 5.

Finally, we proposed several bounds (Section 3.2.1 and 3.3.2.1) that may explain the quality of the complete and relaxed HALP formulations. In our future work, we plan to empirically evaluate their tightness on low-dimensional hybrid decision problems [17, 74] with known optimal value functions.

4.0 SOLVING HYBRID FACTORED MDPS WITH EXPONENTIAL-FAMILY TRANSITION MODELS

The hybrid ALP approach proposed in Section 3.2 imposes a restriction on solved problems. Every continuous variable is bounded on the $[0, 1]$ interval and its transition function is given by a mixture of beta distributions (Section 3.1). Different transition models, such as normal distributions, cannot be used directly and have to be approximated. Note that this restriction only allows for closed-form solutions to the expectation terms in HALP (Section 3.2.2). From the theoretical point of view, the HALP formulation permits arbitrary basis functions and transition models.

In this most general setting, the univariate expectation terms $\mathrm{E}_{P(x)}[f(x)]$ (Section 3.2.2) can be approximated by the empirical mean $\widehat{\mu} = \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)})$, where each $x^{(i)}$ is sampled with respect to the distribution $P(x)$. From the central limit theorem, the error of the estimate $\widehat{\mu} - \mathrm{E}_{P(x)}[f(x)]$ follows the normal distribution $\mathcal{N}(0, \sigma^2/N)$, where $\sigma^2 = \mathrm{var}_{P(x)}[f(x)]$. As a result, to guarantee that the error is smaller than ε with high probability, we require on the order of $O(\sigma^2/\varepsilon^2)$ samples $x^{(i)}$. Since imprecisely evaluated expectation terms may lead to the infeasibility of relaxed HALP formulations, the targeted ε should be always low. Unfortunately, the number of samples N grows quadratically in $1/\varepsilon$. Therefore, at least from the theoretical point of view, Monte Carlo sampling is unsuitable for precise and efficient computation of our expectation terms. Although the real-world performance of Monte Carlo sampling is typically significantly better than suggested by our bound [1], we consider a different approach to evaluating the expectation terms in HALP. We identify rich classes of conjugate basis functions that yield closed-form solutions to transition models based on the exponential family of distributions and their mixtures.

This chapter is organized as follows. First, we review hybrid factored MDPs (Section 3.1) and extend them by exponential-family transition models (Section 4.1). Second, we generalize HALP (Section 3.2) to solve the new class of problems efficiently (Section 4.2). Third, we introduce rich

classes of conjugate basis functions that lead to closed-form solutions to the expectation terms in HALP (Section 4.4). Finally, we evaluate the HALP framework on an autonomous rover planning problem.

4.1 FACTORED TRANSITION AND REWARD MODELS

Discrete-state factored MDPs [14] allow a compact representation of stochastic decision problems by exploiting their structure. In this section, we introduce a new formalism for representing hybrid factored MDPs with exponential-family transition functions. The formalism is based on the HMDP framework [35] and generalizes its transition model for continuous variables (Section 3.1).

A *hybrid factored MDP with an exponential-family transition model (HMDP)* is a tuple $\mathcal{M} = (\mathbf{X}, \mathbf{A}, P, R)$, where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a state space described by a set of state variables, $\mathbf{A} = \{A_1, \ldots, A_m\}$ is an action space represented by action variables, $P(\mathbf{X}' \mid \mathbf{X}, \mathbf{A})$ is an exponential-family transition model of state dynamics conditioned on the preceding state and action, and R is a reward model assigning immediate payoffs to state-action configurations.[1]

State variables: State variables are either discrete or continuous. The state of the system is completely observed and defined by a vector of value assignments $\mathbf{x} = (\mathbf{x}_D, \mathbf{x}_C)$ which partitions along its discrete and continuous components $\mathbf{x}_D$ and $\mathbf{x}_C$.

Action variables: The action space is fully distributed and represented by action variables $\mathbf{A}$. The composite action is defined by a vector of individual action choices $\mathbf{a} = (\mathbf{a}_D, \mathbf{a}_C)$ which partitions along its discrete and continuous components $\mathbf{a}_D$ and $\mathbf{a}_C$. Without loss of generality, we make an assumption that every state variable A_i is either of finite cardinality or bounded.

Transition model: The transition model is characterized by the conditional probability distribution $P(\mathbf{X}' \mid \mathbf{X}, \mathbf{A})$, where $\mathbf{X}$ and $\mathbf{X}'$ denote the state variables at two successive time steps t and $t+1$. We assume that this distribution factors along $\mathbf{X}'$ as $P(\mathbf{X}' \mid \mathbf{X}, \mathbf{A}) = \prod_{i=1}^{n} P(X_i' \mid \mathsf{Par}(X_i'))$ and

[1] *General state and action space MDP* is an alternative term for a hybrid MDP. The term *hybrid* does not refer to the dynamics of the model, which is discrete-time.

can be represented compactly by a DBN [25]. Typically, the parent set $\mathsf{Par}(X_i') \subseteq \mathbf{X} \cup \mathbf{A}$ is a small subset of state and action variables which allows for a local parameterization of the model.

Parameterization of transition model: One-step dynamics of every state variable is represented by the conditional probability $P(X_i' \mid \mathsf{Par}(X_i'))$. The conditionals are chosen from the exponential family of distributions:

$$P(X_i' = x \mid \mathsf{Par}(X_i')) = h(x) \exp[\eta^{\mathsf{T}} t(x)]/Z(\eta) \tag{4.1}$$

based on $\mathrm{Dom}(X_i')$, where η denotes the natural parameters of the distribution, $t(x)$ is a vector of its sufficient statistics, and $Z(\eta)$ is a normalizing function independent of X_i'. The transition model choices that lead to closed-form[2] solutions to the expectation terms in HALP are shown in Figure 23. Our work generalizes to the mixtures of these transition functions (Corollary 2), which provide a very rich class of transition models.

Reward model: The reward model is factored similarly to the transition model. In particular, the reward function $R(\mathbf{x}, \mathbf{a}) = \sum_j R_j(\mathbf{x}_j, \mathbf{a}_j)$ is an additive function of local reward functions defined on the subsets $\mathbf{X}_j$ and $\mathbf{A}_j$. In graphical models, these local functions can be described compactly by reward nodes R_j, which are conditioned on their parent sets $\mathsf{Par}(R_j) = \mathbf{X}_j \cup \mathbf{A}_j$. To allow this representation, we formally extend our DBN into an influence diagram [47]. Note that the form of the reward functions $R_j(\mathbf{x}_j, \mathbf{a}_j)$ is unrestricted.

Optimal value function and policy: The *optimal policy* π^* can be defined greedily with respect to the *optimal value function* V^*, which is a fixed point of the Bellman equation:

$$\begin{aligned} V^*(\mathbf{x}) &= \sup_{\mathbf{a}} \left[R(\mathbf{x}, \mathbf{a}) + \gamma \mathrm{E}_{P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a})} [V^*(\mathbf{x}')] \right] \\ &= \sup_{\mathbf{a}} \left[R(\mathbf{x}, \mathbf{a}) + \gamma \sum_{\mathbf{x}'_D} \int_{\mathbf{x}'_C} P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a}) V^*(\mathbf{x}') \, \mathrm{d}\mathbf{x}'_C \right]. \end{aligned} \tag{4.2}$$

[2]The term *closed-form* refers to a generally accepted set of closed-form operations and functions extended by the gamma functions.

$\mathrm{Dom}(X_i')$	Transition function
$\{0, \ldots, k\}$	**Multinomial distribution** $P(X_i' = j) = \theta_j$ where $\sum_j \theta_j = 1$ and $\theta_j = \phi_{ij}^{\theta}(\mathsf{Par}(X_i'))$
$[0, 1]$	**Beta distribution** $P_{\text{beta}}(X_i' = x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$ where $\alpha = \phi_i^{\alpha}(\mathsf{Par}(X_i'))$ and $\beta = \phi_i^{\beta}(\mathsf{Par}(X_i'))$
$(-\infty, \infty)$	**Normal distribution** $\mathcal{N}(X_i' = x) = \frac{1}{\sigma\sqrt{2\pi}} \exp[-\frac{1}{2\sigma^2}(x-\mu)^2]$ where $\mu = \phi_i^{\mu}(\mathsf{Par}(X_i'))$ and $\sigma = \phi_i^{\sigma}(\mathsf{Par}(X_i'))$
$[0, \infty)$	**Gamma distribution** $P_{\text{gamma}}(X_i' = x) = \frac{1}{\Gamma(\alpha)\beta^{\alpha}} x^{\alpha-1} \exp[-\frac{1}{\beta}x]$ where $\alpha = \phi_i^{\alpha}(\mathsf{Par}(X_i'))$ and $\beta = \phi_i^{\beta}(\mathsf{Par}(X_i'))$

Figure 23: Selecting transition functions based on $\mathrm{Dom}(X_i')$. The functions are parameterized by arbitrary functions $\phi_i(\cdot)$ of their parent sets $\mathsf{Par}(X_i')$.

Accordingly, the *hybrid Bellman operator* $\mathcal{T}^*$ is given by:

$$\mathcal{T}^* V(\mathbf{x}) = \sup_{\mathbf{a}} \left[R(\mathbf{x}, \mathbf{a}) + \gamma \mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})}[V(\mathbf{x}')] \right]. \tag{4.3}$$

In the rest of the chapter, we denote expectation terms over discrete and continuous variables in a unified form:

$$\mathrm{E}_{P(\mathbf{x})}[f(\mathbf{x})] = \sum_{\mathbf{x}_D} \int_{\mathbf{x}_C} P(\mathbf{x}) f(\mathbf{x}) \, \mathrm{d}\mathbf{x}_C. \tag{4.4}$$

4.2 GENERALIZED HYBRID ALP

Since a factored representation of MDPs (Section 4.1) may not guarantee a structure in the optimal value function or policy [54], we resort to the linear value function approximation (Equation 2.20). This approximation restricts the value function $V^{\mathbf{w}}$ to the linear combination of $|\mathbf{w}|$ basis functions $f_i(\mathbf{x})$, where $\mathbf{w}$ is a vector of tunable weights. Each basis function can be defined over the complete state space $\mathbf{X}$, but often is restricted to a subset of state variables $\mathbf{X}_i$ [7, 54].

Similarly to the discrete-state ALP (Section 2.3.2.3), HALP (Section 3.2) optimizes the linear value function approximation. Therefore, it transforms an initially intractable problem of estimating V^* in the hybrid state space $\mathbf{X}$ into a lower dimensional space $\mathbf{w}$. As shown in this section, this approach generalizes to HMDPs with exponential-family transition models (Section 4.1).

The *generalized hybrid ALP (HALP)* formulation is given by a linear program[3]:

$$\begin{aligned} &\text{minimize}_{\mathbf{w}} \sum_i w_i \alpha_i \\ &\text{subject to: } \sum_i w_i F_i(\mathbf{x}, \mathbf{a}) - R(\mathbf{x}, \mathbf{a}) \geq 0 \quad \forall\, \mathbf{x}, \mathbf{a}; \end{aligned} \tag{4.5}$$

where $\mathbf{w}$ represents the variables in the LP, α_i denotes *basis function relevance weight*:

$$\begin{aligned} \alpha_i &= \mathrm{E}_{\psi(\mathbf{x})}[f_i(\mathbf{x})] \\ &= \sum_{\mathbf{x}_D} \int_{\mathbf{x}_C} \psi(\mathbf{x}) f_i(\mathbf{x}) \, \mathrm{d}\mathbf{x}_C, \end{aligned} \tag{4.6}$$

$\psi(\mathbf{x}) \geq 0$ is a *state relevance density function* that weights the quality of the approximation, and $F_i(\mathbf{x}, \mathbf{a}) = f_i(\mathbf{x}) - \gamma g_i(\mathbf{x}, \mathbf{a})$ is the difference between the basis function $f_i(\mathbf{x})$ and its discounted *backprojection*:

$$\begin{aligned} g_i(\mathbf{x}, \mathbf{a}) &= \mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})}[f_i(\mathbf{x}')] \\ &= \sum_{\mathbf{x}'_D} \int_{\mathbf{x}'_C} P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a}) f_i(\mathbf{x}') \, \mathrm{d}\mathbf{x}'_C. \end{aligned} \tag{4.7}$$

Vectors $\mathbf{x}_D$ ($\mathbf{x}'_D$) and $\mathbf{x}_C$ ($\mathbf{x}'_C$) are the discrete and continuous components of value assignments $\mathbf{x}$ ($\mathbf{x}'$) to all state variables $\mathbf{X}$ ($\mathbf{X}'$). The formulation is feasible if the set of basis functions contains a constant function $f_0(\mathbf{x}) \equiv 1$. We assume that such a basis function is always present.

Similarly to our previous discussion in Section 3.2, we address several concerns related to the generalized HALP formulation. First, we analyze the quality of this approximation (Section 4.3) by extending our results from Section 3.2.1. Second, we introduce rich classes of basis functions that lead to closed-form solutions to the expectation terms in the objective function and constraints (Equations 4.6 and 4.7). These terms involve sums and integrals over the complete state space $\mathbf{X}$

[3]More precisely, the HALP formulation (4.5) is a *linear semi-infinite optimization* problem with an infinite number of constraints. Note that the number of basis functions remains finite. For brevity, we refer to this optimization problem as linear programming.

(Section 4.4), and therefore are extremely hard to evaluate. Finally, we discuss approximations to the constraint space in HALP (Section 4.5). Note that complete satisfaction of this constraint space (4.5) may not be possible since every state-action pair $(\mathbf{x}, \mathbf{a})$ induces a constraint.

4.3 ERROR BOUNDS

Recall that HALP minimizes the $\mathcal{L}_1$-norm distance between the optimal value function V^* and its linear approximation $V^{\mathbf{w}}$ (Propositions 2 and 3). Furthermore, the optimization of this error metric is closely related to minimizing the max-norm error $\|V^* - V^{\mathbf{w}}\|_{\infty,1/L}$ (Theorems 2 and 3). Finally, the choice of the state relevance density function $\psi(\mathbf{x})$ significantly impacts the quality of a greedy policy for the value function $V^{\mathbf{w}}$ (Theorem 4).

Interestingly, none of these theoretical results (Section 3.2.1) assumes that the state space $\mathbf{X}$ is bounded. Therefore, our earlier conclusions extend to the generalized HALP formulation (4.5).

4.4 EXPECTATION TERMS

Since our basis functions are often restricted to small subsets of state variables, expectation terms (Equations 4.6 and 4.7) in the generalized HALP formulation (4.5) should be computable without enumerating the complete state space $\mathbf{X}$. Before we justify this claim, similarly to Section 3.2.2, we assume that the state relevance density function $\psi(\mathbf{x})$ factors according to Equation 3.10. Note that this assumption allows us to view the evaluation of the expectation terms in HALP, $\mathrm{E}_{\psi(\mathbf{x})}[f_i(\mathbf{x})]$ and $\mathrm{E}_{P(\mathbf{x}'|\mathbf{x},\mathbf{a})}[f_i(\mathbf{x}')]$, as the same computational problem $\mathrm{E}_{P(\mathbf{x})}[f_i(\mathbf{x})]$, where $P(\mathbf{x})$ is a distribution in a factored form.

Before computing the expectation term $\mathrm{E}_{P(\mathbf{x})}[f_i(\mathbf{x})]$ over the complete state space $\mathbf{X}$, we recall that the basis function $f_i(\mathbf{x})$ is defined on a subset of state variables $\mathbf{X}_i$. Therefore, we immediately conclude that $\mathrm{E}_{P(\mathbf{x})}[f_i(\mathbf{x})] = \mathrm{E}_{P(\mathbf{x}_i)}[f_i(\mathbf{x}_i)]$, where $P(\mathbf{x}_i)$ denotes a factored distribution on a lower dimensional space $\mathbf{X}_i$. If no further assumptions are made, the local expectation term $\mathrm{E}_{P(\mathbf{x}_i)}[f_i(\mathbf{x}_i)]$ may be still hard to compute. Although it can be estimated by a variety of numerical methods, for

instance Monte Carlo [1], these techniques are typically imprecise if the sample size is small, and quite computationally expensive if a high precision is required. Consequently, we try to avoid such an approximation step. Instead, we introduce an appropriate form of basis functions that guarantees closed-form solutions to the expectation term $\mathrm{E}_{P(\mathbf{x}_i)}[f_i(\mathbf{x}_i)]$.

First, let us assume that every basis function $f_i(\mathbf{x}_i)$ factors along the state variables $\mathbf{X}$ as:

$$f_i(\mathbf{x}_i) = \prod_{X_j \in \mathbf{X}_i} f_{ij}(x_j). \tag{4.8}$$

Note that the basis functions remain multivariate despite this independence assumption. We make this presumption for computational purposes and it can be easily relaxed (Section 3.2.2.2).

Based on Equation 4.8, the expectation term $\mathrm{E}_{P(\mathbf{x}_i)}[f_i(\mathbf{x}_i)]$ decomposes as a product:

$$\mathrm{E}_{P(\mathbf{x}_i)}[f_i(\mathbf{x}_i)] = \prod_{X_j \in \mathbf{X}_i} \mathrm{E}_{P(x_j)}[f_{ij}(x_j)] \tag{4.9}$$

of expectations over individual variables X_j. Subsequently, an efficient solution to the expectation term $\mathrm{E}_{P(\mathbf{x}_i)}[f_i(\mathbf{x}_i)]$ is guaranteed by efficient solutions to its univariate components $\mathrm{E}_{P(x_j)}[f_{ij}(x_j)]$.

To obtain closed-form solutions to the expectation terms $\mathrm{E}_{P(x_j)}[f_{ij}(x_j)]$, we consider univariate basis function factors:

$$f(x) = h(x) \exp[\eta^{\mathsf{T}} t(x)]/Z(\eta), \tag{4.10}$$

where η denotes their natural parameters, $t(x)$ is a vector of their sufficient statistics, and $Z(\eta)$ is a normalizing function independent of x. The following proposition provides a recipe for choosing univariate *conjugate* factors $f_{ij}(x_j)$ that complement the univariate distributions $P(x_j)$.

Proposition 7 (Exponential-family basis functions) *Let:*

$$\begin{aligned} P(x) &= h(x) \exp[\eta_P^{\mathsf{T}} t(x)]/Z(\eta_P) \\ f(x) &= h(x) \exp[\eta_f^{\mathsf{T}} t(x)]/Z(\eta_f) \end{aligned}$$

be exponential-family densities over X in the same canonical form, where η_P and η_f denote their natural parameters, $t(x)$ is a vector of their sufficient statistics, and $Z(\cdot)$ is a normalizing function independent of X. If $h(x) \equiv 1$, $\mathrm{E}_{P(x)}[f(x)]$ has a closed-form solution:

$$\mathrm{E}_{P(x)}[f(x)] = \frac{Z(\eta_P + \eta_f)}{Z(\eta_P) Z(\eta_f)}.$$

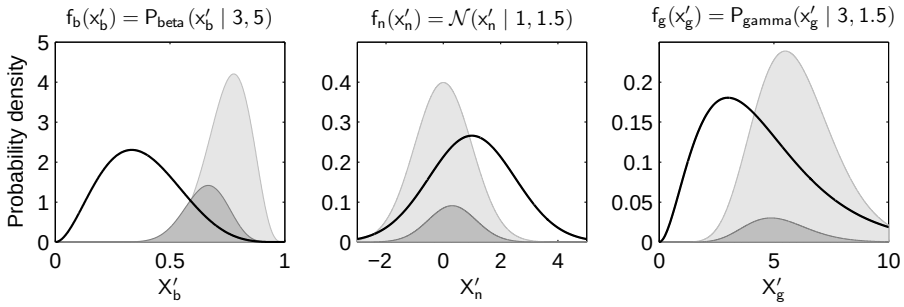


Figure 24: Expectations of three basis functions $f(x')$, denoted by thick black lines, with respect to three transition densities from Example 7, shown in a light gray color. Darker gray lines are given by the product $P(x')f(x')$. The area below corresponds to the expectation terms $\mathrm{E}_{P(x')}[f(x')]$.

Since integration is a distributive operation, our claim straightforwardly generalizes to the mixtures of the densities $P(x)$ and $f(x)$.

Corollary 2 (Mixture of exponential-family basis functions) *Let:*

$$P(x) = \sum_i \pi_i h(x) \exp[\eta_{P_i}^{\mathsf{T}} t(x)] / Z(\eta_{P_i})$$
$$f(x) = \sum_j \rho_j h(x) \exp[\eta_{f_j}^{\mathsf{T}} t(x)] / Z(\eta_{f_j})$$

be mixtures of exponential-family densities over X in the same canonical form, where π_i and ρ_j are the weights assigned to the components of the mixtures, η_{P_i} and η_{f_j} are the natural parameters of the densities, $t(x)$ is a vector of sufficient statistics, and $Z(\cdot)$ is a normalizing function independent of X. If $h(x) \equiv 1$, $\mathrm{E}_{P(x)}[f(x)]$ has a closed-form solution:

$$\mathrm{E}_{P(x)}[f(x)] = \sum_i \sum_j \pi_i \rho_j \frac{Z(\eta_{P_i} + \eta_{f_j})}{Z(\eta_{P_i}) Z(\eta_{f_j})}.$$

Proposition 7 and Corollary 2 have important implications for selecting appropriate basis functions. For instance, we may conclude that normal and beta transition models are complemented by normal and beta basis functions. These in turn guarantee closed-form solutions to Equation 4.7. The same conclusion can be made for gamma transition and basis functions.

Corollary 3 (Normal basis functions) *Let:*

$$\begin{aligned} P(x) &= \mathcal{N}(x \mid \mu_P, \sigma_P) \\ f(x) &= \mathcal{N}(x \mid \mu_f, \sigma_f) \end{aligned}$$

be normal distributions over X. Then $\mathrm{E}_{P(x)}[f(x)]$ has a closed-form solution:

$$\mathrm{E}_{P(x)}[f(x)] = \exp\left[\frac{(\mu_P\sigma_f^2 + \mu_f\sigma_P^2)^2}{2\sigma_P^2\sigma_f^2(\sigma_P^2+\sigma_f^2)} - \frac{\mu_P^2}{2\sigma_P^2} - \frac{\mu_f^2}{2\sigma_f^2} - \frac{\ln 2(\sigma_P^2+\sigma_f^2) - \ln \pi}{2}\right].$$

Corollary 4 (Gamma basis functions) *Let:*

$$\begin{aligned} P(x) &= P_{\text{gamma}}(x \mid \alpha_P, \beta_P) \\ f(x) &= P_{\text{gamma}}(x \mid \alpha_f, \beta_f) \end{aligned}$$

be gamma distributions over X. Then $\mathrm{E}_{P(x)}[f(x)]$ has a closed-form solution:

$$\mathrm{E}_{P(x)}[f(x)] = \frac{1}{\Gamma(\alpha_P)\beta_P^{\alpha_P}} \frac{1}{\Gamma(\alpha_f)\beta_f^{\alpha_f}} \Gamma(\alpha_P + \alpha_f - 1)\left(\frac{\beta_P\beta_f}{\beta_P+\beta_f}\right)^{\alpha_P+\alpha_f-1}.$$

Example 7 *To demonstrate closed-form solutions to the expectation terms in the generalized HALP, we consider a transition model:*

$$P(\mathbf{x}') = P_{\text{beta}}(x_b' \mid 15, 5)\mathcal{N}(x_n' \mid 0, 1)P_{\text{gamma}}(x_g' \mid 12, 0.5)$$

with three state variables $\mathbf{X} = \{X_b, X_n, X_g\}$, where:

$$\begin{aligned} \mathrm{Dom}(X_b) &= [0, 1] \\ \mathrm{Dom}(X_n) &= (-\infty, \infty) \\ \mathrm{Dom}(X_g) &= [0, \infty). \end{aligned}$$

Following Proposition 7, we conclude that basis functions of the form:

$$\begin{aligned} f(\mathbf{x}') &= f_b(x_b')f_n(x_n')f_g(x_g') \\ &= P_{\text{beta}}(x_b' \mid \alpha_b, \beta_b)\mathcal{N}(x_n' \mid \mu_n, \sigma_n)P_{\text{gamma}}(x_g' \mid \alpha_g, \beta_g) \end{aligned}$$

allow for closed-form solutions to the term $\mathrm{E}_{P(\mathbf{x}')}[f(\mathbf{x}')]$. A graphical interpretation of the computation is given in Figure 24. Brief inspection verifies that the univariate products $P(x')f(x')$ have the same canonical forms as the distributions $P(x')$ and $f(x')$.

In line with the previous discussion, Hauskrecht and Kveton [43] have recently identified polynomial basis functions as a conjugate choice for the mixture of beta transition model (Proposition 4). Since any polynomial can be written as a linear combination of the products of beta densities, this result follows from Corollary 2. Similarly to our conjugate choices, piecewise constant functions constitute another suitable class of basis functions. Their expectations can be easily computed as a weighted sum of the cumulative density functions corresponding to the transition model.

4.5 CONSTRAINT SPACE APPROXIMATIONS

An optimal solution $\widetilde{\mathbf{w}}$ to the generalized HALP formulation (4.5) is given by a finite set of *active constraints* at a vertex of the feasible region. Unfortunately, identification of the active constraints is a hard computational problem. In particular, it requires searching through an exponential number of constraints, if the state and action variables are discrete, and an infinite number of constraints, if any of the variables are continuous. Therefore, it is generally infeasible to find the optimal solution $\widetilde{\mathbf{w}}$ to the generalized HALP formulation. Thus, we resort to approximations to the constraint space in HALP whose optimal solution $\widehat{\mathbf{w}}$ is close to $\widetilde{\mathbf{w}}$. This notion of an approximation is formalized as follows.

Definition 5 *The generalized HALP formulation is* relaxed*:*

$$\begin{aligned} &\text{minimize}_{\mathbf{w}} \sum_i w_i \alpha_i \\ &\text{subject to: } \sum_i w_i F_i(\mathbf{x}, \mathbf{a}) - R(\mathbf{x}, \mathbf{a}) \geq 0 \quad (\mathbf{x}, \mathbf{a}) \in \mathcal{C}; \end{aligned} \tag{4.11}$$

if only a subset $\mathcal{C}$ of its constraints is satisfied.

HALP formulations (4.5) can be solved approximately by solving their relaxed formulations (4.11). Several methods for building and solving these approximate LPs have been proposed (Section 3.3): Monte Carlo sampling of constraints [43], their ε-grid discretization [35], and an adaptive MCMC search for a violated constraint [57]. In the rest of this section, we discuss the application of these methods in the context of the generalized HALP formulation.

Monte Carlo methods approximate the constraint space in HALP by its sample. Unfortunately, their efficiency typically depends on an appropriate choice of sampling distributions. The ones that produce good approximations and polynomial sample size bounds are closely related to the optimal solutions and rarely known a priori [24]. On the other hand, constraint sampling is easily applied in continuous domains and its space complexity is proportional to the number of variables. Another way of approximating the constraint space in HALP is by discretizing its continuous variables $\mathbf{X}_C$ and $\mathbf{A}_C$ on an ε-grid. Since the discretized constraint space preserves its original factored structure, it can be compactly satisfied by the methods for discrete-state ALP (Section 2.3.2.3). Note that this relaxation guarantees that the relaxed solution $\widehat{\mathbf{w}}$ converges to the optimal solution $\widetilde{\mathbf{w}}$ when $\varepsilon \longrightarrow 0$. However, it is impractical for small ε (Section 3.3.4). Finally, building of relaxed formulations can be approached as a search for violated constraints [57]. This search can be done efficiently due to the structure in the constraint space.

4.6 EXPERIMENTS

The goal of our experiments is to illustrate the quality of generalized HALP approximations rather than the scale-up potential of the framework. Therefore, we apply generalized HALP to a realistic but low-dimensional autonomous rover problem [17]. For scale-up studies in hybrid domains, refer to Section 3.4. These conclusions fully extend to the generalized HALP.

4.6.1 Experimental Setup

Space exploration and problems arising in this domain have been a very important source of applied AI research in recent years. The design of a planning module for an autonomous Mars rover is one of the challenging problems. Along these lines, Bresina *et al.* [17] outlined requirements for such a planning system. These include the ability to plan under continuous time, with concurrent actions, using limited resources, and all these in the presence of uncertainty. In the same paper, Bresina *et al.* [17] described a simplified rover planning problem, which exhibits these characteristics. In this section, we apply the generalized HALP framework to solve this problem.

The rover problem is a hybrid factored MDP with three state variables S, T, and E, and a single binary action variable A. The discrete state variable S represents 10 exploration stages of the rover, the continuous variable T describes elapsed time, and the continuous variable E reflects the energy level of the rover. Transition functions for the state variables T and E are normal distributions that are conditioned on the action choice a, exploration stage s, elapsed time t, and energy level e [17]. Three branches of the exploration plan yield rewards of 10, 55, and 100. The optimization problem is to choose one of the branches with respect to the elapsed time and remaining energy. A complete description of this example can be found in Bresina *et al.* [17].

The optimal value function V^* for the rover problem is approximated by a linear combination of multivariate basis functions:

$$f(s,t,e) = P(s \mid \theta_1, \ldots, \theta_{10})\mathcal{N}(t \mid \mu_t, \sigma_t)\mathcal{N}(e \mid \mu_e, \sigma_e). \tag{4.12}$$

The univariate basis function factors $\mathcal{N}(t \mid \mu_t, \sigma_t)$ and $\mathcal{N}(e \mid \mu_e, \sigma_e)$ are centered at the vertices of some uniform $n \times n$ grid over the state variables T and E. The parameter n controls the complexity of the approximation. The variance of the univariate factors is computed as $(|\mathrm{Dom}(X_i)|/(n-1))^2$, where $|\mathrm{Dom}(T)| = 4200$ and $|\mathrm{Dom}(E)| = 20$. The $n \times n$ configurations of n^2 basis functions are replicated for all exploration stages s. The weights of the linear approximation $V^{\mathbf{w}}$ are obtained by solving a relaxed HALP formulation whose constraints are restricted to a fixed ε-grid ($\varepsilon = 1/16$). The state relevance density function $\psi(\mathbf{x})$ is uniform. The discount factor γ is 0.95.

As a baseline for our approximations, we consider value functions computed by value iteration on uniformly discretized variables T and E [19, 83]. The value iteration algorithm converges after 5 steps due to the finite-horizon dynamics of the rover problem.

All experiments are done on a Dell Precision 380 workstation with 3.2GHz Pentium 4 CPU and 2GB RAM. Linear programs are solved by the dual-simplex method in CPLEX. Our experimental results are reported in Figures 25, 26, and 27.

4.6.2 Experimental Results

Based on our results, we draw the following conclusions. The generalized HALP is a conceptually valid and practical way of solving hybrid optimization problems. Although our basis functions are

Generalized HALP				Grid-based VI		
Basis configurations	Reward	Time	OV	Grid configurations	Reward	Time
2×2 (41)	27.2 ± 0.5	5	60.5	5×5 (250)	25.5 ± 0.5	< 1
3×3 (131)	27.2 ± 0.5	18	56.3	9×9 (810)	26.2 ± 0.5	2
5×5 (381)	27.2 ± 0.5	75	49.9	17×17 (2 890)	27.2 ± 0.5	20
9×9 (1 191)	27.2 ± 0.5	560	42.8	33×33 (10 890)	27.4 ± 0.5	281

Figure 25: Comparison of two approaches to solving the simplified rover problem. The approaches are compared by the objective value of a relaxed HALP (OV), the expected discounted reward of a corresponding policy, and computation time (in seconds). The expected reward is estimated by the Monte Carlo simulation of 5000 trajectories starting at the initial exploration stage s_1 (Figure 26). The variance of our estimates is due to the natural variance of policies at s_1. The results are shown for different configurations of basis functions (grid points). The numbers in parentheses represent the number of used basis functions (grid points).

placed blindly without prior knowledge, the simplest HALP approximation (2×2 basis configuration) yields a close-to-optimal policy. The approximation is computed faster than a corresponding grid approximation of the same quality. This result is even more encouraging since we may achieve additional several-fold speedup by considering the locality of basis functions in HALP.

Interestingly, although the quality of HALP approximations improves with a larger number of basis functions, the quality of their policies remains the same. Since the optimal value function V^* [17] is monotonically increasing in both T and E, we believe that capturing this behavior is crucial for obtaining a close-to-optimal policy. Note that the simplest HALP approximation (2×2 basis configuration) exhibits this trend.

4.7 CONCLUSIONS

Development of efficient algorithms for solving large factored MDPs is a challenging task. In this chapter, we demonstrated a non-trivial extension to the HALP framework (Chapter 3) that permits efficient solutions to a very general class of hybrid factored MDPs. Moreover, we used the HALP

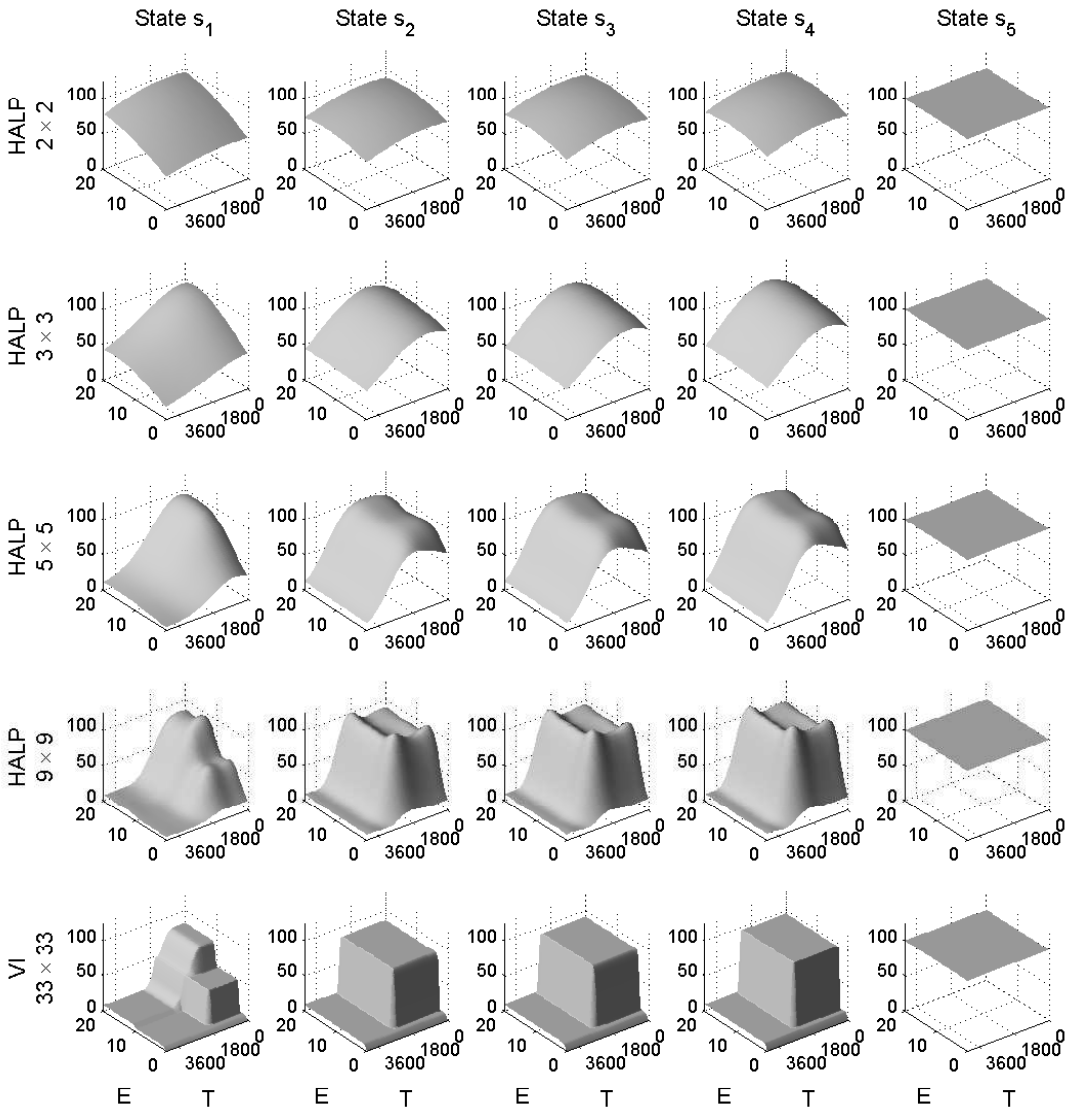


Figure 26: Value function approximations corresponding to the results in Figure 25. The approximations are shown as functions of the elapsed time (T) and energy (E) for every exploration stage $S = \{s_1, \ldots, s_{10}\}$. Note that the most elaborate HALP approximation (9×9 basis configuration) closely resembles the optimal value function [17].

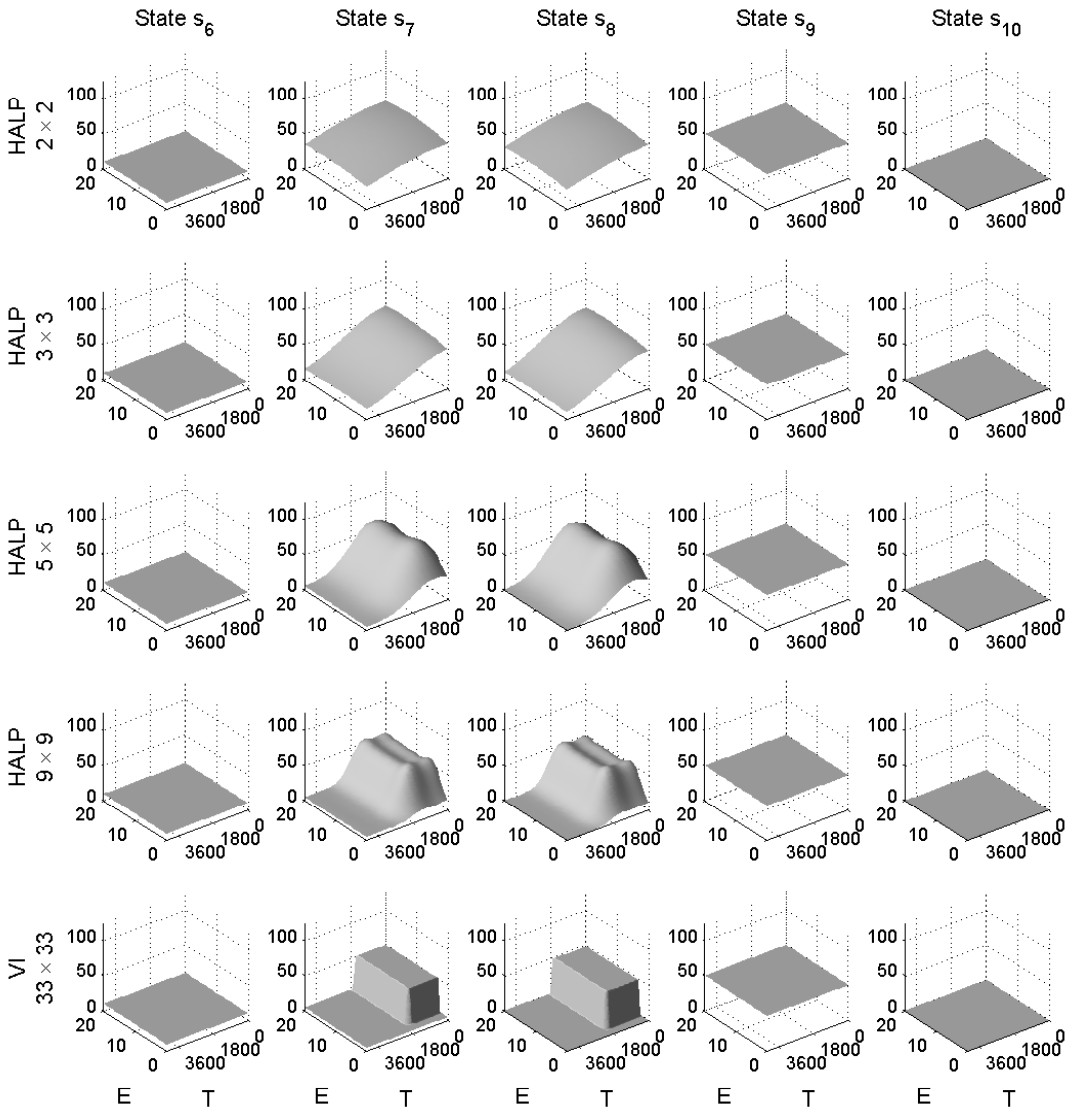


Figure 27: Value function approximations corresponding to the results in Figure 25. The approximations are shown as functions of the elapsed time (T) and energy (E) for every exploration stage $S = \{s_1, \ldots, s_{10}\}$. Note that the most elaborate HALP approximation (9×9 basis configuration) closely resembles the optimal value function [17].

framework to solve an autonomous rover problem and found a close-to-optimal policy with a small set of blindly constructed basis functions.

Unfortunately, such a naive approach to choosing basis functions is infeasible if the number of state variables is larger. To alleviate the concern, we propose an efficient method for learning basis functions in the next chapter.

5.0 LEARNING BASIS FUNCTIONS IN HYBRID DOMAINS

The quality of HALP solutions (Section 3.2) inherently depends on the choice of basis functions (Section 3.2.1). Therefore, it is typically assumed that these are provided as a part of the problem definition, which is unrealistic. The primary goal of this chapter is to alleviate this assumption and learn basis functions automatically.

In the context of discrete-state ALP, Patrascu *et al.* [81] proposed a greedy approach to learning basis functions. This method is based on the dual ALP formulation and its optimization. Although the presented approach is very similar to Patrascu *et al.* [81], it also differs in two aspects. First, it is computationally infeasible to build the complete HALP formulation in hybrid domains. Therefore, we use its relaxed formulations, which may lead to overfitting of learned approximations on active constraints. To solve this problem, we restrict our search to basis functions with a better state-space coverage. Second, instead of choosing from a finite number of basis functions [81], we optimize an infinite class of parametric basis function models. Note that these extensions are clearly non-trivial and pose a number of challenges. For a discussion of other approaches to learning basis functions, refer to Section 2.3.2.4.

The rest of Chapter 5 introduces a new method for the automatic learning of basis functions in hybrid state and action domains. This method starts from an initial set of basis functions and adds new functions *greedily* to improve the quality of the current approximation. Our approach is based on parametric basis function models that are optimized on preselected domains of state variables. These domains represent our initial preference between the quality and complexity of solutions. In what follows, we describe in detail how to score and optimize basis functions.

5.1 OPTIMIZATION OF RELAXED HALP

The quality of the ALP formulation (Section 2.3.2.3) was studied by de Farias and Van Roy [23]. Based on extending their results (Section 3.2.1), we conclude that the optimization of the objective function $\mathrm{E}_{\psi}[V^{\mathbf{w}}]$ in HALP is identical to minimizing the $\mathcal{L}_1$-norm error $\|V^* - V^{\mathbf{w}}\|_{1,\psi}$. Therefore, the objective value $\mathrm{E}_{\psi}\left[V^{\tilde{\mathbf{w}}}\right]$ is a natural measure for evaluating the impact of added basis functions. Unfortunately, computation of this objective value is generally infeasible (Section 3.3). To address this issue, we optimize a surrogate metric, which is given by the objective value of a relaxed HALP $\mathrm{E}_{\psi}\left[V^{\hat{\mathbf{w}}}\right]$. The following proposition relates the optimization of these two objectives.

Proposition 8 *Let $\tilde{\mathbf{w}}$ be an optimal solution to the HALP formulation (3.6) and $\hat{\mathbf{w}}$ be an optimal δ-infeasible solution to its relaxed formulation (3.17). Then the objective value $\mathrm{E}_{\psi}\left[V^{\tilde{\mathbf{w}}}\right]$ is bounded as:*

$$\mathrm{E}_{\psi}\left[V^{\tilde{\mathbf{w}}}\right] \leq \mathrm{E}_{\psi}\left[V^{\hat{\mathbf{w}}}\right] + \frac{\delta}{1-\gamma}.$$

The proposition has an important implication. The objective function $\mathrm{E}_{\psi}\left[V^{\tilde{\mathbf{w}}}\right]$ can be optimized by minimizing the relaxed objective $\mathrm{E}_{\psi}\left[V^{\hat{\mathbf{w}}}\right]$.

5.2 SCORING BASIS FUNCTIONS

To minimize the relaxed objective $\mathrm{E}_{\psi}\left[V^{\hat{\mathbf{w}}}\right]$, we introduce the dual formulation of a relaxed HALP.

Definition 6 *Let every variable in the relaxed HALP formulation (3.17) be subject to the constraint $w_i \geq 0$. Then the* dual relaxed HALP *is given by a linear program:*

$$\begin{aligned} \text{maximize}_{\omega} \quad & \sum_{(\mathbf{x},\mathbf{a})\in\mathcal{C}} \omega_{\mathbf{x},\mathbf{a}} R(\mathbf{x},\mathbf{a}) \\ \text{subject to:} \quad & \sum_{(\mathbf{x},\mathbf{a})\in\mathcal{C}} \omega_{\mathbf{x},\mathbf{a}} F_i(\mathbf{x},\mathbf{a}) - \alpha_i \leq 0 \quad \forall\, i \\ & \omega_{\mathbf{x},\mathbf{a}} \geq 0; \end{aligned} \tag{5.1}$$

where $\omega_{\mathbf{x},\mathbf{a}}$ represents the variables in the LP, one for each constraint in the primal relaxed HALP, and the scope of the index i are all basis functions $f_i(\mathbf{x})$.

Based on the duality theory, we know that the primal (3.17) and dual (5.1) formulations yield the same objective values. Therefore, minimizing the objective of the dual minimizes the objective of the primal. Since the dual formulation is a maximization problem, its objective can be lowered by adding a new constraint, which corresponds to a basis function $f(\mathbf{x})$ in the primal. Unfortunately, to evaluate the true impact of adding $f(\mathbf{x})$ on decreasing $\mathrm{E}_{\psi}\left[V^{\widehat{\mathbf{w}}}\right]$, the primal needs to be resolved with the added basis function. This step is computationally expensive and would significantly slow down any procedure that searches in the space of basis functions.

Similarly to Patrascu *et al.* [81], we consider a different scoring metric. We define *dual violation magnitude* $\tau^{\widehat{\omega}}(f)$:

$$\tau^{\widehat{\omega}}(f) = \sum_{(\mathbf{x},\mathbf{a})\in\mathcal{C}} \widehat{\omega}_{\mathbf{x},\mathbf{a}}[f(\mathbf{x}) - \gamma g_f(\mathbf{x},\mathbf{a})] - \alpha_f, \tag{5.2}$$

which measures the amount by which the optimal solution $\widehat{\omega}$ to a dual relaxed HALP violates the constraint corresponding to the basis function $f(\mathbf{x})$. This score can be interpreted as evaluating the quality of cutting planes in the dual. Intuitively, if $\tau^{\widehat{\omega}}(f)$ is nonnegative, a higher value of $\tau^{\widehat{\omega}}(f)$ is often correlated with a large decrease in the objective $\mathrm{E}_{\psi}\left[V^{\widehat{\mathbf{w}}}\right]$ when the basis function (constraint) $f(\mathbf{x})$ is added to the primal (dual). Following the work of Patrascu *et al.* [81], this criterion prefers meaningful basis functions and it is significantly cheaper than resolving the primal.

Note that computation of the dual violation magnitude $\tau^{\widehat{\omega}}(f)$ requires summation of the terms $g_f(\mathbf{x},\mathbf{a})$ and $f(\mathbf{x})$ for every constraint $(\mathbf{x},\mathbf{a}) \in \mathcal{C}$ in the primal. Fortunately, the scalars $\widehat{\omega}_{\mathbf{x},\mathbf{a}}$ (Equation 5.2) that correspond to inactive primal constraints are equal to zero. Therefore, our summation can be carried out efficiently in $O(|\widehat{\mathbf{w}}|)$ time, where $|\widehat{\mathbf{w}}|$ denotes the number of variables in the primal. In addition, based on the duality theory, the dual solution $\widehat{\omega}$ can be always expressed in terms of the primal solution $\widehat{\mathbf{w}}$. Therefore, the score $\tau^{\widehat{\omega}}(f)$ can be evaluated without formulating the dual at all.

5.3 OPTIMIZATION OF BASIS FUNCTIONS

The dual violation magnitude $\tau^{\widehat{\omega}}(f)$ scores basis functions $f(\mathbf{x})$ and our goal is to find one with a high score. To permit a systematic search among these functions, we assume that they factor along

the state variables $\mathbf{X}$ (Equation 4.8). Moreover, we assume that their univariate factors $f_j(x_j)$ are from the exponential family of distributions:

$$f_j(x_j) = h_j(x_j) \exp[\eta_{f_j}^\intercal t_j(x_j)]/Z_j(\eta_{f_j}), \tag{5.3}$$

where η_{f_j} denotes their natural parameters, $t_j(x_j)$ is a vector of sufficient statistics, and $Z_j(\eta_{f_j})$ is a normalizing function independent of X_j (Section 4.4).

To maximize the score $\tau^{\widehat{\omega}}(f)$ with respect to the natural parameters of the basis function $f(\mathbf{x})$, we use the gradient descent approach. Briefly, from the independence assumption in Equation 4.8, the gradient $\nabla\tau^{\widehat{\omega}}(f)$ can be expressed as a function of the derivatives corresponding to the univariate terms $f_j(x_j)$ and $\mathrm{E}_{P(x'_j|\mathbf{x},\mathbf{a})}\left[f_j(x'_j)\right]$. As demonstrated by Propositions 9 and 10, the derivatives have closed-form solutions for the conjugate choices of basis functions.

Proposition 9 *Let:*

$$f(x) = h(x) \exp[\eta_f^\intercal t(x)]/Z(\eta_f)$$

be an exponential-family density over X, where η_f denotes its natural parameters, $t(x)$ is a vector of sufficient statistics, and $Z(\eta_f)$ is a normalizing function independent of X. Then:

$$\frac{\partial f(x)}{\partial \eta_{f_k}} = f(x)\left[t_k(x) - \frac{1}{Z(\eta_f)}\frac{\partial Z(\eta_f)}{\partial \eta_{f_k}}\right]$$

has a closed-form solution, where η_{f_k} and $t_k(x)$ are the k-th elements of the vectors η_f and $t(x)$, respectively.

Proposition 10 *Let:*

$$\begin{aligned} P(x) &= h(x) \exp[\eta_P^\intercal t(x)]/Z(\eta_P) \\ f(x) &= h(x) \exp[\eta_f^\intercal t(x)]/Z(\eta_f) \end{aligned}$$

be exponential-family densities over X in the same canonical form, where η_P and η_f denote their natural parameters, $t(x)$ is a vector of their sufficient statistics, and $Z(\cdot)$ is a normalizing function independent of X. If $h(x) \equiv 1$, then:

$$\frac{\partial \mathrm{E}_{P(x)}[f(x)]}{\partial \eta_{f_k}} = \frac{1}{Z(\eta_P)Z(\eta_f)}\frac{\partial Z(\eta_P + \eta_f)}{\partial \eta_{f_k}} - \frac{Z(\eta_P + \eta_f)}{Z(\eta_P)Z(\eta_f)^2}\frac{\partial Z(\eta_f)}{\partial \eta_{f_k}}$$

has a closed-form solution, where η_{f_k} denotes the k-th element of the vector η_f.

5.4 OVERFITTING ON ACTIVE CONSTRAINTS

Maximization of the violation magnitude $\tau^{\widehat{\omega}}(f)$ *overfits* on active constraints in the primal relaxed HALP. To explain the phenomenon, we assume that some basis function $f(\mathbf{x})$ is of unit magnitude, unimodal, and centered at an active constraint $(\mathbf{x}', \mathbf{a}')$. If the function $f(\mathbf{x}'')$ is zero at the remaining active constraints $(\mathbf{x}'', \mathbf{a}'')$, $\widehat{\omega}_{\mathbf{x}',\mathbf{a}'} f(\mathbf{x}')$ is the only positive term in $\tau^{\widehat{\omega}}(f)$. Hence, the maximization of the dual score $\tau^{\widehat{\omega}}(f)$ can be achieved by fixing the functional value $f(\mathbf{x}') = 1$ while minimizing the negative terms $g_f(\mathbf{x}'', \mathbf{a}'')$ and α_f. Since the negative terms can be bounded from above as:

$$\begin{aligned} g_f(\mathbf{x}'', \mathbf{a}'') &= \mathrm{E}_{P(\mathbf{x}'|\mathbf{x}'',\mathbf{a}'')}[f(\mathbf{x}')] \\ &\leq \mathrm{E}[f(\mathbf{x})] \max_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}'', \mathbf{a}'') \qquad (5.4) \\ \alpha_f &= \mathrm{E}_{\psi(\mathbf{x})}[f(\mathbf{x})] \\ &\leq \mathrm{E}[f(\mathbf{x})] \max_{\mathbf{x}} \psi(\mathbf{x}), \qquad (5.5) \end{aligned}$$

the score $\tau^{\widehat{\omega}}(f)$ is easily optimized by decreasing the mass $\mathrm{E}[f(\mathbf{x})]$ (variance) corresponding to the basis function $f(\mathbf{x})$. A graphical illustration of a relaxed HALP approximation, which is computed from several overfitted basis functions, is shown in Figure 29.

Although these peaked basis functions may lower the relaxed objective $\mathrm{E}_\psi\left[V^{\widehat{\mathbf{w}}}\right]$, it is not likely that they lower the objective $\mathrm{E}_\psi\left[V^{\widetilde{\mathbf{w}}}\right]$ in HALP. This conclusion can be explained by Proposition 8. Peaked basis functions have a high Lipschitz constant, which translates into the high δ-infeasibility of their relaxed solutions $\widehat{\mathbf{w}}$. If δ is high, the bound in Proposition 8 is loose, and the minimization of the relaxed objective $\mathrm{E}_\psi\left[V^{\widehat{\mathbf{w}}}\right]$ no longer guarantees a low objective value $\mathrm{E}_\psi\left[V^{\widetilde{\mathbf{w}}}\right]$.

To account for this deficiency, we modify our greedy search (Section 5.3). Instead of searching among all basis functions that maximize $\tau^{\widehat{\omega}}(f)$, we focus on those that adhere to a certain Lipschitz factor K. In turn, the new parameter K regulates the smoothness of our approximations.

5.5 EXPERIMENTS

The greedy approach to learning basis functions is studied on two hybrid optimization problems: 6-ring irrigation network (Section 3.4.2) and an autonomous rover planning problem (Section 4.6.1).

5.5.1 Experimental Setup

Irrigation network problems [35] are challenging for state-of-the-art MDP solvers due to the hybrid factored state and action spaces. The objective of an irrigation network operator is to select discrete water-routing actions $\mathbf{A}_D$ to optimize continuous water levels $\mathbf{X}_C$ in connected irrigation channels. The transition model reflects water flows conditioned on the operation modes of regulation devices. It is parameterized locally by beta distributions. The reward model is additive and represented by a mixture of two normal distributions for every channel except for the outflow. The 6-ring irrigation network topology involves 10 continuous state and 10 discrete action variables.

The autonomous rover planning problem [17] involves three state variables S, T, and E, and a binary action variable A. The discrete state variable S represents 10 exploration stages of the rover, the continuous variable T describes elapsed time, and the continuous variable E reflects the energy level of the rover. Transition functions for the state variables T and E are normal distributions that are conditioned on the action choice a, exploration stage s, elapsed time t, and energy level e [17]. Three branches of the exploration plan yield rewards of 10, 55, and 100. The optimization problem is to choose one of the branches with respect to the elapsed time and remaining energy. A complete description of this example can be found in Bresina *et al.* [17].

The optimal value function V^* for the 6-ring irrigation network problem is approximated by a linear combination of univariate basis functions:

$$f(x_i) = P_{\text{beta}}(x_i \mid \alpha, \beta). \tag{5.6}$$

The parameters i, α, and β are initialized randomly and further optimized by two methods: *greedy* (Section 5.3), which maximizes the violation magnitude $\tau^{\widehat{\omega}}(f)$, and *restricted greedy* (Section 5.4), where the greedy optimization is controlled by the Lipschitz factor K. Both methods are evaluated for up to 100 learned basis functions. The smoothness parameter K is slowly increased from 2 to 8 according to a logarithmic schedule. After a new basis function is learned, the weights of the linear approximation $V^{\mathbf{w}}$ are obtained from a relaxed HALP formulation whose constraints are restricted to a fixed ε-grid ($\varepsilon = 1/8$). The state relevance density function $\psi(\mathbf{x})$ is assumed uniform similarly to Section 3.4.2. The discount factor γ is 0.95. The quality of learned approximations is compared to an ε-HALP approximation ($\varepsilon = 1/16$) computed from 40 basis functions suggested by Guestrin *et al.* [35] (Figure 16c).

The optimal value function V^* for the rover problem is approximated by a linear combination of multivariate basis functions:

$$f(s,t,e) = P(s \mid \theta_1,\ldots,\theta_{10})\mathcal{N}(t \mid \mu_t,\sigma_t)\mathcal{N}(e \mid \mu_e,\sigma_e). \tag{5.7}$$

where $P(s \mid \theta_1,\ldots,\theta_{10})$ is a multinomial distribution over 10 exploration stages of the rover. The parameters $\theta_1,\ldots,\theta_{10}$, μ_t, σ_t, μ_e, and σ_e are initialized randomly and optimized as described in the previous paragraph. The quality of learned approximations is compared to two different baselines: an ε-HALP approximation ($\varepsilon = 1/16$) with 381 basis functions (Section 4.6), and a value function obtained by value iteration on uniformly discretized variables T and E (Section 4.6). The variables are discretized on the 17×17 grid.

All experiments are done on a Dell Precision 380 workstation with 3.2GHz Pentium 4 CPU and 2GB RAM. Linear programs are solved by the dual-simplex method in CPLEX. Our experimental results are reported in Figures 28 and 29.

5.5.2 Experimental Results

Based on our results, we draw the following conclusions. First, Figure 28 demonstrates the benefits of automatic basis function learning. On the 6-ring irrigation network problem, we can learn better policies than the existing baseline in a short time (150 seconds). On the autonomous rover problem, we learn as good policies as our baselines in a comparable time. These empirical results are even more encouraging since we may achieve additional several-fold speedup by caching relaxed HALP formulations when adding new basis functions.

Second, Figure 28 also confirms our hypothesis that the minimization of the relaxed objective $\mathrm{E}_{\psi}\left[V^{\widehat{\mathbf{w}}}\right]$ without restricting the search yields suboptimal policies. As the number of learned basis functions grows, we observe a correlation between dropping objectives and rewards, and a growing upper bound on the Lipschitz factor of the approximations.

Finally, Figure 29 illustrates value functions learned on the 6-ring irrigation network problem. We observe the phenomenon of overfitting (the second row from the top) and the gradual improvement of the approximations constructed by the restricted greedy search (the last two rows).

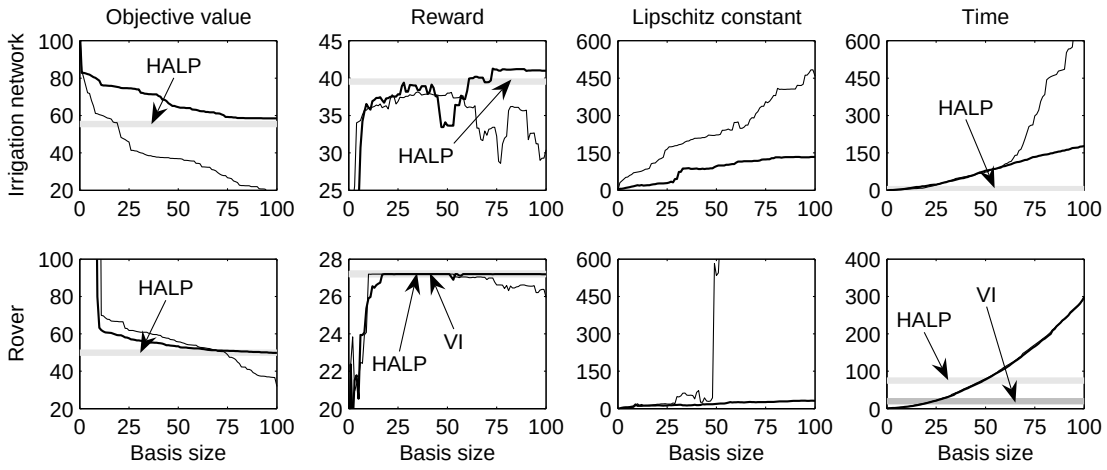


Figure 28: Comparison of the greedy (thin black lines) and restricted greedy (thick black lines) methods on the 6-ring irrigation network and rover problems. The approaches are compared by the objective value of a relaxed HALP, the expected reward of a corresponding policy, an upper bound on the Lipschitz constant of $V^{\hat{\mathbf{w}}}$, and computation time (in seconds). The upper bound is computed as $\sum_i \widehat{w}_i \sum_{X_j \in \mathbf{X}_i} K_{ij}$, where K_{ij} represents the Lipschitz constant of the univariate basis function factor $f_{ij}(x_j)$. Thick gray lines denote our baselines.

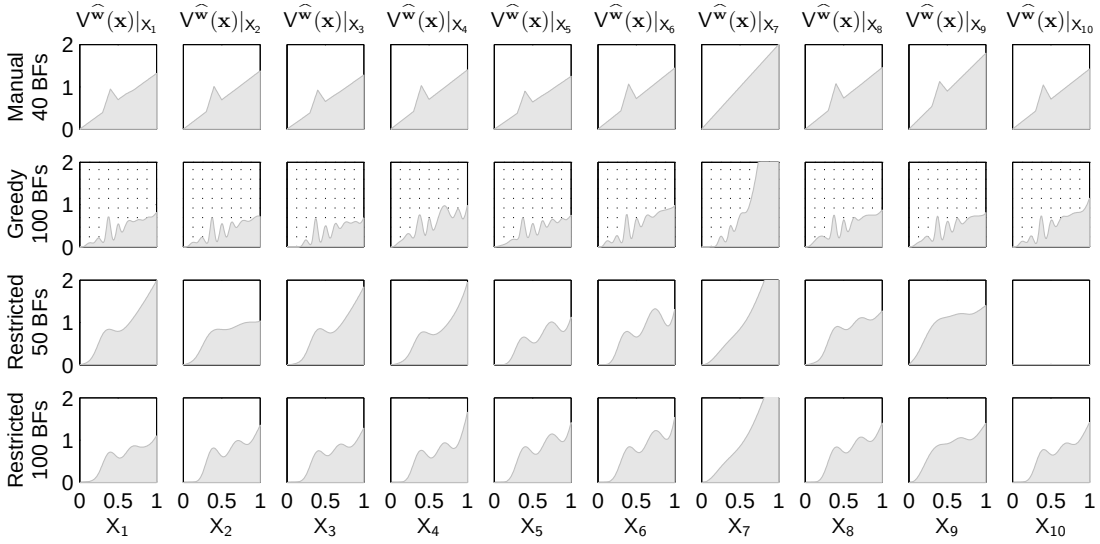


Figure 29: Univariate projections $V^{\widehat{\mathbf{w}}}(\mathbf{x})|_{X_j} = \sum_{i:X_j=X_i} \widehat{w}_i f_i(x_i)$ of approximate value functions $V^{\widehat{\mathbf{w}}}$ on the 6-ring irrigation network problem. From top down, we show value functions computed from 40 basis functions (BFs) suggested by Guestrin *et al.* [35], 100 greedily learned BFs, and 50 and 100 BFs learned by the restricted greedy method. Note that the greedy approximation overfits on the ε-grid ($\varepsilon = 1/8$), which is denoted by dotted lines.

5.6 CONCLUSIONS

Learning of basis functions is a step towards applying MDPs to real-world problems. In this work, we introduced a greedy method that automatically learns basis functions in hybrid state and action domains. On the two presented hybrid MDP problems, the method surpasses existing baselines by the quality of policies or their computation time. An interesting open question is the combination of our greedy search with the state space analysis of Mahadevan *et al.* [64, 65, 66].

6.0 CONCLUSIONS AND FUTURE WORK

Compact representations and efficient solutions for large-scale decision problems with discrete and continuous variables are among the most important challenges faced by the designers of automated decision support systems. In this thesis (Chapters 3 and 4), we introduced a novel hybrid factored Markov decision process (MDP) model that allows for a compact representation of these problems, and a new hybrid approximate linear programming (HALP) framework that permits their efficient solutions. Comparing to the existing work (Section 2.3), the introduced approach has several nice properties. First, unlike other parametric value function approximation methods (Section 2.3.2), it cannot diverge (or oscillate) in principle and its feasibility is easily guaranteed. Second, in contrast to parametric policy approximations (Section 2.3.3), HALP policies can be derived without relying on local optimization methods and a potentially suboptimal parametric class of policies (Section 5). Finally, comparing to the nearest neighbor methods (Section 2.3.4), HALP has a potential to scale up to large distributed decision problems.

Although our work focused specifically on linear programming methods, note that many of our ideas generalize beyond them. For instance, our efficient closed-form solutions to the expectations terms in HALP (Sections 3.2.2 and 4.4) extend to any technique for solving hybrid factored MDPs that employs the linear value function approximation (Equation 2.20). Moreover, our work builds a foundation for the efficient evaluation of the hybrid Bellman operator (Equation 3.4) in the domains of discrete and continuous variables (Section 3.3.3).

Based on our empirical results, we claim that the HALP framework scales up with the dimensionality of studied problems (Section 3.4). Furthermore, it is capable of solving decision problems with non-trivial dynamics (Section 4.6) with no prior knowledge at all (Section 5.5). At this point, further empirical evidence is necessary to make stronger claims. Ultimately, we believe that HALP can solve structured decision problems with hundreds of variables. Note that the irrigation network

problems (Example 6) are among the largest factored MDPs with hybrid state and action variables that were ever solved by an approximate dynamic programming technique [10, 91].

Future work on the HALP framework should definitely focus on the automatic learning of basis functions and faster constraint space approximations. For instance, we believe that the incremental learning of basis functions (Section 5) may provide a good prior for the greedy search for the most violated constraint. This local optimization step is expected to consume a significantly shorter time than our global constraint satisfaction methods (Section 3.3) while delivering competitive results.

Interestingly, the majority of our empirical time complexity results can be reduced on the order of ten or more. For example, the LP_SOLVE package applied in Section 3.4 is significantly slower than later used CPLEX (Sections 4.6 and 5.5). Moreover, our basis functions are local. As a result, most of the expectation terms in HALP formulations (Sections 3.2.2 and 3.3) are typically zero. A smarter representation of basis functions, which avoids these unnecessary computations, can speed up the building of relaxed HALP formulations by several times.

In the rest of this chapter, we discuss the extension of our ideas to hybrid factored semi-MDPs [50, 51] and partially-observable MDPs [2]. In the light of these results, the HALP approach seems to be a promising and general framework for solving large-scale dynamic decision problems under uncertainty.

6.1 SEMI-MARKOV DECISION PROCESSES

In the course of this work, we assumed that Markov decision processes are discrete-time. However, the nature of many real-world problems cannot be captured by these models. For instance, although the actions are usually applied at instant times, rewards often accumulate between these times [21]. Moreover, the duration of a transition between two states is typically a random variable conditioned on the preceding state and action choice. Semi-Markov decision processes [50, 51] are a formalism that permits the modeling of these new challenges while preserving the Markov property.

Formally, a *semi-Markov decision process* [9] is given by a 4-tuple $\mathcal{M}_\tau = (\mathcal{S}, \mathcal{A}, P, r)$, where $\mathcal{S} = \{s_1, \ldots, s_{|\mathcal{S}|}\}$ is a set of states, $\mathcal{A} = \{a_1, \ldots, a_{|\mathcal{A}|}\}$ is a set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \tau \to \mathbb{R}$ is a joint probability distribution over the state and its transition time conditioned on the preceding

state and action choice, and $r : \mathcal{S} \times \mathcal{A} \times t \to \mathbb{R}$ is a reward model assigning immediate payoffs to state-action configurations at a particular time t.

Under mild restrictions on the transition model, we can derive the following fixed point equation:

$$V^*(s) = \max_a \left[R(s,a) + \sum_{s' \neq s} \left\{ \int_{\tau=0}^{\infty} \exp[-\gamma\tau] P(s', \tau \mid s, a) \, \mathrm{d}\tau \right\} V^*(s') \right] \tag{6.1}$$

for the optimal value function V^*, where:

$$R(s,a) = \sum_{s' \neq s} \int_{\tau=0}^{\infty} \left\{ \int_{t=0}^{\tau} \exp[-\gamma t] r(s,a,t) \, \mathrm{d}t \right\} P(s', \tau \mid s, a) \, \mathrm{d}\tau. \tag{6.2}$$

Based on the similarity to the Bellman equation (Equation 2.4), it is possible to write down the LP and ALP formulations (2.17 and 2.25) that allow the computation of the optimal value function V^* (Equation 6.1) and its linear approximation $V^{\mathbf{w}}$ (Equation 2.20):

$$\begin{aligned} \text{minimize}_{\mathbf{w}} \quad & \sum_s \psi(s) V^{\mathbf{w}}(s) \\ \text{subject to:} \quad & V^{\mathbf{w}}(s) - \sum_{s' \neq s} \left\{ \int_{\tau=0}^{\infty} \exp[-\gamma\tau] P(s', \tau \mid s, a) \, \mathrm{d}\tau \right\} V^{\mathbf{w}}(s') - R(s,a) \geq 0 \\ & \forall \; s \in \mathcal{S}, a \in \mathcal{A}; \end{aligned} \tag{6.3}$$

An interesting open question is whether the expectation terms $\int_{\tau=0}^{\infty} \exp[-\gamma\tau] P(s', \tau \mid s, a) \, \mathrm{d}\tau$ and $R(s,a)$ can be evaluated efficiently for a rich class of transition functions. To simplify the problem, we may consider rewriting the transition model $P(s', \tau \mid s, a)$ as a kernel $P(\tau \mid s, a, s') P(s' \mid s, a)$.

For representing the temporal dynamics of semi-MDPs, Younes and Simmons [101] suggested continuous phase-type distributions. These transition functions are computationally feasible due to the memoryless properties of exponential distributions, which are the building blocks of phase-type distributions. In addition, note that these transition models have a clear semantics when combined with factored representations [77].

6.2 PARTIALLY-OBSERVABLE MARKOV DECISION PROCESSES

The standard MDP framework assumes that the state space is completely observed. This paradigm is valid for modeling a variety of real-world optimization problems. However, the framework cannot be applied to domains with incomplete state information, such as navigating robots in unknown environments. Partially-observable Markov decision processes [2] are a formalism that permits the modeling of these new challenges.

Formally, a *partially-observable Markov decision process (POMDP)* [2] is given by a 5-tuple $\mathcal{M}_o = (S, A, \Theta, P, R)$, where $\mathcal{S} = \left\{s_1, \ldots, s_{|\mathcal{S}|}\right\}$ is a set of unobserved states, $\mathcal{A} = \left\{a_1, \ldots, a_{|\mathcal{A}|}\right\}$ is a set of actions, $\Theta = \left\{o_1, \ldots, o_{|\Theta|}\right\}$ is a set of observations, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a stochastic model of state dynamics conditioned on the preceding state and action, $P : S \times A \times \Theta \rightarrow [0, 1]$ is the probability of observations conditioned on the unobserved state and action, and $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a reward model assigning immediate payoffs to state-action configurations.

Assuming the *belief-state* representation of a POMDP [42], we can derive the following fixed point equation:

$$V^*(\mathbf{b}) = \max_a \left[\sum_s b_s R(s, a) + \gamma \sum_o P(o \mid \mathbf{b}, a) V^*(\mathbf{b}_o^a) \right] \tag{6.4}$$

for the optimal value function V^*. The vector $\mathbf{b} = (b_1, \ldots, b_{|\mathcal{S}|})$ represents a belief over the states s such that $\sum_s b_s = 1$, and $\mathbf{b}_o^a$ is its update after taking an action a and observing o. This update is governed by the equation:

$$(\mathbf{b}_o^a)_{s'} = \frac{P(o \mid s', a) \sum_s P(s' \mid s, a) b_s}{P(o \mid \mathbf{b}, a)}, \tag{6.5}$$

where the normalizing constant $P(o \mid \mathbf{b}, a)$ is computed as:

$$P(o \mid \mathbf{b}, a) = \sum_{s'} P(o \mid s', a) \sum_s P(s' \mid s, a) b_s. \tag{6.6}$$

Note that the belief state $\mathbf{b}$ is continuous and can be characterized by a set of $|\mathcal{S}|$ dependent random variables $\mathbf{B} = \left\{B_1, \ldots, B_{|\mathcal{S}|}\right\}$. Therefore, the belief-state representation of a POMDP is a special

form of a continuous-state MDP. As a result, we can modify the HALP formulation (3.6) to find a linear approximation $V^{\mathbf{w}}$ (Equation 2.20):

$$\begin{aligned} \text{minimize}_{\mathbf{w}} \quad & \mathrm{E}_{\psi}[V^{\mathbf{w}}] \\ \text{subject to:} \quad & V^{\mathbf{w}}(\mathbf{b}) - \gamma \sum_{o} P(o \mid \mathbf{b}, a) V^{\mathbf{w}}(\mathbf{b}_o^a) - \sum_{s} b_s R(s,a) \geq 0 \\ & \forall\, \mathbf{b} \in \mathbf{B} : \sum_{s} b_s = 1 \\ & \forall\, a \in \mathcal{A}; \end{aligned} \tag{6.7}$$

to the optimal value function V^* (Equation 6.4).

In line with our previous discussion, solving POMDPs by HALP can be approached as follows. First, the optimal value function of a POMDP is always convex in the belief state $\mathbf{b}$ [2]. Therefore, to approximate it, it suffices to consider convex basis function $f_i(\mathbf{b})$ and their convex combinations:

$$V^{\mathbf{w}}(\mathbf{b}) = \sum_{i} w_i f_i(\mathbf{b}) \quad \forall\, i : w_i \geq 0. \tag{6.8}$$

Intuitively, these convex basis functions can be learned similarly to those in Chapter 5. Second, for every belief state $\mathbf{b}$ and action a, there are only $|\Theta|$ successive states $\mathbf{b}_o^a$ that a POMDP can transfer to. Therefore, the evaluation of the expectation terms in HALP (Section 3.2.2) reduces to a trivial problem. Finally, similarly to the constraint space in HALP (Section 3.3), we hypothesize that the POMDP constraints can be satisfied compactly even if the action and observation spaces $\mathcal{A}$ and Θ are exponentially large but structured.

APPENDIX

PROOFS

Proof of Proposition 2: The Bellman operator $\mathcal{T}^*$ is a contraction mapping. From its monotonicity, for any value function V, $V \geq \mathcal{T}^* V$ implies $V \geq \mathcal{T}^* V \geq \cdots \geq V^*$. Since constraints in the HALP formulation (3.6) enforce $V^{\tilde{\mathbf{w}}} \geq \mathcal{T}^* V^{\tilde{\mathbf{w}}}$, we conclude $V^{\tilde{\mathbf{w}}} \geq V^*$. ■

Proof of Proposition 3: Based on Proposition 2, we note that the constraint $V^{\mathbf{w}} \geq \mathcal{T}^* V^{\mathbf{w}}$ guarantees that $V^{\mathbf{w}} \geq V^*$. Subsequently, our claim is proved by realizing:

$$\arg\min_{\mathbf{w}} \mathrm{E}_\psi[V^{\mathbf{w}}] = \arg\min_{\mathbf{w}} \mathrm{E}_\psi[V^{\mathbf{w}} - V^*]$$

and

$$\begin{aligned} \mathrm{E}_\psi[V^{\mathbf{w}} - V^*] &= \mathrm{E}_\psi|V^{\mathbf{w}} - V^*| \\ &= \mathrm{E}_\psi|V^* - V^{\mathbf{w}}| \\ &= \|V^* - V^{\mathbf{w}}\|_{1,\psi} . \end{aligned}$$

The proof generalizes from the discrete-state case [23] without any alternations. ■

Proof of Theorem 2: Similarly to Theorem 2 by de Farias and Van Roy [23], this claim is proved in three steps. First, we find a point $\overline{\mathbf{w}}$ in the feasible region of the HALP such that $V^{\overline{\mathbf{w}}}$ is within $O(\epsilon)$ distance from $V^{\mathbf{w}^*}$, where:

$$\begin{aligned} \mathbf{w}^* &= \arg\min_{\mathbf{w}} \|V^* - V^{\mathbf{w}}\|_\infty \\ \epsilon &= \left\|V^* - V^{\mathbf{w}^*}\right\|_\infty. \end{aligned}$$

Such a point $\overline{\mathbf{w}}$ is given by:

$$\overline{\mathbf{w}} = \mathbf{w}^* + \frac{(1+\gamma)\epsilon}{1-\gamma} e,$$

where $e = (1, 0, \ldots, 0)$ is an indicator of the constant basis function $f_0(\mathbf{x}) \equiv 1$. This point satisfies all requirements and its feasibility can be handily verified by solving:

$$\begin{aligned} V^{\overline{\mathbf{w}}} - \mathcal{T}^* V^{\overline{\mathbf{w}}} &= V^{\mathbf{w}^*} + \frac{(1+\gamma)\epsilon}{1-\gamma} - \left(\mathcal{T}^* V^{\mathbf{w}^*} + \frac{\gamma(1+\gamma)\epsilon}{1-\gamma}\right) \\ &= V^{\mathbf{w}^*} - \mathcal{T}^* V^{\mathbf{w}^*} + (1+\gamma)\epsilon \\ &\geq 0, \end{aligned}$$

where the last step follows from the inequality:

$$\begin{aligned} \left\|V^{\mathbf{w}^*} - \mathcal{T}^* V^{\mathbf{w}^*}\right\|_\infty &\leq \left\|V^{\mathbf{w}^*} - V^*\right\|_\infty + \left\|V^* - \mathcal{T}^* V^{\mathbf{w}^*}\right\|_\infty \\ &= \left\|V^* - V^{\mathbf{w}^*}\right\|_\infty + \left\|\mathcal{T}^* V^* - \mathcal{T}^* V^{\mathbf{w}^*}\right\|_\infty \\ &\leq (1+\gamma)\epsilon. \end{aligned}$$

Subsequently, we bound the max-norm error of $V^{\overline{\mathbf{w}}}$ by using the triangle inequality:

$$\begin{aligned} \left\|V^* - V^{\overline{\mathbf{w}}}\right\|_\infty &\leq \left\|V^* - V^{\mathbf{w}^*}\right\|_\infty + \left\|V^{\mathbf{w}^*} - V^{\overline{\mathbf{w}}}\right\|_\infty \\ &= \left(1 + \frac{1+\gamma}{1-\gamma}\right)\epsilon \\ &= \frac{2\epsilon}{1-\gamma}, \end{aligned}$$

which yields a bound on the weighted $\mathcal{L}_1$-norm error of $V^{\tilde{\mathbf{w}}}$:

$$\begin{aligned} \left\|V^* - V^{\tilde{\mathbf{w}}}\right\|_{1,\psi} &\leq \left\|V^* - V^{\overline{\mathbf{w}}}\right\|_{1,\psi} \\ &\leq \left\|V^* - V^{\overline{\mathbf{w}}}\right\|_\infty \\ &\leq \frac{2\epsilon}{1-\gamma}. \end{aligned}$$

The proof generalizes from the discrete-state case [23] without any alternations. ■

Proof of Theorem 3: Similarly to Theorem 2, this claim is proved in three steps: finding a point $\overline{\mathbf{w}}$ in the feasible region of the HALP, bounding the max-norm error of $V^{\overline{\mathbf{w}}}$, which yields a bound on the $\mathcal{L}_1$-norm error of $V^{\tilde{\mathbf{w}}}$. A comprehensive proof for the discrete-state case was done by de Farias and Van Roy [23]. The proof generalizes to structured state and action spaces with hybrid variables. ∎

Proof of Proposition 4: The proposition is proved in a sequence of steps:

$$\begin{aligned}
\mathrm{E}_{P(x)}[f(x)] &= \int_x P_{\mathrm{beta}}(x \mid \alpha, \beta) x^n (1-x)^m \,\mathrm{d}x \\
&= \int_x \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} x^n (1-x)^m \,\mathrm{d}x \\
&= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \int_x x^{\alpha+n-1}(1-x)^{\beta+m-1} \,\mathrm{d}x \\
&= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \frac{\Gamma(\alpha+n)\Gamma(\beta+m)}{\Gamma(\alpha+\beta+n+m)} \int_x \frac{\Gamma(\alpha+\beta+n+m)}{\Gamma(\alpha+n)\Gamma(\beta+m)} x^{\alpha+n-1}(1-x)^{\beta+m-1} \,\mathrm{d}x \\
&= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \frac{\Gamma(\alpha+n)\Gamma(\beta+m)}{\Gamma(\alpha+\beta+n+m)} \underbrace{\int_x P_{\mathrm{beta}}(x \mid \alpha+n, \beta+m) \,\mathrm{d}x}_{1} .
\end{aligned}$$

Since integration is a distributive operation, our claim straightforwardly generalizes to the mixture of beta distributions $P(x)$. ∎

Proof of Proposition 5: The proposition is proved in a sequence of steps:

$$\begin{aligned}
\mathrm{E}_{P(x)}[f(x)] &= \int_x P_{\mathrm{beta}}(x \mid \alpha, \beta) \sum_i \mathbf{1}_{[l_i, r_i]}(x)(a_i x + b_i) \,\mathrm{d}x \\
&= \sum_i \int_{l_i}^{r_i} P_{\mathrm{beta}}(x \mid \alpha, \beta)(a_i x + b_i) \,\mathrm{d}x \\
&= \sum_i \left[a_i \int_{l_i}^{r_i} P_{\mathrm{beta}}(x \mid \alpha, \beta) x \,\mathrm{d}x + b_i \int_{l_i}^{r_i} P_{\mathrm{beta}}(x \mid \alpha, \beta) \,\mathrm{d}x \right] \\
&= \sum_i \left[a_i \frac{\alpha}{\alpha+\beta} \int_{l_i}^{r_i} P_{\mathrm{beta}}(x \mid \alpha+1, \beta) \,\mathrm{d}x + b_i \int_{l_i}^{r_i} P_{\mathrm{beta}}(x \mid \alpha, \beta) \,\mathrm{d}x \right] \\
&= \sum_i \left[a_i \frac{\alpha}{\alpha+\beta} (F^+(r_i) - F^+(l_i)) + b_i (F(r_i) - F(l_i)) \right] .
\end{aligned}$$

Since integration is a distributive operation, our claim straightforwardly generalizes to the mixture of beta distributions $P(x)$. ∎

Proof of Proposition 6: This claim is proved in three steps. First, we find a point $\overline{\mathbf{w}}$ in the feasible region of the HALP such that $V^{\overline{\mathbf{w}}}$ is within $O(\delta)$ distance from $V^{\widehat{\mathbf{w}}}$. Such a point $\overline{\mathbf{w}}$ is given by:

$$\overline{\mathbf{w}} = \widehat{\mathbf{w}} + \frac{\delta}{1-\gamma} e,$$

where $e = (1, 0, \ldots, 0)$ is an indicator of the constant basis function $f_0(\mathbf{x}) \equiv 1$. This point satisfies all requirements and its feasibility can be handily verified by solving:

$$\begin{aligned} V^{\overline{\mathbf{w}}} - \mathcal{T}^* V^{\overline{\mathbf{w}}} &= V^{\widehat{\mathbf{w}}} + \frac{\delta}{1-\gamma} - \left(\mathcal{T}^* V^{\widehat{\mathbf{w}}} + \frac{\gamma\delta}{1-\gamma} \right) \\ &= V^{\widehat{\mathbf{w}}} - \mathcal{T}^* V^{\widehat{\mathbf{w}}} + \delta \\ &\geq 0, \end{aligned}$$

where the inequality $V^{\widehat{\mathbf{w}}} - \mathcal{T}^* V^{\widehat{\mathbf{w}}} \geq -\delta$ holds from the δ-infeasibility of $\widehat{\mathbf{w}}$. Since the solution $\widetilde{\mathbf{w}}$ is feasible in the relaxed HALP, we conclude $\mathrm{E}_\psi\left[V^{\widehat{\mathbf{w}}}\right] \leq \mathrm{E}_\psi\left[V^{\widetilde{\mathbf{w}}}\right]$. Subsequently, this inequality yields a bound on the weighted $\mathcal{L}_1$-norm error of $V^{\overline{\mathbf{w}}}$:

$$\begin{aligned} \left\| V^* - V^{\overline{\mathbf{w}}} \right\|_{1,\psi} &= \mathrm{E}_\psi \left| V^{\widehat{\mathbf{w}}} + \frac{\delta}{1-\gamma} - V^* \right| \\ &= \mathrm{E}_\psi\left[V^{\widehat{\mathbf{w}}}\right] + \frac{\delta}{1-\gamma} - \mathrm{E}_\psi[V^*] \\ &\leq \mathrm{E}_\psi\left[V^{\widetilde{\mathbf{w}}}\right] + \frac{\delta}{1-\gamma} - \mathrm{E}_\psi[V^*] \\ &= \left\| V^* - V^{\widetilde{\mathbf{w}}} \right\|_{1,\psi} + \frac{\delta}{1-\gamma}. \end{aligned}$$

Finally, we combine this result with the triangle inequality:

$$\begin{aligned} \left\| V^* - V^{\widehat{\mathbf{w}}} \right\|_{1,\psi} &\leq \left\| V^* - V^{\overline{\mathbf{w}}} \right\|_{1,\psi} + \left\| V^{\overline{\mathbf{w}}} - V^{\widehat{\mathbf{w}}} \right\|_{1,\psi} \\ &\leq \left\| V^* - V^{\widetilde{\mathbf{w}}} \right\|_{1,\psi} + \frac{2\delta}{1-\gamma}, \end{aligned}$$

which leads to a bound on the weighted $\mathcal{L}_1$-norm error of $V^{\widehat{\mathbf{w}}}$. ■

Proof of Proposition 7: The proposition is proved in a sequence of steps:

$$\begin{aligned}
\mathrm{E}_{P(x)}[f(x)] &= \int_x P(x) f(x)\,\mathrm{d}x \\
&= \int_x \frac{\exp[\eta_P^\mathsf{T} t(x)]}{Z(\eta_P)} \frac{\exp[\eta_f^\mathsf{T} t(x)]}{Z(\eta_f)}\,\mathrm{d}x \\
&= \frac{Z(\eta_P + \eta_f)}{Z(\eta_P) Z(\eta_f)} \underbrace{\int_x \frac{\exp[(\eta_P + \eta_f)^\mathsf{T} t(x)]}{Z(\eta_P + \eta_f)}\,\mathrm{d}x}_{1}.
\end{aligned}$$

The final step is a consequence of integrating a distribution. ■

Proof of Corollary 2: The proposition is proved in a sequence of steps:

$$\begin{aligned}
\mathrm{E}_{P(x)}[f(x)] &= \int_x P(x) f(x)\,\mathrm{d}x \\
&= \int_x \left(\sum_i \pi_i \frac{\exp[\eta_{P_i}^\mathsf{T} t(x)]}{Z(\eta_{P_i})} \right) \left(\sum_j \rho_j \frac{\exp[\eta_{f_j}^\mathsf{T} t(x)]}{Z(\eta_{f_j})} \right) \mathrm{d}x \\
&= \sum_i \sum_j \pi_i \rho_j \int_x \frac{\exp[\eta_{P_i}^\mathsf{T} t(x)]}{Z(\eta_{P_i})} \frac{\exp[\eta_{f_j}^\mathsf{T} t(x)]}{Z(\eta_{f_j})}\,\mathrm{d}x \\
&= \sum_i \sum_j \pi_i \rho_j \frac{Z(\eta_{P_i} + \eta_{f_j})}{Z(\eta_{P_i}) Z(\eta_{f_j})}.
\end{aligned}$$

The final step is a consequence of applying Proposition 7. ■

Proof of Corollary 3: This proposition is proved in two steps. First, note that the normal distribution can be written in the canonical form as:

$$\begin{aligned}
\mathcal{N}(x \mid \mu, \sigma) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(x-\mu)^2\right] \\
&= \exp\left[-\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x\right] \exp\left[-\frac{\mu^2}{2\sigma^2} - \frac{\ln 2\sigma^2 + \ln \pi}{2}\right] \\
&= \frac{\exp\left[-\eta_1 x^2 + \eta_2 x\right]}{Z(\eta_1, \eta_2)},
\end{aligned}$$

where:

$$\begin{aligned}
\eta_1 &= \frac{1}{2\sigma^2} \\
\eta_2 &= \frac{\mu}{\sigma^2} \\
Z(\eta_1, \eta_2) &= \exp\left[\frac{\eta_2^2}{4\eta_1} - \frac{\ln \eta_1 - \ln \pi}{2}\right].
\end{aligned}$$

Second, by combining this result with Proposition 7, we conclude:

$$\begin{aligned}
\mathrm{E}_{P(x)}[f(x)] &= \frac{Z(\eta_P + \eta_f)}{Z(\eta_P)Z(\eta_f)} \\
&= \frac{\exp\left[\frac{(\eta_{P_2}+\eta_{f_2})^2}{4(\eta_{P_1}+\eta_{f_1})} - \frac{\ln(\eta_{P_1}+\eta_{f_1}) - \ln \pi}{2}\right]}{\exp\left[\frac{\eta_{P_2}^2}{4\eta_{P_1}} - \frac{\ln \eta_{P_1} - \ln \pi}{2}\right] \exp\left[\frac{\eta_{f_2}^2}{4\eta_{f_1}} - \frac{\ln \eta_{f_1} - \ln \pi}{2}\right]} \\
&= \exp\left[\frac{\left(\frac{\mu_P \sigma_f^2 + \mu_f \sigma_P^2}{\sigma_P^2 \sigma_f^2}\right)^2}{2\frac{\sigma_P^2 + \sigma_f^2}{\sigma_P^2 \sigma_f^2}} - \frac{\mu_P^2}{2\sigma_P^2} - \frac{\mu_f^2}{2\sigma_f^2} - \frac{\ln \frac{\sigma_P^2+\sigma_f^2}{2\sigma_P^2\sigma_f^2} - \ln \frac{1}{2\sigma_P^2} - \ln \frac{1}{2\sigma_f^2} + \ln \pi}{2}\right] \\
&= \exp\left[\frac{(\mu_P \sigma_f^2 + \mu_f \sigma_P^2)^2}{2\sigma_P^2 \sigma_f^2 (\sigma_P^2 + \sigma_f^2)} - \frac{\mu_P^2}{2\sigma_P^2} - \frac{\mu_f^2}{2\sigma_f^2} - \frac{\ln 2(\sigma_P^2 + \sigma_f^2) + \ln \pi}{2}\right],
\end{aligned}$$

where $\eta_P = (\eta_{P_1}, \eta_{P_2})$ and $\eta_f = (\eta_{f_1}, \eta_{f_2})$ are the natural parameters of the distributions $P(x)$ and $f(x)$. ■

Proof of Corollary 4: The proposition is proved in two steps. First, note that the gamma distribution can be written in the canonical form as:

$$\begin{aligned} P_{\text{gamma}}(x \mid \alpha, \beta) &= \frac{1}{\Gamma(\alpha)\beta^{\alpha}} x^{\alpha-1} \exp\left[-\frac{1}{\beta}x\right] \\ &= \exp\left[(\alpha-1)\ln x - \frac{1}{\beta}x\right] \exp[-\ln\Gamma(\alpha) - \alpha\ln\beta] \\ &= \frac{\exp\left[\eta_1 \ln x - \eta_2 x\right]}{Z(\eta_1, \eta_2)}, \end{aligned}$$

where:

$$\begin{aligned} \eta_1 &= \alpha - 1 \\ \eta_2 &= \frac{1}{\beta} \\ Z(\eta_1, \eta_2) &= \exp[\ln\Gamma(\eta_1+1) - (\eta_1+1)\ln\eta_2]. \end{aligned}$$

Second, by combining this result with Proposition 7, we conclude:

$$\begin{aligned} \mathrm{E}_{P(x)}[f(x)] &= \frac{Z(\eta_P + \eta_f)}{Z(\eta_P)Z(\eta_f)} \\ &= \frac{\exp[\ln\Gamma(\eta_{P_1}+\eta_{f_1}+1) - (\eta_{P_1}+\eta_{f_1}+1)\ln(\eta_{P_2}+\eta_{f_2})]}{\exp[\ln\Gamma(\eta_{P_1}+1) - (\eta_{P_1}+1)\ln\eta_{P_2}]\exp[\ln\Gamma(\eta_{f_1}+1) - (\eta_{f_1}+1)\ln\eta_{f_2}]} \\ &= \frac{\eta_{P_2}^{\eta_{P_1}+1}}{\Gamma(\eta_{P_1}+1)} \frac{\eta_{f_2}^{\eta_{f_1}+1}}{\Gamma(\eta_{f_1}+1)} \frac{\Gamma(\eta_{P_1}+\eta_{f_1}+1)}{(\eta_{P_2}+\eta_{f_2})^{\eta_{P_1}+\eta_{f_1}+1}} \\ &= \frac{1}{\Gamma(\alpha_P)\beta_P^{\alpha_P}} \frac{1}{\Gamma(\alpha_f)\beta_f^{\alpha_f}} \Gamma(\alpha_P+\alpha_f-1) \left(\frac{\beta_P\beta_f}{\beta_P+\beta_f}\right)^{\alpha_P+\alpha_f-1}, \end{aligned}$$

where $\eta_P = (\eta_{P_1}, \eta_{P_2})$ and $\eta_f = (\eta_{f_1}, \eta_{f_2})$ are the natural parameters of the distributions $P(x)$ and $f(x)$. ■

Proof of Proposition 8: This claim is proved in two steps. First, we find a point $\overline{\mathbf{w}}$ in the feasible region of the HALP such that $V^{\overline{\mathbf{w}}}$ is within $O(\delta)$ distance from $V^{\widehat{\mathbf{w}}}$. Such a point $\overline{\mathbf{w}}$ is given by:

$$\overline{\mathbf{w}} = \widehat{\mathbf{w}} + \frac{\delta}{1-\gamma} e,$$

where $e = (1, 0, \ldots, 0)$ is an indicator of the constant basis function $f_0(\mathbf{x}) \equiv 1$. This point satisfies all requirements and its feasibility can be handily verified by solving:

$$\begin{aligned} V^{\overline{\mathbf{w}}} - \mathcal{T}^* V^{\overline{\mathbf{w}}} &= V^{\widehat{\mathbf{w}}} + \frac{\delta}{1-\gamma} - \left(\mathcal{T}^* V^{\widehat{\mathbf{w}}} + \frac{\gamma\delta}{1-\gamma} \right) \\ &= V^{\widehat{\mathbf{w}}} - \mathcal{T}^* V^{\widehat{\mathbf{w}}} + \delta \\ &\geq 0, \end{aligned}$$

where the inequality $V^{\widehat{\mathbf{w}}} - \mathcal{T}^* V^{\widehat{\mathbf{w}}} \geq -\delta$ holds from the δ-infeasibility of $\widehat{\mathbf{w}}$. Since the solution $\overline{\mathbf{w}}$ is feasible in the HALP, we conclude $\mathrm{E}_\psi\left[V^{\tilde{\mathbf{w}}}\right] \leq \mathrm{E}_\psi\left[V^{\overline{\mathbf{w}}}\right]$. Subsequently, this observation leads to our final result:

$$\begin{aligned} \mathrm{E}_\psi\left[V^{\tilde{\mathbf{w}}}\right] &\leq \mathrm{E}_\psi\left[V^{\overline{\mathbf{w}}}\right] \\ &= \mathrm{E}_\psi\left[V^{\widehat{\mathbf{w}}} + \frac{\delta}{1-\gamma}\right] \\ &= \mathrm{E}_\psi\left[V^{\widehat{\mathbf{w}}}\right] + \frac{\delta}{1-\gamma}. \end{aligned}$$

■

Proof of Proposition 9: The proposition is proved in a sequence of steps:

$$\begin{aligned}
\frac{\partial f(x)}{\partial \eta_{f_k}} &= \frac{\partial}{\partial \eta_{f_k}} \left(\frac{h(x) \exp[\eta_f^{\intercal} t(x)]}{Z(\eta_f)} \right) \\
&= h(x) \left[\frac{1}{Z(\eta_f)} \frac{\partial \exp[\eta_f^{\intercal} t(x)]}{\partial \eta_{f_k}} + \exp[\eta_f^{\intercal} t(x)] \frac{\partial}{\partial \eta_{f_k}} \left(\frac{1}{Z(\eta_f)} \right) \right] \\
&= h(x) \left[\frac{t_k(x) \exp[\eta_f^{\intercal} t(x)]}{Z(\eta_f)} - \frac{\exp[\eta_f^{\intercal} t(x)]}{Z(\eta_f)^2} \frac{\partial Z(\eta_f)}{\partial \eta_{f_k}} \right] \\
&= f(x) \left[t_k(x) - \frac{1}{Z(\eta_f)} \frac{\partial Z(\eta_f)}{\partial \eta_{f_k}} \right],
\end{aligned}$$

which follow from fundamental differentiation laws. ■

Proof of Proposition 10: Based on Proposition 7, we conclude:

$$\mathrm{E}_{P(x)}[f(x)] = \frac{Z(\eta_P + \eta_f)}{Z(\eta_P) Z(\eta_f)}.$$

Subsequently, our claim is proved in a sequence of steps:

$$\begin{aligned}
\frac{\partial f(x)}{\partial \eta_{f_k}} &= \frac{\partial}{\partial \eta_{f_k}} \left(\frac{Z(\eta_P + \eta_f)}{Z(\eta_P) Z(\eta_f)} \right) \\
&= \frac{1}{Z(\eta_P) Z(\eta_f)} \frac{\partial Z(\eta_P + \eta_f)}{\partial \eta_{f_k}} + \frac{Z(\eta_P + \eta_f)}{Z(\eta_P)} \frac{\partial}{\partial \eta_{f_k}} \left(\frac{1}{Z(\eta_f)} \right) \\
&= \frac{1}{Z(\eta_P) Z(\eta_f)} \frac{\partial Z(\eta_P + \eta_f)}{\partial \eta_{f_k}} - \frac{Z(\eta_P + \eta_f)}{Z(\eta_P) Z(\eta_f)^2} \frac{\partial Z(\eta_f)}{\partial \eta_{f_k}},
\end{aligned}$$

which follow from fundamental differentiation laws. ■

BIBLIOGRAPHY

[1] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50:5–43, 2003.

[2] Karl Astrom. Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174–205, 1965.

[3] Andrew Barto, Richard Sutton, and Charles Anderson. Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5):835–846, 1983.

[4] Jonathan Baxter and Peter Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

[5] Jonathan Baxter, Peter Bartlett, and Lex Weaver. Experiments with infinite-horizon, policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:351–381, 2001.

[6] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[7] Richard Bellman, Robert Kalaba, and Bella Kotkin. Polynomial approximation – a new computational technique in dynamic programming: Allocation processes. *Mathematics of Computation*, 17(82):155–161, 1963.

[8] Dimitri Bertsekas. A counterexample for temporal differences learning. *Neural Computation*, 7(2):270–279, 1995.

[9] Dimitri Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, 1995.

[10] Dimitri Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[11] Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, Belmont, MA, 1997.

[12] Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[13] Craig Boutilier and Richard Dearden. Approximating value trees in structured dynamic programming. In *Proceedings of the 13th International Conference on Machine Learning*, pages 54–62, 1996.

[14] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Exploiting structure in policy construction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1104–1111, 1995.

[15] Justin Boyan and Michael Littman. Exact solutions to time-dependent MDPs. In *Advances in Neural Information Processing Systems 13*, pages 1026–1032, 2001.

[16] Justin Boyan and Andrew Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems 7*, pages 369–376, 1995.

[17] John Bresina, Richard Dearden, Nicolas Meuleau, Sailesh Ramakrishnan, David Smith, and Rich Washington. Planning under continuous time and resource uncertainty: A challenge for AI. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 77–84, 2002.

[18] George Casella and Christian Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

[19] Chee-Seng Chow and John Tsitsiklis. An optimal one-way multigrid algorithm for discrete-time stochastic control. *IEEE Transactions on Automatic Control*, 36(8):898–914, 1991.

[20] Gregory Cooper. A method for using belief networks as influence diagrams. In *Proceedings of the Workshop on Uncertainty in Artificial Intelligence*, pages 55–63, 1988.

[21] Robert Crites and Andrew Barto. Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems 8*, pages 1017–1023, 1996.

[22] Peter Dayan and Terry Sejnowski. TD(λ) converges with probability 1. *Machine Learning*, 14:295–301, 1994.

[23] Daniela Pucci de Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–856, 2003.

[24] Daniela Pucci de Farias and Benjamin Van Roy. On constraint sampling for the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3):462–478, 2004.

[25] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150, 1989.

[26] Rina Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 211–219, 1996.

[27] Simon Duane, A. D. Kennedy, Brian Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

[28] Zhengzhu Feng, Richard Dearden, Nicolas Meuleau, and Richard Washington. Dynamic programming for structured continuous Markov decision problems. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 154–161, 2004.

[29] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for Markov decision processes with infinite state spaces. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, 2005.

[30] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[31] Peter Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, 1990.

[32] Geoffrey Gordon. *Approximate Solutions to Markov Decision Processes*. PhD thesis, Carnegie Mellon University, 1999.

[33] Gregory Grudic and Lyle Ungar. Localizing policy gradient estimates to action transitions. In *Proceedings of 17th International Conference on Machine Learning*, pages 343–350, 2000.

[34] Carlos Guestrin. *Planning Under Uncertainty in Complex Structured Environments*. PhD thesis, Stanford University, 2003.

[35] Carlos Guestrin, Milos Hauskrecht, and Branislav Kveton. Solving factored MDPs with continuous and discrete variables. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 235–242, 2004.

[36] Carlos Guestrin, Daphne Koller, Chris Gearhart, and Neal Kanodia. Generalizing plans to new environments in relational MDPs. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 1003–1010, 2003.

[37] Carlos Guestrin, Daphne Koller, and Ronald Parr. Max-norm projections for factored MDPs. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 673–682, 2001.

[38] Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems 14*, pages 1523–1530, 2002.

[39] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

[40] Carlos Guestrin, Shobha Venkataraman, and Daphne Koller. Context specific multiagent coordination and planning with factored MDPs. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 253–259, 2002.

[41] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their application. *Biometrika*, 57:97–109, 1970.

[42] Milos Hauskrecht. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.

[43] Milos Hauskrecht and Branislav Kveton. Linear program approximations for factored continuous-state Markov decision processes. In *Advances in Neural Information Processing Systems 16*, pages 895–902, 2004.

[44] David Higdon. Auxiliary variable methods for Markov chain Monte Carlo with applications. *Journal of the American Statistical Association*, 93(442):585–595, 1998.

[45] Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutilier. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 279–288, 1999.

[46] Ronald Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

[47] Ronald Howard and James Matheson. Influence diagrams. In *Readings on the Principles and Applications of Decision Analysis*, volume 2, pages 719–762. Strategic Decisions Group, Menlo Park, CA, 1984.

[48] Harold Jeffreys and Bertha Jeffreys. *Methods of Mathematical Physics*. Cambridge University Press, Cambridge, United Kingdom, 1988.

[49] Frank Jensen, Finn Jensen, and Søren Dittmer. From influence diagrams to junction trees. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 367–373, 1994.

[50] William Jewell. Markov-renewal programming. I: Formulation, finite return models. *Operations Research*, 11(6):938–948, 1963.

[51] William Jewell. Markov-renewal programming. II: Infinite return models, example. *Operations Research*, 11(6):949–971, 1963.

[52] Leonid Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979.

[53] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[54] Daphne Koller and Ronald Parr. Computing factored value functions for policies in structured MDPs. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 1332–1339, 1999.

[55] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems 12*, pages 1008–1014, 2000.

[56] Branislav Kveton and Milos Hauskrecht. Heuristic refinements of approximate linear programming for factored continuous-state Markov decision processes. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling*, pages 306–314, 2004.

[57] Branislav Kveton and Milos Hauskrecht. An MCMC approach to solving hybrid factored MDPs. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1346–1351, 2005.

[58] Branislav Kveton and Milos Hauskrecht. Learning basis functions in hybrid domains. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1161–1166, 2006.

[59] Branislav Kveton and Milos Hauskrecht. Solving factored MDPs with exponential-family transition models. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling*, pages 114–120, 2006.

[60] Branislav Kveton, Milos Hauskrecht, and Carlos Guestrin. Solving factored MDPs with hybrid state and action variables. *Journal of Artificial Intelligence Research*, 27:153–201, 2006.

[61] Michail Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[62] Lihong Li and Michael Littman. Lazy approximation for solving continuous finite-horizon MDPs. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 1175–1180, 2005.

[63] Michael Littman. *Algorithms for Sequential Decision Making*. PhD thesis, Brown University, 1996.

[64] Sridhar Mahadevan. Samuel meets Amarel: Automating value function approximation using global state space analysis. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 1000–1005, 2005.

[65] Sridhar Mahadevan and Mauro Maggioni. Value function approximation with diffusion wavelets and Laplacian eigenfunctions. In *Advances in Neural Information Processing Systems 18*, pages 843–850, 2006.

[66] Sridhar Mahadevan, Mauro Maggioni, Kimberly Ferguson, and Sarah Osentoski. Learning representation and control in continuous Markov decision processes. In *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.

[67] Alan Manne. Linear programming and sequential decisions. *Management Science*, 6(3):259–267, 1960.

[68] Peter Marbach and John Tsitsiklis. Simulation-based optimization of Markov reward processes. *IEEE Transactions on Automatic Control*, 46(2):191–209, 2001.

[69] Janusz Marecki, Sven Koenig, and Milind Tambe. A fast analytical algorithm for solving Markov decision processes with continuous resources. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007.

[70] Nicholas Metropolis, Arianna Rosenbluth, Marshall Rosenbluth, Augusta Teller, and Edward Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[71] Nicolas Meuleau, Leonid Peshkin, and Kee-Eung Kim. Exploration in gradient-based reinforcement learning. Technical Report 1713 (AI Memo 2001-003), Massachusetts Institute of Technology, 2001.

[72] Andrew Moore. Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In *Proceedings of the 8th International Conference on Machine Learning*, 1991.

[73] Remi Munos and Andrew Moore. Variable resolution discretization for high-accuracy solutions of optimal control problems. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 1348–1355, 1999.

[74] Remi Munos and Andrew Moore. Variable resolution discretization in optimal control. *Machine Learning*, 49:291–323, 2002.

[75] Andrew Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Inverted autonomous helicopter flight via reinforcement learning. In *International Symposium on Experimental Robotics*, 2004.

[76] Andrew Ng and Michael Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 406–415, 2000.

[77] Uri Nodelman, Christian Shelton, and Daphne Koller. Continuous time Bayesian networks. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 378–387, 2002.

[78] Luis Ortiz. *Selecting Approximately-Optimal Actions in Complex Structured Domains*. PhD thesis, Brown University, 2002.

[79] James Park and Adnan Darwiche. Approximating MAP using local search. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, pages 403–410, 2001.

[80] James Park and Adnan Darwiche. Solving MAP exactly using systematic search. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, pages 459–468, 2003.

[81] Relu Patrascu, Pascal Poupart, Dale Schuurmans, Craig Boutilier, and Carlos Guestrin. Greedy linear value-approximation for factored Markov decision processes. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 285–291, 2002.

[82] Martin Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York, NY, 1994.

[83] John Rust. Using randomization to break the curse of dimensionality. *Econometrica*, 65(3):487–516, 1997.

[84] Scott Sanner and Craig Boutilier. Approximate linear programming for first-order MDPs. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, 2005.

[85] Dale Schuurmans and Relu Patrascu. Direct value-approximation for factored MDPs. In *Advances in Neural Information Processing Systems 14*, pages 1579–1586, 2002.

[86] Paul Schweitzer and Abraham Seidmann. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110:568–582, 1985.

[87] Christian Shelton. Policy improvement for POMDPs using normalized importance sampling. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, pages 496–503, 2001.

[88] Edward Sondik. *The Optimal Control of Partially Observable Markov Decision Processes*. PhD thesis, Stanford University, 1971.

[89] Richard Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[90] Richard Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, pages 1038–1044, 1996.

[91] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[92] Richard Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, pages 1057–1063, 2000.

[93] Gerald Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8(3-4):257–277, 1992.

[94] Gerald Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

[95] Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[96] Michael Trick and Stanley Zin. A linear programming approach to solving stochastic dynamic programs. Technical report, Carnegie Mellon University, 1993.

[97] John Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.

[98] Benjamin Van Roy. *Planning Under Uncertainty in Complex Structured Environments*. PhD thesis, Massachusetts Institute of Technology, 1998.

[99] Ronald Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[100] Ronald Williams and Leemon Baird III. Tight performance bounds on greedy policies based on imperfect value functions. Technical Report NU-CCS-93-14, Northeastern University, 1993.

[101] Hakan Younes and Reid Simmons. Solving generalized semi-Markov decision processes using continuous phase-type distributions. In *Proceedings of the 19th National Conference on Artificial Intelligence*, pages 742–747, 2004.

[102] Changhe Yuan, Tsai-Ching Lu, and Marek Druzdzel. Annealed MAP. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 628–635, 2004.

[103] Wei Zhang and Thomas Dietterich. A reinforcement learning approach to job-shop scheduling. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1114–1120, 1995.

[104] Wei Zhang and Thomas Dietterich. High-performance job-shop scheduling with a time-delay TD(λ) network. In *Advances in Neural Information Processing Systems 8*, pages 1024–1030, 1996.