An algorithm for solving infinite horizon Markov dynamic programmes

Regan Baucke[a,*]

[a]The Department of Engineering Science, The University of Auckland, 70 Symonds Street, Auckland 1010, New Zealand

Abstract

We consider a general class of infinite horizon dynamic programmes where state and control sets are convex and compact subsets of Euclidean spaces and (convex) costs are discounted geometrically. The aim of this work is to provide a convergence result for these problems under as few restrictions as possible. Under certain assumptions on the cost functions, infinite horizon cost-to-go functions can be bounded by a pair of convex, Lipschitz-continuous bounding functions; we seek to refine these bounding functions until an ϵ convergence criteria is met. We prove a convergence result for a simplified version of our problem, and then apply this result for the stochastic version problem where uncertainty is governed by a discrete Markov process. Further, our algorithm is deterministic and requires no Monte-carlo simulation to estimate an upper bound on the cost of a given policy.

Keywords: dynamic programming, decomposition, multistage, stochastic programming, infinite horizon

1. Introduction

Dynamic programming was first developed by Bellman [2] and is a popular methodology in mathematical programming. The method seeks to compute *cost-to-go* functions which encode the cost of future optimal decisions; once the cost-to-go functions have been computed, the optimal policy for a given state is then easily computed as the minimiser of the current stage's cost plus the cost-to-go. In this paper, we focus on a particular class of dynamic programmes; where the state and control spaces are compact and convex subsets of $\mathbb{R}^n$ and $\mathbb{R}^m$ respectively and the modeller seeks to minimise a discounted infinite sum of convex costs. Often infinite horizon models of this type are referred to as Markov decision processes (MDPs). These formulations are useful for modelling problems which have a natural steady state nature: for instance inventory problems or machine maintenance problems, where a control is based on the state of the system regardless of its age or history.

The classical literature of dynamic programming concerns problems where state and action spaces are finite. When state and actions spaces are convex, many of the classical approaches fail as the exact computation of value-functions is impossible in general. As a first attempt, discretisation of the state and action spaces may seem attractive, but in general this approach suffers from the curse of dimensionality.

Benders decomposition from Benders [3] has been a respite against the aforementioned curse; by taking advantage of the convexity of subproblems, global lower bounds can be formed over the (uncountably infinite) domain of the cost-to-go function. This idea was extended to multistage problems in Birge [4]. Rather than performing the hopeless task of computing cost-to-go functions exactly everywhere, piecewise linear lower bounds can be iteratively constructed as an estimate of the cost-to-go function. Under certain assumptions, these lower bound estimates become arbitrarily close to the true function at areas of interest. Since then, many researchers have contributed to the field: methods such as SDDP in [9], DOASA from [10] and CUPPS in [5], all fall under this general category.

The method we introduce lives in the domain of dual dynamic programming methods; at each iteration of our algorithm, we compute a state and control trajectory using the solution to a stage problem which contains our best lower approximation of the *cost-to-go* function. These bounding functions are then updated using zeroth and first-order information from the proceeding optimisation problem, often referred to as *cuts*. This method is well studied in the finite-horizon case where a boundary condition is enforced on the value function at the final stage. In the absence of this terminal boundary condition, a new approach must be taken in order to achieve convergence.

The general idea of the new approach is to allow the length of the forward and backward pass grow larger as the algorithm progresses. Throughout this work we refer to this as 'looking ahead'. The work of Nannicini et al. [8] is of particular interest; they develop an algorithm which looks ahead further one stage each time. Our work uses a similar approach but differs in several key ways: we study a general non-linear (but convex) version of the problem, where cost functions and constraint sets need not be linear. Further, we prove that under a certain assumption, the look-ahead process need not be monotonically increasing nor even require that its limit tend to infinity. Finally, similar to Baucke et al. [1], in the stochastic case, we prove that our algorithm converges deterministically. This makes any requirement to estimate an upper bound on the best policy using Monte-carlo simulation redundant. These key differences form the main contributions of this work.

The paper is laid out as follows: in Section 2 we introduce the bounding functions which are used in our algorithms; their

*Corresponding author
Email address: r.baucke@auckland.ac.nz (Regan Baucke)

properties are the key ingredient for convergence. In Section 3 we study a special case of our problem and state and prove the main theorem in this work. In Section 4, we extend our method to the stochastic case where we study problems where the uncertainty is governed by a discrete Markov process. In Section 5 we provide a discussion and make our conclusions.

2. Bounding functions

As the main feature of our algorithm is the refinement of *bounding functions* we shall dedicate this opening section toward their exposition. Much of this section will also introduce our notation and elementary objects. Consider a proper lower semi-continuous convex function $C : \mathcal{U} \mapsto \bar{\mathbb{R}}$, where $\mathcal{U} \subset \mathbb{R}^n$ and $\bar{\mathbb{R}}$ denotes the extended real numbers. We are interested in forming bounding functions $\bar{C}$ and $\underline{C}$, which use zero-order and first-order information from a finite and non-empty set of sample points $S \subset \mathcal{U}$.

Definition 1. For a finite, non-empty set of sample points S, let

$$\underline{C}^S(u) = \max_{u_s \in S} \; C(u_s) + \langle d_s, u - u_s \rangle.$$

where d_s is a subgradient of $C(u)$ at u_s.

The linear programming formulation of the above (known as *epigraph* form) can be written as

$$\begin{aligned} \underline{C}^S(u) = \min_{\mu \in \mathbb{R}} \quad & \mu \\ \text{s.t.} \quad & \mu \geq C(u_s) + \langle d_s, u - u_s \rangle, \; \forall u_s \in S. \end{aligned}$$

This lower bound function is a key part of many decomposition algorithms and dates back to Kelley [7].

Definition 2. For a finite, non-empty set of sample points S, let

$$\begin{aligned} \bar{C}^S(u) = \max_{\mu \in \mathbb{R}, \lambda \in \mathbb{R}^n} \quad & \mu + \langle \lambda, u \rangle \\ \text{s.t.} \quad & \mu + \langle \lambda, u_s \rangle \leq C(u_s), \; \forall u_s \in S, \\ & \|\lambda\|_* \leq \alpha. \end{aligned}$$

This upper bound function was first introduced in Baucke et al. [1]; notice the similar structure to the epigraph formulation. Indeed, if the $\|\cdot\|_\infty$-norm is considered, the upper bound function is a linear programme.

Proposition 1. *The bounding functions $\bar{C}$ and $\underline{C}$ have the following properties:*

1. *bounding functions are valid;*
2. *bounding functions are equal at each sample point;*
3. *additional sample points do not worsen bounds;*
4. *if C is Lipschitz-continuous then the bounding functions are also Lipschitz-continuous.*

These properties are proved across several Lemmas in Baucke et al. [1]; they are required if our ensuing algorithm is to be convergent.

3. Single vertex case

The main result of this work is a convergence theorem; for clarity, we present the proof on a simplified version of the problem – the result is then leveraged for the general Markovian case.

3.1. Problem formulation

Consider the following optimisation problem with $\delta \in [0, 1)$:

$$\begin{aligned} V(x^0) = \min_{u^l} \quad & \sum_{l=0}^{\infty} \delta^l C(x^l, u^l) \\ \text{s.t.} \quad & x^{l+1} = f(x^l, u^l), \; \forall l \in \mathbb{N}, \\ & x^l \in \mathcal{X}, \; \forall l \in \mathbb{N}, \\ & u^l \in \mathcal{U}(x^l), \; \forall l \in \mathbb{N}. \end{aligned} \tag{1}$$

We call x^l the state of the system (prior to any control), while u^l is called a control. Our objective is to minimise the discounted infinite sum of costs, for a given initial state x^0. We require the functions and multifunctions that make up the above optimisation problem to take a specific form – these conditions ensure that the optimisation problem is convex, feasible, and that subgradients are bounded. For the following analysis, we will use

$$\tilde{\mathcal{U}}(x) = \{u \in \mathcal{U}(x) \mid f(x, u) \in \mathcal{X}\},$$

and $\text{Aff}(\mathcal{X})$ to denote the affine hull of $\mathcal{X}$.

Condition 1. *We assume that the optimisation problem* (1) *has the following properties:*

1. *$\mathcal{X}$ is a non-empty compact and convex subset of $\mathbb{R}^n$;*
2. *the multifunction $\mathcal{U} : \mathbb{R}^n \rightrightarrows \mathbb{R}^m$ is convex and non-empty compact valued;*
3. *the cost function $C(x, u)$ is a convex lower semicontinuous proper function of x and u;*
4. *f is affine in x and u;*
5. *there exists $\gamma > 0$, defining $\mathcal{X}' := \mathcal{X} + B(\gamma)$, where*

$$B(\gamma) = \{y \in \mathit{Aff}(\mathcal{X}) \mid \|y\| \leq \gamma\}$$

such that

   (a) $C(x, u) < \infty, \; \forall x \in \mathcal{X}', \; \forall u \in \mathcal{U}(x)$;
   (b) $f(x, \mathcal{U}(x))) \cap \mathcal{X}$ is non-empty, $\forall x \in \mathcal{X}'$.

These conditions are the same as those considered in Girardeau et al. [6]. Under these conditions, we will layout several properties of the *cost-to-go* functions induced by (1). Noting the self-similar nature of the formulation above, we define the following *cost-to-go* function $V(x) : \text{Aff}(\mathcal{X}) \mapsto \bar{\mathbb{R}}$ as

$$V(x) = \begin{cases} \min\limits_{u \in \tilde{\mathcal{U}}(x)} \; C(x, u) + \delta V(f(x, u)), \; x \in \mathcal{X}, \\ +\infty, \; \text{otherwise.} \end{cases} \tag{2}$$

We can pictorially represent the problem in (2) as that in Figure 1. Here the vertex represents the cost function and the arc represents the transition to the new state. Diagrams like in Figure 1 are a useful method for describing more complicated Markov stochastic processes.

Figure 1: A graph representation of our motivating problem

3.2. Properties of value functions

Our eventual algorithm relies on several properties of the cost-to-go function $V(x)$ in order to converge; here we will introduce these properties. Consider the *extended cost-to-go* function $\tilde{V}(x) : \mathrm{Aff}(\mathcal{X}) \mapsto \bar{\mathbb{R}}$ defined as

$$\tilde{V}(x) = \inf_{u \in \tilde{\mathcal{U}}(x)} C(x,u) + \delta \tilde{V}(f(x,u)). \tag{3}$$

Lemma 1.

$$V(x) = \tilde{V}(x), \ \forall x \in \mathcal{X}.$$

Proof. Because $x \in \mathcal{X}$,

$$V(x) = \min_{u \in \tilde{\mathcal{U}}(x)} C(x,u) + \delta V(f(x,u)).$$

Because $C(x,u)$ is lower semicontinuous and $\tilde{\mathcal{U}}(x)$ is non-empty and compact-valued, the (infinitely many) minimums above are attained and therefore are equal to the infimums in (3). □

Lemma 2. *The function $\tilde{V}(x)$ is convex.*

Proof. The function $\tilde{V}(x)$ is convex as it is the infimum over u of a sum (albeit infinite) of convex functions of x and u. □

Lemma 3. *The function $\tilde{V}(x)$ is bounded on $\mathcal{X}'$.*

Proof. From Condition 1.5(a), $C(x,u) < \infty$, $\forall x \in \mathcal{X}', u \in \mathcal{U}(x)$. So $\bar{M}_C := \sup_{u \in \mathcal{U}(x), x \in \mathcal{X}'} C(x,u) < \infty$. From Condition (1.3), $C(x,u)$ is a proper function of x and u so $\underline{M}_C := \inf_{u \in \mathcal{U}(x), x \in \mathcal{X}'} C(x,u) > -\infty$. The extended cost-to-go function $\tilde{V}(x)$ cannot exceed the geometric sum of the supremum of the cost function, nor fall below the geometric sum of the infimum of the cost function. So

$$-\infty < \frac{1}{1-\delta}\underline{M}_C \leq \tilde{V}(x) \leq \frac{1}{1-\delta}\bar{M}_C < \infty, \ \forall x \in \mathcal{X}', \tag{4}$$

concluding the proof. □

Theorem 1. *The function $V(x)$, as defined above, is convex and Lipschitz-continuous on $\mathcal{X}$.*

Proof. From Lemma 3, $\tilde{V}(x)$ is bounded on $\mathcal{X}'$. So $\tilde{V}(x)$ is Lipschitz on $\mathcal{X}$, as from Lemma 2, $\tilde{V}(x)$ is convex, and bounded on $\mathcal{X}'$, and $\mathcal{X}$ is a compact subset of the relative interior of its domain. From Lemma 1, we have $\tilde{V}(x) = V(x)$, $\forall x \in \mathcal{X}$, so $V(x)$ is convex and Lipschitz on $\mathcal{X}$. □

Because $V(x)$ is convex, we can use the bounding functions introduced earlier to bound our value function $V(x)$. Furthermore, because $V(x)$ is Lipschitz-continuous, we can be sure that our bounding functions are Lipschitz-continuous also.

3.3. Algorithm

Consider a forward pass in a standard DDP algorithm as in Pereira and Pinto [9]; a forward pass is made until a terminal value function is reached, and then a backward pass of value function updates is performed. As there is no 'end point' in an infinite horizon problem, this approach would never be able to complete an iteration. An important element in our algorithm which addresses this pitfall is the 'look-ahead' function $L : \mathbb{N} \mapsto \mathbb{N}$. For the k^{th} iteration, the algorithm generates a state and control trajectory of length $L(k)$, and updates the value function at these points. This allows each iteration to contain a finite amount of computation. We present the following algorithm which solves problems in the form of (1).

Algorithm 1 (The singleton infinite horizon algorithm).

Initialisation. Define $\bar{V}^0(x) = \frac{\bar{M}_C}{1-\delta}$ and $\underline{V}^0(x) = \frac{\underline{M}_C}{1-\delta}$. For all k, set $x^{k,0}$ as x^0. Finally, set $k = 1$.

Iteration k.

Step 1. Set $l = 0$.

Step 2. Solve

$$\begin{aligned} \bar{\theta}^{k,l} = \min_{x^{l+1}, u^l} \quad & C(x^{k,l}, u^l) + \delta \bar{V}^{k-1}(x^{l+1}) \\ \text{s.t.} \quad & x^{l+1} = f^x(x^{k,l}, u^l) \\ & x^{l+1} \in \mathcal{X} \\ & u^l \in \mathcal{U}(x^{k,l}). \end{aligned} \tag{5}$$

Step 3. Solve

$$\begin{aligned} \underline{\theta}^{k,l} = \min_{x^{l+1}, u^l} \quad & C(x^{k,l}, u^l) + \delta \underline{V}^{k-1}(x^{l+1}) \\ \text{s.t.} \quad & x^{l+1} = f^x(x^{k,l}, u^l) \\ & x^{l+1} \in \mathcal{X} \\ & u^l \in \mathcal{U}(x^{k,l}), \end{aligned} \tag{6}$$

storing the minimisers as $(x^{k,l+1}, u^{k,l})$. Compute a sub-gradient $\beta^{k,l}$ with respect to $x^{k,l}$.

Step 4. If $l = L(k)$, move to **Step 5**. Otherwise move to **Step 2** with $l = l + 1$.

Step 5. Update the upper bound as

$$\begin{aligned} \bar{V}^k(x) = \max_{\mu \in \mathbb{R}, \lambda \in \mathbb{R}^n} \quad & \mu + \langle \lambda, x \rangle \\ \text{s.t.} \quad & \mu + \langle \lambda, x^{k',l} \rangle \leq \bar{\theta}^{k',l}, \ \forall l \leq L(k'), \ \forall k' \leq k, \\ & \|\lambda\|_* \leq \alpha. \end{aligned} \tag{7}$$

Step 6. Update the lower bound as

$$\begin{aligned} \underline{V}^k(x) = \min_{\mu \in \mathbb{R}} \quad & \mu \\ \text{s.t.} \quad & \mu \geq \underline{\theta}^{k',l} + \langle \beta^{k',l}, x - x^{k',l} \rangle, \ \forall l \leq L(k'), \ \forall k' \leq k. \end{aligned} \tag{8}$$

Step 7. Move on to iteration $k + 1$. ⋈

We note that by using the most up-to-date information about our value functions at any given point in the algorithm, we could compute dominant cuts which would improve the speed of convergence. We have refrained from outlining such a procedure here, as it is not necessary for the convergence of our algorithm. In some sense, our convergence result is more general as we are using weakest set of cuts possible.

Theorem 2. *Consider the sequence of functions $\bar{V}^k, \underline{V}^k$, state trajectories $x^{k,l}$, associated controls $u^{k,l}$ and function $L : \mathbb{N} \times \mathbb{N}$. If $\limsup_{k \in \mathbb{N} \to \infty} L(k) = \infty$, then*

$$\lim_{k \to \infty} (\bar{V}^k(x^{k,l}) - \underline{V}^k(x^{k,l})) = 0, \ \forall l \in \mathbb{N}. \tag{9}$$

For a notational convenience we denote the iteration counter k as k_0 and let $\bar{\underline{V}} = \bar{V} - \underline{V}$. We will first prove the following technical lemmas which will be useful for the proof of Theorem 2.

Lemma 4. *Consider the sequence of functions $\bar{V}^{k_0}, \underline{V}^{k_0}$ and state trajectories $x^{k_0,l}$ and associated controls $u^{k_0,l}$. We have*

$$\underline{V}^{k_0}(x^{k_0,l}) \leq \delta \underline{V}^{k_0-1}(x^{k_0,l+1}), \quad \forall k_0, \ \forall l \leq L(k_0).$$

Proof. From (6) and (8) in Algorithm 1, we obtain the following inequality:

$$\underline{V}^{k_0}(x^{k_0,l}) \geq \underline{\theta}^{k_0,l} = C(x^{k_0,l}, u^{k_0,l}) + \delta \underline{V}^{k_0-1}(x^{k_0,l+1}).$$

Similarly for the upper bound, from (5) and (7) we obtain

$$\bar{V}^{k_0}(x^{k_0,l}) \leq \bar{\theta}^{k_0,l} \leq C(x^{k_0,l}, u^{k_0,l}) + \delta \bar{V}^{k_0-1}(x^{k_0,l+1}).$$

By taking the difference of these inequalities, we obtain our desired result. □

Lemma 5. *Consider a 'look-ahead' function $L : \mathbb{N} \mapsto \mathbb{N}$ with $\limsup_{k \in \mathbb{N} \to \infty} L(k) = \infty$. The set*

$$\mathbb{N}_L^* := \{k^* \in \mathbb{N} \mid L(k^*) > L(k), \ \forall k < k^* \in \mathbb{N}\}$$

is countably infinite.

Proof. Because of the limit property of L, for any $k^* \in \mathbb{N}_L^*$ there will always exist an $k^\circ \in \mathbb{N}$ for which $L(k^\circ) > L(k^*)$. So $k^\circ \in \mathbb{N}_L^*$, and so $\mathbb{N}_L^*$ must be infinite. Because $\mathbb{N}_L^* \subseteq \mathbb{N}$, the set $\mathbb{N}_L^*$ must be countably infinite. □

Lemma 6. *Consider a 'look-ahead' function $L : \mathbb{N} \mapsto \mathbb{N}$ with $\limsup_{n \in \mathbb{N} \to \infty} L(n) = \infty$ and the corresponding set $\mathbb{N}_L^*$ defined above. For any $N \in \mathbb{N}$ there exists a $\tilde{k}(N) \in \mathbb{N}_L^*$ such that*

$$\min\{L(k_0), L(k_1) - 1, \ldots, L(k_s) - (s-1)\} > N, \\ \forall k_0 > k_1 > \ldots > k_s > \tilde{k}(N) \in \mathbb{N}_L^*.$$

Proof. Notice that by construction, $L(k)$ is strictly monotonically increasing $\forall k \in \mathbb{N}_L^*$. So $\min\{L(k_0), L(k_1) - 1, \ldots, L(k_s) - (s-1)\} = L(k_s) - (s-1), \forall k_0 > k_1 > \ldots k_s \in \mathbb{N}_L^*$. From its strict monotonicity, we have $\lim_{k_s \in \mathbb{N}_L^* \to \infty} L(k_s) = \infty$. So $\lim_{k_s \in \mathbb{N}_L^* \to \infty} L(k_s) - (s-1) = \infty$. So there must exist a $\tilde{k}(N) \in \mathbb{N}_L^*$ for which this lemma's hypothesis holds. □

We proceed with the proof of Theorem 2. The proof utilises a fact that agrees with a certain intuition; our bounding functions should eventually converge as long as we look far enough into the future often enough. If $\limsup_{k \in \mathbb{N} \to \infty} L(k) = \infty$, then we have our desired 'look-ahead' property.

Proof of Theorem 2. Suppose for the sake of contradiction that this theorem's hypothesis is false, i.e. there exists an $\epsilon > 0$ for which

$$\epsilon \leq \bar{\underline{V}}^{k_0}(x^{k_0,l}), \ \forall k_0 \in \mathbb{N}, \ \forall l \in \mathbb{N}.$$

From Lemma 4 and the monotonicity of $\bar{V}^k(x)$ with respect to k for all $x \in \mathcal{X}$ from Proposition 1.3, we have

$$\epsilon \leq \bar{\underline{V}}^{k_0}(x^{k_0,l}) \leq \delta \bar{\underline{V}}^{k_1}(x^{k_0,l+1}), \quad \forall k_0 > k_1 \in \mathbb{N}, \ \forall l \leq L(k_0). \tag{10}$$

Here the universal quantification $\forall k_0 > k_1 \in \mathbb{N}$ should be understood as: for all $k_1 \in \mathbb{N}$ and for all $k_0 \in \mathbb{N}$ such that $k_0 > k_1$. From the Lipschitz-continuity of both $\bar{V}^k(x)$ and $\underline{V}^k(x)$, we have

$$\bar{\underline{V}}^{k_1}(x^{k_0,l+1}) \leq \bar{\underline{V}}^{k_1}(x^{k_1,l+1}) + 2\alpha \|x^{k_1,l+1} - x^{k_0,l+1}\|, \\ \forall k_0, k_1 \in \mathbb{N}, \ \forall l \leq \min\{L(k_0), L(k_1)\}.$$

We note here that because $L(k)$ need not be monotonically increasing $\forall k \in \mathbb{N}$, $x^{k_0,l+1}$ and $x^{k_1,l+1}$ may not be both defined at a given iteration of our algorithm. However we can be sure that $x^{k_0,l+1}$ and $x^{k_1,l+1}$ both exist for those $l \leq \min\{L(k_0), L(k_1)\}$. By combining the two previous inequalities, we obtain

$$\epsilon \leq \delta[\bar{\underline{V}}^{k_1}(x^{k_1,l+1}) + 2\alpha \|x^{k_1,l+1} - x^{k_0,l+1}\|], \\ \forall k_0 > k_1 \in \mathbb{N}, \ \forall l \leq \min\{L(k_0), L(k_1)\}. \tag{11}$$

We can form a similar inequality by the same process as above – from Lemma 4, we have

$$\bar{\underline{V}}^{k_1}(x^{k_1,l+1}) \leq \delta \bar{\underline{V}}^{k_2}(x^{k_1,l+2}), \quad \forall k_1 > k_2 \in \mathbb{N}, \ \forall l \leq L(k_1) - 1,$$

and from the Lipschitz-continuity property of the bounding functions, we have

$$\bar{\underline{V}}^{k_2}(x^{k_1,l+2}) \leq \bar{\underline{V}}^{k_2}(x^{k_2,l+2}) + 2\alpha \|x^{k_2,l+2} - x^{k_1,l+2}\|, \\ \forall k_1, k_2 \in \mathbb{N}, \ \forall l \leq \min\{L(k_1) - 1, L(k_2) - 1\}.$$

By merging the two inequalities above, we obtain

$$\bar{\underline{V}}^{k_1}(x^{k_1,l+1}) \leq \delta[\bar{\underline{V}}^{k_2}(x^{k_2,l+2}) + 2\alpha \|x^{k_2,l+2} - x^{k_1,l+2}\|], \\ \forall k_1 > k_2 \in \mathbb{N}, \ \forall l \leq \min\{L(k_1) - 1, L(k_2) - 1\}. \tag{12}$$

By substituting (12) into (11), we have

$$\epsilon \leq \delta\Big[\delta[\bar{\underline{V}}^{k_2}(x^{k_2,l+2}) + 2\alpha \|x^{k_2,l+2} - x^{k_1,l+2}\|] + 2\alpha \|x^{k_1,l+1} - x^{k_0,l+1}\|\Big], \\ \forall k_0 > k_1 > k_2 \in \mathbb{N}, \ \forall l \leq \min\{L(k_0), L(k_1) - 1, L(k_2) - 1\}.$$

Note the rather subtle restrictions on l, k_0, k_1, and k_2 for which this inequality applies. By performing this expansion s times, we obtain

$$\epsilon \leq \delta^s \bar{\underline{V}}^{k_s}(x^{k_s,l+s}) + 2\alpha \sum_{s'=1,\ldots,s} \delta^{s'} \|x^{k_{s'},l+s'} - x^{k_{s'-1},l+s'}\|, \\ \forall k_0 > k_1 > \ldots > k_s \in \mathbb{N}, \\ \forall l \leq \min\{L(k_0), L(k_1) - 1, L(k_2) - 2, \ldots, L(k_s) - (s-1)\}.$$

Because $\delta \in [0,1)$ and $\underline{\bar{V}}^{k_s}(x^{k_s,l+s})$ is bounded, there must exist an $\tilde{s}(l) \in \mathbb{N}$ for which $\delta^{\tilde{s}(l)}\underline{\bar{V}}^{k_{\tilde{s}(l)}}(x^{k_{\tilde{s}(l)},l+\tilde{s}(l)}) \leq \frac{\epsilon}{2}$. So by subtracting this term from above and dividing through by 2α, we obtain

$$\frac{\epsilon}{4\alpha} \leq \sum_{s'=1,\ldots,\tilde{s}(l)} \delta^{s'} \|x^{k_{s'},l+s'} - x^{k_{\tilde{s}-1},l+s'}\|,$$

$$\forall k_0 > k_1 > \ldots > k_{\tilde{s}(l)} \in \mathbb{N},\ \forall l \leq \min\{L(k_0), \ldots, L(k_{\tilde{s}(l)}) - (\tilde{s}(l)-1)\}.$$

By Lemma 6, we have for any $l \in \mathbb{N}$ there must exist a $\tilde{k}(l) \in \mathbb{N}_L^*$ for which $l \leq \min\{L(k_0), L(k_1)-1, \ldots, L(k_{\tilde{s}(l)})-(\tilde{s}(l)-1)\}$, $\forall k_0 > k_1 > \ldots > k_s > \tilde{k}(l) \in \mathbb{N}_L^*$. Furthermore, the set $\mathbb{N}_L^*$ is a subset of $\mathbb{N}$. So

$$\frac{\epsilon}{4\alpha} \leq \sum_{s'=1,\ldots,\tilde{s}(l)} \delta^{s'} \|x^{k_{s'},l+s'} - x^{k_{s'-1},l+s'}\|,$$
$$\forall k_0 > k_1 > \ldots > k_{\tilde{s}(l)} > \tilde{k}(l) \in \mathbb{N}_L^*,\ \forall l \in \mathbb{N}.$$

By Lemma 5, the set $\mathbb{N}_L^*$ is countably infinite so the above is a contradiction of the compactness of $\mathcal{X}$ as at least one of the sequences in $\{x^{k,l+1}, \ldots, x^{k,l+\tilde{s}(l)}\}, \forall k > \tilde{k}(l) \in \mathbb{N}_L^*$ contains a non-convergent subsequence. So there exists no $\epsilon > 0$ for which

$$\epsilon \leq \underline{\bar{V}}^{k_0}(x^{k_0,l}),\ \forall k_0,\ \forall l \in \mathbb{N},$$

concluding the proof. □

4. Markovian case

In this section, we present the algorithm on the general problem class. On some generic, countably infinite filtered probability space $(\Omega, \mathscr{F}, \{\mathscr{F}_l\}_{l\in\mathbb{N}}, \mathbb{P})$, we can describe a stochastic infinite horizon optimisation problem for an initial state $x_{n_0}^0$ as

$$\begin{aligned} V_{n_0}^{\mathrm{Ex}}(x_{n_0}^0) = \min_{u_{\omega(l)}} \quad & \mathbb{E}_{\mathbb{P}}\Big[\sum_{l=0}^{\infty} \delta^l C_{\omega(l)}(x_{\omega(l)}, u_{\omega(l)})\Big] \\ \text{s.t.} \quad & x_{\omega(l+1)} = f_{\omega(l+1)}(x_{\omega(l)}, u_{\omega(l)}),\ \forall l \in \mathbb{N},\ \omega \in \Omega, \\ & x_{\omega(l+1)} \in \mathcal{X}_{\omega(l+1)},\ \forall l \in \mathbb{N},\ \omega \in \Omega, \\ & u_{\omega(l)} \in \mathcal{U}_{\omega(l)}(x_{\omega(l)}),\ \forall l \in \mathbb{N},\ \omega \in \Omega. \end{aligned} \tag{13}$$

The set Ω is the set of all paths of a countably infinite scenario tree (which encodes the non-anticipativity), and $\omega(l)$ is the l^{th} vertex along path ω. This implies that $n_0 = \omega(0), \forall \omega \in \Omega$ is the root vertex. Here the modeller seeks to minimise the expectation over all paths possible of the infinite discounted accumulated costs through that particular path. In the Markov case, rather than associating our probability space with an infinite scenario tree, we can 'compress' the tree and describe the problem's uncertainty with a connected di-graph with every vertex having an in-degree and out-degree of at least one. Denote the vertices on the graph $\mathcal{N}$ with N vertices as $\{n_0, \ldots, n_N\}$, with the set of child vertices of vertex n as $R(n)$. Figure 2 depicts an example of such a graph. Rather than stating the Markov problem in extensive form i.e. (13), it is quite natural to express the problem formulation directly through its dynamic programming equations. The infinite horizon dynamic programming equation for an arbitrary vertex n is given by

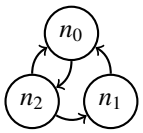

Figure 2: A graph representation of our general problem

$$\begin{aligned} V_n(x_n) = \min_{x_m, u_n} \quad & C_n(x_n, u_n) + \delta \sum_{m\in R(n)} p_{n,m} V_m(x_m) \\ \text{s.t.} \quad & x_m = f_{n,m}(x_n, u_n),\ \forall m \in R(n), \\ & x_m \in \mathcal{X}_m,\ \forall m \in R(n), \\ & u_n \in \mathcal{U}_n(x_n). \end{aligned} \tag{14}$$

where $p_{n,m}$ is the *conditional probability* of reaching vertex m from vertex n. Although we do not show it here, we have that the extensive formulation and dynamic programming formulation coincide i.e. $V_{n_0}^{\mathrm{Ex}}(x_{n_0}^0) = V_{n_0}(x_{n_0}^0)$. Once again, we restrict the cost functions and state and control sets to take the form of those considered in the single vertex case. It is also worth mentioning here that a stochastic process of independent and identically distributed random variables can be considered a special case of a Markov process.

4.1. Algorithm

The algorithm presented here is similar to Algorithm 1, except we make use of the *problem-child* criterion outlined in Baucke et al. [1]; this is the key ingredient to achieving deterministic convergence. As we 'look ahead' during an iteration of our algorithm, we record the child vertex with the largest bound difference for the new state. We label this vertex $\phi_n^{k,l}$, the problem-child of vertex n. Similar to the analysis in Lemma 3, it possible to show that the value functions are convex and Lipschitz-continuous and we can compute initial bounds on the cost-to-go functions $(\bar{M}_n, \underline{M}_n)$ at each vertex. Our algorithm for computing $V_{n_0}(x_{n_0}^0)$ is outlined below.

Algorithm 2 (The Markov infinite horizon algorithm).

Initialisation. Define $\bar{V}_n^0(x) = \bar{M}_n$ and $\underline{V}_n^0(x) = \underline{M}_n$. For all k, set $x_{n_0}^{k,0}$ as $x_{n_0}^0$. Finally, set $k = 1$.

Iteration k.

Step 1. Set $l = 0$ and $n = n_0$.

Step 2. Solve

$$\begin{aligned} \bar{\theta}_n^{k,l} = \min_{x_m^{l+1}, u_n^l} \quad & C_n(x_n^{k,l}, u_n^l) + \delta \sum_{m\in R(n)} p_{n,m} \bar{V}_m^{k-1}(x_m^{l+1}) \\ \text{s.t.} \quad & x_m^{l+1} = f_{n,m}(x^{k,l}, u^l),\ \forall m \in R(n), \\ & x_m^{l+1} \in \mathcal{X}_m,\ \forall m \in R(n), \\ & u_n^l \in \mathcal{U}_n(x_n^{k,l}). \end{aligned}$$

Step 3. Solve

$$\begin{aligned}\underline{\theta}_n^{k,l} = \min_{x_m^{l+1}, u_n^l} \quad & C_n(x_n^{k,l}, u_n^l) + \delta \sum_{m \in R(n)} p_{n,m} \underline{V}_m^{k-1}(x_m^{l+1}) \\ \text{s.t.} \quad & x_m^{l+1} = f_{n,m}(x_n^{k,l}, u_n^l),\ \forall m \in R(n), \\ & x_m^{l+1} \in \mathcal{X}_m,\ \forall m \in R(n), \\ & u_n^l \in \mathcal{U}_n(x_n^{k,l}),\end{aligned}$$

storing the control minimiser as $u_n^{k,l}$. Compute a sub-gradient $\beta_n^{k,l}$ with respect to $x_n^{k,l}$.

Step 4. Update $\Phi_n^{k,l}$ as

$$\Phi_n^{k,l} = \arg\max_{m \in R(n)} \ p_{n,m}(\bar{V}_m^{k-1}(f_{n,m}(x_n^{k,l}, u_n^{k,l})) - \underline{V}_m^{k-1}(f_{n,m}(x_n^{k,l}, u_n^{k,l}))). \tag{15}$$

Further, set $x_{\Phi_n^{k,l}}^{k,l} = f_{n,m}(x_n^{k,l}, u_n^{k,l})$.

Step 5. If $l = L(k)$, move to **Step 6**. Otherwise move to **Step 2** with $l = l + 1$ and $n = \Phi_n^{k,l}$.

Step 6. For all $n \in \mathcal{N}$, update the upper bound function as

$$\begin{aligned}\bar{V}_n^k(x) = \max_{\mu \in \mathbb{R}, \lambda \in \mathbb{R}^n} \quad & \mu + \langle \lambda, x \rangle \\ \text{s.t.} \quad & \mu + \langle \lambda, x_n^{k',l} \rangle \leq \bar{\theta}_n^{k',l},\ \forall l \leq L(k'),\ \forall k' \leq k, \\ & \|\lambda\|_* \leq \alpha.\end{aligned}$$

Step 6. For all $n \in \mathcal{N}$, update the lower bound functions as

$$\begin{aligned}\underline{V}_n^k(x) = \min_{\mu \in \mathbb{R}} \quad & \mu \\ \text{s.t.} \quad & \mu \geq \underline{\theta}_n^{k',l} + \langle \beta_n^{k',l}, x - x_n^{k',l} \rangle,\ \forall l \leq L(k'),\ \forall k' \leq k.\end{aligned}$$

Step 7. If a particular object has not recieved an update in this iteration, give that object the null update i.e. $x_n^{k,l} = x_n^{k-1,l}$. Move on to iteration $k + 1$. ⋈

Theorem 3. *Consider the sequence of functions $\bar{V}_n^k, \underline{V}_n^k$, state trajectories $x_n^{k,l}$, associated controls $u_n^{k,l}$, problem-children $\Phi_n^{k,l}$, and function $L : \mathbb{N} \mapsto \mathbb{N}$. If $\limsup_{k \in \mathbb{N} \to \infty} L(k) = \infty$, then*

$$\lim_{k \to \infty} \left(\bar{V}_n^k(x_n^{k,l}) - \underline{V}_n^k(x_n^{k,l}) \right) = 0, \quad \forall l \in \mathbb{N},\ \forall n \in \mathcal{N}.$$

Proof. By the arguments of Lemma 4, from our algorithm we can construct the following inequalities:

$$\underline{V}_n^{k_0}(x_n^{k_0,l}) \geq C_n(x_n^{k_0,l}, u_n^{k_0,l}) + \delta \sum_{m \in R(n)} p_{n,m} \underline{V}_m^{k_0-1}(f_{n,m}(x_n^{k_0,l}, u_n^{k_0,l})), \quad \forall k_0,\ \forall l \leq L(k_0),\ \forall n \in \mathcal{N},$$

and

$$\bar{V}_n^{k_0}(x_n^{k_0,l}) \leq C_n(x_n^{k_0,l}, u_n^{k_0,l}) + \delta \sum_{m \in R(n)} p_{n,m} \bar{V}_m^{k_0-1}(f_{n,m}(x_n^{k_0,l}, u_n^{k_0,l})), \quad \forall k_0,\ \forall l \leq L(k_0),\ \forall n \in \mathcal{N}.$$

So

$$\bar{\underline{V}}_n^{k_0}(x^{k_0,l}) \leq \delta \sum_{m \in R(n)} p_{n,m} \bar{\underline{V}}_m^{k_0-1}(f_{n,m}(x_n^{k_0,l}, u_n^{k_0,l})), \quad \forall k_0, \forall l \leq L(k_0),\ \forall n \in \mathcal{N}.$$

By the problem-child criterion (15), we have

$$\bar{\underline{V}}_n^{k_0}(x^{k_0,l}) \leq |R(n)| \delta p_{n,\phi_n^{k_0,l}} \bar{\underline{V}}_{\phi_n^{k_0,l}}^{k_0-1}(x_{\phi_n^{k_0,l}}^{k_0,l+1}), \quad \forall k_0,\ \forall l \leq L(k_0),\ \forall n \in \mathcal{N}.$$

By the arguments in Theorem 2, and the finiteness of $R(n)$ argument employed in Lemma 4.2 in Baucke et al. [1], we have that such an equality gives this theorem's result. □

Note here that our algorithm doesn't require the use of random sampling; the problem-child criterion requires the algorithm to look ahead to the vertex which contains the largest bound gap.

5. Discussion and conclusion

A closely related problem to the convex problems studied in this work are the minimax dynamic programmes studied in [1]. Rather than iteratively refining convex bounding functions, they develop piecewise bilinear saddle bounding functions and show how a certain iterative procedure can yield convergence in the finite horizon case. By adapting their formulation and algorithm into the infinite-horizon setting, we claim that arguments of Theorem 2 apply in almost the same way.

In summary, we have shown that under a small number of mild assumptions on the form of (13) and the look-ahead function $L : \mathbb{N} \mapsto \mathbb{N}$, the Markov infinite horizon algorithm converges deterministically to the optimal solution $V_{n_0}(x_{n_0}^0)$.

References

[1] Baucke, R., Downward, A., and Zakeri, G. (2018). A deterministic algorithm for solving stochastic minimax dynamic programmes. *Optimization Online*.

[2] Bellman, R. (1954). The theory of dynamic programming.

[3] Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252.

[4] Birge, J. R. (1985). Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research*, 33(5):989–1007.

[5] Chen, Z. and Powell, W. (1999). Convergent Cutting-Plane and Partial-Sampling Algorithm for Multistage Stochastic Linear Programs with Recourse 1. *Optimization*, 102(3):497–524.

[6] Girardeau, P., Leclere, V., and Philpott, A. (2014). On the Convergence of Decomposition Methods for Multistage Stochastic Convex Programs. *Mathematics of Operations Research*, 40(1):130–145.

[7] Kelley, J. (1960). The cutting-plane method. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712.

[8] Nannicini, G., Traversi, E., and Calvo, R. W. (2017). A Benders squared (B2) framework for infinite-horizon stochastic linear programs. *Optimization Online*, (Umr 7538).

[9] Pereira, M. V. F. and Pinto, L. M. V. G. (1991). Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52(1-3):359–375.

[10] Philpott, A. and Guan, Z. (2008). On the convergence of stochastic dual dynamic programming and related methods. *Operations Research Letters*, 36(4):450–455.