Linear Dynamic Programs for Resource Management

Marek Petrik
IBM T.J. Watson Research Center
1101 Kitchawan Road, Route 134
Yorktown Heights, NY 10598, USA
mpetrik@us.ibm.com

Shlomo Zilberstein
Department of Computer Science
University of Massachusetts
Amherst, MA 01003, USA
shlomo@cs.umass.edu

Abstract

Sustainable resource management in many domains presents large continuous stochastic optimization problems, which can often be modeled as Markov decision processes (MDPs). To solve such large MDPs, we identify and leverage linearity in state and action sets that is common in resource management. In particular, we introduce *linear dynamic programs* (LDPs) that generalize resource management problems and partially observable MDPs (POMDPs). We show that the LDP framework makes it possible to adapt point-based methods–the state of the art in solving POMDPs–to solving LDPs. The experimental results demonstrate the efficiency of this approach in managing the water level of a river reservoir. Finally, we discuss the relationship with dual dynamic programming, a method used to optimize hydroelectric systems.

1 Introduction

Sequential decision making under uncertainty is ubiquitous in many fields, including planning in artificial intelligence and inventory management in operations research. A convenient framework for modeling these problems is the Markov decision process (MDP), which assumes that future states are independent of the history of the process *given* its current state. While small MDPs are easy to solve, most resource management domains involve MDPs with an infinite number of states and actions. We present and analyze novel point-based methods for solving such problems.

Consider, for example, the problem of managing water discharge from a river reservoir to produce hydroelectric power. Among renewable sources, hydroelectricity has the lowest carbon footprint per MWh and is the most significant source of renewable energy, supplying about 20% of the world electricity according to the U.S. Geological Survey. Increasing the efficiency of reservoir management can therefore significantly reduce carbon emissions.

Hydroelectric power is usually produced by discharging water from a reservoir to activate a turbine. The operator of the reservoir must decide how much water to discharge at each time in order to maximize energy production, while satisfying important environmental constraints such as irrigation requirements and flood prevention. This sequential decision problem results in a very large MDP–the state and action spaces are both continuous. The state space, in a simple model, may represent the water level in the reservoir, while actions correspond to the amount of water discharge. A richer model could include additional factors such as the weather or the demand for electricity, but even a basic model cannot be solved directly using standard algorithms such as value or policy iteration.

Large MDPs are typically solved using reinforcement learning methods such as approximate dynamic programming [Powell, 2007; Petrik and Zilberstein, 2009]. These methods are very general and can be applied to problems with an infinite number of states and actions. However, they require good approximation features and state samples to achieve good performance. Designing such features is non-trivial and the guarantees provided by these methods are often quite weak. Furthermore, approximate dynamic programming often cannot take advantage of additional structure present in the problem domain.

To address these deficiencies, we propose a new approach that–unlike standard reinforcement learning techniques–takes advantage of the structure in the domain. Specifically, actions and transitions often influence the state space linearly (e.g. water inflow and discharge change the reservoir volume linearly) and can be modeled as a *linearly controlled problem* (LCP) [Woerner, 2010]–a type of a linear stochastic program. We show that this property restricts the value function, which is the value of being in each state, to be piecewise linear and convex. Such value functions can than be approximated efficiently using methods for solving POMDPs.

Linearly controlled problems are apparently very different from POMDPs. The properties of POMDPs that make them amenable to point-based dynamic programming are: 1) the number of actions is small, 2) each action can be taken in all *belief* states, and 3) the value function update for each action is linear. In contrast, linearly controlled problems have a continuous action space and some of the actions cannot be executed in certain states. For example, the amount of water discharged is limited by the demand for electricity and the available water. Nevertheless, we show that the Fenchel dual of a linearly controlled problem closely resembles a POMDP and can be solved using similar methods.

The remainder of the paper is organized as follows. First, we describe the MDP model and some basic notation in Section 2. Section 3 introduces linear dynamic programs, a special case of dynamic programs. Next, we describe partially observable Markov decision processes in Section 4 and linearly controlled problems in Section 5, showing how to formulate them as linear dynamic programs. Section 6 describes a class of point-based solvers for linear dynamic programs, which are inspired by point-based methods for solving POMDPs. We describe an application of these point-based solvers to reservoir management in Section 7, and discuss related work from the hydroelectricity management literature in Section 8.

2 Notation and Framework

This section describes the basic concepts and notation that we use. A *Markov Decision Process* (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, W, P, r, \alpha)$. Here, $\mathcal{S}$ is a *compact* set of states, $\mathcal{A}$ is a *compact* set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the transition function ($P(s, a, s')$ is the probability of transiting to state s' from state s given action a), and $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}_+$ is a *continuous* reward function. The set-valued function $W : \mathcal{S} \rightrightarrows \mathcal{A}$ maps each state to the set of allowed actions. The initial distribution is: $\alpha : \mathcal{S} \mapsto [0, 1]$, such that $\sum_{s\in\mathcal{S}} \alpha(s) = 1$. The set of time steps is denoted as $\mathcal{T} = \{0 \dots T - 1\}$.

A *decision rule* $d : \mathcal{S} \to \mathcal{A}$ determines the action to be taken in each possible state of the MDP. The set of all decision rules is $\mathcal{D}$. A collection of decision rules, one per time step, is a deterministic Markov *policy* $\pi : \mathcal{T} \times \mathcal{S} \mapsto \mathcal{A}$, which assigns an action to each state of the MDP for every time step. The set of all deterministic policies is denoted as Π.

The rewards and transitions for a decision rule d are defined as follows: $r_d(s) = r(s, d(s))$ and $P_d(s, s') = P(s, d(s), s')$. The focus of this paper is on the *finite* horizon γ-discounted expected sum of rewards–called the *return*–over T steps. The policy π is a collection of decision rules such that $\pi(t) = d_t$. Let S_t be a random variable that represents the state at time t, distributed according to the probability distributions $u_\pi(t) : \mathcal{S} \to [0, 1]$. The return of a policy π is then defined as follows:

$$\rho(\pi) = \mathbf{E}\left[\sum_{t\in\mathcal{T}} \gamma^t r(S_t, d_t(S_t))\right].$$

The distributions $u_\pi(t)$ are unique and depend on the policy as follows: $u_\pi(t + 1) = P_{d_t}^{\mathsf{T}} u_\pi(t)$ and $u_\pi(0) = \alpha$. A policy π^* is optimal if it maximizes the cumulative return ρ.

The finite-horizon value function $v : \mathcal{T} \times \mathcal{S} \to \mathbb{R}$ for a policy π represents the return obtained when starting in a given state and is formally defined for $\bar{t} \in \mathcal{T}$ as follows:

$$v_\pi(\bar{t}, s) = \mathbf{E}\left[\sum_{t=\bar{t}\dots T} \gamma^{t-\bar{t}} r(S_t, d_t(S_t))\right],$$

where $u_\pi(\bar{t}, s) = 1$ and $v(T, s) = \mathbf{0}$. We use $v(t)$ to denote a vector of values for each state. The return for a policy π with a value function v_π is $\mathbf{E}_\alpha[v_\pi(0)]$. The optimal value function v^* is the value function of the optimal policy π^*. We use $\mathcal{V} = \mathbb{R}^{\mathcal{S}}$ to denote the set of possible value functions for any given time step.

The action-value function $q_\pi : \mathcal{T} \times \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ for a policy π, also known as Q-function, represents the value function in a state after taking action a. This function is defined similarly to the value function:

$$q_\pi(\bar{t}, s, a) = \mathbf{E}\left[\sum_{t=\bar{t}+1\dots T} \gamma^{t-\bar{t}-1} r(S_t, d_t(S_t))\right],$$

where $u_\pi(\bar{t}, s) = 1$ and $d_{\bar{t}} = a$. An optimal action-value function q^* is the action-value function of an optimal policy.

An important property of Markov decision processes is that an optimal value function must satisfy the Bellman optimality condition, which is defined as follows.

Theorem 1 (Bellman optimality condition [Bellman, 1957])**.** *A finite-horizon value function $v^*(t)$ is optimal if and only if it satisfies for all $s \in \mathcal{S}$ and $t \in \mathcal{T}$ the following equality:*

$$\begin{aligned} v^*(t, s) &= \max_{a\in\mathcal{A}} \left(r(s, a) + \gamma \sum_{s'\in\mathcal{S}} P(s, a, s') v^*(t + 1, s') \right) \\ &= \max_{a\in\mathcal{A}} \left(r(s, a) + \gamma q^*(t, s, a) \right) \end{aligned}$$

In addition, when $q^*(t, s, a)$ is continuous in a for every s and t, there exists an optimal Markov deterministic policy.

Minor technical assumptions may be needed to ensure the existence of an optimal Markov policy due to the continuous action spaces; please refer to Section 4.3 and 4.4 of [Puterman, 2005] for more details. In general, a policy satisfying the Bellman optimality condition is hard to compute for continuous MDPs. In fact, when the action space is infinite, solving the optimization problem $\max_{a\in\mathcal{A}} r(s, a) + \gamma q^*(t, s, a)$ may be nontrivial. Approximate dynamic programming overcomes this problem by restricting value functions to a small linear space [Petrik and Zilberstein, 2009]. We take an alternative approach that exploits the linear structure of the state and action spaces.

3 Linear Dynamic Programs

In this section, we define *linear dynamic programs* (LDPs) and the main properties of their value functions. LDPs serve to model the Bellman optimality equations; they can be used to compute solutions, but are impractical for simulation. As we show in the remainder of the paper, LDPs bridge resource management problems and POMDPs and make it possible to leverage point-based solvers.

The optimality equations for a linear dynamic program are defined as follows:

$$v(t, s) = \max_{a\in\mathcal{A}} \left(r(s, a) + \gamma q(t, s, a) \right) \tag{1}$$

$$q(t, s, a) = Z(v(t + 1), s, a), \tag{2}$$

where $v(T) = \mathbf{0}$ and $Z : \mathcal{V} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a value function transition functional. The function Z represents the change in the value function due to the stochastic transitions of the dynamic program. Its precise form depends on the specific problem; we define the particular structure for POMDPs and LCPs later in Sections 4 and 5.

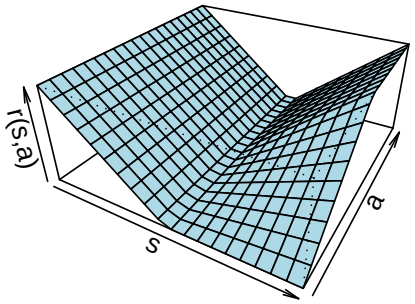

Figure 1: Example of a valid reward function in an LDP.

Definition 1 (Linear Dynamic Program)**.** *Equations* (1) *and* (2) *are called a* linear dynamic program *if they satisfy the following properties:*

1. *States and actions are real vectors:* $\mathcal{S} \subseteq \mathbb{R}^n$, $\mathcal{A} \subseteq \mathbb{R}^m$.
2. *The set* $\mathcal{A}$ *is a convex polyhedral set. Note that this set is* independent *of the state.*
3. *The reward function* r *is piecewise linear concave in* a *for any* s*, and is piecewise linear convex in* s *for any* a*. Moreover,* r *is in particular of the following form:*

$$r(s,a) = \max_{c \in \mathbb{R}^{n'}} \min_{d \in \mathbb{R}^{m'}} u(s,a,c,d)$$

$$u(s,a,c,d) = c^\mathsf{T}(Cs+q_1) + d^\mathsf{T}(Da+q_2) + c^\mathsf{T}Qd + q,$$

where $n' \geq n$*,* $m' \geq m$*;* C*,* D*,* Q *are given matrices;* q_1*,* q_2*,* q_3 *are given vectors; and* q *is a given scalar. We also assume that* $r(s,a) < M$ *for some* M*.*

4. *Let* $f(s) = \max_{i \in \mathcal{I}} \alpha_i^\mathsf{T} s + \beta_i$ *be a piecewise linear function. Then, the function* Z *satisfies the following properties:*
   (a) *For any* $a' \in \mathcal{A}$ *there exists a set* $\mathcal{J}$ *such that:*

$$Z(f,s,a') = \max_{j \in \mathcal{J}} \alpha_j^\mathsf{T} s + \beta_j.$$

   (b) *For any fixed* $s' \in \mathcal{S}$ *there exists a set* $\mathcal{K}$ *such that:*

$$Z(f,s',a) = \min_{k \in \mathcal{K}} \alpha_k^\mathsf{T} a + \beta_k.$$

Conditions 1, 2, 3, 4(b) in Definition 1 ensure that the optimal greedy action a in Equation (1) can be computed by linear programming. In particular, computing the greedy action involves maximizing a piecewise linear concave function over a polytope. Condition 4(a) ensures that the functional Z maps piecewise linear value functions to piecewise linear functions. Note that Conditions 3 and 4 require convexity in the state space but concavity in the action space. Finally, the condition $r(s,a) < M$ ensures that value functions do not become infinitely large.

To understand the formulation of Condition 3, notice that the function $r(s,a)$ is a general representation of the Lagrangian of a linear program [Vanderbei, 2001]. In particular, consider $\mathcal{S} = \mathcal{A} = [-1,1]$ and assume that the reward function $r(s,a)$ is represented by the following linear program:

$$\begin{array}{ll} \max\limits_{c} & c * s \\ \text{s.t.} & -1 \leq c \leq 1 \\ & c \leq a + 1 \end{array} \tag{3}$$

The Lagrangian of this linear program is: $\max_c \min_{d \geq \mathbf{0}} cs - d_1 c + d_1 + d_2 c + d_2 + -d_3 c - d_3 a - d_3$, where $d \in \mathbb{R}^3$; Figure 1 shows the plot of this function. The formulation in Condition 3 also captures simpler functions, such as $\max_{c \in \mathcal{C}} \min_{d \in \mathcal{D}} c^\mathsf{T} s + d^\mathsf{T} a$, where $\mathcal{C}$ and $\mathcal{D}$ are finite sets.

To simplify the notation, we assume that any piecewise linear function can be represented by some *finite* set of indices $i \in \mathcal{I}$, vectors α_i, and constants β_i. That means that the set $\mathcal{I}$ is a subset of a universal set of indices.

The reason for treating the value function v and action-value function q separately in Equations (1) and (2) is that the approximation errors in each of the equations have different sources and can be treated independently.

Linear dynamic programs are important because the optimal value function at any step is piecewise linear. We will later use this piecewise linearity to develop an approximation. The following proposition summarizes the basic properties of linear dynamic programs.

Proposition 1. *Given a linear dynamic program, the optimal value function* $v^*(t)$ *is piecewise linear convex; that is, for some* finite *set* $\mathcal{I}$*:*

$$v^*(t,s) = \max_{i \in \mathcal{I}} \alpha_i^\mathsf{T} s + \beta_i.$$

In addition, the action value function for any state $s \in \mathcal{S}$ *is piecewise linear concave; that is, for some* finite *set* $\mathcal{J}$*:*

$$q^*(t,s,a) = \min_{j \in \mathcal{J}} \alpha_j^\mathsf{T} a + \beta_j,$$

and the optimization problem (1) *reduces to a linear program.*

Proof. We prove the proposition by backward induction on the time step. The proposition follows for time T directly from the definition since $q(T,s,a) = v(T,s) = 0$. Now, assume that $v(t+1) = \max_{i \in \mathcal{I}} \alpha_i^\mathsf{T} s + \beta_i$. From 4(a) in Definition 1, we get: $q^*(t,s',a) = \min_{k \in \mathcal{K}} \alpha_k^\mathsf{T} a + \beta_k$ for a fixed state s'. The greedy action optimization $\max_{a \in \mathcal{A}} r(s,a) + q(t,s,a)$ can be represented as the following linear program:

$$\begin{array}{ll} \max\limits_{a \in \mathcal{A}, d \in \mathbb{R}^{m'}} & \gamma y + \min\limits_{d \in \mathbb{R}^{m'}} u(s,a,c,d) \\ \text{s.t.} & y \leq \alpha_k^\mathsf{T} a + \beta_k \qquad k \in \mathcal{K} \end{array} \tag{4}$$

Since d is minimized over a cone, it can be easily translated to a set of linear constraints, since the function represents a Lagrangian [Boyd and Vandenberghe, 2004]. Following standard linear programming theory [Vanderbei, 2001], we get:

$$v(t,s) = \max_{a \in \mathcal{A}} r(s,a) + \gamma q(t,s,a) = \max_{\bar{a} \in \operatorname{ext} \mathcal{A}} r(s,\bar{a}) + \gamma q(t,s,\bar{a}).$$

Now, let $o(\bar{a})$ denote the objective value for the basic feasible solution. Here, $\operatorname{ext} \mathcal{A}$ is a shorthand for the set of

basic feasible solutions of linear program (4); there is always a finite number of them and the optimal solution is bounded above since $r(s,a) < M$. Then from 4(b): $v(t,s) = \max_{\bar{a} \in \text{ext } \mathcal{A}} \max_{c \in \mathcal{C}} c^{\mathsf{T}} s + \gamma o(\bar{a})$, which implies that the value function is piecewise linear since it is a maximum over a *finite* set of linear functions. □

4 Modeling POMDPs Using LDPs

In this section, we show that partially observable Markov decision processes (POMDPs) can be modeled as linear dynamic programs. We use this connection to adapt POMDP solvers to linear dynamic programs.

Partially observable Markov decision processes (POMDPs) are an extension of MDPs. Like an MDP, a POMDP has an underlying state, but this state cannot be directly observed. Formally, a POMDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, P, O, r, \alpha)$, where states $\mathcal{S}$, actions $\mathcal{A}$, and rewards r are defined identically as for MDPs. In addition, the set of observations is $\mathcal{O}$ with the observation probabilities $O : \mathcal{S} \times \mathcal{O} \to [0,1]$. The transition function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ represents the state transition probabilities.

It is well known that a POMDP can be recast as a so called *belief-state* MDP (e.g. [Hauskrecht, 2000]), with the following states and actions: $(\bar{\mathcal{S}}, \mathcal{A}, \bar{P}, \bar{r}, \bar{\alpha})$. The states

$$\bar{\mathcal{S}} = \left\{ b \in [0,1]^{|\mathcal{S}|} \;\middle|\; \sum_{s \in \mathcal{S}} b(s) = 1 \right\}$$

represent probability distributions over the POMDP states. The rewards and transitions are defined for $b \in \bar{\mathcal{S}}$ as follows:

$$\bar{P}(b, a, b'_o) = \sum_{s \in \mathcal{S}} O(s,o) b(s) \text{ for each } o \in \mathcal{O}, \text{ where}$$

$$b'_o(s') = \frac{\sum_{s \in \mathcal{S}} P(s,a,s') O(s,o) b(s)}{\sum_{s \in \mathcal{S}} O(s,o) b(s)} \text{ for all } s' \quad (5)$$

$$\bar{r}(b) = \sum_{s \in \mathcal{S}} r(s,a) b(s). \quad (6)$$

It is well known that the optimal finite-horizon value functions in POMDPs are piecewise linear convex. We adapt this result to show that the optimality equations for POMDPs can be cast in terms of linear dynamic programs. First, define the linear operator that represents the transitions $T(o,a) : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ for all $a \in \mathcal{A}$ and $o \in \mathcal{O}$ as follows: $T(a,o)[s,s'] = P(s,a,s')O(s,o)$, where $[s,s']$ represents the index of the matrix. Also, define actions that represent the convex combination of the original action space as follows:

$$\bar{\mathcal{A}} = \left\{ \bar{a} \in [0,1]^{|\mathcal{A}|} \;\middle|\; \sum_{a \in \mathcal{A}} \bar{a}(a) = 1 \right\}.$$

Proposition 2. *The following LDP expresses the optimality equations of a t-step POMDP value function:*

$$v(t,b) = \max_{\bar{a} \in \bar{\mathcal{A}}} \left(\sum_{s \in \mathcal{S}, a \in \mathcal{A}} \bar{a}(a) \bar{r}(s,\bar{a}) \right) + \gamma q(t,b,\bar{a})$$

$$q(t,b,\bar{a}) = \sum_{a \in \mathcal{A}} \bar{a}(a) \sum_{o \in \mathcal{O}} v(t+1, T(o,a)b).$$

Proof. The above definition clearly satisfies Conditions 1, 2, 3 of Definition 1; the reward function is linear in both $\bar{a}$ and s. Condition 4(a) holds because q is a linear combination of a finite number of piecewise linear functions when $\bar{a}$ is fixed. Condition 4(b) holds because q is linear in $\bar{a}$. Therefore, we need to show that the optimal solution of this LDP is identical to an optimal solution of the POMDP.

Let $v(t)$ denote a t-step optimal value function. This value function is piecewise linear from Proposition 1. To simplify the notation denote the probability of an observation o in a belief state b as $g(o,b) = \sum_{s \in \mathcal{S}} O(s,o) b(s)$. The Bellman optimality equation for the belief-state MDP model is then:

$$\begin{aligned}
v(t,b) &= \max_{a \in \mathcal{A}} \sum_{s \in \mathcal{S}} r(s,a) b(s) + \gamma \sum_{o \in \mathcal{O}} g(o,b) v(t+1, b') \\
&= \max_{a \in \mathcal{A}} \sum_{s \in \mathcal{S}} r(s,a) b(s) + \gamma \sum_{o \in \mathcal{O}} \max_{i \in \mathcal{I}} \alpha_i^{\mathsf{T}} \left(g(o,b) b' \right) + \beta_i \\
&= \max_{a \in \mathcal{A}} \sum_{s \in \mathcal{S}} r(s,a) b(s) + \gamma \sum_{o \in \mathcal{O}} v\left(t+1, T(a,o)b\right) \\
&= \max_{\bar{a} \in \bar{\mathcal{A}}} \sum_{s \in \mathcal{S}, a \in \mathcal{A}} \bar{a}(a) r(s,a) b(s) + \\
&\qquad \sum_{a \in \mathcal{A}} \bar{a}(a) \gamma \sum_{o \in \mathcal{O}} v\left(t+1, T(a,o)b\right),
\end{aligned}$$

which is identical to the LDP definition. □

Linear dynamic programs can be used to model problems that exhibit similar linear properties as POMDPs. For example, linear predictive state representations [Littman, Sutton, and Singh, 2002] can also be cast and solved as linear dynamic programs.

5 Modeling Linearly Controlled Problems

In this section, we describe linearly controlled problems (LCPs) and show how to model their optimal solutions as linear dynamic programs. LCPs are common in inventory and resource management and are closely related to stochastic linear programs [Kall and Mayer, 2005].

Linearly controlled problems, unlike POMDPs, are traditionally cast as minimization problems. In fact, we show that minimization of linearly controlled problems can be recast as maximization of linear dynamic programs.

Definition 2. *A linearly controlled problem is a tuple $(\mathcal{S}, \mathcal{A}, W, c, \Omega, \mu, T)$ where the compact set of admissible states $\mathcal{S} \subseteq \mathbb{R}^n$ and the compact set of admissible actions $\mathcal{A} \in \mathbb{R}^m$ are closed convex polyhedral sets. The set Ω represents possible* external *stochastic events. The set-valued function $W : \mathcal{S} \rightrightarrows \mathcal{A}$ maps states to their admissible actions and is defined as:*

$$W(s) = \{a \mid Es + Fa \leq g\} \neq \emptyset;$$

there exists an admissible action for every state. The measure $\mu : \mathcal{B}(\Omega) \to \mathbb{R}$ represents the probabilities of $\omega \in \Omega$. Transitions $P : \Omega \times \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ are defined as:

$$P(\omega, s, a) = A_\omega s + B_\omega a + c_\omega.$$

Finally, the cost function $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as:

$$c(s,a) = \max_{j \in \mathcal{J}} d_j^{\mathsf{T}} a + \max_{k \in \mathcal{K}} c_k^{\mathsf{T}} s.$$

Note that the cost function c is independently piecewise linear for a and s; it is not possible to represent a function that is jointly piecewise linear in the space of $[sa]$.

The Bellman optimality equations for LCPs are as follows:

$$v^*(t,s) = \min_{a \in W(s)} c(s,a) + \gamma q^*(t,s,a)$$

$$q^*(t,s,a) = \sum_{\omega \in \Omega} \mu(\omega) v^*(t+1, A_\omega s + B_\omega a + c_\omega)$$

These Bellman optimality equations cannot be directly modeled as an LDP, because the available actions depend on the state; that is, the set W depends linearly on the state. We show next that by taking the dual of the optimality conditions, LCPs become compatible with LDPs.

We use a convex conjugate function of the cost and state–action value to define a new action space $\bar{A} = \mathbb{R}^m$. A convex conjugate function $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is defined as follows:

$$f^*(x^*) = \sup_{x \in \mathbb{R}^n} x^\mathsf{T} x^* - f^*(s).$$

It is an alternative representation of convex functions. The conjugate dual of the cost function is defined as follows:

$$c^*(s,\bar{a}) = \sup_{a \in W(s)} \bar{a}^\mathsf{T} a - c(s,a).$$

Proposition 3. *The optimality equations for a t-step value function of a linearly controlled problem can be expressed as the following* linear dynamic program:

$$v(t,s) = \max_{\bar{a} \in \bar{A}} -c^*(s,\bar{a}) + \gamma q(t,s,\bar{a})$$

$$q(t,s,\bar{a}) = \inf_{a \in W(s)} \bar{a}^\mathsf{T} a + \sum_{\omega \in \Omega} \mu(\omega) v(t+1, A_\omega s + B_\omega a + c_\omega).$$

Proof. The dynamic program in the proposition trivially satisfies the linearity requirements of Definition 1 from the fact that a convex conjugate of a piecewise linear convex function is a piecewise linear convex function. To prove the correspondence to the optimality equations in the proposition, we use Fenchel duality [Rockafellar, 1996; Borwein and Lewis, 2000], which shows that a minimum of a sum of two functions equals the maximum of the sum of their conjugate duals. Formally, for given functions $f : E \to \mathbb{R} \cup \{\infty\}$ and $g : Y \to \mathbb{R} \cup \{\infty\}$ and a linear map $A : E \to Y$. If f and g are convex and satisfy the condition $0 \in \operatorname{core}(\operatorname{dom} g - A \operatorname{dom} f)$ then:

$$\inf_{x \in E} f(x) + g(Ax) = \sup_{\phi \in Y} -f^*(A^*\phi) + g^*(-\phi).$$

The theorem follows by using the conjugate dual of the state-action value function $q(t,s,a)$ to obtain a Fenchel dual representation of v.

Now, Conditions 1 and 2 of Definition 1 are satisfied trivially. To show Condition 3, note that the conjugate dual c^* of the cost function can be written as the following linear program using the fact that $W(s)$ is a polyhedral set:

$$\begin{aligned} c^*(s,\bar{a}) &= \sup_{a \in W(s)} \bar{a}^\mathsf{T} a - c(s,a) \\ &= \max_{a \in W(s)} \left(\bar{a}^\mathsf{T} a - \max_{j \in \mathcal{J}} d_j^\mathsf{T} a \right) - \max_{k \in \mathcal{K}} c_k^\mathsf{T} s. \end{aligned}$$

The formulation required in Condition 3 then follows by simply writing the Lagrangian of this linear program. Note that $-\infty < -c^*(s,\bar{a}) < M$ for some M since $W(s)$ is compact and non-empty. Condition 4 follows by simple algebraic manipulation. □

Note that linear control problems have several important limitations. First, the uncertainty is external and is not influenced by the states or actions. This is applicable in domains in which the influence of the decision maker on the environment is small. Second, the stochasticity may only influence the availability of the actions but not their costs or rewards. For example, while LCPs can be used easily to model stochastic water inflows, they cannot be easily used to model varying electricity prices.

6 Solving Linear Dynamic Programs

In this section, we describe a general point-based method for solving linear dynamic programs. This point-based method is motivated by methods for solving POMDPs [Lovejoy, 1991; Spaan and Vlassis, 2005; Pineau, Gordon, and Thrun, 2006]. We also summarize some theoretical properties of the point-based methods for LDPs, which closely mirror the results for similar POMDP algorithms.

Point based algorithms iteratively compute the value function in the order $v(T), v(T-1), \ldots, v(1)$. Since this value function is piecewise linear, as Proposition 1 shows, it is in principle possible to compute it precisely. However, the number of linear segments grows very quickly, making such an approach impractical. Instead, we approximate the value function by its lower bound $\underline{v}$ and an upper bound $\bar{v}$. As we show below, this approach also guarantees that the computed policy is close to optimal.

Both the upper bound $\bar{v}$ and lower bound $\underline{v}$ on the value function are piecewise linear convex functions and can be represented as follows:

$$\bar{v}(t,s) = \max_{i \in \bar{I}_t} \alpha_i^\mathsf{T} s + \beta_i \qquad \underline{v}(t,s) = \max_{i \in \underline{I}_t} \alpha_i^\mathsf{T} s + \beta_i.$$

Here, we assume that the indexed values α_i, β_i are determined only by the respective sets $\underline{I}$ and $\bar{I}$.

A generic point-based method is summarized in Algorithm 1 and can be seen as an extension of the approach proposed in [Pineau, Gordon, and Thrun, 2006]. Note that the algorithm is presented with a focus on simplicity; we use $(\alpha, \beta) \leftarrow$ and $s \rightarrow$ to represent individual piecewise linear segments.

Algorithm 1 assumes that samples used to compute the value function are already available. In practice, these samples may be obtained by forward simulation or by considering a uniform sample of the state space. Often, after a value function is computed, a new set of samples can be gathered by simulating the greedy policy.

Proposition 4. *The value function $\underline{v}_t$ in Algorithm 1 is a lower on bound on the optimal value function v_t^*. The value function $\bar{v}_t$ is an upper bound on the optimal value function.*

The proof follows similarly to corresponding results for POMDPs [Hauskrecht, 2000].

Algorithm 1: Point-based algorithm for linear dynamic programs. Here, $Z(\underline{I}, s)$ is assumed to appropriately modify the value function.

$\bar{\mathcal{S}}_{\mathcal{T}} \leftarrow$ Sampled states using a greedy policy;
$\underline{I}_T = \bar{I}_T \leftarrow \{(\mathbf{0}, 0)\}$;
$t \leftarrow T$;
for *t=T-1…0* **do**
  `// Select a sampled state`
  foreach $s \in \bar{\mathcal{S}}$ **do**
    `// Compute action value q for the lower and upper bounds`
    $\underline{J} \leftarrow Z(\underline{I}, s), \bar{J} \leftarrow Z(\bar{I}, s)$;
    `// Compute greedy actions using a linear program`
    $\bar{a} \leftarrow \arg\max_{a \in \mathcal{A}} r(s,a) + \gamma \min_{j \in \bar{J}} \alpha_i^\intercal a + \beta_i$;
    $\underline{a} \leftarrow \arg\max_{a \in \mathcal{A}} r(s,a) + \gamma \min_{j \in \underline{J}} \alpha_i^\intercal a + \beta_i$;
    `// This is a piecewise linear function`
    $\underline{I}_t \leftarrow \underline{I}_t \cup \{(\alpha, \beta) \leftarrow r(s,\underline{a}) + \gamma q(s,\underline{a})\}$;
  `// Update the upper bound once the value is computed for linear combinations of all sampled states`
  $\bar{I}_t \leftarrow \left\{ s \rightarrow \sum_{s' \in \mathcal{K}} \bar{v}(t,s') \,\middle|\, \sum_{s' \in \mathcal{K}} s' = s, \mathcal{K} \subseteq \bar{\mathcal{S}} \right\}$;

Note that, when Algorithm 1 is applied to POMDPs, Z can be simply computed by backward propagation of the transition functions. Computing the value of Z in LCPs entails solving a linear program, which is derived according to the proof of Proposition 3. In some sense, the actions computed then represent the dual variables of the linear program. While this makes LCPs somewhat harder to solve than POMDPs, our experiments indicate that this is not an issue in practice.

7 Application to Reservoir Management

We implemented the point-based approach and applied it to a realistic reservoir management problem. While in practice, multiple reservoirs would typically be managed simultaneously–with outflow of upstream reservoirs affecting the inflow of downstream reservoirs–we consider a single-reservoir setting for the sake of simplicity. We do model both variable inflows and variable electricity prices.

The state space in this reservoir management problem is defined by the tuple (l, i, e), where $l \in \mathbb{R}$ is the water volume in the reservoir available for discharge, $i \in \mathbb{R}$ is the inflow to the reservoir, and $e \in \mathbb{R}^m$ are the energy demands for m different electricity prices. The actions represent the discharge $d \in \mathbb{R}^m$ and are assumed to be discretized. The actions must satisfy the constraints on the water available for discharge and the demands for electricity:

$$\left\{ d \in \mathbb{R}^m \,\middle|\, \sum_{i=1\ldots m} d(i) \leq i,\ d(i) \leq e(i),\ i = 1 = m, \ldots, \right\}.$$

The transitions $(l, i, e) \rightarrow (l', i', e')$ are defined as follows. The water level in the next period is the current water level plus the inflow minus the discharge: $l' \leftarrow l + i - \sum_{k=1\ldots m} d(k)$. The random events are Ω_i and Ω_e that represent the uncertainties in the inflows and electricity prices.

The costs in the model are defined as follows:

$$c(l, i, e, d) = \sum_{k=1\ldots m} -p(k)d(k) + c_d \left(\sum_{k=1\ldots m} d(k) \right) + c_l l,$$

where $p(k)$ are the electricity prices at the appropriate demand levels, $c_d \in \mathbb{R} \rightarrow \mathbb{R}$ and $c_l : \mathbb{R} \rightarrow \mathbb{R}$ are piecewise linear convex penalty functions for the discharge and water level being outside the desirable bounds.

The data for inflows, desirable volume, and discharge came from the Yellowtail River Dam in Montana and were fit using generalized linear models. The inflows and energy demands were assumed to follow a log-normal distribution. We discretized the distribution into 14 intervals that cover the observed inflows during the past 50 years.

The decision captures the planned discharge over 10 days, with an upper bound due to electricity demand and structural limitations of the dam. The planning horizon was 10 time steps, which corresponds to 100 days. The policies were then executed in a rolling horizon fashion, always using the greedy policy for the 10-step value function.

The desirable water levels were obtained by matching historical levels. We assigned a piecewise linear penalty for water levels that are below 10% or above 90% of historical averages, with an additional very high penalty for overflowing the dam. Determining the right tradeoff between the penalty for water levels and the revenue for electricity is an important, hard modeling question. Computationally, this assumption did not influence the solution properties significantly, although achieving the right water level is slightly harder because it requires more planning. The results that we report assign 70% of the value to the water level penalty and 30% to revenue.

Figure 2 compares the performance of the proposed point-based method with a greedy policy in terms of the value function of the LDP. The results in the figure are averaged over 10 full executions. Note that with only 10 states sampled, the solution quality is close to optimal (upper bound shown by the top dotted line). Figure 3 shows the increase in computation time as a function of the number of states sampled. While the increase is quadratic as expected, we can obtain in a few seconds a near-optimal solution of this problem with an infinite number of states and actions.

8 Discussion

We introduce linear dynamic programs for solving complex resource management problems, and show how to extend point-based methods–originally developed for solving partially observable MDPs–to linearly controlled problems. Just like in POMDP solvers, these point-based methods take advantage of the linear structure of the input problem. The performance of the approach on a realistic reservoir management problem is very encouraging, returning near-optimal results within a few seconds.

The proposed point-based methods are closely related to dual dynamic programming and linear stochastic programs [Pereira, 1985; 1989; Kall and Mayer, 2005; Shapiro,

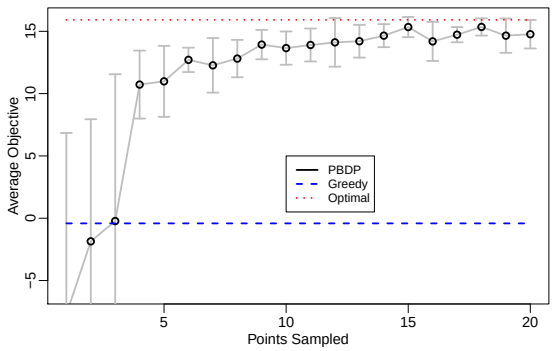

Figure 2: Comparison of point-based dynamic programming, greedy, and an upper bound on the optimal policy.

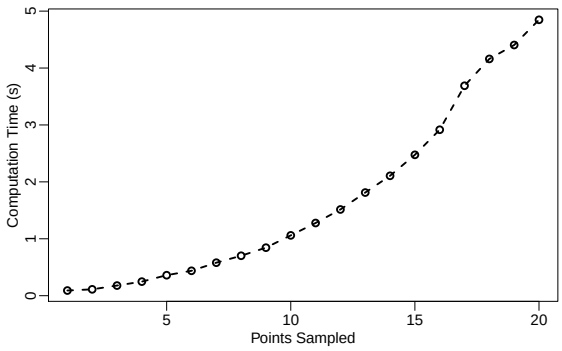

Figure 3: Computation time as a function of the number of sampled points.

2011]. Linear stochastic programs do not have the notion of states, actions, and value functions. Instead, the solution is obtained by computing cuts. These cuts correspond to the linear segments of the value function, but generally only the lower bound is used. Recently, [Weinstein, Mansley, and Littman, 2010] proposed a method for solving MDPs with large action spaces, but they do not take advantage of linear problem structures.

This new linear dynamic programming framework makes it possible to adapt powerful POMDP solvers to a range of sustainable resource management applications.

Acknowledgments

We thank Stefan Woerner and Dharmashankar Subramanian for helpful discussions of linearly controlled problems and stochastic dual dynamic programming.

References

Bellman, R. 1957. *Dynamic Programming*. Princeton University Press.

Borwein, J., and Lewis, A. S. 2000. *Convex Analysis and Nonlinear Optimization: Theory and Examples*.

Boyd, S., and Vandenberghe, L. 2004. *Convex Optimization*. Cambridge University Press.

Hauskrecht, M. 2000. Value-function approximation for partially observable Markov decision processes. *Journal of Artificial Intelligence Research* 13:33–94.

Kall, P., and Mayer, J. 2005. *Stochastic Linear Programming: Models, Theory, and Computation*. Springer.

Littman, M.; Sutton, R.; and Singh, S. 2002. Predictive representations of state. In *Advances in Neural Information Processing Systems*.

Lovejoy, W. S. 1991. Computationally feasible bounds for partially observed Markov decision processes. *Operations Research* 39:162–175.

Pereira, M. V. F. 1985. Stochastic optimization of a multireservoir hydroelectric system: A decomposition approach. *Water Resource Research* 21:779–792.

Pereira, M. V. F. 1989. Stochastic operation scheduling of large hydroelectric systems. *Electric Power and Energy Systems* 11(3):161–169.

Petrik, M., and Zilberstein, S. 2009. Robust value function approximation using bilinear programming. In *Advances in Neural Information Processing Systems (NIPS)*.

Pineau, J.; Gordon, G.; and Thrun, S. 2006. Anytime point-based approximations for fast POMDP solving. *Journal of Artificial Intelligence Research* 27(SOCS-TR-2005.4):335–380.

Powell, W. B. 2007. *Approximate Dynamic Programming*. Wiley-Interscience.

Puterman, M. L. 2005. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc.

Rockafellar, R. T. 1996. *Convex Analysis*. Princeton University Press.

Shapiro, A. 2011. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research* 209:63–72.

Spaan, M. T. J., and Vlassis, N. 2005. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research* 24:195–220.

Vanderbei, R. J. 2001. *Linear Programming: Foundations and Extensions*. Springer, 2nd edition.

Weinstein, A.; Mansley, C.; and Littman, M. 2010. Sample-based planning for continuous action Markov decision processes. In *ICML Workshop on Reinforcement Learning and Search in Very Large Spaces*.

Woerner, S. 2010. Approximate parametric dynamic programing in inventory management. Master's thesis, ETH.