A deterministic algorithm for stochastic minimax dynamic programmes

Regan Baucke[a,b,*], Anthony Downward[a], Golbon Zakeri[a]

[a] *Department of Engineering Science, University of Auckland, New Zealand*
[b] *CERMICS Laboratory, Ecole des Ponts ParisTech, Champs-sur-Marne, France*

Abstract

In this paper, we present an algorithm for solving stochastic minimax dynamic programmes where state and action sets are convex and compact. A feature of the formulations studied is the simultaneous non-rectangularity of both 'min' and 'max' feasibility sets. We begin by presenting upper and lower bound representations of saddle functions which can be formulated as convex programmes. These bounding functions extend outer representations from traditional convex cutting plane algorithms. Our algorithm is similar in spirit to existing stochastic dual dynamic programming (SDDP) type algorithms; bounding functions are iteratively updated in order to compute cost-to-go functions. However, special consideration must be taken in ensuring the validity of the cost-to-go bounding functions (now saddle functions) over the domain. We apply the theory developed in this paper to multistage risk-averse optimisation. Through the dual representation of coherent risk measures, we represent risk-averse stochastic optimisation problems as minimax stochastic dynamic programmes, and provide two formulations for these problems: the *nested risk* formulation, and the *end-of-horizon* formulation. Finally, we demonstrate our algorithm on a risk-averse portfolio optimisation problem; providing numerical validation and a discussion.

Keywords: dynamic programming, decomposition, multistage, minimax, stochastic programming, risk aversion

1. Introduction.

Since the seminal work of Pereira and Pinto [22], multistage stochastic programmes and their decomposition has been studied extensively by researchers in stochastic programming. These problems have many applications in energy planning and scheduling; the hydro-thermal scheduling problem has become a classic in the field. However, a valid criticism of these initial stochastic programming models is the focus minimizing *expected* cost. This criticism is the premise for risk-averse stochastic optimisation and sister concept robust optimisation – rather than simply minimizing expected cost, improvements in worst-case outcomes can be traded off against outcomes in expectation.

*Corresponding author
Email address: regan.baucke@enpc.fr (Regan Baucke)

The theory of coherent risk measures developed by Artzner et al. [1] has given stochastic programmers a clear and consistent framework for incorporating risk aversion into objective functions. This theory has been developed and formalised by several authors, most notably in Ruszczyński and Shapiro [32], but more recently, this theory has been extended to the multistage setting. The authors believe that Ruszczyński and Shapiro [31] and Artzner et al. [2] are important works in this area. If these multistage risk measures have a conditional-coherency property, then a duality result applies in their representation – that is, risk is the maximum value of an expectation over a set of conditional probability measures, often referred to as risk sets. A brief introduction to these concepts is provided in Shapiro [33]. Using the risk set concept, it is possible to represent a multistage stochastic optimisation problem with dynamic risk measures as a multistage robust optimisation problem or minimax problem.

The class of problems studied in this paper is similar to those studied in Ruszczyński [30] and Shapiro [34]. However, a subtle but crucial difference exists between the earlier class and the problems considered in this work. Lest we prematurely introduce notation, a generic representation of the minimax dynamic programme studied previously can be formulated as

$$V_n(x_n) = \min_{u \in \mathcal{U}(x_n)} \max_{\mathbb{P} \in \mathcal{Q}} C(x_n, u_n) + \mathbb{E}_{\mathbb{P}}[V_m(x_m)],$$

where $V_n(x_n)$ is a value function (or equivalently cost-to-go function) for a given vertex in a scenario tree, which represents the stochasticity of the problem. Notice the 'maximisation' control $\mathbb{P}$'s feasible set $\mathcal{Q}$ has no dependence on a 'state'. The algorithms which solve these problems currently are based upon SDDP techniques with a slight modification when constructing the cutting plane which forms a lower bound on the cost-to-go functions. Briefly, when computing the average cut on an update phase of the algorithm, the conditional probabilities of the child vertices are re-weighted according to the solution of an auxiliary problem, – children with worse outcomes are seen as more probable. The approaches laid out in Shapiro et al. [35] and Philpott and De Matos [24] give an introduction to solving these problems.

Our formulation allows variables from both sets to depend on a 'state' and the expectation is taken with respect to some base measure as in

$$G_n(x_n, y_n) = \min_{u \in \mathcal{U}(x_n)} \max_{v \in \mathcal{V}(y_n)} C(x_n, y_n, u_n, v_n) + \mathbb{E}[G_m(x_m, y_m)].$$

While the rectangular problems studied in previous literature can be cast in our framework, in general, problems of the above form are considerably more difficult and require a new algorithm in order to solve them. This problem class encapsulates those studied in several fields, in particular: robust optimisation, risk-averse stochastic programming and zero-sum sub-game perfect equilibria. However, throughout this paper we will provide a perspective from risk-averse optimisation.

We present a new algorithm to solve these minimax stochastic dynamic programmes. The algorithm is similar to many multistage stochastic cutting plane methods – our algorithm can be seen as a generalisation from convex formulations to saddle formulations. We study saddle functions closely, and develop piecewise bilinear upper and lower approximations. We then show how, under a certain iterative procedure, these approximations can be refined in order to compute a saddle point. Another feature of our algorithm is the ability to achieve deterministic convergence, even in the case where our

formulation is stochastic. This is achieved by making use of the both the lower and upper bounds, and during the sampling phase of our procedure, choosing the next realisation of data that gives the greatest potential bound closure for the given state. The work of Georghiou et al. [12] utilise this idea in their algorithm, however their scenario selection problem is NP-hard, and requires the solution to a mixed-integer programming problem. Our scenario selection procedure (called the problem-child selection step) is a polynomial time procedure in the number of realisation of data in the next stage of the problem (or equivalently, the cardinality of the set of children vertices in the scenario tree).

In this work, we will also address another concern shared among the stochastic programming community: the interpretation of the nested conditional risk measure as an objective function. Indeed, this multistage risk measure may not reflect the modeller's actual risk aversion. It is argued that a more natural risk-averse formulation is simply a 'one-time' risk measure on the accumulated costs over the whole probability space. We call these the *end-of-horizon* risk measures. Such risk measures are discussed in Pflug and Pichler [23] and Asamov and Ruszczyński [3] and are charged with being 'time-inconsistent'. Furthermore, Pflug and Pichler provide a set of dynamic programming equations which give the aforementioned optimal 'one-time' risk adjusted value of the accumulated costs. We argue that labelling these problems as 'time-inconsistent' may be premature, since the existence of dynamic programming equations provide such risk measures with some sense of time-consistency. We present our own dynamic programming formulation, compatible with our algorithm, which solves these problems exactly.

We highlight the following as the main contributions of this paper:

- The construction of piecewise bilinear convex programmes that yield lower and upper bounding functions of saddle functions. These functions are novel and provide the key to our proposed algorithm;
- The description of a cutting-plane type algorithm to solve a general class of minimax dynamic programmes; we also prove its convergence;
- The development of minimax dynamic programming formulations that fall within our general class which coincide with popular formulations of risk-averse stochastic dynamic programmes. We also develop a minimax dynamic programming formulation which computes a minimum end-of-horizon risk-adjusted accumulation of costs.

The paper is structured as follows: Section 2 presents some preliminary concepts required for the construction of our bounding functions and the description of our algorithm. In Section 3, we develop our saddle bounding functions – we use our constructions on simplified version of the general multistage saddle problem to build intuition. In Section 4, we extend this algorithm to multistage problems, first deterministic and then stochastic. Section 5 presents dynamic programming formulations for different risk-averse optimisation problems. In Section 6 we apply our algorithm on a portfolio optimisation problem using end-of-horizon risk measures. Section 7 provides a discussion and concluding remarks.

2. Saddle functions.

In this section, we discuss several preliminary concepts which are necessary for the description of our algorithm. To keep this paper self-contained, we will recall the definition of a saddle function and saddle points, and discuss an extended Lipschitz-continuity concept on the Cartesian product of two vector spaces. We shall postpone any discussion on the relationship between these concepts and risk aversion until Section 4.

We begin by reviewing definitions and notation of various elementary concepts in order to keep this work self contained.

Definition 2.1. Given sets $\mathcal{U} \subset \mathbb{R}^n$ and $\mathcal{V} \subset \mathbb{R}^m$, a function $g : \mathcal{U} \times \mathcal{V} \mapsto \mathbb{R}$ is a saddle function if $g(\cdot, v)$ is a convex function for all $v \in \mathcal{V}$ and $g(u, \cdot)$ is a concave function for all $u \in \mathcal{U}$.

Definition 2.2. A point $(\dot{u}, \dot{v}) \in \mathcal{U} \times \mathcal{V}$ is a saddle point of a saddle function $g : \mathcal{U} \times \mathcal{V} \mapsto \mathbb{R}$ if

$$g(\dot{u}, v) \leq g(\dot{u}, \dot{v}) \leq g(u, \dot{v}),\ \forall (u, v) \in \mathcal{U} \times \mathcal{V}.$$

Let $\mathcal{U}$ and $\mathcal{V}$ now be compact and convex subsets of their respective Euclidean spaces. Consider the functions $g_v(u) = \max_{v \in \mathcal{V}} g(u, v)$ and $g_u(v) = \min_{u \in \mathcal{U}} g(u, v)$; their existence is guaranteed because g is continuous (being a finite-valued saddle function) and $\mathcal{U}$ and $\mathcal{V}$ are both compact. It is easy to see that $g_v(u)$ is a convex function (and symmetrically, $g_u(v)$ is a concave function). These functions are useful in the next lemma.

Lemma 2.1. *If $g : \mathcal{U} \times \mathcal{V} \mapsto \mathbb{R}$ is a saddle function, as given in Definition 2.1 and $\mathcal{U}$ and $\mathcal{V}$ are both convex and compact, then the set of saddle points is given by*

$$\mathcal{G} = \left\{ \left\{ \arg\min_u g_v(u) \times \mathcal{V} \right\} \cap \left\{ \arg\max_v g_u(v) \times \mathcal{U} \right\} \right\} \subseteq \mathcal{U} \times \mathcal{V}.$$

Proof. From the minimax theorem of von Neumann [36], a point $(\dot{u}, \dot{v})$ is a saddle point if and only if $g_v(\dot{u}) = g(\dot{u}, \dot{v}) = g_u(\dot{v})$. From the definition of g_v and g_u, we have $g_v(u) \geq g_u(v),\ \forall u \in \mathcal{U}, \forall v \in \mathcal{V}$. So

$$g_v(u) \geq g_v(\dot{u}) = g_u(\dot{v}) \geq g_u(v),\ \forall u \in \mathcal{U}, \forall v \in \mathcal{V}.$$

If $g_v(u) \geq g_v(\dot{u}),\ \forall u \in \mathcal{U}$, then $\dot{u}$ must be a minimiser of g_v (and symmetrically $\dot{v}$ a maximiser for g_u), satisfying the condition for its inclusion in $\mathcal{G}$. This completes the proof. □

Following from Lemma 2.1, it is easy to see that $\mathcal{G}$ is a convex set (being the intersection of two convex sets), and that all elements of $\mathcal{G}$ attain the same value under g. So far we have established that the set of saddle points of our function must be convex, and all saddle points have the same function value. We will now introduce the concept of an *ϵ-saddle point*. These points extend the idea of a saddle point and will aid in describing the convergent properties of the proposed algorithm.

Definition 2.3. A point (u, v) is an ϵ-saddle point of a function $g : \mathcal{U} \times \mathcal{V} \mapsto \mathbb{R}$ if $g_v(u) - \epsilon \leq g(\dot{u}, \dot{v}) \leq g_u(v) + \epsilon,\ \epsilon \geq 0$, where $(\dot{u}, \dot{v})$ is a saddle point. Denote the set of ϵ-saddle points as

$$\mathcal{G}_g(\epsilon) = \{(u, v) \in \mathcal{U} \times \mathcal{V} \mid g_v(u) - \epsilon \leq g(\dot{u}, \dot{v}) \leq g_u(v) + \epsilon\}.$$

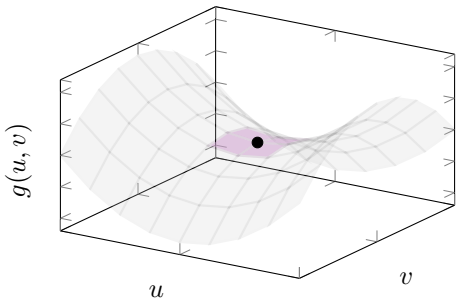


Figure 1: A saddle function $g(u, v)$ with $\mathcal{G}$ (a singleton in this instance) as the mark and $\mathcal{G}(\epsilon)$, $\epsilon > 0$ as the shaded region.

It immediately follows that $\mathcal{G}(0) = \mathcal{G}$; that is, the most strict set of ϵ-saddle points is the set of 'true' saddle points $\mathcal{G}$. As ϵ increases, the qualification of inclusion is relaxed so we obtain the property that $\epsilon_1 \leq \epsilon_2 \iff \mathcal{G}(\epsilon_1) \subseteq \mathcal{G}(\epsilon_2)$. Similar to $\mathcal{G}(0)$, it is easy to see that $\mathcal{G}(\epsilon)$ is a convex set for all $\epsilon \geq 0$. Figure 1 provides a graphical representation of a saddle function and associated sets of saddle points.

We will now turn our attention to the subgradient properties of saddle functions. For any (including non-smooth) saddle function $g : \mathcal{U} \subset \mathbb{R}^n \times \mathcal{V} \subset \mathbb{R}^m \mapsto \mathbb{R}$, we recall and provide a notational extension of concepts of subgradients from convex analysis. We remind the reader that $g(u, v)$ is convex in u, for a fixed v. Recall the definition of the subdifferential set of a convex function $f(u)$ is given by

$$\partial f(u) := \left\{ d^u \in \mathbb{R}^n \mid f(u') \geq f(u) + \langle d^u, u' - u \rangle, \ \forall u' \in \mathcal{U} \right\}.$$

We extend this notion to allow this multifunction to be defined over the Cartesian product of two vector spaces i.e. $\mathcal{U} \times \mathcal{V}$. Our notation is given by

$$\partial^u g(u, v) := \left\{ d^u \in \mathbb{R}^n \mid g(u', v) \geq g(u, v) + \langle d^u, u' - u \rangle, \ \forall u' \in \mathcal{U} \right\}.$$

This multifunction is a mapping from a point (u, v) in $\mathcal{U} \times \mathcal{V}$ to the set of subgradients of the convex function $g(\cdot, v) : \mathcal{U} \subset \mathbb{R}^n \mapsto \mathbb{R}$ at u. Because $g(\cdot, v)$ is convex in u we retain all the usual properties of subgradients for purely convex functions such as convexity and compactness of $\partial^u g(u, v)$ over $\mathcal{U}$. Of course, the symmetric mapping exists in the vector space where $g(u, \cdot)$ is concave as

$$\partial^v g(u, v) := \left\{ d^v \in \mathbb{R}^m \mid g(u, v') \leq g(u, v) + \langle d^v, v' - v \rangle, \ \forall v' \in \mathcal{V} \right\}.$$

We now present and extend the notion of Lipschitz-continuity of saddle functions – the convergence of our proposed algorithm relies on certain Lipschitz-continuity features. First we present the usual definition of Lipschitz-continuity, and then extend in a natural way to allow for different Lipschitz constants in their respective vector spaces.

Definition 2.4. Recall that a Lipschitz saddle function is a saddle function $g(u, v)$ such that there exists some $\alpha \in \mathbb{R}$ for which

$$|g(u_1, v_1) - g(u_2, v_2)| \leq \alpha ||(u_1, v_1) - (u_2, v_2)||, \quad \forall (u_1, v_1), (u_2, v_2) \in \mathcal{U} \times \mathcal{V}. \tag{2.1}$$

We say that the saddle function is α-Lipschitz if the definition above holds for a particular value α.

Definition 2.5. A (α_u, α_v)-Lipschitz saddle function is a saddle function $g(u, v)$ for which there exists some $\alpha_u, \alpha_v \in \mathbb{R}$ where

$$|g(u_1, v) - g(u_2, v)| \leq \alpha_u ||(u_1, v) - (u_2, v)||, \quad \forall u_1, u_2 \in \mathcal{U}, \forall v \in \mathcal{V}, \tag{2.2}$$

$$|g(u, v_1) - g(u, v_2)| \leq \alpha_v ||(u, v_1) - (u, v_2)||, \quad \forall u \in \mathcal{U}, \forall v_1, v_2 \in \mathcal{V}. \tag{2.3}$$

We say a function is (α_u, α_v)-Lipschitz if the above definition holds for the two values α_u and α_v in their respective spaces.

Remark 2.1. We remark here that is it can be shown that these two Lipschitz properties are equivalent; that is, a saddle function satisfying Definition 2.4 also satisfies Definition 2.5 (albeit for different values of their associated constants) and vice-versa. Nevertheless, it is useful to make a distinction between the particular constants α, α_u and α_v relating to their corresponding vector spaces.

3. Bounding functions.

Bounding functions form the basis of many iterative algorithms in dynamic programming where the state and control spaces are uncountably infinite. Early work in dynamic programming e.g. Bellman [4], dealt with dynamic programmes where state and action spaces were finite; backward recursion could then be employed in order to construct exact value functions. In our 'infinite-state' case, it is impossible in general to construct value functions which are exact over their entire domain. A more realistic goal perhaps would be to generate approximate bounding functions which can be refined in certain areas of interest. This idea has its roots in the work of Kelley [16] and Benders [5].

In this section, we review bounding functions for convex functions with the aim of leveraging their properties in the development of saddle function bounds. We assume the standard 'black-box oracle' model of Nesterov [21] when constructing our bounding functions; we are able to request zeroth-order (point) and first-order (subgradient) information of our function at any point in our domain. We call the set of requested points S *sample points*. If our ensuing algorithm is to be successful, we submit that bounding functions in general should have the following properties: that they are valid; that they are equal at a sample point; that additional sample points do not worsen bounds anywhere, that the bounding functions can evaluated 'easily'; and that if they bound a Lipschitz-continuous function then they themselves should be Lipschitz-continuous. Although we have described these conditions are qualitatively here, we will make these concepts mathematically precise on these concepts through this section.

In order to motivate this section, we present the following saddle point optimisation where the bounding functions we will develop become useful: for a given saddle function $g(u, v)$ and tolerance ϵ, we wish to compute a point $(\dot{u}, \dot{v})$ such that

$$(\dot{u}, \dot{v}) \in \mathcal{G}_g(\epsilon) \subseteq \mathcal{U} \times \mathcal{V}, \tag{3.1}$$

where $\mathcal{U} \subset \mathbb{R}^n$ and $\mathcal{V} \subset \mathbb{R}^m$ are both convex and compact. The set $\mathcal{G}_g(\epsilon)$ is the set of ϵ-saddle points for the saddle function $g(u, v)$. Further, we require that the saddle function is Lipschitz-continuous on $\mathcal{U} \times \mathcal{V}$.

3.1. Bounds for convex functions.

Before we develop our bounding functions for saddle functions, we will first review and develop our notation for bounding functions for the convex case. The lower bound in particular will be familiar to many readers. Consider an α-Lipschitz continuous convex function $C : \mathcal{U} \mapsto \mathbb{R}$, where $\mathcal{U}$ is a compact subset of $\mathbb{R}^n$. We are interested in forming bounding functions which use zero-order and first order information from a finite and non-empty set of sample points $S \subset \mathcal{U}$. We denote these ensuing upper and lower bounding functions $\bar{C}$ and $\underline{C}$ respectively.

Definition 3.1. For a finite, non-empty set of sample points S, let

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

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

The above can be written be written in *epigraph* form as

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

which can be solved as a linear programme. This lower bound function is a key part of many decomposition algorithms and dates back to Kelley [16]. Note that if instead $C(u)$ is a concave function, then by a simple revision of Definition 3.1, we obtain an upper bound of the concave function. Complimentary to the lower bound function, we present the following definition of an upper bound convex function.

Definition 3.2. For a finite, non-empty set of sample points S, let

$$\begin{aligned} \bar{C}^S(u) = \max_{\mu \in \mathbb{R}, \lambda \in \mathbb{R}^n} \quad & \mu + \langle \lambda, u \rangle \\ \text{s.t.} \quad & \mu + \langle \lambda, u_s \rangle \leq C(u_s), \; \forall s \in S, \\ & ||\lambda||_* \leq \alpha. \end{aligned} \tag{3.2}$$

An attractive property of this upper bound function is that it is the optimal value of a convex optimisation problem. Furthermore, when the norm considered is the $||\cdot||_\infty$-norm, the resulting optimisation problem is linear. The feasible region represents the coefficients of all hyperplanes in $\mathbb{R}^{n+1}$ which support all the sample points $C(u_s)$ and have a bounded gradient. Upper bounds similar to this have been discussed in several previous works. We relate our upper to Philpott et al. [25] where they consider a set of sample points in $\mathcal{U}$ and form their bound by taking the minimal convex combination of sample points as in

$$\begin{aligned} \bar{C}^S(u) = \min_{\sigma_s} \quad & \sum_{s \in S} C(u_s)\sigma_s \\ \text{s.t.} \quad & \sum_{s \in S} \sigma_s = 1, \\ & \sum_{s \in S} \sigma_s u_s = u, \\ & \sigma_s \geq 0, \; \forall s \in S. \end{aligned} \tag{3.3}$$

One of the draw backs of this upper bound function is that its effective domain is limited to the convex hull of S. If $u \notin \text{Conv}(S)$, then the above linear programme is infeasible i.e. $\bar{C}(u) = \infty$. So if $\text{Conv}(S) \neq \mathcal{U}$, then $\bar{C}(u)$ is not Lipschitz on $\mathcal{U}$; a property required by our ensuing algorithm. This idea can be traced back to Madansky [18] where upper bounds on convex functions are also sought. Madansky correctly notes that in order to attain a bound where $\bar{C}^S(u) < \infty$ over the entire domain, the domain needs to be closed and bounded, and all extreme points need to be included in S. Even in the best case of a polyhedral domain (where the number of extreme points is bounded), the number of corner points grows exponentially in the dimensionality of $\mathcal{U}$. We circumvent this obstacle by making use of the Lipschitz-continuity of our function of concern C. We arrive at Definition 3.2 by taking the linear programming dual of 3.3 and further constraining the gradient coefficients λ. Our upper bound definition has our desired Lipschitz continuity property directly encoded within its convex programming formulation. We will delay any proof of the claims that our lower and upper bound functions have our aforementioned desired properties until we consider special cases of our saddle bounding functions.

3.2. Bounds for saddle functions.

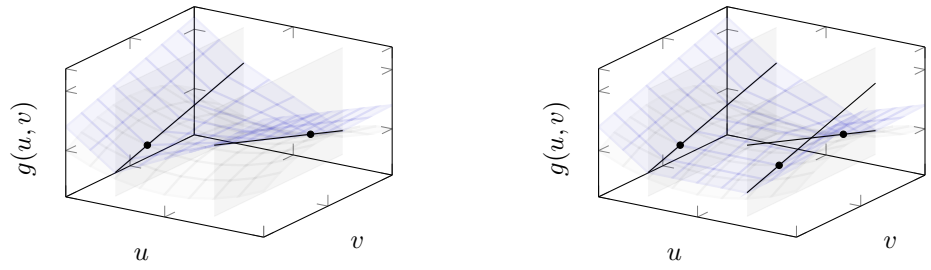
This section builds upon the upper and lower bounds on convex to saddle functions. The general idea is to develop a convex programming formulation which borrows aspects of the two convex bounding functions discussed earlier. The domain of interest here is the Cartesian product $\mathcal{U} \times \mathcal{V}$, so the set of sample points S contains ordered pairs (u_s, v_s).

Definition 3.3. Consider a saddle function $g : \mathcal{U} \times \mathcal{V} \mapsto \mathbb{R}$. For a finite, non-empty set of sample points S let

$$\begin{aligned} \bar{G}^S(u,v) = \max_{\mu,\lambda} \quad & \mu + \langle \lambda, u \rangle \\ \text{s.t.} \quad & \mu + \langle \lambda, u_s \rangle \leq g(u_s, v_s) + \langle d_s^v, v - v_s \rangle, \ \forall s \in S, \\ & ||\lambda||_q \leq \alpha_u. \end{aligned} \tag{3.4}$$

Let us first develop an intuition of the upper bound function $\bar{G}^S$ before laying out its properties. Consider Definition 3.3 when there is no v dependence. Then problem (3.4) reduces to (3.2). Alternatively, when the u dependence is dropped, we obtain the concave version of the epigraph formulation (3.1), which is an upper bound. The optimisation problem defining $\bar{G}^S(u,v)$ utilises both upper bound features from the convex/concave bounding functions in the appropriate vector spaces. Note that $\bar{G}^S(u,v)$ is a saddle function; it is the optimal value of a convex programme (which is maximizing) with an affine function of v in its right-hand side and an affine function of u in the objective function. Furthermore, in the case of the $||\cdot||_\infty$-norm, where a linear program is rendered, $\bar{G}^S(u,v)$ is a piecewise bilinear function of u and v. We formalise this in the following lemma.

Lemma 3.1. *Consider the function $\bar{G}^S(u,v)$ where the norm considered is the $||\cdot||_\infty$-norm. Further, assume that the defining linear programme has an optimal solution over the domain of $\bar{G}^S(u,v)$. Then $\bar{G}^S(u,v)$ is a piecewise bilinear form over its domain. Furthermore $\bar{G}^S(u,v)$ is convex in v and concave in u.*



(a) $|S_1| = 2$ (b) $|S_2| = 3$ and $S_1 \subset S_2$

Figure 2: For the saddle function $g(u, v)$ in grey, the blue surfaces are the upper bounding function $\bar{G}(u, v)$.

Proof. It is well known that, in any parametric linear program of the form $\bar{G}^S$, the domain of $\bar{G}^S$ may be divided into a finite number of so-called critical regions, indexed by $i \in \{1, \cdots, I\}$, characterised by the various combinations of variables forming optimal bases, with the property that $\bar{G}^S$ is continuous on each critical region separately. The specific form above also constitutes that the critical regions are closed (see [19] and references therein, particularly [10]). Let us denote one such basis corresponding critical region i as B_i, for $i \in \{1, \cdots, I\}$, and note that while these may not be unique, the corresponding optimal value is unique. The value of $\bar{G}^S$ on region i is then given by $c^T {B_i}^{-1} b$ where c is the vector of cost coefficients (containing u) and b is the right-hand side vector (containing v). It immediately follows that $\bar{G}^S$ is a piecewise bilinear form. Convexity of $\bar{G}^S$ in v relies on the convexity of a linear program's optimal objective as function of its right-hand side; a well known result utilised widely in stochastic programming (see e.g. [7]) and the concavity in u is a simple analogue. □

Further intuition of the upper bound function in the univariate case can be obtained by studying Figure 2. We proceed to prove several properties of $\bar{G}^S$ which are utilised in the convergence proofs for the proposed saddle point algorithm.

Lemma 3.2. *The optimisation problem defining $\bar{G}^S(u, v)$ for any non-empty S is bounded and feasible.*

Proof. The point in (μ, λ) space

$$\mu = \min_{s \in S} G(u_s, v_s) + \langle d_s^v, v - v_s \rangle, \text{ and } \lambda = \mathbf{0},$$

is always feasible for any S. Secondly, the vector λ is bounded above and below, while μ is bounded from above in a maximisation problem, so clearly the optimisation problem defining $\bar{G}^S(u, v)$ is bounded. □

Lemma 3.3. *If $g(u, v)$ is a saddle-function and (α_u, α_v)-Lipschitz on $\mathcal{U} \times \mathcal{V}$ under the $||\cdot||_p$-norm, then $\bar{G}^S(u, v)$ is (α_u, α_v)-Lipschitz.*

Proof. For all $v \in \mathcal{V}$, the function $\bar{G}^S(u, v)$ is clearly α_u-Lipschitz. For all u, the value of $|\bar{G}^S(u, v_1) - \bar{G}^S(u, v_2)|$ is bounded by

$$\max_{s \in S} \langle d_s^v, (v_1 - v_2) \rangle.$$

The function $g(u,\cdot)$ is α_v-Lipschitz, so its subgradients are uniformly bounded (under the $||\cdot||_q$-norm, where $\frac{1}{p}+\frac{1}{q}=1,\ \forall p,q\in(1,\infty)$). So $||d^v(u,v)||_q \leq \alpha_v,\ \forall (u,v)\in\mathcal{U}\times\mathcal{V}$. This gives the result. □

Lemma 3.4. *If $g(u,v)$ is a saddle-function and (α_u,α_v)-Lipschitz on $\mathcal{U}\times\mathcal{V}$ under the $||\cdot||_p$-norm, and $(u',v')\in S$, then $\bar{G}^S(u',v')=g(u',v')$.*

Proof. For a given S, the pair (μ,λ) in the feasible region of (3.4) represent the coefficients of a supporting hyperplane of the points $\big(u_s, G(u_s,v_s)+\langle d_s^v, v'-v_s\rangle\big),\ \forall s\in S$ where μ is the intercept and λ is the gradient of the hyperplane. Recall from Lemma 3.2 that the feasible region is always non-empty. Because the function $g(u,v)$ is convex in u and (α_u,α_v)-Lipschitz, there exists a supporting hyperplane for which

$$g(u',v')+\langle \tilde{d}^u, u-u'\rangle \leq g(u,v'),\ \forall u\in\mathcal{U},$$

where $||\tilde{d}^u||_q\leq\alpha_u$ and $\tilde{d}^u$ is in the subdifferential set of $g(u,v)$ at (u',v'). So a candidate solution for $\bar{G}^S(u',v')$ is $\tilde{\mu}=g(u',v')-\langle\tilde{d}^u,u'\rangle$ and $\tilde{\lambda}=\tilde{d}^u$ which gives $\bar{G}^S(u',v')=g(u',v')$. This solution is feasible because if the plane supports $(u_s,g(u_s,v')),\ \forall s\in S$, then by the concavity of $g(u,v)$ in v, it also supports $(u_s,G(u_s,v_s)+\langle d_s^v,v'-v_s\rangle),\ \forall s\in S$. If another candidate (μ,λ) gives $\mu+\langle\lambda,\hat{u}\rangle<g(\hat{u},\hat{v})$, then the pair is not optimal because of the existence of $(\tilde{\mu},\tilde{\lambda})$. If a further candidate (μ,λ) gives $\mu+\langle\lambda,u'\rangle>g(u',v')$, then the pair does not support the point $(u',G(u',v')+\langle d_s^v,v'-v'\rangle)$, making it infeasible. This completes the proof. □

Lemma 3.5. *If $S^1\subseteq S^2$, then $\bar{G}^{S^1}(u,v)\geq\bar{G}^{S^2}(u,v)$.*

Proof. The function $\bar{G}^S(u,v)$ is defined by a maximisation problem which is feasible and bounded. The relation $S^1\subseteq S^2$ implies that the feasibility set of S^2 is contained within S^1, giving the result. □

Lemma 3.6. *If $g(u,v)$ is a saddle function and α-Lipschitz on $\mathcal{U}\times\mathcal{V}$ under the $||\cdot||_p$-norm, then $\bar{G}^S(u,v)\geq g(u,v)$, for all $(u,v)\in\mathcal{U}\times\mathcal{V}$.*

Proof. For the sake of contradiction, suppose there exists a $(u',v')\in\mathcal{U}\times\mathcal{V}$ for which $\bar{G}^S(u',v')<g(u',v')$. By defining $S^*=S\cup(u',v')$, and by Lemma 3.4, we have $g(u',v')=\bar{G}^{S^*}(u',v')$. So $\bar{G}^S(u',v')<\bar{G}^{S^*}(u',v')$, which is precluded by Lemma 3.5, completing the proof. □

Remark 3.1. We note here that the lemmas established above hold symmetrically for the lower bounding function $\underline{G}^S(u,v)$. Further, by considering the convex function upper bound (as in Definition 3.2) as a special case of a saddle function upper bound i.e. with no dependence on v, we can see that $\bar{C}(u)$ does indeed have the properties claimed previously.

With these bounding functions introduced and their properties understood, we establish a pair of useful bounding results with respect the bounding functions' saddle points. We will then be in a position to solve the motivating problem of this section.

Lemma 3.7. *Under the conditions of Definition 3.3, we have*

$$\min_{u\in\mathcal{U}}\bar{G}^S(u,v)\geq\min_{u\in\mathcal{U}}g(u,v),\ \forall v\in\mathcal{V}.$$

Proof. Suppose the statement of this lemma is false, and therefore there exists a $v' \in \mathcal{V}$ for which $\min_{u\in\mathcal{U}} \bar{G}^S(u,v) < \min_{u\in\mathcal{U}} g(u,v)$. Now define u' as an arbitrary minimiser of $\bar{G}^S(u,v')$; this gives the following inequality:

$$\bar{G}^S(u',v') < \min_{u\in\mathcal{U}} g(u,v') \leq g(u',v').$$

This inequality is precluded by Lemma 3.6, completing the proof. □

Following directly from this lemma, we obtain the following corollary.

Corollary 3.1.

$$\max_{v\in\mathcal{V}} \min_{u\in\mathcal{U}} \bar{G}^S(u,v) \geq \max_{v\in\mathcal{V}} \min_{u\in\mathcal{U}} g(u,v).$$

Corollary 3.1 states the saddle point of the upper bound function always lies above the saddle point of $g(u,v)$, even if the saddle points are not coincident. We do not prove this here, but note that it can be easily proven by the arguments of Lemma 3.7.

We now present and prove the convergence of the procedure to compute the saddle point of a saddle function:

$$(\dot{u},\dot{v}) \in \mathcal{G}_g(\epsilon) \subseteq \mathcal{U}\times\mathcal{V}, \tag{3.5}$$

This can be considered the saddle point analogy of Kelly's Cutting Plane algorithm for convex functions; we evaluate our target function in several places and update bounding functions iteratively. Although novel, it is not the main result of the paper but serves as a useful introduction to the upper and lower bound functions, and gives the format for further proofs. We require the conditions set out in (3.1) on $g(u,v)$ and $\mathcal{U}$ and $\mathcal{V}$ for the following theorem.

Theorem 3.1. *Consider the sequences*

$$v^k = \arg\max_{v\in\mathcal{V}} \min_{u\in\mathcal{U}} \bar{G}^{S_{k-1}}(u,v), \tag{3.6}$$

$$u^k = \arg\min_{u\in\mathcal{U}} \max_{v\in\mathcal{V}} \underline{G}^{S_{k-1}}(u,v), \tag{3.7}$$

$$S_k = S_{k-1} \cup (u^k, v^k),$$

with

$$\bar{g}^k = \min_{u\in\mathcal{U}} \max_{v\in\mathcal{V}} \bar{G}^{S_{k-1}}(u,v),\ \ \underline{g}^k = \min_{u\in\mathcal{U}} \max_{v\in\mathcal{V}} \underline{G}^{S_{k-1}}(u,v). \tag{3.8}$$

We have that

$$\lim_{k\to\infty} \left(\bar{g}^k - \underline{g}^k\right) = 0.$$

To ease our notational burden, we will use $\bar{G}^k(u,v)$ as a shorthand for $\bar{G}^{S_k}(u,v)$. We proceed with the proof of Theorem 3.1.

Proof. From (3.6) and (3.8), we have

$$\bar{g}^k \leq \bar{G}^{k-1}(u,v^k),\ \forall k, \forall u \in \mathcal{U}.$$

For $\bar{g}^k$, from (3.8), we have

$$\underline{g}^k \geq \underline{G}^{k-1}(u^k,v),\ \forall k, \forall v \in \mathcal{V}.$$

It follows then that,

$$\bar{g}^k - \underline{g}^k \leq \bar{G}^{k-1}(u^k, v^k) - \underline{G}^{k-1}(u^k, v^k), \ \forall k. \tag{3.9}$$

We will now show that the right-hand side of (3.9) converges to zero as k tends to infinity – this will complete the proof as Corollary 3.1 bounds the difference of the saddle points of these functions from below at zero. Suppose there exist some $\epsilon > 0$ for which

$$\epsilon \leq \bar{G}^{k-1}(u^k, v^k) - \underline{G}^{k-1}(u^k, v^k), \ \forall k.$$

Subtracting $\bar{G}^{k-1}(u^{\hat{k}}, v^{\hat{k}}) - \underline{G}^{k-1}(u^{\hat{k}}, v^{\hat{k}})$ from both sides gives

$$\epsilon - \bar{G}^{k-1}(u^{\hat{k}}, v^{\hat{k}}) + \underline{G}^{k-1}(u^{\hat{k}}, v^{\hat{k}}) \leq$$
$$\bar{G}^{k-1}(u^k, v^k) - \underline{G}^{k-1}(u^k, v^k) - \bar{G}^{k-1}(u^{\hat{k}}, v^{\hat{k}}) + \underline{G}^{k-1}(u^{\hat{k}}, v^{\hat{k}}), \ \forall \hat{k}, k.$$

From Lemma 2.1, there exists a constant α for which both $\underline{G}^k$ and $\bar{G}^k$ are α-Lipschitz. So

$$\epsilon - \bar{G}^{k-1}(u^{\hat{k}}, v^{\hat{k}}) + \underline{G}^{k-1}(u^{\hat{k}}, v^{\hat{k}}) \leq 2\alpha ||(u^k, v^k) - (u^{\hat{k}}, v^{\hat{k}})||, \ \forall \hat{k}, k.$$

From Lemma 3.4, $\bar{G}^{k-1}(u^{\hat{k}}, v^{\hat{k}}) - \underline{G}^{k-1}(u^{\hat{k}}, v^{\hat{k}}) = 0, \ \forall \hat{k} < k$, so,

$$\frac{\epsilon}{2\alpha} \leq ||(u^k, v^k) - (u^{\hat{k}}, v^{\hat{k}})||, \ \forall \hat{k} < k, \forall k,$$

which is a contradiction of the compactness of $\mathcal{U} \times \mathcal{V}$, as (u^k, v^k) must contain a convergent subsequence. It must follow then, that no $\epsilon > 0$ exists and $\bar{g}^k - \underline{g}^k \to 0$ as $k \to \infty$. □

Remark 3.2. The convergence result above requires the successive computation of saddle points. From a computational perspective, it is important to consider the cost of making these saddle point evaluations. Fortunately, methods such as Newton's Method or barrier methods can easily be adapted for saddle functions at zero additional computational cost – the complexity of computing the saddle point of a saddle function is no more than computing the minimum of a convex function. That is to say, provided fairly relaxed regularity conditions, this can be achieved in polynomial time: see Nemirovski [20] as an example.

Unlike traditional optimisation, where simply a minimum or maximum is sought, in saddle point optimisation, the function $g(u, v)$ can take the value of the saddle point in several places without satisfying the saddle point conditions i.e. Definition 2.2. Convergence of the bounding functions is not sufficient to say that a saddle point has been located – we need to show that the algorithm converges to the saddle point value, and that some sequence of iterates converge toward a saddle point as in Definition 2.3. The following two lemmas give the desired result.

Lemma 3.8. *For $\epsilon \geq 0$, and the functions $g_v(u)$ and $g_u(v)$ defined earlier, we have*

$$\epsilon \geq g_v(u) - g_u(v) \implies (u, v) \in \mathcal{G}_g(\epsilon), \quad \forall (u, v) \in \mathcal{U} \times \mathcal{V}.$$

Proof. Restating Definition 2.3, we have

$$\mathcal{G}(\epsilon) = \{(u, v) \in \mathcal{U} \times \mathcal{V} \mid g_v(u) - \epsilon \leq g(\dot{u}, \dot{v}) \leq g_u(v) + \epsilon\}.$$

Substituting $\epsilon \geq g_v(u) - g_u(v)$ into the definition yields

$$g_u(v) \leq g(\dot{u}, \dot{v}) \leq g_v(u),$$

which is true for all $(u, v) \in \mathcal{U} \times \mathcal{V}$ because $g(u, v)$ is a saddle function. This completes the proof. □

Lemma 3.9. *Consider the sequence of functions $\bar{G}^k$ and $\underline{G}^k$ generated by the algorithm. Then*

$$\mathcal{G}_g^k := \left\{ \left\{ \arg\min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \bar{G}^{k-1}(u, v) \times \mathcal{V} \right\} \cap \left\{ \arg\max_{v \in \mathcal{V}} \min_{u \in \mathcal{U}} \underline{G}^{k-1}(u, v) \times \mathcal{U} \right\} \right\} \subseteq \mathcal{G}_g(\bar{g}^k - \underline{g}^k).$$

Proof. From Lemma 3.8, for a point (u, v) to be a member of $\mathcal{G}_g(\epsilon)$, we require that ϵ satisfies

$$\epsilon \geq g_v(u) - g_u(v).$$

From Corollary 3.1, we have that $\bar{g}^k \geq g_v(u'),\ \forall u' \in \arg\min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \bar{G}^{k-1}(u, v)$ and $\underline{g}^k \leq g_u(v'),\ \forall v' \in \arg\max_{v \in \mathcal{V}} \min_{u \in \mathcal{U}} \underline{G}^{k-1}(u, v)$. So clearly,

$$\bar{g}^k - \underline{g}^k \geq g_v(u) - g_u(v), \quad \forall (u, v) \in \mathcal{G}_g^k,$$

completing the proof. Further, from Lemma 3.1, we have that $\bar{g}^k - \underline{g}^k$ approaches 0, so $\mathcal{G}_g^k$ approaches the set of true saddle points. □

4. Proposed algorithm.

In this section, we extend the ideas in the previous section to the more complex setting of multistage minimax optimisation. As with many multistage methods, we seek to solve the problem by constructing upper and lower bounds to the value functions. A policy can then be computed by solving a *stage problem* at a given state which considers the current cost and a cost-to-go function, which encodes the cost of future optimal decisions. This was first seen in Birge [6] and has seen a lot of attention from researchers since. An apt name for the broader collection of these methods is 'nested Benders decomposition'. We borrow the exploration-update pattern seen in many nested Benders decomposition methods for our algorithm; we generate a state and control trajectory using our best approximations to the cost-to-go functions at hand. From this trajectory, we update our cost-to-go function using zeroth and first order information about the optimal solution of future stage problems. We show by induction that value function approximations (both upper and lower) converge in the limit at the state trajectories generated by the algorithm. We begin by studying the deterministic problem first, and then we move on to the stochastic version of the problem.

4.1. The deterministic case.

In general, a multistage optimisation problem is one whose solution are indexed by stage. Often these optimisation problems are coupled in their stages: the first stage's control may then impact the feasible decisions and cost function for the proceeding stages. The optimisation problem that we will study in this section has the following components:

- the number of stages T, inducing a a time horizon $\mathcal{T} := \{0, \dots, T\}$;
- the minimising state x_t and maximising state y_t for each stage t;
- the minimising control u_t and maximising state v_t for each stage $t < T$;
- the feasible set for the minimising control $\mathcal{U}_t(x_0)$ for each stage $t < T$;
- the feasible set for the maximising control $\mathcal{V}_t(y_0)$ for each stage $t < T$;
- the transition function for the next stage's minimising state $f_t^x(x_t, u_t)$ for each stage $t < T$;
- the transition function for the next stage's minimising state $f_t^y(y_t, v_t)$ for each stage $t < T$;
- a stage cost function $C_t(x_t, y_t, u_t, v_t)$ for each stage $t < T$;
- a terminal value function $G_T(x_T, y_T)$.

The goal of the optimisation problem is to compute the saddle point of the sum of the stage costs functions over the time horizon plus a terminal value function. Mathematically, we desire:

$$\begin{aligned} G_0(x_0, y_0) = \min_{x_t, u_t} \max_{y_t, v_t} \quad & \sum_{t=0}^{T-1} C_t(x_t, y_t, u_t, v_t) + G_T(x_T, y_T) \\ \text{s.t.} \quad & x_{t+1} = f_t^x(x_t, u_t),\ \forall t < T, \\ & y_{t+1} = f_t^y(y_t, v_t),\ \forall t < T, \\ & (u_t, v_t) \in \mathcal{U}_t(x_t) \times \mathcal{V}_t(y_t),\ \forall t < T, \\ & (x_t, y_t) \in \mathcal{X}_t \times \mathcal{Y}_t,\ \forall t \leq T. \end{aligned} \tag{4.1}$$

We require several technical conditions of the functions and multifunctions that make up the problem formulation. Briefly, these conditions ensure that the problem is 'convex' and admits subgradients; we refer the reader to Appendix B for their full description. It is important to note here that the feasibility region of (4.1) is disjoint in the minimisation variables x, u and maximisation variables y, v. We can write the above optimisation problem in its dynamic programming form as

$$\begin{aligned} G_t(x_t, y_t) = \min_{x_{t+1}, u_t} \max_{y_{t+1}, v_t} \quad & C_t(x_t, y_t, u_t, v_t) + G_{t+1}(x_{t+1}, y_{t+1}) \\ \text{s.t.} \quad & x_{t+1} = f_t^x(x_t, u_t), \\ & y_{t+1} = f_t^y(y_t, v_t), \\ & (x_{t+1}, y_{t+1}) \in \mathcal{X}_{t+1} \times \mathcal{Y}_{t+1}, \\ & (u_t, v_t) \in \mathcal{U}_t(x_t) \times \mathcal{V}_t(y_t), \end{aligned} \tag{4.2}$$

These dynamic programming equations can be thought of as a multistage simultaneous-play zero-sum game; when solved, the solution corresponds to a sub-game perfect equilibrium. By a similar analysis to that conducted in Girardeau et al. [13], value functions $G_t(x_t, y_t)$ defined as above are Lipschitz-continuous and saddle functions with respect to the state x_t, y_t for all $t < T$. These properties are crucial as they allow the value functions to be bounded correctly by our saddle bounding functions presented earlier. Based upon the saddle algorithm in the previous section, we outline our algorithm for solving (4.1) below.

Algorithm 4.1 (The multistage deterministic minimax algorithm)**.**

Initialisation. *Define* $\underline{G}_t^0 := -\infty, \bar{G}_t^0 := \infty, \ \forall\ t < T$. *Because the final value function is given, we set* $\underline{G}_T^k = G_T, \bar{G}_T^k = G_T, \ \forall k \in \mathbb{N}$. *Set the initial state* $(x_0^k, y_0^k) = (x_0, y_0), \ \forall k$. *Finally set* $k = 1$.

Iteration k**.**

Step 1. *Set* $t = 0$.

Step 2. *Solve*

$$\begin{aligned} \underline{\theta}_t^k = \min_{x_{t+1}, u_t} \max_{y_{t+1}, v_t} \quad & C_t(x_t^k, y_t^k, u_t, v_t) + \underline{G}_{t+1}^{k-1}(x_{t+1}, y_{t+1}) \\ \text{s.t.} \quad & x_{t+1} = f_t^x(x_t^k, u_t), \\ & y_{t+1} = f_t^y(y_t^k, v_t), \\ & (x_{t+1}, y_{t+1}) \in \mathcal{X}_{t+1} \times \mathcal{Y}_{t+1}, \\ & (u_t, v_t) \in \mathcal{U}_t(x_t^k) \times \mathcal{V}_t(y_t^k), \end{aligned} \tag{4.3}$$

storing (x_{t+1}^k, u_t^k) *as the minimisers. Compute a subgradient* β_n^k *with respect to* x_n^k.

Step 3. *Solve*

$$\begin{aligned} \bar{\theta}_t^k = \max_{y_{t+1}, v_t} \min_{x_{t+1}, u_t} \quad & C_t(x_t, y_t, u_t, v_t) + \bar{G}_{t+1}^{k-1}(x_{t+1}, y_{t+1}) \\ \text{s.t.} \quad & x_{t+1} = f_t^x(x_t^k, u_t), \\ & y_{t+1} = f_t^y(y_t^k, v_t), \\ & (x_{t+1}, y_{t+1}) \in \mathcal{X}_{t+1} \times \mathcal{Y}_{t+1}, \\ & (u_t, v_t) \in \mathcal{U}_t(x_t^k) \times \mathcal{V}_t(y_t^k), \end{aligned} \tag{4.4}$$

storing (y_{t+1}^k, v_t^k) *as the maximisers. Compute a subgradient* γ_t^k *with respect to* y_t^k.

Step 4. *Update the lower bound value function as*

$$\begin{aligned} \underline{G}_t^k(x, y) = \min_{\mu, \lambda} \quad & \mu + \langle \lambda, y \rangle \\ \text{s.t.} \quad & \mu + \langle \lambda, y_t^{\hat{k}} \rangle \geq \underline{\theta}_t^{\hat{k}} + \langle \beta_t^{\hat{k}}, x - x_t^{\hat{k}} \rangle, \quad \forall \hat{k} \leq k, \\ & ||\lambda|| \leq \alpha_y. \end{aligned} \tag{4.5}$$

Step 5. *Update the upper bound value function as*

$$\begin{aligned} \bar{G}_t^k(x,y) = \max_{\mu,\lambda} \quad & \mu + \langle \lambda, x \rangle \\ \text{s.t.} \quad & \mu + \langle \lambda, x_t^{\hat{k}} \rangle \leq \bar{\theta}_t^{\hat{k}} + \langle \gamma_t^{\hat{k}}, y - y_t^{\hat{k}} \rangle, \quad \forall \hat{k} \leq k, \\ & ||\lambda|| \leq \alpha_x. \end{aligned} \tag{4.6}$$

Step 6. *If $t = T$, move on to iteration $k+1$. Otherwise, continue from* **Step 2** *with $t = t+1$.* ⋈

We note here that during our algorithm, it could be possible to delay the value function update steps our algorithm until the entire state trajectory has been computed; this would strengthen the bounding functions generated at each iteration. However, this is not a necessity for convergence and obfuscates the description of the algorithm and proof. Indeed, this particular presentation strengthens our convergence result; we are able to achieve convergence using the weakest possible information. It is the focus for the remainder of this section to prove the convergence of our algorithm.

Our proposed convergence proof relies on two lemmas. The first is presented below, while the second (Appendix A.1) is consigned to the appendix on the account of its technicality. The first develops a crucial inequality obtained from our algorithm, and the second leverages the Lipschitz-continuity of value functions to give a conditional convergence result. For the following lemma we will define $\tilde{\mathcal{U}}(x) = \{u \in \mathcal{U}(x) \mid f_t^x(x,u) \in \mathcal{X}_{t+1}\}$ for notational convenience.

Lemma 4.1. *Consider the sequences of functions $\bar{G}_t^k$ and $\underline{G}_t^k$, and state trajectories (x_t^k, y_t^k) generated by the algorithm. For all $t = 0, \ldots, T-1$, we have*

$$\bar{G}_t^k(x_t^k, y_t^k) \leq C_t(x_t^k, y_t^k, u_t, v_t^k) + \bar{G}_{t+1}^{k-1}(f_t^x(x_t^k, u_t), y_{t+1}^k), \ \forall u_t \in \mathcal{U}_t(x_t^k).$$

Proof. From Step 5 in Algorithm 4.1, we have

$$\begin{aligned} \bar{G}_t^k(x_t^k, y_t^k) = \max_{\mu,\lambda} \quad & \mu + \langle \lambda, x_t^k \rangle \\ \text{s.t.} \quad & \mu + \langle \lambda, x_t^{\hat{k}} \rangle \leq \bar{\theta}_t^{\hat{k}} + \langle \gamma_t^{\hat{k}}, y_t^k - y_t^{\hat{k}} \rangle, \quad \forall \hat{k} \leq k \\ & ||\lambda|| \leq \alpha_x. \end{aligned}$$

So $\bar{G}_t^k(x_t^k, y_t^k) \leq \bar{\theta}_t^k$. From Step 3, we have

$$\begin{aligned} \bar{\theta}_t^k &= \min_{u_t \in \tilde{\mathcal{U}}(x_t^k)} \max_{v_t \in \tilde{\mathcal{V}}(y_t^k)} C_t(x_t^k, y_t^k, u_t, v_t) + \bar{G}_{t+1}^{k-1}(f_t^x(x_t^k, u_t), f_t^y(y_t^k, v_t)), \\ &= \min_{u_t \in \tilde{\mathcal{U}}(x_t^k)} C_t(x_t^k, y_t^k, u_t, v_t^k) + \bar{G}_{t+1}^{k-1}(f_t^x(x_t^k, u_t), y_{t+1}^k), \\ &\leq C_t(x_t^k, y_t^k, u_t, v_t^k) + \bar{G}_{t+1}^{k-1}(f_t^x(x_t^k, u_t), y_{t+1}^k), \ \forall u_t \in \mathcal{U}_t(x_t^k). \end{aligned}$$

This concludes the proof. □

Theorem 4.1. *Consider the sequence of functions $\bar{G}_t^k$ and $\underline{G}_t^k$, and state and control trajectories $\big((x_t^k, y_t^k), (u_t^k, v_t^k)\big)$ generated by the algorithm. For all $t \leq T$, we have*

$$\lim_{k \to \infty} \big(\bar{G}_t^k(x_t^k, y_t^k) - \underline{G}_t^k(x_t^k, y_t^k)\big) = 0.$$

Proof. The proof proceeds with a backwards induction. At time $t+1$, the induction hypothesis is

$$\lim_{k\to\infty} \left(\bar{G}^k_{t+1}(x^k_{t+1}, y^k_{t+1}) - G^k_{t+1}(x^k_{t+1}, y^k_{t+1})\right) = 0.$$

This is true for last stage by definition, which establishes the base case for our induction. Now we want to show

$$\lim_{k\to\infty} \left(\bar{G}^k_{t}(x^k_{t}, y^k_{t}) - G^k_{t}(x^k_{t}, y^k_{t})\right) = 0, \ \forall t < T,$$

assuming the induction hypothesis. From Lemma 4.1, we have

$$\bar{G}^k_t(x^k_t, y^k_t) \leq C_t(x^k_t, y^k_t, u^k_t, v^k_t) + \bar{G}^{k-1}_{t+1}(x^k_{t+1}, y^k_{t+1}), \ \forall t < T, \ \forall k,$$

and symmetrically

$$G^k_t(x^k_t, y^k_t) \geq C_t(x^k_t, y^k_t, u^k_t, v^k_t) + G^{k-1}_{t+1}(x^k_{t+1}, y^k_{t+1}), \ \forall t < T, \ \forall k.$$

Subtracting these, we obtain

$$\bar{G}^k_t(x^k_t, y^k_t) - G^k_t(x^k_t, y^k_t) \leq \bar{G}^{k-1}_{t+1}(x^k_{t+1}, y^k_{t+1}) - G^{k-1}_{t+1}(x^k_{t+1}, y^k_{t+1}), \ \forall t < T, \ \forall k.$$

We are now in a position to invoke Lemma Appendix A.1. By applying Lemma Appendix A.1 where the finite (one, in this case) collection of function sequences is $\bar{G}^k_{t+1}$ and G^k_{t+1} and the sequence of iterates on the compact set is $(x^k_{t+1}, y^k_{t+1}) \in \mathcal{X}_{t+1} \times \mathcal{Y}_{t+1}$, coupled with the induction hypothesis, we have that the right-hand side of the above approaches zero as k tends to infinity, completing the proof. □

This result extends the result of Theorem 3.1, and states that if upper and lower approximations are 'nested', by sampling them in this procedure, we obtain convergence. In this sense, we can say that this result mimics the convergence of dual dynamic programming of convex functions.

As an extension of Lemma 3.9, we will now show that a particular sequence of controls converge to the saddle sets of the saddle function defined at the states generated by the algorithm, thereby constituting a solution to the dynamic programming equations as in (4.2). This requires the use of new notation: let $\mathcal{G}_t[x_t, y_t](\epsilon)$ define the set of ϵ-saddle points for the function $C_t(x_t, y_t, \cdot, \cdot) + G_{t+1}(f^x_t(x_t, \cdot), f^y_t(y_t, \cdot))$, over $\tilde{\mathcal{U}}_t(x_t) \times \tilde{\mathcal{V}}_t(y_t)$. The following corollary statement provides the result that each control does indeed define an ϵ-optimal control.

Corollary 4.1. *Consider the sequence of functions $\bar{G}^k_t$ and G^k_t, and state and control pairs $\left((x^k_t, y^k_t), (u^k_t, v^k_t)\right)$ generated by the algorithm. For all $t < T$, we then have*

$$\Bigg\{ \Big\{ \operatorname*{arg\,min}_{u_t \in \tilde{\mathcal{U}}(x^k_t)} C(x^k_t, y^k_t, u_t, v^k_t) + \bar{G}^{k-1}_{t+1}(f^x_t(x^k_t, u_t), y^k_{t+1}) \times \mathcal{V}_t(y^k_t) \Big\} \cap$$
$$\Big\{ \operatorname*{arg\,max}_{v_t \in \tilde{\mathcal{V}}(y^k_t)} C(x^k_t, y^k_t, u^k_t, v_t) + G^{k-1}_{t+1}(x^k_{t+1}, f^y_t(y^k_t, v_t)) \times \mathcal{U}_t(x^k_t) \Big\} \Bigg\}$$
$$\subseteq \mathcal{G}_t[x^k_t, y^k_t]\left(\bar{G}^k_t(x^k_t, y^k_t) - G^k_t(x^k_t, y^k_t)\right), \ \forall k.$$

We do not provide a proof for Corollary 4.1 in this paper, but conjecture that such a proof would employ arguments similar to Lemma 3.9.

4.2. The stochastic setting.

In this section, we extend the multistage deterministic model by incorporating stochasticity. We consider a discrete filtered process, and therefore, describe our algorithm with respect to a scenario tree object; this provides several notational benefits. We consider a scenario tree with vertices $\mathcal{N}$ containing leaf vertices $\mathcal{L} \subset \mathcal{N}$. Each vertex $n \in \mathcal{N} \setminus \mathcal{L}$ has a cost function $C_n(x, y, u, v)$ and an associated state feasibility set $\mathcal{X}_n \times \mathcal{Y}_n$ and control set $\mathcal{U}_n(x) \times \mathcal{V}_n(y)$. The arcs on the scenario tree have weight p_n which are interpreted as the probability of reaching vertex n from the root vertex. Finally, the leaves have terminal value functions $G_m(x_m, y_m),\ \forall m \in \mathcal{L}$ defined over $\mathcal{X}_m \times \mathcal{Y}_m$. We require the same conditions as those laid out for the deterministic problem in (4.2), but rather than the conditions holding for every stage of the problem, we have them for every vertex in the tree. The extensive form of the problem considered in this section is given by

$$\begin{aligned} G_0(x_0, y_0) = \min_{x_n, u_n} \max_{y_n, v_n} \quad & \sum_{n \in \mathcal{N} \setminus \mathcal{L}} p_n C_n(x_n, y_n, u_n, v_n) + \sum_{m \in \mathcal{L}} p_m G_m(x_m, y_m) \\ \text{s.t.} \quad & x_m = f_m^x(x_n, u_n),\ \forall m \in R(n),\ \forall n \in \mathcal{N} \setminus \mathcal{L}, \\ & y_m = f_m^y(y_n, v_n),\ \forall m \in R(n),\ \forall n \in \mathcal{N} \setminus \mathcal{L}, \\ & (u_n, v_n) \in \mathcal{U}_n(x_n) \times \mathcal{V}_n(y_n),\ \forall n \in \mathcal{N} \setminus \mathcal{L}, \\ & (x_n, y_n) \in \mathcal{X}_n \times \mathcal{Y}_n,\ \forall n \in \mathcal{N}. \end{aligned} \tag{4.7}$$

The sum in the objective function represents the inner product of cost functions and their probabilities (i.e. an expectation). Meanwhile, the structure of the scenario encodes the non-anticipativity of the problem. We can describe the associated dynamic programming equations as

$$\begin{aligned} G_n(x_n, y_n) = \min_{x_m, u_n} \max_{y_m, v_n} \quad & C_n(x_n, y_n, u_n, v_n) + \sum_{m \in R(n)} \frac{p_m}{p_n} G_m(x_m, y_m) \\ \text{s.t.} \quad & x_m = f_m^x(x_n, u_n), \quad \forall m \in R(n), \\ & y_m = f_m^y(y_n, v_n), \quad \forall m \in R(n), \\ & (x_m, y_m) \in \mathcal{X}_m \times \mathcal{Y}_m, \quad \forall m \in R(n), \\ & (u_n, v_n) \in \mathcal{U}_n(x_n) \times \mathcal{V}_n(y_n), \end{aligned} \tag{4.8}$$

where $R(n)$ is the set of immediate children of vertex n.

Similar to many stochastic extensions in multistage methods, our algorithm follows that presented earlier for the deterministic case; we generate state and control trajectories for a given path in our scenario tree and compute updates to our bounding functions. Many methods generate a state and control trajectory and perform updates along a randomly selected path in the scenario tree, making these algorithms probabilistic. Proofs of convergence of these random sampling methods rely on well known theorems such as the Borel-Contelli Lemma in Philpott and Guan [26] and the Strong Law of Large Numbers in Girardeau et al. [13]. Depending on any independence assumptions on the data process in these problems, value function information can be shared between vertices in the scenario tree. The canonical example of this is the in the SDDP algorithm of Pereira and Pinto [22]. Further 'cut-sharing' techniques are discussed in Infanger and Morton [15]. Our ensuing algorithm does not assume nor exploit any particular structure in the

scenario tree; the tree object is simply used as a means to describe a proof in the most general setting in the hope of proving a stronger convergence result. In practice, one could not hope to solve large instances of (4.7); what makes these problems tractable is the effective compression of the scenario trees brought on by a particular structure in the uncertainty (i.e. independence).

Contrary to existing methods, our algorithm does not take a random sampling approach; we make use of the bound gap information to instead deterministically select which path in the tree should be taken. We introduce the object Φ_n^k, which keeps track of the child vertex of vertex n with the maximal difference in bounds at iteration k during the algorithm; Φ_n^k is critical to the convergence of our algorithm, as it ensures that the path in the scenario tree which is sampled at any given iteration is one where the bound gap is guaranteed to reduce and converge in the limit. Our algorithm for solving problems in the form of (4.7) is presented below.

Algorithm 4.2 (The multistage stochastic minimax algorithm)**.**

Initialisation. *Define* $\underline{G}_n^0 := -\infty, \bar{G}_n^0 := \infty,\ \forall n \in \mathcal{N} \setminus \mathcal{L}$. *Because the leaf value functions are given, set* $\underline{G}_m^k = G_m(x,y),\ \bar{G}_m^k(x,y) = G_m(x,y),\ \forall m \in \mathcal{L},\ \forall k$. *Set the initial state* $(x_0^k, y_0^k) = (x_0, y_0),\ \forall k$. *Finally, set* $k = 1$.

Iteration k**.**

Step 1. *Set* $n = 0$.

Step 2. *Solve*

$$\begin{aligned} \underline{\theta}_n^k = \min_{x_m,u_n} \max_{y_m,v_n} \quad & C_n(x_n^k, y_n^k, u_n, v_n) + \sum_{m \in R(n)} \frac{p_m}{p_n} \underline{G}_m^{k-1}(x_m, y_m) \\ \text{s.t.} \quad & x_m = f_m^x(x_n^k, u_n),\ \forall m \in R(n), \\ & y_m = f_m^y(y_n^k, v_n),\ \forall m \in R(n), \\ & (x_m, y_m) \in \mathcal{X}_m \times \mathcal{Y}_m,\ \forall m \in R(n), \\ & (u_n, v_n) \in \mathcal{U}_n(x_n^k) \times \mathcal{V}_n(y_n^k), \end{aligned} \tag{4.9}$$

storing v_n^k *as the minimiser. Compute a subgradient* β_n^k *with respect to* x_n^k.

Step 3. *Solve*

$$\begin{aligned} \bar{\theta}_n^k = \max_{y_m,v_n} \min_{x_m,u_n} \quad & C_n(x_n^k, y_n^k, u_n, v_n) + \sum_{m \in R(n)} \frac{p_m}{p_n} \bar{G}_m^{k-1}(x_m, y_m) \\ \text{s.t.} \quad & x_m = f_m^x(x_n^k, u_n),\ \forall m \in R(n), \\ & y_m = f_m^y(y_n^k, v_n),\ \forall m \in R(n), \\ & (x_m, y_m) \in \mathcal{X}_m \times \mathcal{Y}_m,\ \forall m \in R(n), \\ & (u_n, v_n) \in \mathcal{U}_n(x_n^k) \times \mathcal{V}_n(y_n^k), \end{aligned} \tag{4.10}$$

storing v_n^k *as the maximiser. Compute a subgradient* γ_n^k *with respect to* y_n^k.

Step 4. *Update the lower bound value function as*

$$\begin{aligned} \underline{G}_n^k(x,y) = \min_{\mu,\lambda} \quad & \mu + \langle \lambda, y \rangle \\ \text{s.t.} \quad & \mu + \langle \lambda, y_n^{\hat{k}} \rangle \geq \underline{\theta}_t^{\hat{k}} + \langle \beta_n^{\hat{k}}, x - x_n^{\hat{k}} \rangle, \quad \forall \hat{k} \leq k, \\ & ||\lambda|| \leq \alpha_y. \end{aligned} \tag{4.11}$$

Step 5. *Update the upper bound value function as*

$$\begin{aligned} \bar{G}_n^k(x,y) = \max_{\mu,\lambda} \quad & \mu + \langle \lambda, x \rangle \\ \text{s.t.} \quad & \mu + \langle \lambda, x_n^{\hat{k}} \rangle \leq \bar{\theta}_n^{\hat{k}} + \langle \gamma_n^{\hat{k}}, y - y_n^{\hat{k}} \rangle, \quad \forall \hat{k} \leq k, \\ & ||\lambda|| \leq \alpha_x. \end{aligned} \tag{4.12}$$

Step 6. *Update* Φ_n^k *as*

$$\Phi_n^k = \underset{m \in R(n)}{\arg\max} \ \frac{p_m}{p_n} \big(\bar{G}_m^{k-1}(f_m^x(x_n^k, u_n^k), f_m^y(y_n^k, v_n^k)) - \underline{G}_m^{k-1}(f_m^x(x_n^k, u_n^k), f_m^y(y_n^k, v_n^k)) \big). \tag{4.13}$$

If Φ_n^k *is not unique, then select one arbitrarily. Further, update* $(x_{\Phi_n^k}^k, y_{\Phi_n^k}^k)$ *as* $(f_{\Phi_n^k}^x(x_n^k, u_n^k), f_{\Phi_n^k}^y(y_n^k, v_n^k))$. *Set* $n = \Phi_n^k$.

Step 7. *If* n *is a leaf node, then move on to iteration* $k+1$. *Otherwise, continue from* ***Step 2*** *with the updated vertex* n. *Note that all state, controls, value functions and problem children that have not been updated in this iteration receive the null update i.e.* $(x_n^{k+1}, y_n^{k+1}) = (x_n^k, y_n^k)$. ⋈

We note the similarities between Algorithm 4.1 and Algorithm 4.2, both algorithms update along a certain path in the tree (in the deterministic case, there is only one path). Algorithm 4.2 selects the next vertex to visit in the tree (defining a path) according to the problem-child criterion. This idea is similar to that presented in Georghiou et al. [12]. However, the key difference is that our problem-child selection problem in Step 6 of our algorithm is 'easy' in the complexity sense. As the bounding functions are convex programmes, their evaluation is not 'hard'; however, we require $|R(n)|$ of these evaluations per vertex visited in each iteration.

We now state and prove our deterministic convergence theorem.

Theorem 4.2. *Consider the sequence of functions* $\bar{G}_n^k$ *and* $\underline{G}_n^k$, *and state and control trajectories* $\big((x_n^k, y_n^k), (u_n^k, v_n^k)\big)$, *and problem-children* Φ_n^k *generated by Algorithm 4.2. For all* $n \in \mathcal{N}$, *we have*

$$\lim_{k \to \infty} \big(\bar{G}_n^k(x_n^k, y_n^k) - \underline{G}_n^k(x_n^k, y_n^k) \big) = 0.$$

Proof. The proof proceeds again with a backwards induction. For the problem-child vertex Φ_n^k, the induction hypothesis is

$$\lim_{k \to \infty} \big(\bar{G}_{\Phi_n^k}^k(x_{\Phi_n^k}^k, y_{\Phi_n^k}^k) - \underline{G}_{\Phi_n^k}^k(x_{\Phi_n^k}^k, y_{\Phi_n^k}^k) \big) = 0.$$

This is true for all problem-children who are leaves, as their bounding functions are equal by definition. Now that the base case is established, we want to show

$$\lim_{k \to \infty} \big(\bar{G}_n^k(x_n^k, y_n^k) - \underline{G}_n^k(x_n^k, y_n^k) \big) = 0, \ \forall n \in \mathcal{N} \setminus \mathcal{L},$$

assuming the induction hypothesis. By arguments similar to those in Lemma 4.1, we have

$$\bar{G}_n^k(x_n^k, y_n^k) \leq C_n(x_n^k, y_n^k, u_n^k, v_n^k) + \sum_{m \in R(n)} \frac{p_m}{p_n} \bar{G}_m^{k-1}(f_m^x(x_n^k, u_n^k), f_m^y(y_n^k, v_n^k)),\ \forall n \in \mathcal{N}\backslash\mathcal{L},\ \forall k,$$

and symmetrically

$$\underline{G}_n^k(x_n^k, y_n^k) \geq C_n(x_n^k, y_n^k, u_n^k, v_n^k) + \sum_{m \in R(n)} \frac{p_m}{p_n} \underline{G}_m^{k-1}(f_m^x(x_n^k, u_n^k), f_m^y(y_n^k, v_n^k)),\ \forall n \in \mathcal{N}\backslash\mathcal{L},\ \forall k.$$

Subtracting these, we have

$$\bar{G}_n^k(x_n^k, y_n^k) - \underline{G}_n^k(x_n^k, y_n^k) \leq$$
$$\sum_{m \in R(n)} \frac{p_m}{p_n} \left[\bar{G}_m^{k-1}(f_m^x(x_n^k, u_n^k), f_m^y(y_n^k, v_n^k)) - \underline{G}_m^{k-1}(f_m^x(x_n^k, u_n^k), f_m^y(y_n^k, v_n^k)) \right],\ \forall n \in \mathcal{N}\backslash\mathcal{L},\ \forall k.$$

But by the problem-child selection principle (Step 6 of Algorithm 4.2), we must have

$$\bar{G}_n^k(x_n^k, y_n^k) - \underline{G}_n^k(x_n^k, y_n^k) \leq |R(n)| \frac{p_{\Phi_n^k}}{p_n} \left[\bar{G}_{\Phi_n^k}^{k-1}(x_{\Phi_n^k}^k, y_{\Phi_n^k}^k) - \underline{G}_{\Phi_n^k}^{k-1}(x_{\Phi_n^k}^k, y_{\Phi_n^k}^k) \right],\ \forall n \in \mathcal{N}\backslash\mathcal{L},\ \forall k.$$

Similar to the proof of Theorem 4.1, we invoke Lemma Appendix A.1 to conclude the proof; our finite collection of function sequences is now $\bar{G}_m^k, \underline{G}_m^k,\ \forall m \in R(n)$, our finite collection of sequences of iterates on their respective compact sets are given by $(x_m^k, y_m^k),\ \forall m \in R(n)$, and finally our map from $\mathbb{N} \mapsto R(n)$ is given by Φ_n^k. This proves the induction hypothesis. Coupled with our base case, this concludes the proof. □

Remark 4.1. We remark that in the special case where the cost functions are solely convex (concave) and only a minimisation (maximisation) is considered, Algorithm 4.2 provides a deterministic guarantee of convergence for these problems also. For the convex case, our formulation (4.7) is identical to that presented in Girardeau et al. [13]. This convergence result is interesting in its own right.

Stochastic dual dynamic programming algorithms form a lower bound (in the convex case) of the value functions by randomly sampling the scenario tree. It is the main result of Girardeau et al. [13] that this method converges *almost surely*. Our result above extends this result to a *sure convergence* result, and also makes obsolete the use of Monte Carlo estimation to form an upper bound.

5. Risk-averse formulations.

In this section we present multistage stochastic programming problems under different risk-averse objective functions. We present these formulations in the context of our minimax formulation from the previous section – that is, we require these formulations to comply with that presented in (4.2). Technically, in this section we refer to *acceptability* measures, rather than risk measures – however these formulations require only a small revision to apply as risk measures. Briefly, a risk measure $\rho : \mathcal{Z} \mapsto \mathbb{R}$ is a function

which maps random variables from some random variable space (on some measure space) to a real number. The seminal work of Artzner et al. [1] introduces a class of risk measures called *coherent* risk measures. Many researchers have since contributed to the development of the theory of coherent risk measures and their use in optimisation; we provide the following references for further reading: Delbaen [9] and Ruszczyński and Shapiro [32]. A particular focus of this theory is the representation of these risk measures; these alternative representations are frequently exploited when coherent risk measures are used in optimisation. We defer to the work of Rockafellar [27] to provide an intuitive introduction to these concepts. In our work, we focus on the CV@R measure, which was proved to be coherent by Rockafellar et al. [29]. Coherency is an important property here, as it permits the following *dual* representation: when considering a finite probability space $(\Omega, \mathscr{F}, \mathbb{P})$ and associated random variable space $\mathcal{Z}$ whose elements $|Z(\omega)| < \infty,\ \forall \omega \in \Omega,\ \forall Z \in \mathcal{Z}$, we have for a *quantile* $\beta \in (0, 1]$ that

$$\begin{aligned} \text{CV@R}_\beta(Z(\omega)) = \min_{\eta_\omega} \quad & \sum_{\omega \in \Omega} \mathbb{P}(\omega) Z(\omega) \eta_\omega \\ \text{s.t.} \quad & 0 \leq \eta_\omega \leq \frac{1}{\beta},\ \forall \omega \in \Omega, \\ & \sum_{\omega \in \Omega} \mathbb{P}(\omega) \eta_\omega = 1. \end{aligned} \tag{5.1}$$

At the optimal solution, it is possible to compute the *risk-adjusted probabilities* as

$$\mathbb{Q}(\omega) = \dot{\eta}_\omega \mathbb{P}(\omega),$$

which famously gives $\mathbb{E}_{\mathbb{Q}}[Z(\omega)] = \text{CV@R}(Z(\omega))$, where $\dot{\eta}_\omega$ is the optimal η_ω. Suppose the random variable $Z(\omega)$ is now a function of a control vector $v \in \mathbb{R}^n$ i.e. $Z(v, \omega)$. The usual risk-averse optimisation problem is to maximise the risk-adjusted value of a random variable whose outcomes are a function of a control vector v. This problem can be written as the following minimax optimisation problem:

$$\begin{aligned} \max_{v \in \mathcal{V}} \text{CV@R}_\beta(Z(v, \omega)) = \max_{v} \min_{\eta_\omega} \quad & \sum_{\omega \in \Omega} \mathbb{P}(\omega) Z(v, \omega) \eta_\omega \\ \text{s.t.} \quad & v \in \mathcal{V}, \\ & 0 \leq \eta_\omega \leq \frac{1}{\beta},\ \forall \omega \in \Omega, \\ & \sum_{\omega \in \Omega} \mathbb{P}(\omega) \eta_\omega = 1. \end{aligned}$$

If $Z(v, \omega)$ is concave in $v,\ \forall \omega \in \Omega$, then the objective is a saddle function. Consequently, this formulation complies with our general minimax formulation presented in (4.2); our control in the maximisation vector space is the vector v, while our control in the minimisation vector space is the vector of *probability scalers* η_ω. One could consider formulating this problem into a single maximisation problem, utilising the results of Rockafellar and Uryasev [28]. While useful in the two-stage setting, there exists no natural analogue of this reformulation in the multistage extension.

In contrast to evaluating the risk of two-stage random variables, evaluating risk of random variable in a multistage setting requires significant further consideration. The

work of Artzner et al. [2] extends their earlier work of two-stage risk measures into dynamic risk measures. Complementary to this work, Ruszczyński and Shapiro [31] provide a comprehensive technical foundation for measurement of multistage random variables through the use of conditional risk measures. The key take-away from these works is that risk-averse multistage optimisation obeys a time-consistency property when risk measures are 'nested' between stages. This theory led to the incorporation of CV@R in SDDP type algorithms as shown in Philpott and De Matos [24] and Shapiro et al. [35]. We present two minimax formulations that can be interpreted as risk-averse stochastic programmes. First, we will consider the traditional rectangular (or nested risk) formulation (similar to that considered by Philpott and De Matos [24]). Secondly, we present a novel dynamic programming formulation which computes an optimal policy over a single CV@R evaluation of accumulated profits.

5.1. Rectangular.

The most natural way to present the rectangular or nested formulation is through the definition of its value functions. A nested CV@R dynamic programming formulation is given by

$$\begin{aligned} V_n(y_n) = \max_{y_m, v_n} \quad & C_n(y_n, v_n) + \text{CV@R}_{\beta_n}\left[V_m(y_m)\right]_{m\in R(n)} \\ \text{s.t.} \quad & y_m = f_m(y_n, v_n),\ \forall m \in R(n), \\ & y_m \in \mathcal{Y}_m,\ \forall m \in R(n), \\ & v_n \in \mathcal{V}_n(y_n), \end{aligned}$$

for all $n \in \mathcal{N} \setminus \mathcal{L}$. The notation $\text{CV@R}[\,\cdot\,]_{m\in R(n)}$ denotes the *conditional value at risk* of the conditional random variable generated by the children of n. For the remainder of this section, we conduct our analysis for *fixed feasible* y and v, so the dynamic programming equations above become

$$V_n = C_n + \text{CV@R}_{\beta_n}\left[V_m\right]_{m\in R(n)}. \tag{5.2}$$

Here, $\beta_n \in (0, 1]$ are predefined quantiles; one for each vertex in the tree. These formulations are termed rectangular because the risk sets generated by the conditional CV@R measures are independent of each other. This key observation is critical for the algorithm presented in Philpott and De Matos [24]. Figure 3 gives a diagrammatic representation of how such an objective function might be visualised; here the nested structure of formulation is made apparent. It is our goal now to cast this formulation into a form that

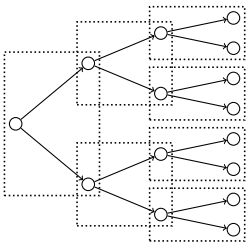


Figure 3: A nested risk measure on a scenario tree.

complies with the form of (4.8). Unfortunately, we cannot directly invoke the dual representation of CV@R (from (5.1)) to produce dynamic programming equations of the form

$$\begin{aligned} V_n = \min_{\eta_m} \quad & C_n + \sum_{m \in R(n)} \eta_m \frac{p_m}{p_n} V_m \\ \text{s.t.} \quad & 0 \leq \eta_m \leq 1/\beta_n, \quad \forall m \in R(n), \\ & \sum_{m \in R(n)} \frac{p_m}{p_n} \eta_m = 1. \end{aligned}$$

Our general problem formulation (4.8) requires that the *cost* function contains the dependence on control η_m, *not* the value-to-go function as is exhibited above. To remedy this, we move the 'risk adjustment' η_m to the cost functions of the children vertices. In doing so, we define a different set of dynamic programming equations $G_n^{\text{Nested}}(\eta)$; the value of arriving at vertex n, with probability scalar η. The full form of these dynamic programming equations is given by

$$\begin{aligned} G_n^{\text{Nested}}(\eta_n) = \min_{\eta_m} \quad & \eta_n C_n + \sum_{m \in R(n)} \frac{p_m}{p_n} [G_m^{\text{Nested}}(\eta_m)] \\ \text{s.t.} \quad & 0 \leq \eta_m \leq \frac{\eta_n}{\beta_n}, \quad \forall m \in R(n), \qquad (5.3.1) \\ & \sum_{m \in R(n)} \frac{p_m}{p_n} \eta_m = \eta_n. \end{aligned} \tag{5.3}$$

Here, η_m, the probability scalers, are considered the control and the states; the value of a particular η_n at each vertex in the tree represents the amount that the conditional probability of arriving at vertex n is scaled. We encourage the reader to relate this interpretation of η_n to η_ω in (5.1).

The following theorem shows that the dynamic programming formulation above coincides with the solution to (5.2).

Theorem 5.1. *Consider the two formulations given in* (5.2) *and* (5.3)*. We have*

$$G_0^{\text{N}}(1) = V_0.$$

We direct the reader to Appendix A for the proof since the evaluation of recursion becomes notationally cumbersome.

5.2. End-of-horizon.

As argued in the introduction of this paper, perhaps a more natural method of developing risk-averse policies would be to optimise an objective function which performs a single risk evaluation of every path of accumulated costs. In contrast to Figure 3, Figure 4 shows a diagrammatic representation of such an end-of-horizon (EOH) risk measure. We require new notation to express our idea: let Ω denote the set of all paths in the scenario tree from the root node to the leaf vertices. Further, let $\Omega(n) \subseteq \Omega$ denote the set of paths that contain vertex n. In mathematical form, we seek to compute

$$\text{CV@R}_{\beta_0}\left[\sum_{n \in \omega} C_n\right]_{\omega \in \Omega} \tag{5.4}$$

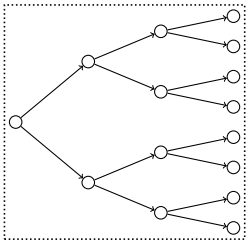


Figure 4: A *end-of-horizon* risk measure on a scenario tree.

for some $\beta_0 \in (0, 1]$. In the above formulation, costs are accumulated over each of the paths, and then CV@R evaluation is taken over these paths. In Pflug and Pichler [23], the authors lay down several technical foundations for a possible decomposition of this risk measure. They also present a set of dynamic programming equations whose solution coincides with a solution to (5.4). Here we present, a similar set of dynamic programming equations, and importantly, one which complies with the structure of (4.8):

$$\begin{aligned} G_n^{\mathrm{EOH}}(\eta_n) = \min_{\beta_m} \quad & \eta_n C_n + \sum_{m \in R(n)} \frac{p_m}{p_n} [G_m^{\mathrm{EOH}}(\eta_m)] \\ \text{s.t.} \quad & 0 \leq \eta_m \leq \frac{1}{\beta_0}, \quad \forall m \in R(n), \quad (5.5.1) \\ & \sum_{m \in R(n)} \frac{p_m}{p_n} \eta_m = \eta_n. \end{aligned} \tag{5.5}$$

Once again, the probability scalers η_n have the same interpretation as (5.3), however, their feasible set differs slightly. The constraint (5.5.1) in (5.5) dictates that the probability scalers η_n at any vertex on the tree can not exceed the amount prescribed by the risk set determined at the the root vertex. By formulating the problem in this way, probability scalers are 'distributed' over the tree in such a way that their composition yeilds the equivalent of a single CV@R evaluation over the entire tree. Further intuition of the formulation can be gained when the proof to Theorem 5.2 is studied.

Theorem 5.2. *Given the end-of-horizon dynamic programming equations defined in* (5.5)*, we have*

$$G_0^{\mathrm{EOH}}(1) = \mathrm{CV@R}_{\beta_0}\left[\sum_{n \in \omega} C_n\right]_{\omega \in \Omega}.$$

Once again, we direct the reader to Appendix A for the proof as the evaluation of the recursion becomes cumbersome.

Remark 5.1. We wish to comment on the *time consistency* of our end-of-horizon risk formulation. Time consistency is a developing concept in mathematical programming and as such, there are many competing notions of time consistency. We argue, that our end-of-horizon risk formulation is time consistent, because, by Theorem 5.2, the formulation permits a dynamic programming formulation. This notion of time consistency is the same as that argued in Carpentier et al. [8] and Lara and Leclere [17].

6. Numerical validation.

In this section we will present a risk-averse multistage stochastic portfolio optimisation problem in order to demonstrate our algorithm, and validate our end-of-horizon risk formulation. These numerical results are based upon a preliminary implementation of our algorithm written in Julia and interfaces with Gurobi as a linear programme solver (see [11] and [14] respectively).

We note that for the purposes of numerically solving our ensuing minimax dynamic programmes, the norm considered in the definition of our bounding functions is the $||\cdot||_\infty$-norm. As mentioned in Lemma 3.1, the resulting bounding function definitions become linear programmes. Similar to many SDDP type algorithms, we do not explicitly compute our bounding functions over their domain, but rather we store the bounding functions implicitly in larger optimisation problems (i.e. (4.9) and (4.10)).

In many instances, it may be difficult to determine an appropriate bound on the magnitude of the gradient λ inside our bounding function definitions (i.e. the Lipschitz constant). Indeed, a more accurate constant α will strengthen the bounding functions while still remaining valid. Without performing any prior analysis, we take a 'big-M' approach for bounding the hyperplane gradient coefficients in (4.11) and (4.12). We proceed by introducing a portfolio optimisation problem to illustrate our algorithm and end-of-horizon risk concept.

6.1. A portfolio optimisation problem.

Consider a general multistage portfolio optimisation problem with the following structure:

- A collection of S securities $\mathcal{S} = \{s_1, s_2, \ldots, s_S\}$;
- A portfolio is a vector $y_n \in \mathbb{R}^S$ which represents the value of each of the holdings at vertex n;
- the control vector $v_t \in \mathbb{R}^S$ represents the transactions made on the portfolio at vertex n;
- the fluctuation of value security s from vertex n to m, $\Psi_{m,s}$;
- the cost $C_n(y_n, v_n)$ incurred at vertex n.

In this section we will report on the results of a instance of this portfolio optimisation problem. In our problem, we will consider a collection of securities $\mathcal{S} := \{\text{cash}, \text{bonds}, \text{stocks}\}$. We impose several constraints on the managment of the portfolio: the portfolio must be *long only*, meaning that the portfolio manager cannot form a portfolio whose value falls below zero; the portfolio manager cannot withdraw from the security $s_3 :=$ stock, until the end of the period; lastly, the portfolio manager can only fund their purchases of new securities from selling other securities. We can write these constraints down mathematically (for each vertex n) as

$$
\begin{aligned}
& y_{n,s} + v_{n,s} \geq 0, \quad \forall s \in \mathcal{S}, && \text{(long only portfolio)} \\
& v_{n,s_3} \geq 0, && \text{(cannot withdraw from } s_3 = \text{stock)} \\
& \sum_{s \in \mathcal{S}} v_{n,s} = 0. && \text{(portfolio must be internally funded)}
\end{aligned}
$$

Finally, we constrain the state of our portfolio with the portfolio dynamics, which requires that the new value of stock s at vertex m is the previous value $y_{n,s}$ plus the previous transaction $v_{n,s}$ both multiplied by the fluctuation parameter $\Psi_{m,s}$:

$$y_{m,s} = \Psi_{m,s} \times (y_{n,s} + v_{n,s}).$$

The fluctuations, $\Psi_{m,s}$ are sampled from a multivariate normal distribution with parameters shown in Table 1. The accumulated annual mean and standard deviation are also included in the table for reference. Note that securities s_2 and s_3 are correlated (monthly) with correlation coefficient $\rho = 0.3$. Finally, the portfolio manager faces no stagewise transaction costs.

While initially holding a unit of s_1 (cash), the portfolio manager's objective is to maximise the risk adjusted asset value at the maturity date. With an end-of-horizon risk

Table 1: Multivariate normal distribution parameters.

$\mathcal{S}$	Monthly		Annual	
	μ	σ	μ	σ
s_1	0.000%	0.000%	0.00%	0.00%
s_2	0.247%	0.281%	3.00%	1.00%
s_3	0.327%	0.700%	4.00%	2.51%

measure given by CV@R_{β_0}, we obtain the following optimisation problem

$$\begin{aligned}
\max_{y,v} \quad & \text{CV@R}_{\beta_0}\left[\sum_{s\in\mathcal{S}} y_{n,s}\right]_{n\in\mathcal{L}} \\
\text{s.t.} \quad & y_{m,s} = \Psi_{m,s} \times (y_{n,s} + v_{n,s}), \quad \forall m \in R(n),\ \forall s \in \mathcal{S},\ \forall n \in \mathcal{N}\setminus\mathcal{L}, \\
& \sum_{s\in\mathcal{S}} v_{n,s} = 0, \quad \forall n \in \mathcal{N}\setminus\mathcal{L}, \\
& v_{n,s_3} \geq 0, \quad \forall n \in \mathcal{N}\setminus\mathcal{L}, \\
& y_{n,s} + v_{n,s} \geq 0, \quad \forall s \in \mathcal{S},\ \forall n \in \mathcal{N}\setminus\mathcal{L}.
\end{aligned} \tag{6.1}$$

By Theorem 5.2, we are able to formulate a set of minimax dynamic programming equations to supply to our solver.

6.2. Solution interpretation.

Figure 5 shows the convergence toward the optimal saddle point value for a CV@R quantile of $\beta_0 = 0.4$. Our convergence profile is similar to many cutting plane algorithms. As discussed in Nesterov [21], even the traditional Kelley Cutting Plane method can yield arbitrarily poor convergence. Because of the complex nature of our class of problems, careful consideration is required when reflecting on their solutions. Figures 5 and 6 pertain to our portfolio optimisation example where $\beta_0 = 0.4$ and the portfolio manager begins with a unit of s_1. We will report our results in terms of *percent increase yeild* beyond the starting balance, rather than absolute terms. Figure 5 shows that the optimal policy obtains a risk-adjusted annual increase between 2.0% and 2.15%. From the

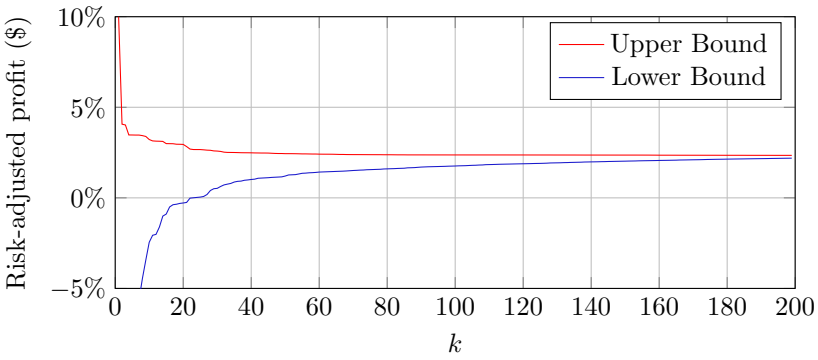


Figure 5: Convergence plot for portfolio optimisation example with $\beta_0 = 0.4$.

perspective of robust optimisation, the following interpretation could be offered: given the risk adjusted probabilities on the tree, there is no better policy that maximises expected profit, and simultaneously, there is no set of probabilities on the tree (in the risk set defined by the multistage risk measure) that could worsen the expected payoff – that is, we have computed a set of conditional saddle points. Similar to the two-stage case, we can compute the risk adjusted probabilities on the scenario tree as

$$\mathbb{Q}(\omega) = \dot{\eta}_n p_n, \quad \forall n \in \mathcal{L},$$

where $\dot{\eta}_n$ are the optimal leaf probability scalers. Figure 6 plots a histogram of a 2000 sample Monte-Carlo simulation which estimates the distribution of accumulated costs for two different optimal policies.

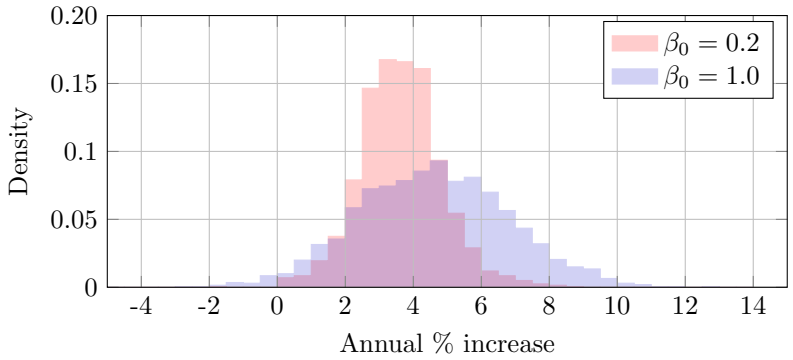


Figure 6: Profit distribution for different policies

With respect to the profit distribution, we observe a familiar phenomenon; that risk-averse policies increase profits in the worst outcomes, generally at the cost of profits in expectation. We note here that because of our end-of-horizon objective function, if a particular sample path results in 'good' outcomes thus far, then the policy dictates that there is no cost in having a risk neutral outlook from now on. Because the path has been successful so far, this path will not feature in the worst β_0 proportion of outcomes.

This is in contrast to the traditional nested formulation where risk aversion does not depend on history. We are enthusiastic about exploring these concepts further, however an extensive study on policy comparisons between different risk formulations would be best suited as a standalone work.

Table 2 contains the results of investigation in which our portfolio optimisation problem is parameterised in the end-of-horizon CV@R quantile parameter β_0. In this study, we use our algorithm to compute five policies, each optimal for their respective quantile β_0, ranging from $\beta_0 = 0.2$ (risk averse) to $\beta_0 = 1$ (risk neutral). For each policy, we estimate (using Monte Carlo simulation) the expectation of the worst proportion of accumulated profits in each of the five β-quantiles. Each Monte Carlo estimate uses 2000 random samples. In Table 2, each column presents the results of the five Monte Carlo estimation for a single optimal policy. If our generated policies are indeed optimal, then

Table 2: Monte Carlo estimation of CV@R quantiles for several policies.

CV@R_β Estimation	Optimal Policy				
	$\beta_0 = 0.2$	$\beta_0 = 0.4$	$\beta_0 = 0.6$	$\beta_0 = 0.8$	$\beta_0 = 1.0$
$\beta = 0.2$	1.52%	1.50%	1.30%	0.92%	1.05%
$\beta = 0.4$	2.01%	2.12%	2.13%	1.86%	1.96%
$\beta = 0.6$	2.38%	2.50%	2.68%	2.57%	2.66%
$\beta = 0.8$	2.71%	2.85%	3.16%	3.21%	3.21%
$\beta = 1.0$	3.14%	3.28%	3.73%	4.00%	4.09%

we ought to observe that for each policy, the estimation at the *same quantile* will be the best (highest) out of all the policies. From our results, it can be observed that each policy does indeed perform the best for its given quantile, except for policy $\beta_0 = 0.4$. Policy $\beta_0 = 0.6$ achieves an estimate of the expectation of the worst 40% of scenarios as a 2.13% yield, while policy $\beta_0 = 0.4$ achieves a yield of 2.12%. We attribute this artifact to sampling effects; the number of paths on the complete scenario tree is in the order of $6^{12} \approx 2 \times 10^9$, while our Monte-Carlo simulation only samples 2000 of these paths.

7. Conclusion

In this work, we have presented a cutting-plane type decomposition algorithm for solving a general class of minimax dynamic programmes. With our algorithm we are able to provide both an upper and lower bound on value functions, and obtain a deterministic guarantee of convergence, by way of a guided sampling approach. We have focused on minimax formulations which coincide with natural risk-averse stochastic programming formulations. We have demonstrated how two different risk measures over finite time-horizons (nested risk, and end-of-horizon risk) can be cast in our minimax framework and subsequently solved using our algorithm. The implementation of the end-of-horizon risk measure, in this time-consistent manner is a further novel contribution of this work. We demonstrate its ability to shape the end-of-horizon distribution of profits.

We have illustrated the application of our algorithm on a portfolio management problem with risk aversion over various end-of-horizon quantiles, showing that the method is able to converge on policies which maximise the specified risk-adjusted accumulated profits. Furthermore, we show the convergence of the upper and lower bounds for an example problem, and the corresponding distribution of profits. Given this general framework, a broad class of multistage optimisation problems has been made accessible by our algorithm.

Acknowledgements

Regan Baucke would like to thank the members of the Electrical Power Optimisation Centre for fruitful discussion. He would also like to thank both the University of Auckland and the Energy Education Trust New Zealand for their financial support during the period this work was conducted.

Appendix A. Proofs.

The following lemmas, theorems, and their proofs best serve the comprehension of this body of work as items seperated from the main article.

Lemma Appendix A.1. *Consider a finite collection of $|N|$ sequences of α-Lipschitz functions $\bar{H}_n^k$ and $\underline{H}_n^k$ on $\mathcal{Z}_n \subset \mathbb{R}^{d_n}$ and $\mathcal{Z}_n$ is compact. Further, for all $k \in \mathbb{N}$ we have $\underline{H}_n^k(z) \leq \underline{H}_n^{k+1}(z) \leq \bar{H}_n^{k+1}(z) \leq \bar{H}_n^k(z),\ \forall z \in \mathcal{Z}_n,\ \forall n \in N$. For any collection of $|N|$ sequences z_n^k on $\mathcal{Z}_n$, and any mapping $n : \mathbb{N} \mapsto N$, we have*

$$\lim_{k\to\infty} \left(\bar{H}_{n(k)}^k(z_{n(k)}^k) - \underline{H}_{n(k)}^k(z_{n(k)}^k)\right) = 0 \implies \lim_{k\to\infty} \left(\bar{H}_{n(k)}^{k-1}(z_{n(k)}^k) - \underline{H}_{n(k)}^{k-1}(z_{n(k)}^k)\right) = 0.$$

Proof. Suppose, for the sake of contradiction, that there exists an $\epsilon > 0$ for which

$$\epsilon \leq \bar{H}_{n(k)}^{k-1}(z_{n(k)}^k) - \underline{H}_{n(k)}^{k-1}(z_{n(k)}^k),\ \forall k \in \mathbb{N}.$$

Now because N is finite, there must exist at least one $n^* \in N$ for which the sequence $n(k) = n^*$ infinitely many times. Denote the infinite subset of the naturals where this equality holds as $\mathbb{N}^*$. So

$$\epsilon \leq \bar{H}_{n^*}^{k-1}(z_{n^*}^k) - \underline{H}_{n^*}^{k-1}(z_{n^*}^k),\ \forall k \in \mathbb{N}^*.$$

Subtracting $\left(\bar{H}_{n^*}^{k-1}(z_{n^*}^{\hat{k}}) - \underline{H}_{n^*}^{k-1}(z_{n^*}^{\hat{k}})\right)$ from both sides gives

$$\begin{aligned}\epsilon - \left(\bar{H}_{n^*}^{k-1}(z_{n^*}^{\hat{k}}) - \underline{H}_{n^*}^{k-1}(z_{n^*}^{\hat{k}})\right) \leq \\ \bar{H}_{n^*}^{k-1}(z_{n^*}^k) - \underline{H}_{n^*}^{k-1}(z_{n^*}^k) - \left(\bar{H}_{n^*}^{k-1}(z_{n^*}^{\hat{k}}) - \underline{H}_{n^*}^{k-1}(z_{n^*}^{\hat{k}})\right),\ \forall k, \hat{k}, \in \mathbb{N}^*.\end{aligned}$$

Because each of $\bar{H}_n^{k-1}$ and $\underline{H}_n^{k-1}$ is α-Lipschitz on $\mathcal{Z}_n,\ \forall k,\ \forall n \in N$, we have

$$\epsilon - \left(\bar{H}_{n^*}^{k-1}(z_{n^*}^{\hat{k}}) - \underline{H}_{n^*}^{k-1}(z_{n^*}^{\hat{k}})\right) \leq 2\alpha||z_{n^*}^k - z_{n^*}^{\hat{k}}||,\ \forall k, \hat{k} \in \mathbb{N}^*.$$

From our lemma's hypothesis, we have that

$$\lim_{k\in\mathbb{N}\to\infty} \left(\bar{H}_{n_k}^k(z_{n_k}^k) - \underline{H}_{n_k}^k(z_{n_k}^k)\right) = 0,$$

so our subsequence must have the same limit i.e.

$$\lim_{k\in\mathbb{N}^*\to\infty}\left(\bar{H}^k_{n^*}(z^k_{n^*})-\underline{H}^k_{n^*}(z^k_{n^*})\right)=0.$$

This limit property, combined with the monotonicity of $\left(\bar{H}^k_n(z)-\underline{H}^k_n(z)\right)$ with respect to k for all $z\in\mathcal{Z}_n$ and for all $n\in N$, means there exists some $\tilde{k}$ large enough for which $\left(\bar{H}^{k-1}_{n^*}(z^{\hat{k}}_{n^*})-\underline{H}^{k-1}_{n^*}(z^{\hat{k}}_{n^*})\right)\leq\frac{\epsilon}{2},\ \forall\tilde{k}\leq k,\ \tilde{k}\leq\hat{k}<k$, and both $k,\hat{k}$ are in $\mathbb{N}^*$. So

$$\frac{\epsilon}{2}\leq 2\alpha||z^k_{n^*}-z^{\hat{k}}_{n^*}||,\ \forall\tilde{k}\leq k,\ \tilde{k}\leq\hat{k}<k,\ \text{and } k,\hat{k}\in\mathbb{N}^*.$$

Dividing through, we have

$$\frac{\epsilon}{4\alpha}\leq||z^k_{n^*}-z^{\hat{k}}_{n^*}||,\ \forall\tilde{k}\leq k,\ \tilde{k}\leq\hat{k}<k,\ \text{and } k,\hat{k}\in\mathbb{N}^*.$$

which is a contradiction of the compactness of $\mathcal{Z}_{n^*}$, because $\mathbb{N}^*$ is countably infinite, and $z^k_{n^*}$ contains no convergent subsequence. So there exists no such $\epsilon>0$ for which

$$\epsilon\leq\bar{H}^{k-1}_{n_k}(z^k_{n_k})-\underline{H}^{k-1}_{n_k}(z^k_{n_k}),\ \forall k,$$

concluding the proof. □

Lemma Appendix A.2. *Under the conditions laid out in* (5.1)*, for* $\eta_0\geq 0$*, we have*

$$\begin{aligned}\mathrm{CV@R}_\beta(Z(\omega))\times\eta_0=\min_{\eta_\omega}\quad&\sum_{\omega\in\Omega}\mathbb{P}(\omega)Z(\omega)\eta_\omega\\ \text{s.t.}\quad&0\leq\eta_\omega\leq\frac{\eta_0}{\beta},\quad\forall\omega\in\Omega,\\ &\sum_{\omega\in\Omega}\mathbb{P}(\omega)\eta_\omega=\eta_0.\end{aligned}\tag{A.1}$$

Proof. The proof is simple when a observing a re-scaling of η_ω in (A.1) as $\frac{\eta_\omega}{\eta_0}$. For the case where $\eta_0=0$, the equality holds trivially. □

Theorem (5.1). *Consider the two formulations given in* (5.2) *and* (5.3)*. We have*

$$G^N_0(1)=V_0.$$

Proof of Theorem 5.1. The strategy of this proof is to evaluate a recursion from the leaves of the tree, toward the root vertex. We remind the reader that $P(n)$ denotes the parent vertex of vertex n. For all $n\in P(m),\forall m\in\mathcal{L}$, we have,

$$\begin{aligned}G^{\text{Nested}}_n(\eta_n)=\min_{\eta_m}\quad&C_n\eta_n+\sum_{m\in R(n)}\frac{p_n}{p_m}\left[C_m\eta_m\right]\\ \text{s.t.}\quad&0\leq\eta_m\leq\frac{\eta_n}{\beta_n},\quad\forall m\in R(n),\\ &\sum_{m\in R(n)}\frac{p_m}{p_n}\eta_m=\eta_n,\quad\forall m\in R(n),\end{aligned}$$

which by Lemma Appendix A.2 gives

$$G_n^{\text{Nested}}(\eta_n) = \eta_n \times \text{CV@R}_{\beta_n}[C_m]_{m\in R(n)} + C_n\eta_n = \eta_n\big[C_n + \text{CV@R}_{\beta_n}[C_m]_{m\in R(n)}\big].$$

So $\forall n \in P(P(m)), \forall m \in \mathcal{L}$, we have

$$\begin{aligned} G_n(\eta_n) = \min_{\eta_m} \quad & C_n\eta_n + \sum_{m\in R(n)} \frac{p_n}{p_m}\eta_m\left[C_m + \text{CV@R}_{\beta_m}[C_{\hat{m}}]_{\hat{m}\in R(m)}\right] \\ \text{s.t.} \quad & 0 \le \eta_m \le \frac{\eta_n}{\beta_n}, \quad \forall m \in R(n), \\ & \sum_{m\in R(n)} \frac{p_m}{p_n}\eta_m = \eta_n, \quad \forall m \in R(n), \end{aligned}$$

which again by Lemma Appendix A.2 gives

$$\begin{aligned} G_n^{\text{Nested}}(\eta_n) &= \eta_n \times \text{CV@R}_{\beta_n}\big[C_m + \text{CV@R}_{\beta_m}[C_{\hat{m}}]_{\hat{m}\in R(m)}\big]_{m\in R(n)} + C_n\eta_n \\ &= \eta_n\left[C_n + \text{CV@R}_{\beta_n}\big[C_m + \text{CV@R}_{\beta_m}[C_{\hat{m}}]_{\hat{m}\in R(m)}\big]_{m\in R(n)}\right]. \end{aligned}$$

By the recursive application of the above, we obtain

$$G_0^{\text{Nested}}(1) = C_0 + \text{CV@R}_{\beta_0}\left[C_m + \text{CV@R}_{\beta_m}\left[C_{\hat{m}} + \dots\right]\right] = V_0$$

which is exactly the evaluation of the recursive relation (5.2). This completes the proof. □

Theorem (5.2). *Given the end-of-horizon dynamic programming equations defined in* (5.5)*, we have*

$$G_0^{\text{EOH}}(1) = \text{CV@R}_{\beta_0}\left[\sum_{n\in\omega} C_n\right]_{\omega\in\Omega}$$

Proof of Theorem 5.2. Once again, our strategy for this proof is to evaluate the recursion. Further we prove the case where the root vertex children's children are the leaf vertices; for larger trees, the proof is the same. By evaluating the recursion in (5.5) and merging on mins, we obtain

$$\begin{aligned} G_0^{\text{EOH}}(1) = \min_{\eta} \quad & C_0 + \sum_{m\in R(0)} p_m\left[C_m\eta_m + \sum_{\dot{m}\in R(m)} \frac{p_{\hat{m}}}{p_m}\big[C_{\hat{m}}\eta_{\hat{m}}\big]\right] \\ \text{s.t.} \quad & 0 \le \eta_m \le \frac{1}{\beta_0}, \quad \forall m \in R(n),\ \forall n \in \mathcal{N}\setminus\mathcal{L}, \qquad (A.2) \\ & \sum_{m\in R(n)} \frac{p_m}{p_n}\eta_m = \eta_n, \quad \forall m \in R(n),\ \forall n \in \mathcal{N}\setminus\mathcal{L}. \qquad (A.2.1) \end{aligned}$$

By the recursive evaluation of constraint (A.2.1), we have

$$\eta_n = \sum_{\omega\in\Omega(n)} \frac{p_{\mathcal{L}(\omega)}}{p_n}\eta_{\mathcal{L}(\omega)}. \qquad (A.3)$$

Here, the notation $\Omega(n) \subseteq \Omega$ denotes the set of all paths that contain vertex n and $\mathcal{L}(\omega)$ is the leaf vertex associated with path $\omega \in \Omega$. From (A.3), we have

$$\begin{aligned} G_0^{\mathrm{EOH}}(1) = \min_{\eta} \quad & C_0 + \sum_{m \in R(0)} p_m \Big[C_m \Big(\sum_{\omega \in \Omega(m)} \frac{p_{\mathcal{L}(\omega)} \eta_{\mathcal{L}(\omega)}}{p_m} \Big) + \sum_{\hat{m} \in R(m)} \frac{p_{\hat{m}}}{p_m} [C_{\hat{m}} \Big(\sum_{\omega \in \Omega(\hat{m})} \frac{p_{\mathcal{L}(\omega)} \eta_{\mathcal{L}(\omega)}}{p_{\hat{m}}} \Big)] \Big] \\ \text{s.t.} \quad & 0 \leq \eta_{\mathcal{L}(\omega)} \leq \frac{1}{\beta_0}, \quad \forall \omega \in \Omega, \\ & \sum_{\omega \in \Omega} p_{\mathcal{L}(\omega)} \eta_{\mathcal{L}(\omega)} = 1. \end{aligned}$$

By expanding the above and cancelling through probabilities, we obtain

$$\begin{aligned} G_0^{\mathrm{EOH}}(1) = \min_{\eta} \quad & C_0 \sum_{\omega \in \Omega(0)} p_{\mathcal{L}(\omega)} \eta_{\mathcal{L}(\omega)} + \sum_{m \in R(0)} C_m \sum_{\omega \in \Omega(m)} p_{\mathcal{L}(\omega)} \eta_{\mathcal{L}(\omega)} + \sum_{m \in R(0)} \sum_{\hat{m} \in R(m)} C_{\hat{m}} \sum_{\omega \in \Omega(\hat{m})} p_{\mathcal{L}(\omega)} \eta_{\mathcal{L}(\omega)} \\ \text{s.t.} \quad & 0 \leq \eta_{\mathcal{L}(\omega)} \leq \frac{1}{\beta_0}, \quad \forall \omega \in \Omega, \\ & \sum_{\omega \in \Omega} p_{\mathcal{L}(\omega)} \eta_{\mathcal{L}(\omega)} = 1. \end{aligned} \tag{A.4}$$

Now, in each of the terms in the objective above, the nested sums accumulate each $p_{\mathcal{L}(\omega)} \eta_{\mathcal{L}(\omega)}, \ \forall \omega \in \Omega$ exactly once. By taking a particular $p_{\mathcal{L}(\omega^*)} \eta_{\mathcal{L}(\omega^*)}$ out as a factor of each term, we obtain $[C_0 + C_{m^*} + C_{\hat{m}^*}] p_{\mathcal{L}(\omega^*)} \eta_{\mathcal{L}(\omega^*)}$ where m^*, and $\hat{m}^*$ are on the path defined by ω^*. By performing this factorisation for each $p_{\mathcal{L}(\omega^*)} \eta_{\mathcal{L}(\omega^*)}$, we obtain

$$\begin{aligned} G_0^{E}(1) = \min_{\eta_\omega} \quad & \sum_{\omega \in \Omega} \eta_{\mathcal{L}(\omega)} p_{\mathcal{L}(\omega)} \Big[\sum_{n \in \omega} C_n \Big] \\ \text{s.t.} \quad & 0 \leq \eta_{\mathcal{L}(\omega)} \leq \frac{1}{\beta_0}, \quad \forall \omega \in \Omega, \\ & \sum_{\omega \in \Omega} p_{\mathcal{L}(\omega)} \eta_{\mathcal{L}(\omega)} = 1. \end{aligned}$$

The above is the risk envelope representation of the CV@R risk measure over the sum of accumulated rewards, completing the proof. □

Appendix B. Technical conditions.

Consider the extensive form of the optimisation problem presented in Section 4:

$$\begin{aligned} G_0(x_0, y_0) = \min_{x_t, u_t} \max_{y_t, v_t} \quad & \sum_{t=0}^{T-1} C_t(x_t, y_t, u_t, v_t) + G_T(x_T, y_T) \\ \text{s.t.} \quad & x_{t+1} = f_t^x(x_t, u_t), \ \forall t < T, \\ & y_{t+1} = f_t^y(y_t, v_t), \ \forall t < T, \\ & (u_t, v_t) \in \mathcal{U}_t(x_t) \times \mathcal{V}_t(y_t), \ \forall t < T, \\ & (x_t, y_t) \in \mathcal{X}_t \times \mathcal{Y}_t, \ \forall t \leq T. \end{aligned} \tag{B.1}$$

with the final value function $G_T(x_T, y_T)$ is given. We require the following technical conditions for our algorithm to converge:

1. for all t, $\mathcal{X}_t$ and $\mathcal{Y}_t$ are non-empty subsets of $\mathbb{R}^{n_x}$ and $\mathbb{R}^{m_y}$ respectively;
2. for all t, multifunctions $\mathcal{U}_t : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_u}$ and $\mathcal{V}_t : \mathbb{R}^{m_y} \mapsto \mathbb{R}^{m_v}$ and are convex and non-empty compact valued;
3. the final cost function $G_T(x_T, y_T)$ is a saddle function. The cost functions $C_t(x, y, u, v)$, $\forall t < T$ are convex in x and u and concave in y and v;
4. for all $t < T$, functions $f_t^x(x_t, u_t)$ and $f_t^y(y_t, v_t)$ are affine in u_t, x_t and v_t, y_t respectively;
5. the final value function $G_T(x_T, y_T)$ is finite-valued and Lipschitz-continuous on $\mathcal{X}_T \times \mathcal{Y}_T$;
6. for all $t < T$, there exists $\delta_t^x > 0$ and $\delta_t^y > 0$, defining $\mathcal{X}_t' := \mathcal{X}_t + B_t^x(\delta_t^x)$ and $\mathcal{Y}_t' := \mathcal{Y}_t + B_t^y(\delta_t^y)$, where

$$B_t^z(\delta) = \{z \in \text{Aff}(\mathcal{Z}_t) \mid ||z|| \leq \delta\}$$

such that

(a) $\forall y \in \mathcal{Y}_t,\ \forall v \in \mathcal{V}_t(y),\ \forall x \in \mathcal{X}_t',\ \mathcal{X}_t' \times \mathcal{U}_t(x) \subseteq \text{dom } C_t(\cdot, y, \cdot, v)$,
(b) $\forall x \in \mathcal{X}_t'$

$$f_t^x(x, \mathcal{U}_t(x)) \cap \mathcal{X}_{t+1} \text{ is non-empty,}$$

and

(a) $\forall x \in \mathcal{X}_t,\ \forall u \in \mathcal{U}_t(x),\ \forall y \in \mathcal{Y}_t',\ \mathcal{Y}_t' \times \mathcal{V}_t(y) \subseteq \text{dom } C_t(x, \cdot, u, \cdot)$,
(b) $\forall y \in \mathcal{Y}_t'$

$$f_t^y(y, \mathcal{V}_t(y)) \cap \mathcal{Y}_{t+1} \text{ is non-empty.}$$

Conditions 1-5 ensure that (B.1) and its value function definitions are convex optimisation problems. Condition 6 is designed to give a constraint qualification condition for the optimisation problem (as it is non-linear in general). In words, it says that the saddle function can be defined over an extended domain in a convex/concave fashion – this allows the existence of subgradients at the boundary points of the non-extended domain in the respective directions. In Girardeau et al. [13], the authors call this condition *extended relatively complete recourse*, which is a more general class of recourse (which includes complete recourse). In their analysis, the existence of δ_t allows the construction of a Lipschitz constant to give their Lipschitz-continuity of $V_t(x_t)$ result. We do not conduct such an analysis here, but conjecture a similar analysis can be performed to construct a particular (α_u, α_v) pair. Our proof relies only on the existence of such a pair which is guaranteed by these conditions. We note that these conditions can be considered a saddle function analogue to those required by Girardeau et al. [13] in their paper.

References

[1] P. Artzner, F. Delbaen, J.-M. Eber, D. Heath, Coherent Measures of Risk, Mathematical Finance 9 (1999) 203–228.

[2] P. Artzner, F. Delbaen, J.-M. Eber, D. Heath, H. Ku, Coherent multiperiod risk measurement, 2002.

[3] T. Asamov, A. Ruszczyński, Time-consistent approximations of risk-averse multistage stochastic optimization problems, Mathematical Programming 153 (2015) 459–493.

[4] R. Bellman, The theory of dynamic programming, 1954.
[5] J. F. Benders, Partitioning procedures for solving mixed-variables programming problems, Numerische Mathematik 4 (1962) 238–252.
[6] J. R. Birge, Decomposition and Partitioning Methods for Multistage Stochastic Linear Programs, Operations Research 33 (1985) 989–1007.
[7] J. R. Birge, F. Louveaux, Two-Stage Recourse Problems, Springer New York, New York, NY, 2011, pp. 181–263.
[8] P. Carpentier, J.-P. Chancelier, G. Cohen, M. De Lara, P. Girardeau, Dynamic consistency for stochastic optimal control problems, Annals of Operations Research 200 (2012) 247–263.
[9] F. Delbaen, Coherent risk measures on general probability spaces, Advances in Finance and Stochastics: Essays in Honour of Dieter Sondermann (2002) 1–37.
[10] W. Dinkelbach, Parametrische lineare Programmierung, Springer Berlin Heidelberg, Berlin, Heidelberg, 1969, pp. 90–149.
[11] I. Dunning, J. Huchette, M. Lubin, JuMP: A Modeling Language for Mathematical Optimization, arXiv:1508.01982 [math.OC] (2015) 1–25. arXiv:1508.01982.
[12] A. Georghiou, W. Wiesemann, A. Tsoukalas, Robust Dual Dynamic Programming, Optimization Online 1 (2017).
[13] P. Girardeau, V. Leclere, A. Philpott, On the Convergence of Decomposition Methods for Multistage Stochastic Convex Programs, Mathematics of Operations Research 40 (2014) 130–145.
[14] I. Gurobi Optimization, Gurobi Optimizer Reference Manual, 2016.
[15] G. Infanger, D. P. Morton, Cut Sharing for Multistage Stochastic Linear Programs with Interstage Depency, Mathematical Programming 75 (1996) 241–256.
[16] J. Kelley, The cutting-plane method, Journal of the Society for Industrial and Applied Mathematics 8 (1960) 703–712.
[17] M. D. Lara, V. Leclere, Building up time-consistency for risk measures and dynamic optimization, European Journal of Operational Research 249 (2016) 177–187.
[18] A. Madansky, Bounds on the Expectation of a Convex Function of a Multivariate Random Variable, The Annals of Mathematical Statistics 30 (1959) 743–746.
[19] D. H. Martin, On the continuity of the maximum in parametric linear programming, Journal of Optimization Theory and Applications 17 (1975) 205–210.
[20] A. Nemirovski, On self-concordant convexconcave functions, Optimization Methods and Software 11 (1999) 303–384.
[21] Y. Nesterov, Introductory Lectures on Convex Programming Volume I: Basic course, Lecture Notes I (1998) 1–211.
[22] M. V. F. Pereira, L. M. V. G. Pinto, Multi-stage stochastic optimization applied to energy planning, Mathematical Programming 52 (1991) 359–375.
[23] G. Pflug, A. Pichler, Time-inconsistent multistage stochastic programs: Martingale bounds, European Journal of Operational Research 249 (2016) 155–163.
[24] A. Philpott, V. De Matos, Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion, European Journal of Operational Research 218 (2012) 470–483.
[25] A. Philpott, V. De Matos, E. Finardi, On Solving Multistage Stochastic Programs with Coherent Risk Measures, Operations Research 51 (2013) 957–970.
[26] A. Philpott, Z. Guan, On the convergence of stochastic dual dynamic programming and related methods, Operations Research Letters 36 (2008) 450–455.
[27] R. T. Rockafellar, Coherent Approaches to Risk in Optimization Under Uncertainty, OR Tools and Applications: Glimpses of Future Technologies (2007) 38–61.
[28] R. T. Rockafellar, S. Uryasev, Optimization of conditional value-at-risk, Journal of Risk 2 (2000) 21–41. arXiv:arXiv:1011.1669v3.
[29] R. T. Rockafellar, S. P. Uryasev, M. Zabarankin, Deviation Measures in Risk Analysis and Optimization, Technical Report, 2002.
[30] A. Ruszczyński, Risk-averse dynamic programming for Markov decision processes, Mathematical Programming 125 (2010) 235–261.
[31] A. Ruszczyński, A. Shapiro, Conditional Risk Mappings, Mathematics of Operations Research 31 (2006) 544–561.
[32] A. Ruszczyński, A. Shapiro, Optimization of Convex Risk Functions, Mathematics of Operations Research 31 (2006) 433–452. arXiv:1111.0194v1.
[33] A. Shapiro, Analysis of stochastic dual dynamic programming method, European Journal of Operational Research 209 (2011) 63–72.
[34] A. Shapiro, Minimax and risk averse multistage stochastic programming, European Journal of

Operational Research 219 (2012) 719–726.
[35] A. Shapiro, D. Dentcheva, A. Ruszczyński, Lectures on stochastic programming: modeling and theory, 2009.
[36] J. von Neumann, Zur Theorie der Gesellschaftsspiele, Mathematische Annalen 100 (1928) 295–320.