

---

# Risk aversion in multistage stochastic optimisation problems

Regan Shay Baucke

A thesis submitted in fulfilment of the requirements for the degree of  
Doctor of Philosophy in Operations Research,  
The University of Auckland, 2018.

---



## Co-Authorship Form

This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

A deterministic Algorithm for Solving stochastic minimax dynamic programmes.  
Chapter 5.

Nature of contribution by PhD candidate: Proofs of Lemmas, Theorems, conception and form of article + numerical model

Extent of contribution by PhD candidate (%): 80%

### CO-AUTHORS

Name	Nature of Contribution
Golbon Zakeri	Discussion and editing.
Anthony Downward	Discussion and editing.

### Certification by Co-Authors

The undersigned hereby certify that:

- ❖ the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- ❖ that the candidate wrote all or the majority of the text.

Name	Signature	Date
Golbon ZAKERI		20/08/18
ANTHONY DOWNWARD		24/08/18



## Co-Authorship Form

This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.	
A deterministic algorithm for solving multistage stochastic programming problems. Optimisation Online. Chapter 4.	
Nature of contribution by PhD candidate	Statement of Proofs of Lemmas and theorems, conception and form of article + numerical model
Extent of contribution by PhD candidate (%)	80%

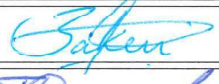
### CO-AUTHORS

Name	Nature of Contribution
Golbon Zakeri	Discussion and editing
Anthony Downward	Discussion and editing

### Certification by Co-Authors

The undersigned hereby certify that:

- ❖ the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- ❖ that the candidate wrote all or the majority of the text.

Name	Signature	Date
Golbon ZAKERI		20/08/18
ANTHONY DOWNWARD		24/08/18



# Abstract

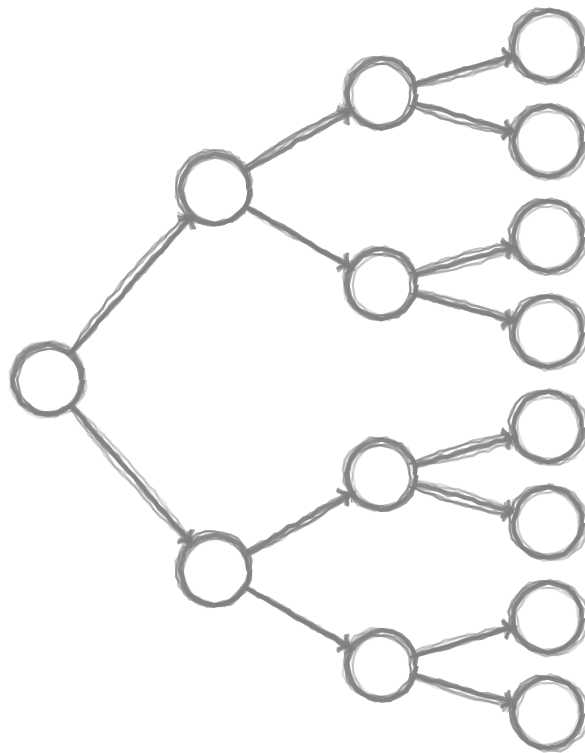
In this thesis, we explore the concept of risk aversion in multistage stochastic optimisation problems. In these optimisation problems, a decision maker must make a sequence of decisions through time, where the ‘state’ of the system can depend on previous decisions, as well as some ‘random noise’ that may occur in-between when decisions are made. This class of problems are notoriously difficult to solve, owing to the size of the scenario trees that these problems induce.

In the classical version of the problem, the decision maker seeks to minimise the expected cost of her decisions. Recently, practitioners have become concerned with not only the expected cost of their policy, but also other features of the resulting cost distribution. A particular concern is the performance of the system in the worst case; this is a characterisation of *risk aversion*.

We begin by reviewing several results in risk aversion in both the static and dynamic cases. A particularly important result is the *dual representation* of coherent risk measures, which relates risk averse optimisation to minimax or *robust* optimisation. Within this minimax framework, we explore different methods of including risk into the objective function of a multistage stochastic optimisation problem. We present the traditional approach which works by composing rectangular risk sets. We also present a new method of incorporating risk that relies on the ‘non-rectangularity’ of the risk sets. Prior to this work, no algorithm existed to solve the latter type of multistage minimax optimisation problems, making these formulations intractable.

We develop a decomposition style algorithm for this class of multistage minimax optimisation problems. This algorithm relies on upper and lower bound representations of saddle functions; these bounding functions are iteratively updated until a deterministic convergence criterion is met.

Lastly, we extend our work into the infinite horizon paradigm; we develop a convergence result for a broad class of Markov decision processes, where the objective function is an infinite geometric sum of costs.



Above is a scenario tree which represents uncertainty in a multistage stochastic optimisation problem. In the first stage, two possible events can occur. With more stages, the problem size grows exponentially which makes these problems difficult to solve.



Dedicated to my family



## Acknowledgements

This thesis is the culmination of at least three years of academic work, all of which wouldn't be possible without my supervision team , Golbon Zakeri and Tony Downward. I give my sincerest thanks to them for their effort, mentorship, and advice. I would like to thank my parents for supporting me throughout this endeavour. To my colleagues: thanks for making the Monday mornings a little easier. I couldn't ask for a more reliable daily source of wit and sarcasm. Lastly, I would like to thank Anna for her patience her support.



# Contents

LIST OF FIGURES	11
LIST OF TABLES	13
1 INTRODUCTION	15
1.1 Recurring mathematical themes	17
1.1.1 Analysis	17
1.1.2 Probability	19
1.1.3 Stochastic optimisation	21
1.1.4 Dynamic programming	22
2 RISK AVERSION	27
2.1 Acceptance sets	27
2.2 Risk measures	29
2.2.1 Coherent risk measures	30
2.2.2 Stochastic dominance and law invariance	32
2.2.3 The conditional value at risk	35
2.3 A duality result	37
2.3.1 Kusuoka representation	40
2.4 Risk in optimisation	42
2.4.1 Minimax evaluation	42
2.4.2 Rockafellar evaluation	44
2.4.3 Risk in constraints	44
3 MULTISTAGE RISK AVERSION	47
3.1 A time-consistency concept	48
3.1.1 Time consistency under uncertainty	51
3.1.2 Relationship with dynamic programming	53

3.2	Filtered probability spaces	54
3.3	Conditional risk mappings	57
3.3.1	Composition of conditional mappings	60
3.3.2	A law invariance issue	61
3.3.3	Parametric conditional risk sets	62
3.3.4	A heuristic interpretation	65
4	MULTISTAGE STOCHASTIC PROGRAMMING ALGORITHMS	69
4.1	Problem description	70
4.1.1	Scenario tree representation	70
4.1.2	Problem formulation	71
4.2	Preliminary concepts	73
4.2.1	Kelley's cutting plane method	73
4.2.2	The stochastic case	76
4.2.3	Forming upper bounds	78
4.3	Nested Benders and SDDP	79
4.3.1	The deterministic case	80
4.3.2	The stochastic case	84
4.3.3	Assumptions regarding uncertainty	86
4.3.4	Forming upper bounds in the stochastic case	88
4.4	A deterministic algorithm	88
4.4.1	An upper bound function	89
4.4.2	The deterministic algorithm	92
4.5	A computational study	95
4.5.1	A test problem	95
4.5.2	Results	96
4.5.3	Solution verification	99
5	RISK AVERSE MULTISTAGE STOCHASTIC PROGRAMMES	101
5.1	Problem class and current techniques	101
5.1.1	Problem formulation	102
5.1.2	Average-cut reweighting	104
5.1.3	The case for a saddle algorithm	107
5.2	An algorithm for minimax problems	108
5.2.1	Saddle functions	108

5.2.2	<i>Bounds for saddle functions</i>	111
5.2.3	<i>The stochastic setting</i>	119
5.3	<i>Risk averse formulations</i>	126
5.3.1	<i>Rectangular risk</i>	127
5.3.2	<i>End-of-horizon risk</i>	131
5.3.3	<i>Rectangularisation</i>	134
5.3.4	<i>Multi-quantile structure</i>	135
5.4	<i>Numerical validation</i>	136
5.4.1	<i>A portfolio optimisation problem</i>	137
5.4.2	<i>Solution interpretation</i>	139
<b>6</b>	<b>RISK IN MARKOV DECISION PROCESSES</b>	<b>143</b>
6.1	<i>The deterministic model</i>	144
6.1.1	<i>Problem formulation</i>	144
6.1.2	<i>Properties of value functions</i>	146
6.1.3	<i>An algorithm and its convergence</i>	147
6.2	<i>The stochastic model</i>	153
6.2.1	<i>Algorithm</i>	154
6.2.2	<i>Risk averse formulations</i>	156
<b>7</b>	<b>CONCLUSIONS AND FURTHER RESEARCH</b>	<b>159</b>
7.1	<i>How this research contributes</i>	159
7.1.1	<i>Challenges</i>	160
7.2	<i>Research questions borne from this thesis</i>	161
<b>REFERENCES</b>		<b>1</b>





## List of Figures

2.1	A convex cone $\mathcal{A}$ in $\mathbb{R}^2$ .	28
2.2	The risk of a random variable.	30
2.3	Acceptance set for $\text{Var}(X)$ .	32
2.4	Acceptance set for $\text{VaR}(X)$ .	36
2.5	A range of acceptance sets for different CVaR quantiles.	37
2.6	A set $A$ and its convex cone closure.	38
2.7	The dual cone $\mathcal{A}^*$ and its subset of probability measures $\mathcal{Q}_{\mathcal{A}}$ .	39
3.1	Coin toss space.	62
3.2	Decomposition on the Coin toss space.	66
4.1	Convex function $W(u)$ with a lower bound function of two sample points.	76
4.2	Compression of scenario tree.	87
4.3	Lower bounds on the Expected Cost for the two algorithms.	97
4.4	Convergence plot with Monte Carlo estimates for an upper bound.	98
5.1	A saddle function $g(u, v)$ with two sets of $\epsilon$ -saddle points.	110
5.2	For the saddle function $g(u, v)$ in grey, the blue surfaces are the upper bounding function $\bar{G}(u, v)$ .	113
5.3	A nested risk measure on a scenario tree.	128
5.4	A <i>end-of-horizon</i> risk measure on a scenario tree.	131
5.5	Convergence plot for portfolio optimisation example with $\beta_0 = 0.4$ .	139
5.6	Profit distribution for different policies.	140
6.1	A graph representation of our motivating problem.	145
6.2	The look-ahead function $L$ and induced set $\mathbb{N}_L(n)$ .	150
6.3	A graph representation of our motivating problem.	154



## List of Tables

4.1	Results for Problem 1	97
4.2	Monte Carlo estimates required for convergence tests	99
5.1	Multivariate normal distribution parameters.	138
5.2	Monte Carlo estimation of CVaR quantiles for several policies.	140



# Chapter 1

## Introduction

Planning over multiple time horizons remains one of the most challenging optimisation problems faced in the mathematical programming community. George Dantzig, one of the pioneers of linear programming, once famously said:

“In retrospect, it is interesting to note that the original problem that started my research is still outstanding – namely the problem of planning or scheduling dynamically over time, particularly planning dynamically under uncertainty. If such a problem could be successfully solved it could eventually, through better planning, contribute to the well-being and stability of the world.”

While it is this author’s opinion that perhaps the usefulness of planning dynamically over time may be overestimated by Dantzig, these problems present an exciting and rewarding challenge for theoreticians and practitioners alike.

Opinions vary between those in the field of optimisation as to whether or not the algorithms and models developed are of practical use in real-life. I would argue that it depends on how close one’s model (mathematical abstraction) reflects the corresponding ‘real-life’ situation. No serious person can deny the utility and prevalence of the shortest path algorithm and corresponding graph structures which attempt to abstract a physical network topology. However, the further our mathematical abstractions stray from ‘real-life’, the less meaningful the optimal solution becomes. Optimisation models are a powerful tool; however, without an appreciation for the complexity of the problem at hand, the optimal solution represents at best a ‘close-enough’ solution, and at worst, something completely impractical. In the hands of a layperson, the optimal solution of a model can be falsely reassuring; after all: how can one contradict the computer?

Very rarely is one faced with a decision making problem in the form of

$$z^* = \max_{v \in \mathcal{V}} F(v),$$

where  $F$  and  $\mathcal{V}$  both take a closed form. More often than not, any large optimisation problem will have parameters that are not known in advance – that is, we have ‘missing information’. Stochastic programming is a optimisation methodology that seeks to resolve the ‘missing information’ problem. In doing so, it hopes to move mathematical abstractions ‘closer’ to their real-life counterparts, making their optimal solutions more meaningful.

An inherent assumption of stochastic programming is that optimising agent is concerned with the average performance of her decision over the distribution of missing information. However, even this assumption can be an assumption too far. Consider a fair coin toss, where heads earns the player \$10 and tails earns \$0. The expected return of this coin toss is \$5. Consider another coin toss where heads earns \$6 and tails earns \$2. Then the expected return this coin toss is \$4. An interesting question to ask her is: “If offered a choice between these two coin tosses, which coin toss would you choose?” Clearly, there is no correct answer, as it is a matter of preference. In order to declare a correct answer, one needs a method of ordering ‘coin tosses’ and then choosing the better one. Within an optimisation problem, the job of ordering coin tosses is performed by the objective function. Ordering the tosses by their expected return is one way to perform the ordering, however there is no reason to not use other objective functions. An optimising agent who performs an ordering where the expected return is not paramount, is considered to have a risk attitude. Perhaps the agent likes to gamble, in which case we would call her a risk seeker. Otherwise, she may be more conservative; we would then call her risk averse.

This thesis is a study of multistage stochastic optimisation, how risk aversion can impact these optimisation problems, and how we can solve these problems. This thesis is laid out as follows: Chapter 2 introduces risk on simple probability spaces, and introduces risk averse optimisation. Chapter 3 extends these ideas to filtered probability spaces, where we seek to ‘order’ returns (random variables) dynamically. In Chapter 4, we introduce the multistage stochastic programming problem and review the algorithms used to solve these problems. We present our own algorithm which aims to improve on the current methods. Chapter 5 extends the formulations

in Chapter 4 by adapting the objective functions to contain multistage risk measures. The resulting problems are more complicated and require a new algorithm to solve. In Chapter 6, we introduce the infinite horizon problem and adapt the ideas presented in the preceding chapters to these problems. Finally, Chapter 7 presents the conclusions of this research and we remark on possible future research directions. As a note to the reader, the material in this thesis is best understood when read in the order that it appears.

## 1.1 Recurring mathematical themes

Throughout this thesis we will appeal to many classical mathematical results. This section is dedicated to their discussion so the thesis remains self-contained. These theorems range from those in convex analysis to real analysis, to probability theory. We note here that this is not an exhaustive list of mathematical notions employed in this thesis. This section also serves to introduce common notations employed throughout the thesis.

### 1.1.1 Analysis

Throughout this thesis, we will develop several convergence results. These results rely heavily on concepts within analysis. Here we present the most essential results required for understanding our convergence results later on.

**Theorem 1.1 (The Bolzano-Weierstrass theorem).**

Every bounded sequence in  $\mathbb{R}^n$  has a convergent subsequence.

An alternative formulation of the theorem states that if a subset of  $\mathbb{R}^n$  is closed and bounded, then it is sequentially compact; that is, every sequence contains a convergent subsequence. The proof utilises the monotone convergence theorem which states that a monotonic sequence which is bounded is convergent. All that is left is to show is that any sequence contains a monotonic subsequence which is easily shown. A common proof technique employed in this thesis is to construct a sequence in a compact Euclidean subset that contains no convergent subsequence. Therefore, the assumptions under which the non-convergent subsequence was constructed must be false; proving the hypothesis of our theorem true.

**Theorem 1.2 (The Weierstrass theorem).**

All continuous real-valued functions on a compact subset  $\mathcal{X}$  of a metric space attains a maximum and a minimum on  $\mathcal{X}$ .

An interesting question to ask at this point is: “under what conditions is only a minimum guaranteed?”. This question motivates the following definition

**Definition 1.1.**

A function  $f : \mathcal{X} \mapsto \mathbb{R}$  is lower-semicontinuous on  $S \subseteq \mathcal{X}$  if and only if the sets  $\{x \in S : f(x) \leq a\}$  are closed for all  $a \in \mathbb{R}$ .

Obviously, a continuous function is lower-semicontinuous, but a lower-semicontinuous function need not be continuous. Consider the function

$$f(x) = \begin{cases} x^2 + 1, & x > 0, \\ x^2, & x \leq 0. \end{cases}$$

The function  $f$  has a discontinuity at 0, but notice that it still achieve its minimum of 0 at  $x = 0$ . This leads naturally to an alternative definition of lower-semicontinuous functions: the function may jump up, but it must be equal to its lower part at the discontinuity. A function which is both upper-semicontinuous and lower-semicontinuous is therefore continuous!

Another concept regarding continuity is *Lipschitz-continuity*.

**Definition 1.2.**

A Lipschitz-continuous function (under the  $p$ -norm) is a function  $f : \mathcal{X} \mapsto \mathbb{R}$  for which there exists an  $\alpha < \infty$  such that

$$|f(x_1) - f(x_2)| \leq \alpha \|x_1 - x_2\|_p, \quad \forall x_1, x_2 \in \mathcal{X}.$$

Clearly a Lipschitz-continuous function is continuous and is therefore both lower-semicontinuous and upper-semicontinuous. However the converse relationship does not necessarily hold.

**Example 1.1.**

The following list shows several examples and counter-examples of Lipschitz-continuous functions:



- the function  $f(x) = x$  is Lipschitz-continuous on  $\mathbb{R}$  with Lipschitz constant  $\alpha = 1$ ;
- the function  $f(x) = x^2$  is Lipschitz-continuous on  $[a, b]$  with  $a, b \in \mathbb{R}$  and  $a < b$ . The smallest Lipschitz constant is given by  $\max\{|f'(a)|, |f'(b)|\}$ . Note that Lipschitz-continuous functions need not be differentiable;
- the function  $f(x) = |x|$  is Lipschitz-continuous on  $\mathbb{R}$ ;
- the function  $f(x) = \sqrt{x}$  is not Lipschitz-continuous on  $[0, 1]$ . Consider the derivative of function  $f'(x) = \frac{1}{2\sqrt{x}}$ . As  $x$  approaches zero, there will always exist an  $x$  such that  $f'(x) > \alpha$ ,  $\forall \alpha \in \mathbb{R}$ .

These notions, although rather technical, are important and form the foundation for several convergence results in this work.

### 1.1.2 Probability

Probability is a relatively modern field within mathematics – since this thesis concerns risk, uncertainty, and chance, it is natural to use the language of probability. Here we lay out the most basic objects that are required for mathematics with probability.

#### Definition 1.3.

The triple  $(\Omega, \mathcal{F}, \mathbb{P})$  is a probability space, where  $\Omega$  is a non-empty set of outcomes,  $\mathcal{F}$  is a  $\sigma$ -algebra on  $\Omega$ ,  $\mathbb{P}$  is a measure on  $(\Omega, \mathcal{F})$  and  $\mathbb{P}(\Omega) = 1$ .

The set  $\Omega$  is called the sample space, while the set  $\mathcal{F}$  is the collection events and is a  $\sigma$ -algebra of  $\Omega$ . The function  $\mathbb{P}$  assigns elements of  $\mathcal{F}$  to  $[0, 1]$ .

#### Definition 1.4.

For a non-empty set  $\Omega$ , the set  $\mathcal{F}$  is a  $\sigma$ -algebra on  $\Omega$  if

1.  $\mathcal{F}$  contains the empty set and  $\Omega$ ;
2.  $\mathcal{F}$  is closed under complement i.e.  $A \in \mathcal{F} \implies \Omega \setminus A \in \mathcal{F}$ .
3.  $\mathcal{F}$  is closed under countable union i.e.  $A_1, A_2, \dots \in \mathcal{F} \implies A_1 \cup A_2 \cup \dots \in \mathcal{F}$ .

These conditions ensure that statements about probabilities of events make ‘sense’. For instance, if we are able to measure the event  $A \in \mathcal{F}$ , intuitively, we know that  $\Omega \setminus A$  ought to be measurable too. So  $\Omega \setminus A \in \mathcal{F}$ .

**Example 1.2.**

Consider a situation where a fair two-sided coin is tossed twice. The set of outcomes is  $\Omega = \{HH, HT, TH, TT\}$ , the set of events are  $\mathcal{F} = 2^\Omega$ , and  $\mathbb{P}(\omega) = \frac{1}{4}, \forall \omega \in \Omega$ . Because  $\mathbb{P}$  is a measure, we can construct the probabilities of the remaining events in  $\mathcal{F}$  by the countable additivity property of measures.

In the finite case, these constructions are straightforward. As usual in mathematics, when we consider the infinite case, our constructions require thoughtful consideration. Consider a situation where a number is picked at random (uniformly) from the unit interval. The set of outcomes is  $\Omega = (0, 1)$ . However the set of events is not so clear in this instance. One might be tempted to declare  $\mathcal{F} = 2^{(0,1)}$  such as in the finite case. However such a  $\sigma$ -algebra contains sets which *are not measurable*. Hope is not lost in our endeavour, we can generate a valid  $\sigma$ -algebra where all its elements are measurable sets. Consider the open interval  $(a, b)$ , we generate our required  $\sigma$ -algebra by declaring that the set  $\mathcal{F}$  contains every open interval on  $(0, 1)$ ,  $(a, b)$ ,  $a \leq b$ , and that the  $\sigma$ -algebra is closed under complement and closed under countable unions. The resulting  $\sigma$ -algebra is called the *Borel*  $\sigma$ -algebra generated on  $(0, 1)$ , or  $\mathcal{B}((0, 1))$ . Now for every set in  $\mathcal{F}$ , we can ascribe a probability. In our case of a uniform measure, we can define  $\mathbb{P}((a, b)) = b - a$ .

Next we have *random variables*. Although further abstractions can be made, here we will limit our discussion to *real-valued* random variables. Consider a probability space  $(\Omega, \mathcal{F}, \mathbb{P})$ . A real-valued random variable is a function  $X : \Omega \mapsto \mathbb{R}$ . We require the function to be *measurable* – that is, for every  $B \in \mathcal{B}(\mathbb{R})$ , i.e. every set in the Borel  $\sigma$ -algebra generated by the real line, the *preimage*  $X^{-1}(B)$  is an element of the  $\sigma$ -algebra  $\mathcal{F}$ , where  $X^{-1}(B) = \{\omega \in \Omega \mid X(\omega) \in B\}$ . Heuristically, this means that the random variable cannot tell us more about the probability space than the  $\sigma$ -algebra prescribes. Finally,  $\mathbb{E}_{\mathbb{P}}$  is the expectation operator and by definition we have

$$\mathbb{E}_{\mathbb{P}}[X] = \int_{\Omega} X(\omega) d\mathbb{P}(\omega),$$

the Lebesgue integral of the of the random variable  $X$  with respect to the probability measure  $\mathbb{P}$ .

### 1.1.3 Stochastic optimisation

As mentioned earlier, it is considered a luxury when one is faced with a decision making problem with full knowledge of the future – we would call such a problem deterministic i.e.

$$z^* = \max_{v \in \mathcal{V}} F(v).$$

However, in reality, decision makers are rarely capable of knowing all the information required to make the optimal decision. A popular framework for modelling this ‘missing knowledge’, is stochastic programming. In general, stochastic programming is a methodology which studies mathematical optimisation problems where problem parameters are uncertain; as in, they can be modelled as random variables on some underlying probability space.

Consider some random variable (perhaps not real-valued)  $X : \Omega \mapsto \mathbb{R}^n$ , of ‘problem data’ or parameters and some profit mapping  $F : \mathcal{V} \times \mathbb{R}^n \mapsto \mathbb{R}$ . It is tempting to consider the average data problem as in

$$z^{\text{AV}} = \max_{v \in \mathcal{V}} F(v, \mathbb{E}_{\mathbb{P}}[X(\omega)]).$$

By averaging the data, we recover a deterministic optimisation problem which has obvious attraction. However, it is not hard to contrive an example where the optimisation yields an arbitrarily bad objective value when clearly better solutions are available. Further, the function  $F(v, X(\omega))$  may not even be defined at  $\mathbb{E}_{\mathbb{P}}[X(\omega)]$ . After further consideration, the ‘correct’ approach to solving this problem is to solve

$$z^* = \max_{v \in \mathcal{V}} \mathbb{E}_{\mathbb{P}}[F(v, X(\omega))].$$

By defining  $Z$  as the appropriate composition of  $F$  and  $X$ , we arrive the canonical form of a stochastic optimisation problem:

$$z^* = \max_{v \in \mathcal{V}} \mathbb{E}_{\mathbb{P}}[Z(v, \omega)].$$

Of course, by defining  $\mathbb{E}_{\mathbb{P}}[Z(v, \omega)]$  as a single cost function  $\mathcal{Z}(v)$ , we have, in a technical sense, a deterministic problem. However, forming  $\mathcal{Z}(v)$  is difficult or impossible for several reasons: if the underlying probability space is continuous, the expectation amounts to an integral. Even worse, the dimensionality of the probability space (i.e. the data vector) leads to a high dimensional integral; which can be very difficult to compute. Further, the modeller may not have access to information about

the underlying probability space, and has only partially observed realisations of the data random variable, making the expectation impossible to compute. The review paper of Shapiro and Nemirovski (2005) provides an introduction to some of these concepts.

With these impediments in mind, we introduce the *sample average approximation* (or SAA) approach to solving a stochastic programme. The sample average approximation can be formulated as follows:

$$z_N^* = \max_{v \in \mathcal{V}} \frac{1}{N} \sum_{n=1, \dots, N} Z(v, \omega_n).$$

That is, we take  $N$  independent random samples from our random variable  $Z(v, \omega)$  and average them. We could expect that such a procedure would give a convergence result as the number of samples  $N$  tends to infinity; we see a similar result for the Law of large numbers, in particular, the Strong law. We do indeed obtain the convergence result provided the random variable  $Z(v, \omega)$  is *light-tailed*, a minor technical assumption. There are many interesting works devoted to this topic, of note is Shapiro and Homem-de-Mello (2000). In another work, Shapiro and Nemirovski (2005), develop bounds on the minimum number of samples  $N$ , required to generate an  $\epsilon$ -optimal solution  $z_N^*$ , which is an  $2\epsilon$ -optimal solution to  $z^*$  with some probability  $1 - \alpha$ . Under some mild technical assumptions, they obtain

$$N \geq O(1) \left( \frac{1}{\epsilon} \right)^2 \left[ n \log \left( \frac{1}{\epsilon} \right) + \log \left( \frac{1}{\alpha} \right) \right], \quad (1.1)$$

where  $n$  is the dimension of the control set  $\mathcal{V}$ . However, if the underlying probability space is indeed finite, then the problem can be posed exactly as

$$z^* = \max_{v \in \mathcal{V}} \sum_{\omega \in \Omega} \mathbb{P}(\omega) Z(v, \omega).$$

Notice the similar form between the SAA problem and the problem above. Most of the time throughout this thesis, our algorithms operate on finite problems; if the underlying problem is not finite, then we appeal to the sample average approximation result in order to give probabilistic convergence for the continuous problem.

#### 1.1.4 Dynamic programming

Dynamic programming was first developed by Bellman (1954) and is a popular methodology in mathematical programming. The method seeks to compute *cost-to-go* functions which encode the cost of future optimal decisions; once the cost-to-go

functions have been computed, the optimal policy for a given state is then easily computed as the minimiser of the current stage's cost plus the cost-to-go associated with the new stage.

Dynamic programming only makes sense as a solution methodology if the optimisation problem of concern has what is known as an *optimal substructure*. This usually means the problem has some sort of *state* or *stage* structure. Consider the following generic optimisation problem:

$$\begin{aligned} \min_{u \in \Theta} \quad & C(u) \\ \text{s.t.} \quad & u \in \mathcal{U}. \end{aligned} \tag{1.2}$$

An optimisation problem is said to have optimal substructure if there exist functions  $C_1(u_1)$ ,  $C_2(u_1, u_2)$  and sets  $\mathcal{U}_1$ ,  $\mathcal{U}_2(u_1)$  for which

$$C(u) + \mathbb{I}_{\mathcal{U}}(u) = C_1(u_1) + C_2(u_1, u_2) + \mathbb{I}_{\mathcal{U}_1}(u_1) + \mathbb{I}_{\mathcal{U}_2(u_1)}(u_2), \quad \forall u = (u_1, u_2) \in \Theta, \tag{1.3}$$

where  $\mathbb{I}_X$  is the indicator function for the set  $X$ . Then, by solving

$$\begin{aligned} \min_{u_1, u_2} \quad & C_1(u_1) + C_2(u_1, u_2) \\ \text{s.t.} \quad & u_1 \in \mathcal{U}_1, \\ & u_2 \in \mathcal{U}_2(u_1), \end{aligned} \tag{1.4}$$

i.e. the right-hand side of (1.3), we obtain the solution to the left-hand side of (1.3). As our earlier example suggested, we can form *dynamic programming* equations by noting the 'min' over the second control  $u_2$  can be pushed forward. So (1.4) is equivalent to

$$\begin{aligned} \min_{u_1} \quad & C_1(u_1) + \min_{u_2} C_2(u_1, u_2) \\ \text{s.t.} \quad & u_1 \in \mathcal{U}_1, \\ & u_2 \in \mathcal{U}_2(u_1). \end{aligned}$$

And so by defining

$$\begin{aligned} V_1(u_1) = \min_{u_2} \quad & C_2(u_1, u_2) \\ \text{s.t.} \quad & u_2 \in \mathcal{U}_2(u_1), \end{aligned} \tag{1.5}$$

we obtain

$$\begin{aligned} \min_{u_1} \quad & C_1(u_1) + V_1(u_1) \\ \text{s.t.} \quad & u_1 \in \mathcal{U}_1, \end{aligned}$$

which again solves (1.4) and (1.2). This notion of forming cost-to-go functions can generalise further, where recursive definitions of cost-to-go functions are commonplace.

Let us illustrate with an example. Suppose one seeks a shortest path on a connected graph from vertex  $n_\alpha$  to  $n_\omega$ . One must make a sequence of decisions (moving from vertex  $n$  to  $m$  at some cost  $C(n, m)$ ) until they reach the end vertex  $n_\omega$ . If one is positioned at vertex  $n_a$ , and had two vertices in front of her:  $n_b$  and  $n_c$ , and she knew that the shortest path from  $n_b$  to  $n_\omega$  and  $n_c$  to  $n_\omega$ , then in order to make the optimal decisions she would simply choose

$$\min \{C(n_a, n_b) + V_b, C(n_a, n_c) + V_c\}.$$

The agent simply has to make a greedy decision now (after incorporating the cost-to-go) in order to determine the optimal strategy. This is the intuition behind dynamic programming: a sequence of solutions of easy optimisation problems constitutes the solution to a much larger optimisation problem, which might be difficult to solve. A property of optimal solutions obtained from dynamic programming is the *Principle of Optimality* (Bellman 1954):

“An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision”.

However, nothing comes for free; the we are now tasked with computing the cost-to-go functions. Depending on the form of the optimisation problem of concern, constructing the cost-to-go functions will require a different approach. Different approaches are taken to compute the value function depending on the problem structure; for example, one might obtain a closed form expression for the cost-to-go function. In large scale optimisation problems, this is almost never the case. If  $\mathcal{U}_1$  is a finite set, then generating a ‘look-up’ table for each input into  $V_1$  can be a feasible method of computing the value function. This amounts to solving (1.5) for each input value  $u_1$ . When the dynamic programming equations span multiple stages, we use later cost-to-go functions to generate the ‘look-up’ table for earlier cost-to-go functions, doing so recursively until the problem is solved. This is often referred to as *backward recursion*.

Things become less straight-forward when  $\mathcal{U}_1$  is infinite (and so the domain of  $V_1$  is infinite). Our aforementioned look-up table procedure will be hopeless, at least at

obtaining an exact solution. In these cases, the best we can hope to do is *approximate* the value function. There are three main approaches to generating approximations to cost-to-go functions. Discretising the state space as a first approach seems attractive, but this approach can suffer from the curse of dimensionality; that is, the number of points required to maintain a constant resolution grows exponentially in the number of dimensions.

The approximate dynamic programming method of W. B. Powell (2011) seeks to discover optimal multipliers to basis functions whose linear combinations might approximate the cost-to-go function well. Often, the problem of choosing the form and quantity of basis functions for a given problem is difficult and highly problem-dependant. These methods have also found application in other areas such as machine learning and artificial intelligence.

Benders decomposition is a further method; we will become very familiar with it throughout this thesis. In this method, the value function is approximated by the pointwise maximum of a number of linear functions. This approximation can then be refined in areas where the optimal solution is likely to be. This method requires several assumptions about the cost-to-go function (i.e. convexity) to be applicable.

With these mathematical tools, we are now in a position to discuss in more precision the ideas of risk in multistage optimisation.





## Chapter 2

### Risk aversion

As this thesis intends to utilise the theory of risk aversion, there would be no more appropriate place to begin than to define these terms. The idea of risk is intimately connected to the idea of probability and uncertainty. This is why the language of risk is largely inherited from the language of probability theory; we discuss things in terms of probability spaces, random variables, and filtrations. Intuitively, we can say that risk is present when the result of some uncertain event contains the possibility of unacceptable outcomes. This agrees with common notions of risk – that when a bet is made, if it has a chance of losing money from this bet then the bet has a risk. While the inverse is true – a bet where you are happy with the outcome of every possibility by definition contains no risk.

*Risk aversion* can be thought of as a general preference for some random variables over others. A particular risk attitude can be characterised as a preference relation on the set of random variables. We can say  $X_1 > X_2$  if a risk attitude prefers the random variable  $X_1$  over the random variable  $X_2$ . Many researchers in stochastic programming and financial mathematics have contributed to the mathematical characterisation of these concepts; in this chapter, we focus mainly on the work of Artzner, Delbaen, Eber, and Heath (1999) and Ruszczynski and Shapiro (2006b). The ideas presented in this chapter will be important for the understanding of the remainder of this thesis.

#### 2.1 Acceptance sets

The most fundamental object in the study of risk aversion is an *acceptance set*. In stochastic optimisation and other fields (such as financial mathematics), a decision maker may model uncertain profits (or losses) as a random variable on a particular

probability space. Consider a space of ‘positions’  $\mathcal{X}$  on some probability space  $(\Omega, \mathcal{F}, \mathbb{P})$  that an agent could be exposed to. Here a position refers to a random variable  $X$  on some probability space  $(\Omega, \mathcal{F}, \mathbb{P})$ , offering some financial return. An acceptance set  $\mathcal{A}$  is a subset of  $\mathcal{X}$ , such that the agent would agree to be exposed to any random variable (position) in  $\mathcal{A}$  without compensation. Of course, the agent is free to arbitrarily include any position in his acceptance set. However, Artzner, Delbaen, Eber, and Heath (1999) propose the following axioms of what we call *coherent* acceptance sets; conditions that a sensible agent ought to accept. They are the following:

$$\mathcal{A} \supseteq \mathcal{X}^+, \quad (2.1)$$

$$\mathcal{A} \cap \mathcal{X}^- = \{0\}, \quad (2.2)$$

$$\mathcal{A} \text{ is a convex cone,} \quad (2.3)$$

where

$$\mathcal{X}^+ := \{X \in \mathcal{X} \mid X(\omega) \geq 0, \forall \omega \in \Omega\}, \quad (2.4)$$

and  $\mathcal{X}^- := -\mathcal{X}^+$ . Condition (2.1) says that a position which offers a positive return in every realisation (i.e.  $\forall \omega \in \Omega$ ) should be in the acceptance set. Condition (2.2) says that the only acceptable position offering a non-positive return in every realisation is the null position  $0$ . Condition (2.3) says that if  $X_1$  and  $X_2$  are both acceptable, then  $X_1\alpha + X_2\beta$ ,  $\forall \alpha, \beta > 0$  should be acceptable. Figure 2.1 illustrates such an acceptance set. With this construction, we can immediately form a crude preference relation of

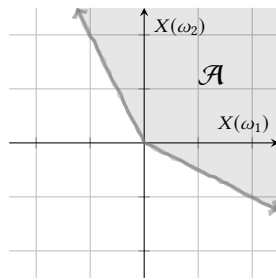


Figure 2.1: A convex cone  $\mathcal{A}$  in  $\mathbb{R}^2$ .

the set  $\mathcal{X}$  (this being our ultimate goal). We can say that

$$X_1 \in \mathcal{A} \text{ and } X_2 \in \mathcal{X} \setminus \mathcal{A} \iff X_1 > X_2, \forall X_1, X_2 \in \mathcal{X}.$$

The above relation is too coarse, it gives us no way to compare random variables that are both within (or both outside) the acceptance set. For this, we will introduce the notion of a *risk measure* – a function which measures how acceptable a position is rather than just whether or not the random variable is acceptable.

## 2.2 Risk measures

The ultimate end of acceptance sets is to define risk measures. Much like an acceptance set, an agent can measure risk in any way she likes. However, in this section we will focus on those (like acceptance sets) which a sensible agent ought to accept. We will use  $\rho : \mathcal{X} \mapsto \mathbb{R}$  to denote a measure of risk, where  $\mathcal{X}$  is a space of random variables on some probability space  $(\Omega, \mathcal{F}, \mathbb{P})$ . The idea of measuring risk can be traced back to the foundational work of Markowitz (1952) who sought to rank portfolios in terms of their mean and variance. We shall see, however, while a foundational piece of work, this approach has several flaws. We define a the risk measure of a particular acceptance set  $\mathcal{A}$  as the following.

**Definition 2.1.**

The function  $\rho_{\mathcal{A}} : \mathcal{X} \mapsto \mathbb{R}$  is a risk measure with

$$\rho_{\mathcal{A}}(X) = \inf_m \{m \mid m \times \mathbb{1} + X \in \mathcal{A}\}$$

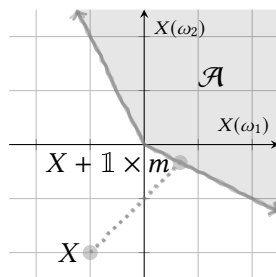
where  $\mathbb{1}(\omega) = 1, \forall \omega \in \Omega$ .

The risk of a position can be thought of as the amount of a ‘sure’ position that would be required to add to the position  $X$  such that it would become acceptable. If the position  $X$  is already acceptable, then the risk of the position  $\rho_{\mathcal{A}}(X)$  is non-positive; it then measures the amount of a sure position that could subtracted from  $X$  and still have the position be acceptable. Figure 2.2 demonstrates this concept on a finite probability space  $(\{\omega_1, \omega_2\}, 2^{\{\omega_1, \omega_2\}}, \mathbb{P})$  with an arbitrary acceptance set  $\mathcal{A}$ . We can also define acceptance sets in terms of risk measures.

**Definition 2.2.**

The set  $\mathcal{A}_{\rho} \subset \mathcal{X}$  is an acceptance set such that

$$\mathcal{A}_{\rho} = \{X \in \mathcal{X} \mid \rho(X) \leq 0\}.$$



It is easy to show that for any acceptance set  $\mathcal{A}$ , the following property holds:

$$\mathcal{A} = \mathcal{A}_{\rho_{\mathcal{A}}}.$$

Risk measures allow us to refine our ordering of random variables. We can now say that

$$\rho(X_1) \leq \rho(X_2) \iff X_1 \geq X_2.$$

In some situations, it may be more intuitive to speak of the acceptability of random variable, rather than its riskiness. We define an acceptance measure  $\nu(X) = -\rho(X)$ . This allows for the somewhat more natural relation

$$v(X_1) \geq v(X_2) \iff X_1 \geq X_2. \quad (2.5)$$

An interesting question to ask now is: “Under what conditions on  $\rho$  do we recover sensible acceptance sets under Definition 2.2?” This question motivates the following subset of risk measures developed in Artzner, Delbaen, Eber, and Heath (1999).

### 2.2.1 Coherent risk measures

An important class of risk measures are those called *coherent risk measures*. Similar to acceptance sets, they restrict the form of risk measures to those which a sensible agent ought to accept. A coherent risk measure is a risk measure which is:

1. monotonic i.e.  $X \geq Y$  *almost surely*  $\implies \rho(X) \leq \rho(Y)$ ;
2. translationally invariant i.e.  $\rho(X + \mathbb{1}a) = \rho(X) - a$ ;
3. convex i.e.  $\rho(X + Y) \leq \rho(X) + \rho(Y)$ ;
4. positively homogeneous i.e.  $\rho(\alpha X) = \alpha \rho(X)$ ,  $\alpha \geq 0$ .

One can easily verify that the convex combination of coherent risk measures is also convex. It can be seen that through the relationship

$$\mathcal{A} = \{X \in \mathcal{X} \mid \rho(X) \leq 0\},$$

that an acceptance set which is a convex cone containing  $\mathcal{X}^+$ . Note that the equivalent conditions for an acceptance measure are given by

1. monotonic i.e.  $X \leq Y$  *almost surely*  $\implies \nu(X) \leq \nu(Y)$ ;
2. translationally invariant i.e.  $\nu(X + \mathbb{1}a) = \nu(X) + a$ ;
3. concave i.e.  $\nu(X + Y) \geq \nu(X) + \nu(Y)$ ;
4. positively homogeneous i.e.  $\nu(\alpha X) = \alpha \nu(X)$ ,  $\alpha \geq 0$ .

$\nu$  is concavity, and the condition of translational invariance gives  $\nu(X + \mathbb{1}a) = \nu(X) + a$ .

#### Example 2.1.

The function  $-\mathbb{E}_{\mathbb{Q}}[\cdot]$  where  $\mathbb{Q}$  is any probability measure on the appropriate measure space is a coherent risk measure. Its acceptance set is all random variables whose expectation under the measure  $\mathbb{Q}$  is greater than zero. The function  $-\text{ess inf}[\cdot]$  is also a coherent risk measure. We use the essential infimum here, since events of probability zero ought to be ignored by our risk measure. The acceptance set of  $-\text{ess inf}[\cdot]$  is  $\mathcal{X}^+$ , necessarily the least accepting risk measure.

In the example above, the risk measure's corresponding acceptance measures are given by  $\mathbb{E}_{\mathbb{P}}[\cdot]$  and  $\text{ess inf}[\cdot]$ <sup>1</sup>

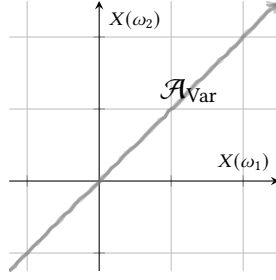
For the following insight we require the next definition.

#### Definition 2.3.

Consider a random variable  $X \in \mathcal{X}$  on some probability space. Define the variance of a random variable  $\text{Var} : \mathcal{X} \mapsto \mathbb{R}$  as

$$\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2].$$

<sup>1</sup>Speaking in terms of acceptance measures has the notational advantage of forgoing the preceding ‘−’ sign; a small gain for those wary of lingering ‘−’ signs.

Figure 2.3: Acceptance set for  $\text{Var}(X)$ .

Note that for a given  $X$ ,  $\text{Var}(X)$  may not be defined, since the integral defining the outer expectation may not be bounded. This is the motivation for the random variable space  $\mathcal{X}$  being restricted to  $\mathcal{L}^2(\Omega, \mathcal{F}, \mathbb{P})$ . The function  $\text{Var}(X)$  is not a coherent risk measure. Figure 2.3 shows the acceptance set of  $\text{Var}(X)$ ; immediately, we can see irrational patterns of acceptance – the agent would rather accept the position  $-\mathbb{1}$  than  $X(\omega_1) = 1, X(\omega_2) = 2$ . Indeed, we can extend this criticism to the seminal work of Markowitz (1952). For a given mean-variance risk measure

$$\rho(X) = \lambda \text{Var}(X) - (1 - \lambda) \mathbb{E}(X),$$

it is trivial to construct a random variable  $X \in \mathcal{X}^+$  which has an arbitrarily large variance and so is not in the risk measure's acceptance set i.e.  $\rho(X) \leq 0$ .

### 2.2.2 Stochastic dominance and law invariance

A useful way of characterising a real-valued random variable is through a *cumulative distribution function*. We provide the usual definition below.

**Definition 2.4.**

Define the cumulative distribution function  $F_X : \mathbb{R} \mapsto [0, 1]$  for a real-valued random variable  $X$  on some probability space  $(\Omega, \mathcal{F}, \mathbb{P})$  as

$$F_X(x) = \mathbb{P}\left(X^{-1}((-\infty, x])\right).$$

A closely related function, called the *quantile function*, can be also be defined.

**Definition 2.5.**

Define the quantile function  $Q_X : (0, 1) \mapsto \mathbb{R}$  for a real-valued random variable  $X$  on some probability space  $(\Omega, \mathcal{F}, \mathbb{P})$  as

$$Q_X(q) = \inf\{x \mid q \leq F_X(x)\}.$$

If  $F_X$  is continuous and strictly monotonically increasing, then  $Q_X = F_X^{-1}$ . With these definitions, we will introduce the concept of *stochastic dominance* and *law invariance*. Stochastic dominance is a partial ordering of random variables; if a random variable  $X_1$  is no less than another random variable  $X_2$ , at every quantile, then  $X_2$  ought not to stochastically dominate  $X_1$ . In terms of quantile functions, we have

$$Q_{X_1}(q) \geq Q_{X_2}(q), \forall q \in (0, 1) \iff X_1 \overset{\text{sd}}{\geq} X_2.$$

A natural strengthening of this property can be given by the following condition;

$$Q_{X_1}(q) \geq Q_{X_2}(q), \forall q \in (0, 1) \text{ and } \exists q^* \in (0, 1) \mid Q_{X_1}(q^*) > Q_{X_2}(q^*) \iff X_1 \overset{\text{sd}}{>} X_2.$$

We say a coherent acceptance measure  $\nu$  preserves stochastic dominance if

$$X_1 \overset{\text{sd}}{\geq} X_2 \implies \nu(X_1) \geq \nu(X_2), \forall X_1, X_2 \in \mathcal{X}.$$

Clearly this is a desirable property for our risk measures to have.

A further condition on ‘sensible’ acceptance sets that is not included in the work of Artzner, Delbaen, Eber, and Heath (1999) is the condition of *law invariability*. We define it as the following:

$$Q_{X_1}(q) = Q_{X_2}(q), \forall q \in (0, 1) \implies \nu(X_1) = \nu(X_2), \forall X_1, X_2 \in \mathcal{X}.$$

Note that some authors define law invariance through the cumulative distribution function i.e.

$$F_{X_1}(x) = F_{X_2}(x), \forall x \in \mathbb{R} \implies \nu(X_1) = \nu(X_2), \forall X_1, X_2 \in \mathcal{X}.$$

Our definition is equivalent as random variables that share the same cumulative distribution function share the quantile function. Note that two random variables can share the same quantile function without being equal to each other in the sense that  $X_1(\omega) = X_2(\omega)$ ,  $\forall \omega \in \Omega$ . Our acceptance measure does not care what outcome occurs on the probability space  $\omega$  but cares only for the resulting value of the random variable of interest  $X(\omega)$ .

In the following lemma, we prove that the law invariance of a coherent acceptance measure is a necessary and sufficient requirement for the acceptance measure to preserve stochastic dominance. This has been proven by Ruszczyński and Shapiro (2006b) as well as other stochastic dominance results. We provide our own proof for illustrative purposes.

**Lemma 2.1.**

Consider an atomless probability space  $(\Omega, \mathcal{F}, \mathbb{P})$  with an associated random variable space  $\mathcal{X}$ . A coherent acceptance measure  $\nu : \mathcal{X} \mapsto \mathbb{R}$  preserves stochastic dominance if and only if it is law invariant.

*Proof.* The proof is elementary. Suppose  $\nu$  preserves stochastic dominance but is not law invariant. Then there exist two random variables  $X_1$  and  $X_2$  with the same distribution (and so are equal in the stochastic dominance sense) for which  $\nu(X_1) \neq \nu(X_2)$ , and so either  $\nu(X_1) < \nu(X_2)$  or  $\nu(X_2) < \nu(X_1)$ . Without loss of generality, say we have  $\nu(X_1) < \nu(X_2)$ . Now consider a new random variable

$$X_3(\omega) = X_1(\omega) + \mathbb{1}(\omega) \times \frac{\nu(X_2) - \nu(X_1)}{2}.$$

Clearly  $X_3$  stochastically dominates  $X_1$  (and so stochastically dominates  $X_2$ ). But from the translational invariance property of coherent acceptance measures, we have

$$\nu(X_3) = \nu(X_1) + \frac{\nu(X_2) - \nu(X_1)}{2} = \frac{\nu(X_1) + \nu(X_2)}{2} < \nu(X_2).$$

This means that the acceptance measure  $\nu$  does not have the stochastic dominance preservation property, creating a contradiction. This establishes the forward direction. The backwards direction is as follows. Suppose  $\nu$  is law invariant, but does not preserve stochastic dominance. Then there exists two random variables for which

$$X_1 \stackrel{\text{sd}}{\preceq} X_2 \text{ and } \nu(X_1) > \nu(X_2), \quad X_1, X_2 \in \mathcal{X}.$$

But by the main result of Lindvall (1999) we have that there exists a random variable  $X'_2 \in \mathcal{X}$  such that  $X'_2$  is equal to  $X_2$  in distribution and (due to the stochastic dominance of  $X_2$  over  $X_1$ ) that  $X_1(\omega) \leq X'_2(\omega)$ ,  $\forall \omega \in \Omega$ . We note here that this result only holds for atomless spaces. By the monotonicity property



of coherent risk measures, we then have

$$v(X'_2) \geq v(X_1) > v(X_2).$$

But from  $v$ 's law invariance we have  $v(X'_2) = v(X_2)$ , forming a contradiction and completing the proof.  $\square$

By this lemma, we should conclude that we ought to only use law invariant risk/acceptance measures in optimisation, lest we prefer a random variable  $X_1$  to  $X_2$  and yet  $X_2$  stochastically dominates  $X_1$ . The next subsection introduces a class of law invariant risk/acceptance measures.

### 2.2.3 The conditional value at risk

An early and popular method of measuring the risk of random variables was *value-at-risk* or VaR<sup>2</sup>. We define the acceptance measure version of VaR at some quantile  $\beta$  as the quantile function of the random variable  $X$  at the quantile  $\beta$  i.e.

$$\text{VaR}_\beta(X) = Q_X(\beta).$$

and so we have  $\text{VaR}_\beta(X) = -Q(\beta)$ , as the risk measure version. In general, VaR is *not* a coherent risk measure; it lacks the convexity property. Consider our simple probability space  $(\{\omega_1, \omega_2\}, 2^{\{\omega_1, \omega_2\}}, \mathbb{P})$  where  $\mathbb{P}$  is the discrete uniform measure. When VaR is greater than  $\frac{1}{2}$ , we have  $\text{VaR} = \max$ . Because, in general,  $\max\{X\} + \max\{Y\} \geq \max\{X + Y\}$ , we fail to obtain the convexity condition required by coherent risk measures. The acceptance set for the  $\text{VaR}_\beta$  risk measure is shown below with  $\beta = \frac{1}{2}$ . As shown in Figure 2.4, the resulting acceptance set  $\mathcal{A}_{\text{VaR}}$  is *not* convex; the agent is willing to accept  $\omega_1 \rightarrow 0, \omega_2 \rightarrow -2$  and  $\omega_1 \rightarrow -2, \omega_2 \rightarrow 0$  but not  $\omega_1 \rightarrow -1, \omega_2 \rightarrow -1$ . For this reason, VaR has fallen out of favour as a means of measuring the risk of random variables. However, a close relative of the *value-at-risk* measure is the *conditional value-at-risk* or CVaR. Conditional value-at-risk is the most widely discussed and used coherent risk/acceptance measure. We can define it in several ways, one of the most intuitive being the following.

#### Definition 2.6.

Consider a probability space  $(\Omega, \mathcal{F}, \mathbb{P})$  with an associated random variable

<sup>2</sup>We use the convention that 'VaR' refers to value-at-risk while Var refers to variance.

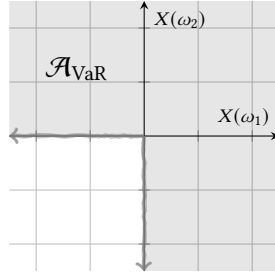


Figure 2.4: Acceptance set for VaR(X).

space  $\mathcal{X}$ . We define the acceptance measure  $\text{CVaR}_\beta^+ : \mathcal{X} \mapsto \mathbb{R}$ , with  $\beta \in (0, 1]$  as

$$\text{CVaR}_\beta^+(X) = \frac{1}{\beta} \int_0^\beta Q_X(q) dq, \quad \forall X \in \mathcal{X}.$$

For  $\beta = 0$ , we define  $\text{CVaR}_\beta^+(X) = \text{ess inf}(X)$ .

We call  $\beta$  the quantile of the CVaR measure; it acts as a parameterisation. The next example below explores some simple properties of the CVaR acceptability measure.

#### Example 2.2.

The CVaR for a given random variable  $X$  for different quantiles are given below.

For  $\beta = 1$ , we have

$$\text{CVaR}_1^+(X) = \frac{1}{1} \int_0^1 Q_X(q) dq = \mathbb{E}_{\mathbb{P}}[X],$$

and for  $\beta = 0$ , we have

$$\lim_{\beta \rightarrow 0} \text{CVaR}_\beta^+(X) = \lim_{\beta \rightarrow 0} \frac{1}{\beta} \int_0^\beta Q_X(q) dq = \text{ess inf}(X).$$

Now consider a random variable  $X$  with quantile function  $Q_X(q) = q$ ,  $\forall q \in (0, 1)$ . Then we have  $\text{CVaR}_0^+(X) = 0$  and  $\text{CVaR}_1^+(X) = 1/2$ . For any quantile  $\beta$ , we have

$$\text{CVaR}_\beta^+(X) = \frac{1}{\beta} \int_0^\beta q dq = \frac{\beta}{2}.$$

By adjusting the parameter  $\beta$ , we are able to capture the most extreme cases of risk aversion. Clearly CVaR is a law invariant acceptability measure, since it depends only on the quantile function.

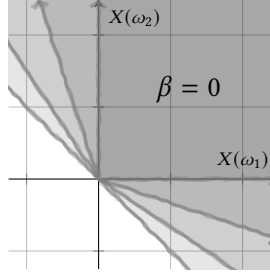


Figure 2.5: A range of acceptance sets for different CVaR quantiles.

There are many (equivalent) definitions of CVaR. On atomless spaces, one natural interpretation which arrives from Definition 2.6, is

$$\text{CVaR}_\beta(X) = \mathbb{E}[X \mid X \leq Q_X(\beta)].$$

where  $\mathbb{E}[X \mid X \leq Q_X(\beta)]$  is a short hand for the expectation under the conditional probability measure generated by the event  $X^{-1}((-\infty, Q_X(\beta)])$ . That is,  $\text{CVaR}(X)$  is the expectation of the random variable  $X$  conditioned on the  $X$  taking values at or below its worst  $\beta$ -quantile. A further definition is given by

$$\text{CVaR}_\beta(X) = \max_{\xi \in \mathbb{R}} \xi - \frac{1}{\beta} \mathbb{E}[\max\{\xi - X, 0\}],$$

which is attributed to Rockafellar and S. Uryasev (2000). Although less natural, this definition has the attractive property of being easier implemented as a linear programme if the underlying probability space is finite. We will discuss some further attractive properties of this definition in a later section. Pflug (2000) proved that CVaR is indeed a coherent risk measure.

## 2.3 A duality result

One of the most important results in the theory of coherent risk measures is the *dual representation* of coherent risk measures. The result gives a way of characterising all coherent risk measures on a given probability space. However, before we present the duality theorem, we will first introduce several concepts from convex analysis.

**Definition 2.7.**

Given some real topological vector space  $\mathcal{X}$ , and dual space  $\mathcal{Y}$  which contains all linear continuous functions  $\langle \mu, \cdot \rangle : \mathcal{X} \mapsto \mathbb{R}$ , let the dual cone  $\mathcal{A}^*$  of a set

$\mathcal{A} \subseteq \mathcal{X}$  be the set

$$\mathcal{A}^* = \{\mu \in \mathcal{Y} \mid \langle \mu, X \rangle \geq 0, \forall X \in \mathcal{A}\}.$$

Notice immediately, that the dual cone of the dual cone of  $\mathcal{A}$  (denoted  $\mathcal{A}^{**}$ ) is the smallest closed convex cone containing :  $\mathcal{A}$ . Figure 2.6 illustrates this graphically.

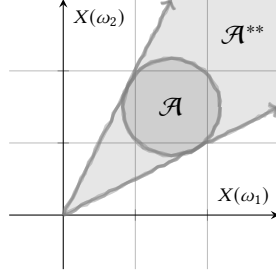


Figure 2.6: A set  $A$  and its convex cone closure.

Theorem 2.1 (The risk duality theorem).

For an acceptance set  $\mathcal{A}$  which is a closed convex cone containing  $\mathcal{X}^+$ , we have

$$\rho_{\mathcal{A}}(X) = \sup_{\mu \in Q_{\mathcal{A}}} \langle \mu, -X \rangle, \quad \forall X \in \mathcal{X},$$

where  $Q_{\mathcal{A}}$  is subset of the dual cone of  $\mathcal{A}$  for which  $\langle \mu, \mathbb{1} \rangle = 1$ .

Ruszczynski and Shapiro (2006b) provide a proof of a similar but equivalent theorem; their theorem is stated in terms of risk measures rather than acceptance sets and makes use of the Fenchel-Moreau Theorem and convex conjugate function  $\rho^*$  of the risk measure  $\rho$ . Equivalently, the same result can be arrived at by considering the acceptance sets and their dual cones. Heuristically, the theorem holds because  $\mathcal{A}$  is a closed convex cone and so  $\mathcal{A} = \mathcal{A}^{**}$ . Stated in terms of acceptance measures, we have

$$v_{\mathcal{A}}(X) = -\rho_{\mathcal{A}}(X) = - \sup_{\mu \in Q_{\mathcal{A}}} \langle \mu, -X \rangle = \inf_{\mu \in Q_{\mathcal{A}}} \langle \mu, X \rangle, \quad \forall X \in \mathcal{X}.$$

The following lemmas show that the set  $Q$  is a special subset of the space  $\mathcal{Y}$ .

Lemma 2.2.

Consider a convex cone  $\mathcal{A}$  containing  $\mathcal{X}^+$ . The dual cone  $\mathcal{A}^*$  is contained within  $\mathcal{Y}^+$ .

*Proof.* By definition, the dual cone is given by

$$\mathcal{A}^* = \{\mu \in \mathcal{Y} \mid \langle \mu, X \rangle \geq 0, \forall X \in \mathcal{A}\}.$$

If  $\mathcal{A}^*$  is not contained within  $\mathcal{Y}^+$ , then there exists a  $\mu' \in \mathcal{A}^*$  and  $\mu' \notin \mathcal{Y}^+$  such that  $\langle \mu', X \rangle < 0$ , for some  $X \in \mathcal{X}^+$ . Because  $\mathcal{A}$  contains  $\mathcal{X}^+$ , this inequality contradicts the condition of  $\mu'$  being a member of  $\mathcal{A}^*$ , concluding the proof.  $\square$

Lemma 2.3.

The set  $\mathcal{Q}_{\mathcal{A}}$  contains only probability measures.

*Proof.* The set  $\mathcal{A}^*$  contains only linear functions which are positive, i.e. in  $\mathcal{Y}^+$ . Secondly, the restriction  $\langle \mu, \mathbb{1} \rangle = 1$  is the satisfaction of  $\mu(\Omega) = 1$ . So  $\mu$  is a probability measure, concluding the proof.  $\square$

The key result here is that all coherent acceptance measures can be represented by a ‘risk set’  $\mathcal{Q}$ ; a set of probabilities against which an expectation can be taken. Further, any set of probability measures over which the minimal expectation is computed will be a coherent risk measure. This has the interpretation of the “Nervous Nellie” effect: risk aversion can be viewed as an individual viewing negative outcomes as more likely. This powerful representation is critical to many of the algorithms presented in this thesis.

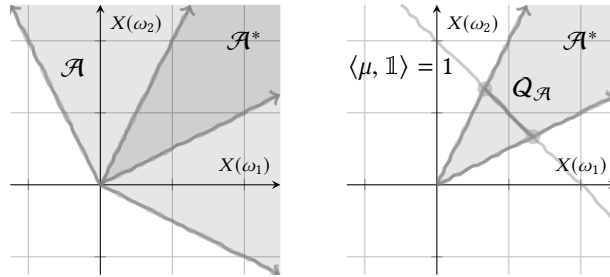


Figure 2.7: The dual cone  $\mathcal{A}^*$  and its subset of probability measures  $\mathcal{Q}_{\mathcal{A}}$ .

Example 2.3 builds intuition of Theorem 2.1 constructs the risk sets of several acceptance measures.

Example 2.3.

The acceptance set of the  $\text{ess inf}[\cdot]$  acceptance measure is  $\mathcal{X}^+$ . The set  $\mathcal{X}^+$  is

*self dual* i.e. the dual cone of  $\mathcal{X}^+$  is  $\mathcal{Y}^+$ . So the risk set of  $\text{ess inf}[\cdot]$  is

$$\mathcal{Q}_{\text{ess sup}} = \{\mu \in \mathcal{Y}^+ \mid \langle \mu, \mathbb{1} \rangle = 1\};$$

this is the set of all possible probability measures! The dual cone of the  $\mathbb{E}[\cdot]$  acceptance measure is the ray

$$\begin{aligned} \mathcal{A}_{\mathbb{E}}^* &= \{\mu \in \mathcal{Y} \mid \langle \mu, X \rangle \geq 0, \mathbb{E}[X] \geq 0, \forall X \in \mathcal{X}\}, \\ &= \{\mu \in \mathcal{Y} \mid \exists \alpha \in [0, \infty) : \mu = \alpha \mathbb{P}\}. \end{aligned}$$

So the risk set becomes

$$\mathcal{Q}_{\mathbb{E}} = \{\mu \in \mathcal{A}_{\mathbb{E}}^* \mid \langle \mu, \mathbb{1} \rangle = 1\} = \{\mathbb{P}\}.$$

This, of course, agrees with our intuition; the relation

$$\mathbb{E}_{\mathbb{P}}[X] = \langle \mathbb{P}, X \rangle = \inf_{\mu \in \mathcal{Q}_{\mathbb{E}}} \langle \mu, X \rangle,$$

only holds for all  $X \in \mathcal{X}$  if  $\mathcal{Q}_{\mathbb{E}}$  is a singleton containing the base measure.

The risk set for CVaR was first outlined in Rockafellar, S. P. Uryasev, et al. (2002) and is given by

$$\mathcal{Q}_{\text{CVaR}_{\beta}} = \{\mu \in \mathcal{Y} \mid 0 \leq \frac{d\mu}{d\mathbb{P}}(\omega) \leq \frac{1}{\beta}, \int_{\Omega} d\mu(\omega) = 1\}.$$

Notice that this representation relies on the existence of the Radon-Nikodym derivative  $d\mu/d\mathbb{P}$ , which requires that  $\mu$  is absolutely continuous with respect to  $\mathbb{P}$ . Because  $\text{CVaR}_{\beta}$  can be viewed as a conditional expectation, we satisfy the absolutely continuous requirement since all conditional probability measures  $\mu$  are absolutely continuous with respect to the base measure  $\mathbb{P}$  i.e.

$$\mathbb{P}(A) = 0 \implies \mu(A) = 0, \forall A \in \mathcal{F}.$$

### 2.3.1 Kusuoka representation

As mentioned earlier, coherent risk measures allow for risk measure which are not law invariant i.e. is *not* true that if two random variables have the same distribution, then same risk. The duality theorem from the previous section characterises all coherent risk measure be they law invariant or not. An interesting question to ask is: “given the definition of coherent risk measure, how can we characterise the entire

space of law invariant coherent risk measures?” An answer to this question was provided by Kusuoka (2001). They showed that any coherent risk measure can be written in the form of the Lebesgue integral

$$\nu(X) = \inf_{f \in \mathcal{P}} \int_0^1 \text{CVaR}_\beta^+(X) df(\beta),$$

where  $\mathcal{P}$  is a set of probability measures on the space  $((0, 1), \mathcal{B}((0, 1)))$ . In this sense, we can consider the  $\text{CVaR}^+$  acceptance measure as a ‘basis function’. This reduces much of the study of law invariant risk measure to the study of  $\text{CVaR}^+$ . Now that we have introduced coherent risk measures and introduced  $\text{CVaR}$ , we can further define certain groups of risk measures. One set of risk measures is called spectral risk measures and is defined as follows.

**Definition 2.8.**

An acceptance measure  $\nu$  is called *spectral* if it can be represented as

$$\nu(X) = \int_0^1 \text{CVaR}_\beta^+(X) df(\beta),$$

where  $f(\beta)$  is a probability measure on the space  $((0, 1), \mathcal{B}([0, 1]))$ .

In terms of the Kusuoka representation, we have that  $\mathcal{P}$  is a singleton. Spectral risk measures have the following property.

**Definition 2.9.**

Two random variables  $X_1$  and  $X_2$  are *comonotonic* if

$$(X_1(\omega) - X_1(\omega'))(X_2(\omega) - X_2(\omega')) \geq 0, \quad (2.6)$$

almost surely under the product measure  $\mathbb{P} \otimes \mathbb{P}$ .

**Definition 2.10.**

An acceptance measure  $\nu$  is called *comonotonic* if for two comonotonic random variables  $X_1$  and  $X_2$  we have

$$\nu(X_1) + \rho(X_2) = \rho(X_1 + X_2).$$

It follows that all spectral risk measures are comonotonic. That is, if two random variables share the the same minimising probability measure, then a comonotonic risk measure measures the risk of the sum as the sum of the risk.

## 2.4 Risk in optimisation

Ultimately, the goal of forming an ordering on random variables is so that some optimisation procedure can select the one which is the most favourable. Being able to correctly characterise the risk measure means that an optimisation procedure does actually select the most favourable random variable. This has been the concern of the previous sections. In this section, we show how these risk measures are incorporated into optimisation problems. Suppose that our random variable  $Z$  is now also a function of some control  $v \in \mathcal{V}$ . The risk neutral problem (the one considered in traditional stochastic programming) is given by

$$\max_{v \in \mathcal{V}} \mathbb{E}_{\mathbb{P}}[Z(v, \omega)]$$

If the agent seeks the most acceptable random variable, rather than the random variable with the greatest expectation, then she simply solves

$$\max_{v \in \mathcal{V}} v(Z(v, \omega)). \quad (2.7)$$

In this section we will outline two common methods of solving (2.7).

### 2.4.1 Minimax evaluation

One of the most natural methods incorporating coherent risk measures into an optimisation problem is through the dual representation of coherent risk measures. On an atomless probability space, recall that the risk set for CVaR is given by

$$\mathcal{Q}_{\text{CVaR}_\beta} = \left\{ \mu \in \mathcal{Y} \mid \int_{\Omega} d\mu(\omega) = 1, 0 \leq \frac{d\mu}{d\mathbb{P}} \leq \frac{1}{\beta} \right\}.$$

For a finite probability space, a discrete analogue holds; the risk set is given by

$$\mathcal{Q}_{\text{CVaR}_\beta} = \left\{ \mu \in \mathcal{Y} \mid \sum_{\omega \in \Omega} \mu(\omega) = 1, 0 \leq \frac{\mu(\omega)}{\mathbb{P}(\omega)} \leq \frac{1}{\beta}, \forall \omega \in \Omega \right\}.$$

So in a standard linear programming form, we have

$$\begin{aligned} \min_{\mu(\omega)} \quad & \sum_{\omega \in \Omega} \mu(\omega) Z(\omega) \\ \text{s.t.} \quad & 0 \leq \frac{\mu(\omega)}{\mathbb{P}(\omega)} \leq \frac{1}{\beta}, \forall \omega \in \Omega, \\ & \sum_{\omega \in \Omega} \mu(\omega) = 1. \end{aligned} \quad (2.8)$$



It can also be useful to frame the problem in *risk envelope* form:

$$\begin{aligned}
\min_{\eta(\omega)} \quad & \sum_{\omega \in \Omega} \eta(\omega) \mathbb{P}(\omega) Z(\omega) \\
\text{s.t.} \quad & 0 \leq \eta(\omega) \leq \frac{1}{\beta}, \quad \forall \omega \in \Omega, \\
& \sum_{\omega \in \Omega} \eta(\omega) \mathbb{P}(\omega) = 1.
\end{aligned} \tag{2.9}$$

The function  $\eta(\omega)$  is called the risk envelope, the amount that each probability is scaled in order to map the original probability to match the minimising probability i.e.  $\eta(\omega) \mathbb{P}(\omega) = \mu(\omega)$ . In the continuous setting, the risk envelope  $\eta(\omega)$  can be thought of as  $d\mu/d\mathbb{P}$  in the Radon-Nikodym derivative sense. Note that not all coherent risk measures have a risk envelope form as envelope form requires that the measures in the risk set  $\mu \in \mathcal{Q}$  are absolutely continuous with respect to the base measure  $\mathbb{P}$ . However, any law invariant risk measure can be written in envelope form.

Recall the optimisation problem in mind; we have

$$\max_{v \in \mathcal{V}} \inf_{\mu \in \mathcal{Q}_{\text{CVaR}}} \langle \mu, Z(v) \rangle.$$

If  $v = \text{CVaR}_\beta$  then by an appeal to the dual representation of risk as given by Theorem 2.1, we can write the above as

$$\begin{aligned}
z^* = \max_{v \in \mathcal{V}} \min_{\eta(\omega)} \quad & \sum_{\omega \in \Omega} \eta(\omega) \mathbb{P}(\omega) Z(v, \omega) \\
\text{s.t.} \quad & 0 \leq \eta(\omega) \leq \frac{1}{\beta}, \quad \forall \omega \in \Omega, \\
& \sum_{\omega \in \Omega} \eta(\omega) \mathbb{P}(\omega) = 1.
\end{aligned} \tag{2.10}$$

Notice that for a given  $v \in \mathcal{V}$ , the inner minimisation problem is linear in its structure; this relies on the finiteness of  $\Omega$ . We can think of this method as modelling a simultaneous game between the agent and ‘nature’. The agent picks the action  $v \in \mathcal{V}$  which maximises her expected payoff for the given probability measure, and nature picks a probability measure  $\mu$  from the risk set which minimises the agent’s expected payoff. Of course, nature is a metaphor for the agent’s own risk aversion, she chooses her own risk set as a method of simulating her mistrust in events occurring in the way that the base measure  $\mathbb{P}$  prescribes.

This type of formulation has several other interpretations, in particular one from *robust optimisation*. In a very general form, we can write a robust optimisation

problem as

$$\max_{v \in \mathcal{V}} \min_{u \in \mathcal{U}} g(u, v).$$

The minimisation set  $\mathcal{U}$  is often called the uncertainty set. In our case, the uncertainty set is the risk set; the set of probability measures over which an expectation is taken over and  $f$  is the expectation operator. Robust optimisation problems of this form are often called *distributionally robust optimisation*.

### 2.4.2 Rockafellar evaluation

As briefly mentioned earlier, Rockafellar and S. Uryasev (2000) developed an equivalent definition of CVaR in the form of a linear optimisation problem:

$$\text{CVaR}_\beta(Z(v, \omega)) = \max_{\xi \in \mathbb{R}} \xi - \frac{1}{\beta} \mathbb{E}[\max\{\xi - Z(v, \omega), 0\}];$$

By using this definition of CVaR and seeking to maximise the acceptability of a random variable  $Z(v, \omega)$  over  $v \in \mathcal{V}$ , we arrive at

$$\max_{v \in \mathcal{V}} \text{CVaR}_\beta(Z(v, \omega)) = \max_{v \in \mathcal{V}} \max_{\xi \in \mathbb{R}} \xi - \frac{1}{\beta} \mathbb{E}[\max\{\xi - Z(v, \omega), 0\}].$$

By ‘merging on maxes’ (owing to the linearity of the inner problem), we arrive at

$$\max_{v \in \mathcal{V}} \text{CVaR}_\beta(Z(v, \omega)) = \max_{(v, \xi) \in (\mathcal{V}, \mathbb{R})} \xi - \frac{1}{\beta} \mathbb{E}[\max\{\xi - Z(v, \omega), 0\}];$$

a single optimisation problem. If  $Z(v, \omega)$  has the appropriate concavity conditions outlined at the beginning of this section, then whole problem is a convex optimisation problem.

### 2.4.3 Risk in constraints

So far, we have only considered risk in the objective function of our optimisation problems. This is because our original philosophy on risk was to perform an ordering of the random variable and use some optimisation routine to select the most preferred one. Another natural use of risk in optimisation problems is the use of risk in constraints. This idea stems directly from the notion of an acceptance set. Rather than seeking the most acceptable random variable, the agent could solve

$$\begin{aligned} \max_{v \in \mathcal{V}} \quad & f(v) \\ \text{s.t.} \quad & \rho(Z(v, \omega)) \leq b \end{aligned} \tag{2.11}$$

which, from the definition of acceptance sets, equivalent to

$$\begin{aligned} \max_{v \in \mathcal{V}} \quad & f(v) \\ \text{s.t.} \quad & Z(v, \omega) - b \times \mathbb{1} \in \mathcal{A}_\rho \end{aligned} \tag{2.12}$$

A drawback of incorporating risk into optimisation problems in the constraints is that it is not clear as to whether or not a given instance of the problem is feasible for a given value of  $b$ . One strategy to mitigate such an infeasibility would be to penalise any infeasibility in the objective function. In doing so, the penalised version of the problem takes the same form as (2.7).

Another type of problem which incorporates risk in constraints is *probabilistic constrained optimisation* or *optimisation with chance constraints*. An agent may seek to maximise some reward function  $f$  subject to the probability of violating some constraint being below some threshold. In mathematical form, we have

$$\begin{aligned} z^* = \max_{v \in \mathcal{V}} \quad & f(v) \\ \text{s.t.} \quad & \mathbb{P}(Z(v, \omega) \notin \mathcal{Z}) \leq p. \end{aligned} \tag{2.13}$$

Here  $p$  is called a failure rate, and  $\mathcal{Z}$  is the set of all feasible values that  $Z(v, \omega)$  can take. When  $Z(v, \omega)$  is real valued and  $\mathcal{Z}$  is an open set  $(-\infty, z)$  then, the set  $\mathbb{P}(Z(v, \omega) \notin \mathcal{Z}) \leq p$  is equivalent to the set  $\text{VaR}_p(Z(v, \omega)) \leq z$ . Often problems of this form are not convex in general; as such, we will not consider problems of this sort in this thesis.

In this chapter, we have introduced risk measures, acceptance sets and risk sets, and showed how they relate to each other. We have argued that law invariant coherent risk measures are sensible since they first-order preserve stochastic dominance. We have shown that all such risk measures have a special representation that is critical for the algorithms that will be presented later in this thesis. Finally, we have shown how an agent may incorporate risk measures into an optimisation problem. Most of the notation, constructions, and theorems developed in this chapter will act as a building block for the analysis of risk for the multistage problem classes which will be considered in this thesis.



## Chapter 3

### Multistage risk aversion

Forming preference relations on random variables in the static case (through coherent risk measures) is a well developed and understood subject within stochastic programming. A much more interesting problem that faces stochastic programming is the ordering of random variables dynamically over a time horizon; we loosely refer to this as *multistage risk*. Similar to risk in the static setting, our goal is to develop a theory on making comparisons between random variables. However, rather than a single random variable, we want to compare sequences of random variables of arbitrary length. Many authors have written on this subject. The authors of Artzner, Delbaen, Eber, and Heath (1999) extend their work into the dynamic setting in Artzner, Delbaen, Eber, Heath, and Ku (2002). The work of Riedel (2004) extends this work and provides two critical observations: that dynamic risk measures must be temporally logical, and the axiom of translational invariance should be recast as the axiom of *predictable* translational invariance. In his words:

“... this should be recast as predictable translational invariance to account for the release of new information”.

However, we favour the perspective on the subject offered by Ruszczyński and Shapiro (2006a) (which incidentally builds upon the two previous works). In Ruszczyński and Shapiro (2006a), a clear framework for analysis of these risk measures is provided, along with several representation theorems. In this chapter, we will introduce the notation and concepts developed in Ruszczyński and Shapiro (2006a), and expand upon some of their ideas within the framework.

As hinted at earlier, a theme of particular importance in this chapter is the concept of *time-consistency*. Time-consistency requires that: does my current assessment of

the risk in front of me agree with assessments of risk I have made in the past? That is, a time-consistent risk measure cannot contradict itself over time. We collate several authors' views, along with our own, and show that that are all connected to concepts from dynamic programming.

### 3.1 A time-consistency concept

Time consistency is a concept that relates to making a sequence of control decisions through *time*. In this section, we will present our definition in a general context, and then show how such a definition relates to those presented by other authors. We shall demonstrate our concept on the smallest non-trivial sequence of optimisation problems, a sequence of length two. Consider a set of optimisation problems given by

$$z_1^* = \min_{u_1 \in \mathcal{U}_1^1, u_2 \in \mathcal{U}_2^1(u_1)} c_1^1(u_1) + c_2^1(u_1, u_2),$$

$$z_2^*(u_1) = c_1^2(u_1) + \min_{u_2 \in \mathcal{U}_2^2(u_1)} c_2^2(u_1, u_2).$$

For the sake of simplicity in order to aid conceptual understanding, assume that all feasibility sets are non-empty and a minimum exists for each problem i.e. we have relatively complete recourse. A member of this sequence is often called a stage; points in time when a control is to be determined and the made. The notation  $c_t^\tau$  denotes the cost  $c$  incurred in stage  $t$  from the perspective of stage  $\tau$  – the same holds for the feasible sets  $\mathcal{U}_t^\tau$ . Given a sequence of optimisation problems, in order to determine the control decision to make in a given stage, the optimising agent solves their stage problem and arbitrarily chooses a control in the minimising set.

We say a sequence of optimisation problems is time consistent if after implementing an optimal control from the perspective of the first stage, in the second stage the optimising agent does not regret the control made in the first stage. The following definition captures this concept mathematically.

#### Definition 3.1.

We say a sequence of optimisation problems is time consistent if

$$\arg \min_{u_1 \in \mathcal{U}_1^1} \left\{ \min_{u_2 \in \mathcal{U}_2^1(u_1)} c_1^1(u_1) + c_2^1(u_1, u_2) \right\} \subseteq \arg \min_{u_1 \in \mathcal{U}_1^2} \left\{ c_1^2(u_1) + \min_{u_2 \in \mathcal{U}_2^2(u_1)} c_2^2(u_1, u_2) \right\},$$

The right-hand side of the above is the set of all first-stage controls that the agent could have hoped for in the second stage; if any of these controls are implemented in the first stage, then there is no regret from the perspective of the second stage problem. The left-hand side is the set of all first-stage controls that are optimal for the first-stage problem; clearly the agent would not regret selecting one of these controls from the perspective of the first stage. We can enforce time-consistency by having

$$c_1^1 = c_1^2, \mathcal{U}_1^1 = \mathcal{U}_1^2, c_2^1 = c_2^2, \text{ and } \mathcal{U}_2^1(u_1) = \mathcal{U}_2^2(u_1), \forall u_1,$$

this is, our objective functions and feasibility sets are consistent across time (hence the name time-consistency). We argue that a concept of ‘no regret’ between any stages of the multistage optimisation process is in itself a condition of optimality.

Let us explore time consistency when the optimising agent is greedy. In the first stage, they solve and implement one of the minimisers. In order to check that the agent has no regret (with any of the arbitrarily selected first stage minimisers), she must then compare the sets

$$\arg \min_{u_1 \in \mathcal{U}_1^2} c_1^2(u_1) + \min_{u_2 \in \mathcal{U}_2^2(u_1)} c_2^2(u_1, u_2),$$

and

$$\arg \min_{u_1 \in \mathcal{U}_1^1} \min_{u_2 \in \mathcal{U}_2^1(u_1)} c_1^1(u_1) + c_2^1(u_1, u_2)$$

and check that the second is contained within the first. Under our definition of time-consistency, in general, this sequence of optimisation problems is not time-consistent. From the perspective of the first-stage problem, the optimising agent does not consider any second-stage costs when making choosing a control now. This could lead to a large second-stage cost from the perspective of the second-stage problem; and from this perspective, the agent may regret her first-stage control. This is the folly of the greedy optimiser.

Let us consider a sequence of optimisation problems that models the somewhat relatable problem of joining the local gym. Recall the form of the sequence of optimisation problems:

$$z_1^* = \min_{u_1 \in \mathcal{U}_1^1, u_2 \in \mathcal{U}_2^1(u_1)} c_1^1(u_1) + c_2^1(u_1, u_2),$$

$$z_2^*(u_1) = c_1^2(u_1) + \min_{u_2 \in \mathcal{U}_2^2(u_1)} c_2^2(u_1, u_2).$$

For our problem, the two controls in the first stage are  $\mathcal{U}_1^1 = \{\text{join gym, do nothing}\}$ .

Our second-stage controls are

$$\mathcal{U}_2^1(u_1) = \mathcal{U}_2^2(u_1) = \begin{cases} \{\text{work out, do nothing}\}, & \text{if } u_1 = \text{join gym}, \\ \{\text{do nothing}\}, & \text{if } u_1 = \text{do nothing}. \end{cases}$$

Our cost functions from the perspective of the first stage might be given by

$$c_1^1(u_1) = \begin{cases} 1, & \text{if } u_1 = \text{join gym}, \\ 0, & \text{if } u_1 = \text{do nothing}, \end{cases}$$

and

$$c_2^1(u_1, u_2) = \begin{cases} -2, & \text{if } (u_1, u_2) = (\text{join gym, work out}), \\ 0, & \text{if } (u_1, u_2) = (\text{join gym, do nothing}), \\ 0, & \text{if } (u_1, u_2) = (\text{do nothing, do nothing}). \end{cases}$$

The cost functions from the perspective of the second stage might be given by

$$c_1^2(u_1) = \begin{cases} 1, & \text{if } u_1 = \text{join gym}, \\ 0, & \text{if } u_1 = \text{do nothing}, \end{cases}$$

and

$$c_2^2(u_1, u_2) = \begin{cases} 1, & \text{if } (u_1, u_2) = (\text{join gym, work out}), \\ 0, & \text{if } (u_1, u_2) = (\text{join gym, do nothing}), \\ 0, & \text{if } (u_1, u_2) = (\text{do nothing, do nothing}). \end{cases}$$

When deciding whether or not to join gym in the first stage, our agent perceives a gain to working out in the second stage. So the optimal first-stage control from the perspective of the first stage is  $u_1 = \text{join gym}$ . In the second stage, when it comes on deciding what to do, the optimiser changes her mind on the cost of working out at the gym; their optimal control at this point is to do nothing! But this contradicts the plan created in the first stage, on which the decision to join the gym was based; the sequences of optimisation problems were not time-consistent, as she regrets her first-stage decision.

### Theorem 3.1.

If a sequence of optimisation problems is not time-consistent, then there exists an implemented optimal control sequence prescribed by the sequence of optimisation problems which will not be optimal for at least one of the problems



in the sequence.

*Proof.* The proof is trivial. If time-consistency does not hold, then there must exist an optimal control from the first stage which is not in the desired first-stage control set from the perspective of the second stage.  $\square$

### 3.1.1 Time consistency under uncertainty

The above notion of time-consistency dealt with the case where the optimisation problems were deterministic. Here, we extend our time-consistency concept to stochastic optimisation problems. We will consider a probability space  $(\Omega, \mathcal{F}, \mathbb{P})$  and associated random variable space  $\mathcal{X}$  measurable with respect to  $\mathcal{F}$ . Our sequence of optimisation problems is given by

$$z_1^* = \min_{u_1 \in \mathcal{U}_1^1, u_2(\omega) \in \mathcal{U}_2^1(u_1, \omega)} c_1^1(u_1) + \mathbb{M}^1(c_2^1(u_1, u_2(\omega), \omega)),$$

$$z_2^*(u_1, \omega) = c_1^2(u_1, \omega) + \min_{u_2(\omega) \in \mathcal{U}_2^2(u_1, \omega)} c_2^2(u_1, u_2(\omega), \omega), \quad \forall \omega \in \Omega,$$

where  $\mathbb{M}^1 : \mathcal{X} \mapsto \mathbb{R}$  and  $\mathbb{M}^1$  preserves its minimums i.e.

$$\min_{u(\omega) \in \mathcal{U}(\omega)} \mathbb{M}^1(X(u(\omega), \omega)) = \mathbb{M}^1\left(\min_{u(\omega) \in \mathcal{U}(\omega)} X(u(\omega), \omega)\right),$$

for all random functions  $X$ . Note that the expectation operator and coherent risk measures are an example of such a mapping. This generalisation allows us to ‘abstract away’ the particular differences between coherent risk measures and other mappings from  $\mathcal{X} \mapsto \mathbb{R}$ .

In general, an agent will always regret her first-stage control in a stochastic optimisation problem, where  $\mathbb{M}^1 = \mathbb{E}$ . That is, the agent will never be able to satisfy the time-consistency definition for deterministic problems for every outcome as in:

$$\begin{aligned} & \arg \min_{u_1 \in \mathcal{U}_1^1} \left\{ \min_{u_2(\omega) \in \mathcal{U}_2^1(u_1, \omega)} c_1^1(u_1) + \mathbb{M}^1(c_2^1(u_1, u_2(\omega), \omega)) \right\} \\ & \subseteq \arg \min_{u_1 \in \mathcal{U}_1^2} \left\{ c_1^2(u_1, \omega) + \min_{u_2(\omega) \in \mathcal{U}_2^2(u_1, \omega)} c_2^2(u_1, u_2(\omega), \omega) \right\}, \quad \forall \omega \in \Omega. \end{aligned}$$

We contend that this feature is natural for stochastic optimisation problems and ought not to disqualify a regret concept from employment in a time-consistency definition.

We use a similar but relaxed idea of ‘regret’ to form our final definition of time consistency in stochastic optimisation problems; we must be sure that the agent is

to accept regret in the case of uncertainty. That is, the agent has to accept that a poor outcome may occur but given a reproduction of the experiment faced on the same probability space, the agent would not change their first-stage control. By this intuition, the agent does not truly regret their first-stage control, as the agent knows she must ‘take the good with the bad’. In mathematical notation, by this definition, we have time consistency when

$$\begin{aligned} & \arg \min_{u_1 \in \mathcal{U}_1^1} \left\{ \min_{u_2(\omega') \in \mathcal{U}_2^1(u_1, \omega')} c_1^1(u_1) + \mathbb{M}^1(c_2^1(u_1, u_2(\omega'), \omega')) \right\} \\ & \subseteq \arg \min_{u_1 \in \mathcal{U}_1^1} \left\{ c_1^2(u_1, \omega) + \mathbb{M}_\omega^2 \left( \min_{u_2(\omega') \in \mathcal{U}_2^2(u_1, \omega')} c_2^2(u_1, u_2(\omega'), \omega') \right) \right\}, \forall \omega \in \Omega. \end{aligned}$$

Here the agent must be consistent in their view of the costs, their view of their feasibility sets and how their view of the random second stage costs are accounted for (through  $\mathbb{M}$ ) over time *and* different outcomes on the probability space. Once again, by enforcing that

$$c_1^1 = c_1^2, \mathcal{U}_1^1 = \mathcal{U}_1^2, c_2^1(\cdot, \cdot, \omega) = c_2^2(\cdot, \cdot, \omega), \text{ and } \mathcal{U}_2^1(u_1, \omega) = \mathcal{U}_2^2(u_1, \omega), \forall u_1, \forall \omega \in \Omega,$$

and finally  $\mathbb{M}^1 = \mathbb{M}_\omega^2$ ,  $\forall \omega \in \Omega$ , we can be sure that our definition of time-consistency holds.

We relate this definition of time-consistency to those of other authors. The definition (taken verbatim) of a time-consistency offered by Carpentier et al. (2012) is:

“A family of optimization problems formulated is said to be dynamically consistent if the optimal strategies obtained when solving the original problem remain optimal for all subsequent problems.”

This view of time-consistency here is similar to one put forward by Shapiro (2009). His can be interpreted as the following:

“If a plan is optimal from  $t_1$  to  $T$ , then that same plan from  $t_2 > t_1$  is optimal from  $t_2$  to  $T$ ”.

Our definition can be considered a relaxation of the above definitions. Our definition allows the plan made in the first stage to change in the future stages, as long as the future stages do not regret earlier decisions.

Where the definitions offered above break down is when the greedy optimiser is considered. Under the above definitions, the greedy optimiser's sequence of optimisation problem is time consistent. Any plan of controls for the problem developed in the first stage mean that *any* second stage decision is optimal as the cost function is a constant i.e.  $c_2^1(u_1, u_2) = 0$ . When it comes to implementing a control in the second stage, any feasible control is 'part of the original plan', and therefore the sequence of optimisation problems is time consistent under the definitions above. We believe that any useful definition of time consistency should exclude the greedy sequence of optimisation problems from being time consistent.

### 3.1.2 Relationship with dynamic programming

Sequences of control problems and dynamic programming have always overlapped throughout the history of operations research. Here we will explore their overlap through our time consistency definition. Consider the dynamic programming equations

$$V_1^* = \min_{u_1 \in \mathcal{U}_1^1} c_1^1(u_1) + V_2^*(u_1),$$

$$V_2^*(u_1) = \min_{u_2 \in \mathcal{U}_2^2(u_1)} c_2^2(u_1, u_2).$$

From these dynamic programming equations, we can construct a sequence of optimisation problems which take the form

$$z_1^* = \min_{u_1 \in \mathcal{U}_1^1, u_2 \in \mathcal{U}_2^2(u_1)} c_1^1(u_1) + c_2^2(u_1, u_2),$$

$$z_2^*(u_1) = c_1^1(u_1) + \min_{u_2 \in \mathcal{U}_2^2(u_1)} c_2^2(u_1, u_2).$$

We now present a theorem that relates dynamic programming with time consistency.

#### Theorem 3.2.

Dynamic programming equations imply time consistency of the corresponding sequence of optimisation problems.

*Proof.* Clearly we have time consistency, because the sequence shares the same objective functions and feasibility sets.  $\square$

So anywhere that dynamic programming equations hold, we automatically have time consistency of the corresponding sequence of optimisation problems. Intuitively,

we should expect this to be the case. If when making her first-stage control, our agent accounts for (through a value function) the impact her control will have in the second stage, then in the second stage, she won't regret her first stage control.

This notion carries over into the stochastic case also; the dynamic equations given by

$$V_1^* = \min_{u_1 \in \mathcal{U}_1^1, u_2(\omega) \in \mathcal{U}_2^1(u_1, \omega)} c_1^1(u_1) + \mathbb{M}^1(V_2^*(u_1, \omega)),$$

and

$$V_2^*(u_1, \omega) = \min_{u_2(\omega) \in \mathcal{U}_2^2(u_1, \omega)} c_2^2(u_1, u_2(\omega), \omega), \quad \forall \omega \in \Omega,$$

ensure that we have time consistency. As we saw in the previous chapter, evaluating the risk of a random variable can be viewed as an optimisation problem (a stochastic optimisation problem in particular). For multistage risk measures (as we shall see throughout this chapter), we obtain a sequence of optimisation problems. We determine whether or not they are time consistent by whether their corresponding sequence of optimisation problems are time consistent. The rest of this chapter introduces the theory of multistage risk measures; at the end of this chapter, we conclude by making a determination on whether our multistage risk measures developed in this chapter are time consistent.

## 3.2 Filtered probability spaces

Similar to the previous chapter, we begin by laying out the fundamental objects which are used to discuss the ideas in this chapter. We consider the same probability space  $(\Omega, \mathcal{F}, \mathbb{P})$ , however, in the multistage case, we are also concerned with another object  $\{\mathcal{F}_t\}_{t \in 1, \dots, T}$ , called a *filtration*. The sequence of  $\sigma$ -algebra must satisfy

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \dots \subset \mathcal{F}_T$$

in order to be a filtration. A filtration represents the discovery of information as a stochastic process evolves over time. Throughout this chapter we will take the convention that  $\mathcal{F}_1 = \{\{\}, \Omega\}$  (i.e. we know nothing) and that  $\mathcal{F}_T = \mathcal{F}$  (i.e. we know everything). Further, we consider here only filtrations of finite length – in Chapter 6 we will extend these ideas to filtrations of countably-infinite length.

We often speak about functions and multifunctions being *measurable*. A function  $X : \Omega \mapsto \mathbb{R}$  is measurable with respect to the  $\sigma$ -algebra  $\mathcal{F}$  if for all  $B \in \mathcal{B}(\mathbb{R})$

$$\{\omega \in \Omega : X(\omega) \in B\} \in \mathcal{F}.$$

Where  $\mathcal{B}(\mathbb{R})$  is the Borel  $\sigma$ -algebra generated by the real numbers. An interpretation of the definition above is the following: a random variable is measurable with respect to a  $\sigma$ -algebra if the value of the random variable is knowable with the information contained in the  $\sigma$ -algebra.

Let us now introduce a sequence of function spaces  $\{\mathcal{X}_t\}_{t \in 1, \dots, T}$ . The space  $\mathcal{X}_t$  is the space of functions  $X : \Omega \mapsto \mathbb{R}$ , that are  $\mathcal{F}_t$  measurable. Immediately, we can note an observation: that all functions  $X \in \mathcal{X}_t$  are also in  $\mathcal{X}_\tau$  where  $t \leq \tau \leq T$ . We also note that the functions in  $\mathcal{X}_1$  are the constant functions – their value must be known without any information. Using these tools, we will now develop the theory of multistage risk aversion. Let us first consider the example below to develop an intuition for these concepts.

### Example 3.1.

Consider the ‘two coin toss’ probability space  $\Omega = \{HH, HT, TH, TT\}$ ,  $\mathcal{F}_3 = 2^\Omega$ , and  $\mathbb{P}$  is the discrete uniform measure. Consider the set of  $\sigma$ -algebras  $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}$ . If  $\mathcal{F}_1 = \{\{\}, \Omega\}$ , and  $\mathcal{F}_2 = \{\{\}, \{HH, HT\}, \{TH, TT\}, \Omega\}$ , then the set of  $\sigma$ -algebras define a filtration on the measure space.  $\mathcal{F}_1$  represents knowing the outcome of no coin tosses,  $\mathcal{F}_2$  represents knowing the outcome of the first coin toss, and  $\mathcal{F}_3$  represents knowledge of the full experiment i.e. both coin tosses. Consider the corresponding set of random variable spaces  $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$ . An example of two random variables  $X_2 \in \mathcal{X}_2$  and  $X_3 \in \mathcal{X}_3$  which are measurable with respect to their respective  $\sigma$ -algebras are

$$X_2(\omega) = \begin{cases} X_2(\omega) = a, & \omega = HH, \\ X_2(\omega) = a, & \omega = HT, \\ X_2(\omega) = b, & \omega = TH, \\ X_2(\omega) = b, & \omega = TT, \end{cases} \quad \text{and} \quad X_3(\omega) = \begin{cases} X_3(\omega) = a, & \omega = HH, \\ X_3(\omega) = b, & \omega = HT, \\ X_3(\omega) = c, & \omega = TH, \\ X_3(\omega) = d, & \omega = TT. \end{cases}$$

Notice how we can only know the outcome of  $X_3$  if we know outcome of the experiment, i.e. all of the information. This is in contrast to  $X_2$ , we only need the information in  $\mathcal{F}_2$  (the outcome of the first coin toss) to determine its value. However  $X_2$  is also measurable with respect to  $\mathcal{F}_3$  also because  $\mathcal{F}_3$  contains  $\mathcal{F}_2$ ’s information. As mentioned earlier, the only random variables for which

we need no information (i.e. measurable with respect to  $\mathcal{F}_1$ ) in order to know their outcome are the constant functions i.e  $X(\omega) = s, \forall \omega \in \Omega$ .

Our final mathematical consideration before the presentation of the main work in this chapter is the notion of a *atom* in a measure space.

**Definition 3.2.**

Consider a measure space  $(\Omega, \mathcal{F})$ . An atom of this space is a non-empty set  $A \in \mathcal{F}$  such that

$$B \in \mathcal{F} \text{ and } B \subseteq A \implies B = \{\} \text{ or } B = A.$$

We denote the set of atoms of a particular  $\sigma$ -algebra  $\mathcal{H}$  as  $\alpha(\mathcal{H})$ .

If the  $\sigma$ -algebra is countably generated (i.e.  $2^\Omega$  if  $\Omega$  is finite or  $\mathcal{F} \subseteq \mathcal{B}(\Omega)$  if  $\Omega = \mathbb{R}$ ) then these  $\sigma$ -algebras have a further property that the atoms of  $\mathcal{F}$  form a partition of  $\Omega$ , and so every  $\omega$  is contained within one atom. Let  $m_{\mathcal{H}} : \Omega \mapsto \alpha(\mathcal{H})$ , be this mapping of  $\omega$  to its atom in  $\mathcal{H}$ . Often, in probability we speak of an *atomless* space. As every measure space contains atoms, the term atomless can be a point of confusion. An atomless space refers to a probability space where the state space  $\Omega$  is uncountably infinite. The standard probability space  $((0, 1), \mathcal{B}((0, 1)), \mathbb{P})$  where  $\mathbb{P}$  is the standard Lebesgue measure, is an example of an atomless space. The set of atoms of this space is the set  $(0, 1)$  and every element in the set has zero measure.

Using these tools, the motivation for this section is develop a theory for forming an ordering on random variables which is consistent through time. Suppose the total value of random can be given by

$$X = X_1 + \dots + X_T, X_1 \in \mathcal{X}_1, \dots, X_T \in \mathcal{X}_T. \quad (3.1)$$

Because  $X_T$  is in  $\mathcal{X}_T$ , the total value  $X$  is in  $\mathcal{X}_T$ . We could measure the risk of the entire portfolio in one time i.e. using a function  $\rho : \mathcal{X}_T \mapsto \mathbb{R}$ , however this approach ignores the dynamic nature of the sequence. As time progresses, the observer may wish to re-evaluate the portfolio's position, however a two-stage approach does not offer dynamic insight to the observer. We require our multistage risk evaluation framework to allow re-evaluations of the portfolio throughout the stochastic process. In particular, if the portfolio were re-evaluated at some time  $t$ , the re-evaluation agrees in some sense with previous evaluations. This is the requirement of time-consistency.

Our goal is to develop a method of evaluating sequences of positions in a way which respects the dynamic nature of the process while ensuring that time-consistency is respected.

### 3.3 Conditional risk mappings

The main object that extends a risk measure from the static case to a dynamic setting is a conditional risk measure. We will often refer to these as conditional risk *mappings* as the term *measure* will be reserved for measurement in the measure-theory sense. Throughout this section we will refer to a general probability space  $(\Omega, \mathcal{F}, \mathbb{P})$ , and consider a filtration  $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}$ , where  $\mathcal{F}_1 = \{\{\}, \Omega\}$ ,  $\mathcal{F}_3 = \mathcal{F}$  and  $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_3$ . The ‘two coin toss’ is an example of such a probability space. We will consider also the space of random variables  $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$  which are measurable with respect to their corresponding  $\sigma$ -algebras.

A conditional risk (or acceptance) mapping is a function  $\rho(v) : \mathcal{X}_3 \mapsto \mathcal{X}_2$ . Instead of  $\rho$  mapping random variables to the reals as in the static case, the  $\rho$  operator maps a random variable to a random variable measurable with respect to some sub- $\sigma$ -algebra. An example of such an acceptance mapping is the conditional expectation operator.

#### Example 3.2.

Consider the above probability space, filtration, and random variable spaces.

Then  $\mathbb{E}[X \mid \mathcal{F}_2]$  is a conditional acceptance mapping.

We will consider conditional acceptance mappings that are *conditionally coherent*. The mapping  $v : \mathcal{X}_3 \mapsto \mathcal{X}_2$  is conditionally coherent if it is

1. monotonic i.e.  $X(\omega) \geq Y(\omega) \implies [v(X)](\omega) \geq [v(Y)](\omega), \forall \omega \in \Omega$ ;
2. translationally invariant i.e. If  $Y \in \mathcal{X}_2$  then  $[v(X + Y)](\omega) = [v(X)](\omega) + Y(\omega)$ ;
3. concave i.e.  $[v(X + Y)](\omega) \leq [v(X)](\omega) + [v(Y)](\omega)$ ;
4. positively homogeneous i.e.  $[v(\alpha X)](\omega) = \alpha[v(X)](\omega), \alpha > 0$ .

We use the  $[v(X)]$  notation to emphasise that  $v(X)$  is itself a random variable (albeit measurable with respect to some sub- $\sigma$ -algebra).

**Example 3.3.**

Consider the two coin toss probability space. The following is a conditional coherent risk measure:

$$[v(X)](\omega) = \begin{cases} \min\{X(HH), X(HT)\}, & \omega \in \{HH, HT\}, \\ \min\{X(TH), X(TT)\}, & \omega \in \{TH, TT\}. \end{cases}$$

It is easy to see that such an conditional acceptance measure obeys the requirements outlined above.

Our analysis throughout this section relies heavily on the dual objects of acceptance mappings, as these dual objects represent optimisation problems. We pair the sequence of random variable spaces with a corresponding sequence of dual spaces  $\mathcal{Y}$  such that  $\mathcal{Y}_t$  contains finite signed measures on the space  $(\Omega, \mathcal{F}_t)$ . This allows for the following inner product to be valid:

$$\int_{\Omega} X_t(\omega) d\mu_t(\omega) < \infty \quad \forall (X_t, \mu_t) \in \mathcal{X}_t \times \mathcal{Y}_t, \forall t.$$

We will often shorten the above to  $\langle \mu_t, X_t \rangle$  for notational convenience. We denote  $\mathcal{P}_{\mathcal{Y}_t} \subset \mathcal{Y}_t$  as the set of probability measures on  $(\Omega, \mathcal{F}_t)$ . For a given  $\omega \in \Omega$  we can also define  $\mathcal{P}_{\mathcal{Y}_3|\mathcal{F}_2}(\omega) \subset \mathcal{P}_{\mathcal{Y}_3}$  as the subset of  $\mu \in \mathcal{P}_{\mathcal{Y}_3}$ , such that for every  $B \in \mathcal{F}_2$  we have

$$\mu(B) = \begin{cases} 1, & \omega \in B, \\ 0, & \omega \notin B. \end{cases}$$

We can think of  $\mathcal{P}_{\mathcal{Y}_3|\mathcal{F}_2}(\omega)$  for a given  $\omega$  as the set of all probability measures on  $\mathcal{F}_3$  that are conditional probability measures with respect to  $\mathcal{F}_2$  and  $\omega$  is an outcome which has not been ‘conditioned out’. Similar to the dual representation of coherent risk measures in the previous chapter, a similar duality result is given by Ruszczyński and Shapiro (2006a). If  $v : \mathcal{X}_3 \mapsto \mathcal{X}_2$  is a conditional coherent acceptance mapping then

$$[v(X)](\omega) = \inf_{\mu \in Q(\omega)} \langle \mu, X \rangle, \quad \forall \omega \in \Omega, \forall X \in \mathcal{X}_3,$$

where  $Q(\omega)$  is a set of conditional probabilities and a subset of  $\mathcal{P}_{\mathcal{Y}_3|\mathcal{F}_2}(\omega)$ . This makes  $[v(\cdot)]$  an  $\mathcal{F}_2$  measurable function. That is, we can determine the value of  $v(X)$  if we know the information contained in  $\mathcal{F}_2$ .



## Example 3.4.

Consider the two coin toss probability space and filtration. The following (arbitrary) probability measure on  $\mathcal{F}_3$  is contained within both  $\mathcal{P}_{\mathcal{G}_3|\mathcal{F}_2}(HH)$  and  $\mathcal{P}_{\mathcal{G}_3|\mathcal{F}_2}(HT)$ :

$$\mathbb{P}_H(A) = \begin{cases} \frac{3}{4}, & A = \{HH\}, \\ \frac{1}{4}, & A = \{HT\}, \\ 0, & A = \{TH\}, \\ 0, & A = \{TT\}. \end{cases}$$

On the other hand, following probability measure on  $\mathcal{F}_3$  is contained within both  $\mathcal{P}_{\mathcal{G}_3|\mathcal{F}_2}(TH)$  and  $\mathcal{P}_{\mathcal{G}_3|\mathcal{F}_2}(TT)$ :

$$\mathbb{P}_T(A) = \begin{cases} 0, & A = \{HH\}, \\ 0, & A = \{HT\}, \\ \frac{2}{3}, & A = \{TH\}, \\ \frac{1}{3}, & A = \{TT\}. \end{cases}$$

Note that because our space is discrete, the probability measure is wholly defined on all elements of  $\mathcal{F}_3$  by defining it on every  $\omega \in \Omega$ . So the acceptance mapping  $\nu : \mathcal{X}_3 \mapsto \mathcal{X}_2$  defined as

$$[\nu(X)](\omega) = \begin{cases} \mathbb{E}_{\mathbb{P}_H}(X), & \omega = HH, \\ \mathbb{E}_{\mathbb{P}_H}(X), & \omega = HT, \\ \mathbb{E}_{\mathbb{P}_T}(X), & \omega = TH, \\ \mathbb{E}_{\mathbb{P}_T}(X), & \omega = TT. \end{cases}$$

is a conditional coherent acceptance measure. Here we can see that  $[\nu(X)](\omega)$  is a  $\mathcal{F}_2$  measurable function, we can know for sure the value of  $[\nu(X)](\omega)$  if we know the information in  $\mathcal{F}_2$ .

A further representation theorem posed in Ruszczyński and Shapiro (2006a) is that

$$[\nu(X)](\omega) = \inf_{\mu(\cdot) \in \mathcal{Q}(\cdot)} \langle \mu(\omega), X \rangle, \quad \forall \omega \in \Omega, \forall X \in \mathcal{X}_3.$$

This is in contrast to the representation above, where we performed an infimum over the measures contained within  $\mathcal{Q}(\omega)$ ; in this case we perform an infimum over

the space of all measures for every  $\omega \in \Omega$  and ‘pick out’ our particular  $\omega$ . This representation is critical to the following composition theory.

### 3.3.1 Composition of conditional mappings

Now that these conditional acceptance mappings are understood, we can use them to form an ordering and subsequent conditional orderings on  $\mathcal{F}_3$ -measurable random variables. We can construct a conditional acceptance mapping  $v_1^3 : \mathcal{X}_3 \mapsto \mathcal{X}_1$  by *composing* two further conditional acceptance mappings i.e.

$$v_1^3 = v_1^2 \circ v_2^3.$$

where  $v_1^2 : \mathcal{X}_2 \mapsto \mathcal{X}_1$  is a coherent conditional acceptance mapping. Ruszczyński and Shapiro (2006a) show that  $v_1^3$  is a coherent conditional mapping if both  $v_1^2$  and  $v_2^3$  are coherent conditional mappings. Recall that by the duality of conditional acceptance mappings we have

$$\begin{aligned} [v_2^3(X)](\omega) &= \inf_{\mu_2^3(\cdot) \in Q_2^3(\cdot)} \langle \mu_2^3(\omega), X \rangle, \quad \forall X \in \mathcal{X}_3, \forall \omega \in \Omega, \\ v_1^2(X) &= \inf_{\mu_1^2 \in Q_1^2} \langle \mu_1^2, X \rangle, \quad \forall X \in \mathcal{X}_2. \end{aligned}$$

Note, we do not employ the square bracket notation for objects which are measurable with respect to  $\mathcal{F}_1$  since these are the constant across  $\omega$ . Because  $[v_2^3(X)](\omega) \in \mathcal{X}_2$ , we can nest the above, and define an acceptance mapping which maps random variables from  $\mathcal{X}_3$  to  $\mathcal{X}_1$  as

$$v_1^3(X) = \inf_{\mu_1^2 \in Q_1^2} \int_{\omega \in \Omega} \inf_{\mu_2^3(\cdot) \in Q_2^3(\cdot)} \langle \mu_2^3(\omega), X \rangle d\mu_1^2(\omega).$$

Ruszczyński and Shapiro (2006a) showed that the inner infimum and integral can be exchanged to allow

$$v_1^3(X) = \inf_{\mu_1^2 \in Q_1^2} \inf_{\mu_2^3(\cdot) \in Q_2^3(\cdot)} \int_{\omega \in \Omega} \langle \mu_2^3(\omega), X \rangle d\mu_1^2(\omega).$$

Via this composition, we can then construct the set  $Q_1^3$  as the set of all probability measures on  $\mathcal{F}_3$  which are  $\mathcal{F}_1$ -measurable such that

$$\inf_{\mu_1^2 \in Q_1^2} \inf_{\mu_2^3(\cdot) \in Q_2^3(\cdot)} \int_{\omega \in \Omega} \langle \mu_2^3(\omega), X \rangle d\mu_1^2(\omega) = \inf_{\mu_1^3 \in Q_1^3} \langle \mu_1^3, X \rangle, \quad \forall X \in \mathcal{X}_3.$$

We can perform this construction explicitly as

$$\mu_1^3(A) = \int_{\omega \in \Omega} [\mu_2^3(A)](\omega) d\mu_1^2(\omega), \quad \forall A \in \mathcal{F}_3.$$

So by composing two acceptance mappings  $\nu_1^2 \circ \nu_2^3$ , we obtain a single risk mapping from  $\mathcal{X}_3$  to  $\mathcal{X}_1$ ; in the dual space, we compose two risk sets  $\mathcal{A}_1^2 \circ \mathcal{A}_2^3$  to obtain a single risk set  $\mathcal{A}_1^3$  over which to take an infimum.

When evaluating the risk of a  $\mathcal{F}_3$  measurable random variable through time, any multistage risk mapping  $\nu_1^3 = \nu_1^2 \circ \nu_2^3$  will be time-consistent according to our earlier definition. This is because the composition can be written as a dynamic programme; defining a sequence of optimisation problems whose feasibility sets (risk sets) are consistent throughout time. Later in this thesis, we will refer to these risk measures as ‘rectangular’; if once considers the cartesian product of the conditional risk sets, the resulting object will take a ‘rectangular’ form.

### 3.3.2 A law invariance issue

In the two-stage setting, under the atomless space assumption, all law invariant coherent risk measures preserve stochastic dominance. However, in the multistage setting, it is easy to show (through our example below) that composition of law invariant coherent risk measures do not, in general, lead to a risk measure which is law invariant. One of the consequences of this is the concession of the (highly desirable) stochastic dominance property.

Consider Figure 3.1. By taking the expectation over the worst case probabilities (governed by the risk sets of  $\nu$ ) of the random variables above, we obtain  $\nu(X_1) = 1.75$ , and  $\nu(X_2) = 1.875$  from Figures 3.1b and 3.1c respectively. The random variables  $X_1$  and  $X_2$  clearly have the same distribution since because the base measure (shown in Figure 3.1c) is uniform, two random variables which are permutations of each other share the same distribution. The multistage risk measure  $\nu$  is not law invariant, since  $\nu$  prefers one more than the other. In fact, it has been proved in Shapiro (2012), that the only acceptance mappings which are law invariant are those which are composition of either  $\mathbb{E}[\cdot]$  or  $\text{ess inf}(\cdot)$ . It important to note here that the definition of law invariance considered in the multistage case is one that follows from a simple analogy from the static case; that the composition of conditional acceptance measures which form a multistage risk measure  $\nu$ , should be invariant under two random variables with the same probability law. This is in line with the definition used

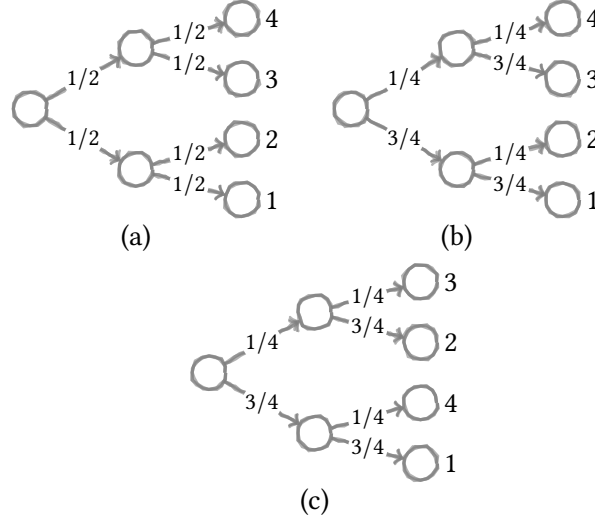


Figure 3.1: Coin toss space

in Shapiro (2012) and Kupper and Schachermayer (2009). Other definitions of law invariance have been proposed for the multistage case i.e Homem-De-Mello and Pagnoncelli (2016), but these are not useful for our considerations.

Similar to the static case, it could be argued that a lack of law invariance renders these risk measures inappropriate for use in optimisation. So we are left in a bind; composed conditional risk mappings give us a way to evaluate risk of random variables consistently through time, but their compositions are not law invariant. We overcome this problem by developing *parametric* conditional risk sets. With these objects we can decompose a law invariant risk set  $Q_1^3$  into

$$Q_1^3 = Q_1^2 \circ Q_2^3(\omega).$$

where  $Q_2^3(\omega)$  is *parametric*; a property not considered by Shapiro (2012) when developing his law invariance result. This concept is a fundamental contribution of this thesis.

### 3.3.3 Parametric conditional risk sets

A parametric risk set is a multifunction  $Q_2^3 : \Omega \times \mathcal{P}_{\mathcal{F}_2|\mathcal{F}_1} \rightrightarrows \mathcal{P}_{\mathcal{F}_3|\mathcal{F}_2}(\omega)$ . Note that these risk sets have the same properties as their non-parametric analogue –  $Q_2^3$  is a  $\mathcal{F}_2$  measurable multifunction mapping outcomes in the state space and a probability measure on  $(\Omega, \mathcal{F}_2)$  to sets of probability measures on  $(\Omega, \mathcal{F}_3)$ . A lemma critical to

our main decomposition result, eventually demonstrating the efficacy of parametric conditional risk sets, comes from Billingsley (1995) and is presented below.

Lemma 3.1 (Theorem 33.3 Billingsley).

Consider a probability space  $(\Omega, \mathcal{F}, \mathbb{P})$ , sub- $\sigma$ -algebra  $\mathcal{H}$ , random variable space  $\mathcal{X}$  (measurable with respect to  $\mathcal{F}$ ) and paired dual space  $\mathcal{Y}$ . There exists a function  $\mu : \mathcal{F} \times \Omega \mapsto [0, 1]$  for which

1.  $\mu(\cdot, \omega)$  is a probability measure from  $\mathcal{P}_{\mathcal{Y}|\mathcal{H}}(\omega)$  on  $(\Omega, \mathcal{F})$ ,  $\forall \omega \in \Omega$ ;
2.  $\mu(A, \cdot)$  is a version of  $\mathbb{E}_{\mathbb{P}}[\mathbb{1}_A | \mathcal{H}]$ ,  $\forall A \in \mathcal{F}$ , and is  $\mathcal{H}$  measurable.

We should think of this function as a conditional probability function. The key result here is that any probability measure from  $\mathcal{P}_{\mathcal{Y}_3|\mathcal{F}_1}$  can be decomposed into a probability measure from  $\mathcal{P}_{\mathcal{Y}_2|\mathcal{F}_1}$  and a function  $\mu(\cdot, \omega)$  returning probability measures from  $\mathcal{P}_{\mathcal{Y}_3|\mathcal{F}_2}(\omega)$ . We provide an illustrative example below.

Example 3.5.

Consider ‘two coin toss’ probability space. Suppose we wish to decompose the measure

$$\mu_1^3(A) = \begin{cases} \frac{1}{3}, & A = \{HH\}, \\ \frac{1}{3}, & A = \{HT\}, \\ \frac{1}{3}, & A = \{TH\}, \\ 0, & A = \{TT\}. \end{cases}$$

Clearly, we must have

$$\mu_1^2(A) = \begin{cases} \frac{2}{3}, & A = \{HH, HT\}, \\ \frac{1}{3}, & A = \{TH, TT\}, \end{cases}$$

So our final component  $\mu_2^3(A, \omega)$  is given by

$$\mu_2^3(A, HH) = \mu_2^3(A, HT) = \begin{cases} \frac{1}{2}, & A = \{HH\}, \\ \frac{1}{2}, & A = \{HT\}, \\ 0, & A = \{TH\}, \\ 0, & A = \{TT\}. \end{cases}$$

and

$$\mu_2^3(A, TH) = \mu_2^3(A, TT) = \begin{cases} 0, & \omega = HH, \\ 0, & \omega = HT, \\ 1, & \omega = TH, \\ 0, & \omega = TT. \end{cases}$$

With this decomposition concept in mind, we present our main risk set decomposition theorem.

Lemma 3.2.

For a convex set  $Q_1^3 \subseteq \mathcal{P}_{\mathcal{Y}_3|\mathcal{F}_1}$  there exists a corresponding  $Q_1^2 \subseteq \mathcal{P}_{\mathcal{Y}_2|\mathcal{F}_1}$  and  $Q_2^3 : \Omega \times \mathcal{P}_{\mathcal{Y}_2|\mathcal{F}_1} \rightrightarrows \mathcal{P}_{\mathcal{Y}_3|\mathcal{F}_2}(\omega)$  such that  $Q_1^3 = Q_1^2 \circ Q_2^3$  where  $Q_1^2$  is a convex set and  $Q_2^3(\omega, \cdot)$  is a convex multifunction.

*Proof.* We will first construct the set  $Q_1^2$ . Define

$$Q_1^2 := \left\{ \mu_1^2 \in \mathcal{P}_{\mathcal{Y}_2|\mathcal{F}_1} \mid \mu_1^2(D) = \mu_1^3(D), \forall \mu_1^3 \in Q_1^3, \forall D \in \mathcal{F}_2 \right\} \quad (3.2)$$

We will now construct our risk sets from partitioning these probability measures. Consider the set

$$Q_2^3(\omega, \mu_1^2) := \bigcup_{\mu_1^3 \in Q_1^3} \{ \mu_2^3(\cdot, \omega) \in \mathcal{P}_{\mathcal{Y}_3|\mathcal{F}_2}(\omega) \mid \mu_1^3(A) = \int_{\bar{\omega} \in \Omega} \mu_2^3(A, \bar{\omega}) d\mu_1^2(\bar{\omega}), \forall A \in \mathcal{F}_3 \}. \quad (3.3)$$

By Lemma 3.1, for any measure  $\mu_1^3(A)$ , the function  $\mu_2^3(A, \omega)$  featured above is guaranteed to exist and is a measure belonging to  $\mathcal{P}_{\mathcal{Y}_3|\mathcal{F}_2}(\omega)$ .

Now that we have constructed  $Q_1^2$  and  $Q_2^3$ , we will show that they have the convexity properties required by this lemma. First, note that because  $Q_1^3$  is convex, then so is  $Q_1^2$ . We next wish to show that the induced  $Q_2^3(\omega, \cdot)$  is a convex multifunction. We proceed by showing the graph of the multifunction is convex. The graph of the multifunction  $Q_2^3(\omega, \mu_1^2)$  is the set

$$G(Q_2^3(\omega, \cdot)) = \left\{ (\mu_1^2(\cdot), \mu_2^3(\cdot, \omega)) \mid \mu_1^2 \in Q_1^2, \mu_2^3(\cdot, \omega) \in Q_2^3(\omega, \mu_1^2) \right\}.$$

Suppose that an arbitrary pair  $\tilde{\mu}_1^3, \hat{\mu}_1^3 \in Q_1^3$  induce

$$(\tilde{\mu}_1^2, \tilde{\mu}_2^3(\cdot, \omega)), (\hat{\mu}_1^2, \hat{\mu}_2^3(\cdot, \omega)) \in G(Q_2^3(\omega, \cdot)), \forall \omega \in \Omega. \quad (3.4)$$

Then an arbitrary third selection  $\dot{\mu}_1^3 = \lambda \tilde{\mu}_1^3 + (1 - \lambda) \hat{\mu}_1^3, \lambda \in [0, 1]$ , induces

$$(\lambda \tilde{\mu}_1^2 + (1 - \lambda) \hat{\mu}_1^2, \dot{\mu}_2^3(\cdot, \omega)),$$

for some  $\dot{\mu}_2^3(\cdot, \omega)$ . Denote  $\lambda \tilde{\mu}_1^2 + (1 - \lambda) \hat{\mu}_1^2$  as  $\dot{\mu}_1^2$  for notational convenience. We are required to show that

$$\dot{\mu}_2^3(\cdot, \omega) \in Q_2^3(\omega, \dot{\mu}_1^2), \forall \omega \in \Omega.$$

Now, by definition,  $\dot{\mu}_2^3(\cdot, \omega)$  satisfies

$$\dot{\mu}_1^3(A) = \int_{\bar{\omega} \in \Omega} \dot{\mu}_2^3(A, \bar{\omega}) d\dot{\mu}_1^2(\bar{\omega}), \forall A \in \mathcal{F}_3.$$

So  $\dot{\mu}_2^3(\cdot, \omega) \in Q_2^3(\omega, \dot{\mu}_1^2)$  if  $\dot{\mu}_1^3 \in Q_1^3$ . Because  $Q_1^3$  is convex, the measure  $\dot{\mu}_1^3 = \lambda \tilde{\mu}_1^3 + (1 - \lambda) \hat{\mu}_1^3, \lambda \in [0, 1]$ , is in  $Q_1^3$  since both  $\tilde{\mu}_1^3, \hat{\mu}_1^3 \in Q_1^3$ , completing the proof.

□

So we now have the ability to decompose a risk set into two further risk sets whose composition gives our initial risk set. With the parametric risk sets, we can now overcome our law invariance problem. By defining our initial risk set  $Q_1^3$  as a risk set which is law invariant, and computing its decomposition, we can use the composite risk sets to define a set of dynamic programming equations which guaranteed us time-consistency.

In Lemma 3.2, we showed that  $Q_2^3(\omega, \mu_1^2)$  is a convex multifunction; although not of direct importance here, this property has significance when incorporating these risk measures in optimisation problems. We note here that in contrast to the non-parametric conditional risk sets; the cartesian product of these parametric conditional risk sets need not be rectangular. This is because the risk set of later stages depends on the particular probability measure selected by a earlier risk set.

### 3.3.4 A heuristic interpretation

Now that all the mathematics of our objects are laid out, we offer a heuristic interpretation of these ideas. Consider our coin toss space with the discrete uniform measure shown in Figure 3.2a. Suppose our initial evaluation of the acceptability of

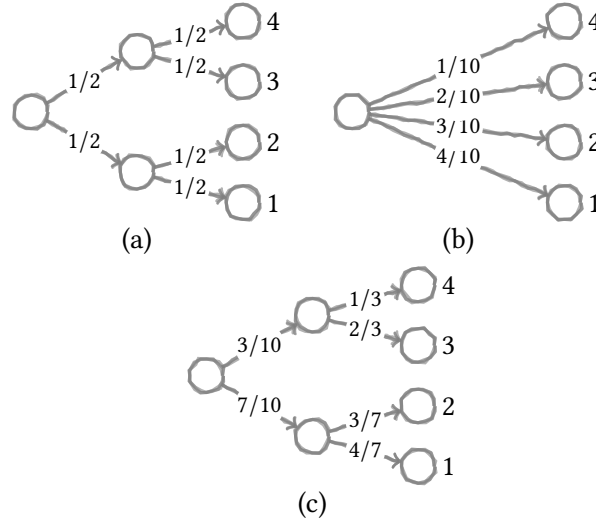


Figure 3.2: Decomposition on the Coin toss space

$X$  is given by some measure  $\mu_1^3$  i.e.  $v_1^3(X) = \mathbb{E}_{\mu_1^3}[X] = 2.5$  – This notion is captured by Figure 3.2b. We can decompose this measure into two measures  $\mu_1^2$  and  $\mu_2^3(\omega)$ , whose composition form our original measure  $\mu_1^3$  as in Figure 3.2c. Suppose we would like to re-evaluate our position given some new information. This is simple, we simply evaluate  $\mathbb{E}_{\mu_1^3}[X|\mathcal{F}_2]$ . So we would have

$$\mathbb{E}_{\mu_1^3}[X|\mathcal{F}_2](\omega) = \begin{cases} 3 + 1/3, & \text{if } \omega = HH, \\ 3 + 1/3, & \text{if } \omega = HT, \\ 1 + 3/7, & \text{if } \omega = TH, \\ 1 + 3/7, & \text{if } \omega = TT. \end{cases}$$

Now the question of time-consistency is all that is left to answer. We have to ensure that the optimisation problem faced when making the re-evaluation agrees with the initial valuation. This is achieved *by construction*.

In the beginning of this chapter, we discussed that dynamic programming equations automatically have a time consistency property. Even though our risk sets are *non-rectangular*, we can still form dynamic programming equations which compute the risk-adjusted value of a random variable in a dynamic fashion. The first value function (and eventual value) is given by

$$v_1^3(X) = \inf_{\mu_1^2 \in \mathcal{Q}_1^2(\omega)} \langle \mu_1^2, [v_2^3(X, \mu_1^2)](\omega) \rangle,$$



and the second (and final) value function is given by

$$[v_2^3(X, \mu_1^2)](\omega) = \inf_{\mu_2^3 \in Q_2^3(\omega, \mu_1^2)} \langle \mu_2^3, X \rangle.$$

Note that  $\langle \mu_2^3, \cdot \rangle$  is equivalent to  $\mathbb{E}_{\mu_2^3}[\cdot]$ ; hence the above dynamic formulation may appear more natural in the following form:

$$v_1^3(X) = \inf_{\mu_1^2 \in Q_1^2(\omega)} \mathbb{E}_{\mu_1^2} [v_2^3(X, \mu_1^2)](\omega),$$

and

$$[v_2^3(X, \mu_1^2)](\omega) = \inf_{\mu_2^3 \in Q_2^3(\omega, \mu_1^2)} \mathbb{E}_{\mu_2^3} [X].$$

The structure of  $Q_2^3(\omega, \mu_1^2)$  ensures that after solving the dynamic programming equations, the probability measures that are selected, when composed, form a measure  $\mu_1^3$  which lies within our original ‘end-of-horizon’ risk set  $Q_1^3$  and indeed solves

$$\inf_{\mu_1^3 \in Q_1^3} \mathbb{E}_{\mu_1^3} [X].$$

We are familiar with the above notion with traditional *rectangular* risk sets; because our conditional risk sets can now be *parametric* or *non-rectangular*, we can still form dynamic equations, and so does not disqualify them from having our time-consistency notion. In conversations with Alexander Shapiro in 2017, he commented that such a construction was a valid approach in creating time-consistent and law invariant mappings.

So by using the parametric risk sets we have developed in this chapter, we are able to form object functions which are both law invariant and time-consistent. We contend that these risk measures are a suitable class of risk measures for use in optimisation; by being law invariant, we can be sure that they will preserve stochastic dominance, and by being time-consistent, they ensure evaluations of risk through time do not contradict each other, a condition argued at the beginning of this chapter which is necessary for optimality.



## Chapter 4

# Multistage stochastic programming algorithms

In this chapter, we will present a class of optimisation problems called *multistage stochastic programmes*. These problems are a natural extension of traditional two-stage stochastic programming problems, where instead of making a control contingent on the outcome of one observation of uncertainty, controls are made contingent on the outcome of a stochastic process adapted to some filtration of some arbitrary length. Early versions of these formulations were studied by Birge (1985) and have since then been a large focus of the stochastic programming community. These problems are notorious in their difficulty; this is due to the exponential growth of the scenario tree that represents the filtration on the problem's probability space. For this reason, decomposition methods are the commonly utilised. Rather than solving the problem in its extensive form; we form dynamic programming equations and construct value/cost-to-go functions.

Because we seek to minimise the expected cost of a policy, we say that these problems are *risk neutral*. We can easily adapt the formulation to include risk averse objective functions, but this risk neutral class of problems are difficult enough in their own right. The risk averse versions of these problems will be the focus of Chapter 5. The work presented here is based in part upon Baucke, Downward, et al. (2017).

## 4.1 Problem description

In this section we will develop a new notation for describing filtered probability space, and will then outline in mathematical precision the form of optimisation problems considered.

### 4.1.1 Scenario tree representation

For a given discrete finite probability space  $(\Omega, \mathcal{F}, \mathbb{P})$ , with a filtration  $\{\mathcal{F}_t\}_{t=0}^T$ , we can form a directed graph with edge weights  $(\{\mathcal{N}, \mathcal{E}\}, p)$  where  $p : \mathcal{E} \mapsto [0, 1]$  which is isomorphic to the filtered probability space. We require a further assumption that  $\mathbb{P}(\omega) > 0, \forall \omega \in \Omega$ . Consider the set of atoms

$$\mathcal{N} = \bigcup_{\omega \in \Omega, t \leq T} \{i(a_{\mathcal{F}_t}(\omega))\}.$$

Let these atoms be the set of vertices on our graph and let  $i$  be the bijection from atoms to vertices. Now define a set of edges in our graph from

$$\mathcal{E} = \bigcup_{\omega \in \Omega, t < T} \{(i(a_{\mathcal{F}_t}(\omega)), i(a_{\mathcal{F}_{t+1}}(\omega)))\}.$$

It can be shown that the graph generated above is a tree in the graph theory sense; we often refer to the graph as a scenario tree. Further, we can consider a mapping of edges to  $p$  to the unit interval such that  $p((n_1, n_2)) = \mathbb{P}(i^{-1}(n_2))/\mathbb{P}(i^{-1}(n_1))$ . It is easy to see how the graph and edge weights  $(\{\mathcal{N}, \mathcal{E}\}, p)$  recover the original probability space; the universal set is given by

$$\Omega^* = \bigcup_{\omega \in \Omega} \{a_{\mathcal{F}_T}(\omega)\},$$

the filtration  $\{\mathcal{F}_t^*\}_{t=0}^T$  can be generated from

$$\left\{ \sigma \left( \bigcup_{\omega \in \Omega} a_{\mathcal{F}_t}(\omega) \right) \right\}_{t=0}^T,$$

and finally, the probability measure is given by

$$\mathbb{P}^*(\omega) = \prod_{t=0, \dots, T-1} p(i(a_{\mathcal{F}_t}(\omega)), i(a_{\mathcal{F}_{t+1}}(\omega))).$$

The graph object provides a convenient notation when describing multistage stochastic optimisations problems and the algorithms which solve them.

### 4.1.2 Problem formulation

Given some discrete probability space  $(\Omega, \mathcal{F}, \mathbb{P})$  and filtration  $\{\mathcal{F}_t\}_{t=0}^T$  on the probability space, we can describe a multistage stochastic optimisation problem as  $V_0(x_0(\omega)) =$

$$\begin{aligned} \min_{u_t} \quad & \mathbb{E}_{\mathbb{P}} [C_0(x_0(\omega), u_0(\omega), \omega) + \dots + C_{T-1}(x_{T-1}(\omega), u_{T-1}(\omega), \omega) + V_T(x_T(\omega), \omega)] \\ \text{s.t.} \quad & u_t(\omega) \in \mathcal{U}_t(x_t(\omega), \omega), \forall t < T, \forall \omega \in \Omega, \\ & x_t(\omega) \in \mathcal{X}_t(\omega), \forall t \leq T, \forall \omega \in \Omega, \\ & x_{t+1}(\omega) = f_{t+1}(x_t(\omega), u_t(\omega), \omega), \forall t < T, \forall \omega \in \Omega, \\ & u_t(\omega) \text{ is } \mathcal{F}_t \text{ measurable}, \forall t < T. \end{aligned} \tag{4.1}$$

The sets  $\mathcal{X}_t(\omega)$ , multifunctions  $\mathcal{U}_t(x_t, \omega)$ , cost functions  $C_t(x, u, \omega)$ , and transition functions  $f_t(x, u, \omega)$  are to be  $\mathcal{F}_t$  measurable for all  $t$ . Finally, for non-anticipativity, we require that our control  $u_t(\omega)$  is a  $\mathcal{F}_t$  measurable random variable; this induces the appropriate measurability in states  $x_t(\omega)$ . The constraint of non-anticipativity is a natural one. It says that the control at time 0 must be the same  $\forall \omega \in \Omega$ ; without this constraint our policy would otherwise be allowed ‘anticipate’ the future in each scenario at time 0 and make the corresponding optimal control for each scenario. This notion generalises to all stages  $t$ . Within the set of optimisation problems, we only consider a subset with the following conditions (holding  $\forall \omega \in \Omega$ ):

1. the state space  $\mathcal{X}_t(\omega)$  is a non-empty subset of  $\mathbb{R}^n$ ,  $\forall t \leq T$ ;
2. multifunctions  $\mathcal{U}_t(\cdot, \omega) : \mathbb{R}^n \rightrightarrows \mathbb{R}^m$  are convex and non-empty compact valued  $\forall t < T$ ;
3. the final cost function  $V_T$  is a convex lower semicontinuous proper function. The cost functions  $C_t(x, u, \omega)$ , are convex lower semicontinuous proper functions of  $x$  and  $u \forall t < T$ .
4. functions  $f_t(x, u, \omega)$  are affine  $\forall t < T$ ;
5. the final cost function  $V_T(x, \omega)$  is finite-valued and Lipschitz-continuous on  $\mathcal{X}_T(\omega)$ ;
6. for all  $t < T$ , there exists  $\delta_t(\omega) > 0$ , defining  $\mathcal{X}'_t(\omega) := \mathcal{X}_t(\omega) + B_t(\delta_t(\omega))$ , where

$$B_t(\delta) = \{y \in \text{Aff}(\mathcal{X}_t) \mid \|y\| \leq \delta\}$$

such that

- a)  $\forall x \in \mathcal{X}'_t(\omega), \forall u \in \mathcal{U}_t(x, \omega), C_t(x, u, \omega) < \infty;$
- b)  $\forall x \in \mathcal{X}'_t(\omega)$

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

Conditions 1-5 ensure that (4.1) and its value function definitions (yet to be introduced) 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 Girardeau et al. (2014), the authors call this condition *extended relatively complete recourse*, which is a more general class of recourse which includes complete recourse. Complete recourse is a condition that says that there is always a feasible control in the future, no matter which controls were made in the past. We note that these conditions are identical to those required by Girardeau et al. (2014) in their paper.

From here, we can produce dynamic programming equations which are required by our decomposition algorithms. By defining

$$\begin{aligned}
 V_1(x_1(\omega), \omega) &= \min_{u_t} \mathbb{E}_{\mathbb{P}}[C_1(x_1(\omega), u_1(\omega), \omega) + \dots + V_T(x_T(\omega), \omega) \mid \mathcal{F}_1](\omega) \\
 \text{s.t. } u_t(\omega) &\in \mathcal{U}_t(x_{t-1}(\omega), \omega), \forall 1 \leq t < T, \forall \omega \in \Omega, \\
 x_t(\omega) &\in \mathcal{X}_t(\omega), \forall 1 \leq t \leq T, \forall \omega \in \Omega, \\
 x_{t+1}(\omega) &= f_{t+1}(x_t(\omega), u_t(\omega), \omega), \forall 1 \leq t < T, \forall \omega \in \Omega, \\
 u_t(\omega) &\text{ is } \mathcal{F}_t \text{ measureable}, \forall 1 \leq t \leq T,
 \end{aligned}$$

we obtain

$$\begin{aligned}
 V_0(x_0(\omega)) &= \min_{u_0} C_0(x_0(\omega), u_0(\omega), \omega) + \mathbb{E}_{\mathbb{P}}[V_1(x_1(\omega), \omega)] \\
 \text{s.t. } u_0(\omega) &\in \mathcal{U}_1(x_1(\omega), \omega), \forall \omega \in \Omega, \\
 x_1(\omega) &\in \mathcal{X}_1(\omega), \forall \omega \in \Omega, \\
 x_1(\omega) &= f_1(x_1(\omega), u_1(\omega), \omega), \forall \omega \in \Omega, \\
 u_0(\omega) &\text{ is } \mathcal{F}_0 \text{ measureable.}
 \end{aligned}$$

By recursively performing this decomposition for all  $t < T$ , we obtain

$$\begin{aligned}
 V_t(x_t(\omega), \omega) = \min_{u_t} \quad & C_t(x_t(\omega), u_t(\omega), \omega) + \mathbb{E}_{\mathbb{P}}[V_{t+1}(x_{t+1}(\omega), \omega) \mid \mathcal{F}_t](\omega) \\
 \text{s.t.} \quad & u_t(\omega) \in \mathcal{U}_t(x_{t-1}, \omega), \forall \omega \in \Omega, \\
 & x_{t+1}(\omega) \in \mathcal{X}_t(\omega), \forall \omega \in \Omega, \\
 & x_{t+1}(\omega) = f_{t+1}(x_t(\omega), u_t(\omega), \omega), \forall \omega \in \Omega, \\
 & u_t(\omega) \text{ is } \mathcal{F}_t \text{ measurable.}
 \end{aligned}$$

By making use of our scenario tree isomorphism, we write the dynamic programming equations for all  $n \in \mathcal{N} \setminus \mathcal{L}$  as

$$\begin{aligned}
 V_n(x_n) = \min_{u_n} \quad & C_n(x_n, u_n) + \sum_{m \in R(n)} p(n, m) V_m(x_m) \\
 \text{s.t.} \quad & u_n \in \mathcal{U}_n(x_n), \\
 & x_m = f_m(x_n, u_n), \forall m \in R(n), \\
 & x_m \in \mathcal{X}_m, \forall m \in R(n);
 \end{aligned} \tag{4.2}$$

a noticeably more compact and intuitive form. The function  $R(n)$  maps the vertex  $n$  to the set of its children nodes. As mentioned earlier, the same dynamic programming formulations are studied in Girardeau et al. (2014) and several key results are developed. In their analysis, the existence of  $\delta_t$  from Condition (6) allows the construction of a Lipschitz constant to show that  $V_t(x_t)$  is the Lipschitz-continuous. As we shall see, this property is required in order to form *bounding functions* which estimate the value to go functions.

## 4.2 Preliminary concepts

With the multistage stochastic formulations in mind, in this section, we present several preliminary concepts required to describe and analyse the ensuing algorithms which solve these problems. We will review several *cutting plane methods* which grow in complexity as we approach our algorithm for solving multistage stochastic optimisation problems in the general case.

### 4.2.1 Kelley's cutting plane method

The genesis of *cutting plane methods* comes from the seminal work of Kelley (1960). In this work, Kelley outlines a procedure for computing the minimum of a convex

function by forming a series of lower approximations to the objective function. As we will discover, this algorithm naturally extends to both the two-stage and then multistage case. For now, we are concerned with solving the following optimisation problem:

$$w^* = \min_{u \in \mathcal{U}} W(u),$$

where  $\mathcal{U} \subset \mathbb{R}^n$  is convex and compact, and  $W(u)$  is a convex function and  $\alpha$ -Lipschitz on  $\mathcal{U}$ . We seek to form a lower bound on  $W$ , defined as the pointwise maximum of linear functions over the affine hull of  $\mathcal{U}$ . These linear functions are often referred to as *cuts*.

For a given  $\alpha$ -Lipschitz convex function  $W : \mathcal{U} \subset \mathbb{R}^n \mapsto \mathbb{R}$  and a finite set of sample points  $S$ , with  $u_s \in \mathcal{U}$ , we define the following lower bound function of  $W(u)$ .

**Definition 4.1.**

For a finite, non-empty set of sample points  $S$ , let

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

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

The lower bound function possesses many attractive qualities for use in decomposition algorithms. Three that we wish to highlight here are:

$$S_1 \subseteq S_2 \implies \underline{W}^{S_1}(u) \leq \underline{W}^{S_2}(u), \quad \forall u \in \mathcal{U}, \quad (4.3)$$

$$\underline{W}^S(u) \leq W(u), \quad \forall u \in \mathcal{U}, \quad (4.4)$$

$$\underline{W}^S(u_s) = W(u_s), \quad \forall s \in S \quad (4.5)$$

These three properties are exploited frequently throughout this paper. Furthermore, it can be seen that because  $W(u)$  is  $\alpha$ -Lipschitz, so is the lower bound function  $\underline{W}^S(u)$ . These facts will not be proved here; we contend that these properties are well known from previous works (for instance, Kelley [1960](#), Benders [1962](#)). The main result from Kelley ([1960](#)) is the convergence theorem presented below.

**Theorem 4.1 (Kelley's Cutting Plane Method).**

Consider the sequences

$$u^k = \arg \min_{u \in \mathcal{U}} \underline{W}^{S_{k-1}}(u), \quad S_k = S_{k-1} \cup \{u^k\}, \quad (4.6)$$



with

$$\underline{w}^k = \min_{u \in \mathcal{U}} \underline{W}^{S_{k-1}}(u). \quad (4.7)$$

We have that

$$\lim_{k \rightarrow \infty} w^* - \underline{w}^k = 0$$

For illustrative purposes, we will present the proof as it provides the conceptual basis for further convergence proofs in this thesis. For notational ease we define  $\underline{W}^{S_k}$  as  $\underline{W}^k$ .

*Proof of Theorem 4.1* From (4.6) and (4.7), we have

$$\underline{w}^k = \underline{W}^{k-1}(u^k), \quad \forall k,$$

and

$$w^* \leq W(u^k), \quad \forall k.$$

It follows then that,

$$w^* - \underline{w}^k \leq W(u^k) - \underline{W}^{k-1}(u^k), \quad \forall k. \quad (4.8)$$

We will now show that the right-hand side of (4.8) converges to 0 as  $k$  tends to infinity – this will complete the proof. Suppose there exist some  $\epsilon > 0$  for which

$$\epsilon \leq W(u^k) - \underline{W}^{k-1}(u^k), \quad \forall k.$$

Subtracting  $W(u^{\hat{k}}) - \underline{W}^{k-1}(u^{\hat{k}})$  from both sides gives

$$\epsilon - W(u^{\hat{k}}) + \underline{W}^{k-1}(u^{\hat{k}}) \leq W(u^k) - \underline{W}^{k-1}(u^k) - W(u^{\hat{k}}) + \underline{W}^{k-1}(u^{\hat{k}}), \quad \forall \hat{k}, k.$$

From the  $\alpha$ -Lipschitz continuity of  $W$  and  $\underline{W}^{k-1}$ , we have

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

From (4.5), the lower bound function is equal to the true function for  $\hat{k} < k$  i.e.  $\underline{W}^{k-1}(u^{\hat{k}}) = W(u^{\hat{k}})$ , so

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

which is a contradiction of the compactness of  $\mathcal{U}$ . It must follow then, that no  $\epsilon > 0$  exists and  $w^* - \underline{w}^k \rightarrow 0$  as  $k \rightarrow \infty$ .  $\square$

The essence of the algorithm is to iteratively refine a lower bound function in ‘areas of interest’; these ‘areas of interest’ are the minimums of the current best lower bound function.

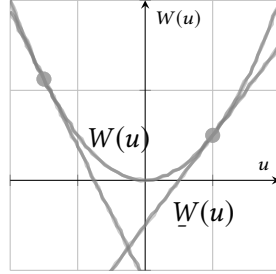


Figure 4.1: Convex function  $W(u)$  with a lower bound function of two sample points.

Although the lower bound function does not converge over the entire domain, we are still able to achieve our desired result of converging toward the *minimum* of  $W(u)$ . This can be seen in Figure 4.1. This philosophy of refining lower bound approximations is the foundation of all cutting plane algorithms.

#### 4.2.2 The stochastic case

Suppose that instead of minimising a single convex function using the above method, one was tasked with minimising the sum of several convex functions. This problem is very familiar to the stochastic programming community; the sum in question is an expectation of value functions over different scenario realisations. Without loss of generality, we wish to compute the value of

$$w^* = \min_{u \in \mathcal{U}} \sum_{\omega \in \Omega} p_{\omega} W_{\omega}(u), \quad (4.9)$$

with the same requirements of convexity and  $\alpha$ -Lipschitz continuity of  $W_{\omega}(u)$ , for all  $\omega \in \Omega$  and a convex and compact control set  $\mathcal{U}$ . Here  $p_{\omega}$  are positive weights; in the case where this is an expectation, these weights must sum to 1. We stress here that  $\Omega$  is finite. The first decomposition method for this problem class was developed in Benders (1962) and is commonly referred to as Benders Decomposition. This decomposition method presents itself in two main variants: the *average-cut* method and the *multi-cut* method. Briefly, an average-cut method brings a single cut back to the first stage problem at every iteration. This cut has gradients equal

to the average of all the second stage subgradients. Similarly, the constant term is the average of the second stage function evaluations. The average-cut lower bound function is given by

$$\underline{W}_\Omega^S(u) = \max_{s \in S} \sum_{\omega \in \Omega} p_\omega W_\omega(u_s) + \left\langle \sum_{\omega \in \Omega} p_\omega d_{\omega,s}, (u - u_s) \right\rangle \leq \sum_{\omega \in \Omega} p_\omega W_\omega(u).$$

In the multi-cut method, each function in the sum carries its own set of cuts i.e.

$$\underline{W}_\omega^S(u) = \max_{s \in S} W_\omega(u_s) + \langle d_{\omega,s}, (u - u_s) \rangle \leq W_\omega(u).$$

From these set of cuts, we can form a lower bound as

$$\sum_{\omega \in \Omega} p_\omega \underline{W}_\omega^S(u) \leq \sum_{\omega \in \Omega} p_\omega W_\omega(u).$$

So clearly both methods of utilising cut information form lower bounds. We present an elementary convergence result for the average-cut method (similar in style to Theorem 4.1) below.

#### Theorem 4.2 (Average-cut).

Consider the sequences

$$u^k = \arg \min_{u \in \mathcal{U}} \underline{W}_\Omega^{S_{k-1}}(u), \quad S_k = S_{k-1} \cup u^k, \quad (4.10)$$

with

$$\underline{w}^k = \min_{u \in \mathcal{U}} \underline{W}_\Omega^{S_{k-1}}(u). \quad (4.11)$$

We have that

$$\lim_{k \rightarrow \infty} w^* - \underline{w}^k = 0$$

*Proof.* From (4.10) and (5.17), we have

$$\underline{w}^k = \underline{W}_\Omega^{k-1}(u^k), \quad \forall k.$$

$$w^* \leq \sum_{\omega \in \Omega} p_\omega W_\omega(u^k), \quad \forall k.$$

It follows then that

$$w^* - \underline{w}^k \leq \sum_{\omega \in \Omega} p_\omega W_\omega(u^k) - \underline{W}_\Omega^{k-1}(u^k), \quad \forall k. \quad (4.12)$$

We will now show that the right-hand side of (5.18) converges to 0 as  $k$  tends to infinity – this will complete the proof. Suppose there exist some  $\epsilon > 0$  for

which

$$\epsilon \leq \sum_{\omega \in \Omega} p_{\omega} W_{\omega}(u^k) - \underline{W}_{\Omega}^{k-1}(u^k), \forall k.$$

Subtracting  $\sum_{\omega \in \Omega} p_{\omega} W_{\omega}(u^{\hat{k}}) - \underline{W}_{\Omega}^{k-1}(u^{\hat{k}})$  from both sides gives

$$\begin{aligned} \epsilon - \sum_{\omega \in \Omega} p_{\omega} W_{\omega}(u^{\hat{k}}) + \underline{W}_{\Omega}^{k-1}(u^{\hat{k}}) &\leq \\ \sum_{\omega \in \Omega} p_{\omega} W_{\omega}(u^k) - \underline{W}_{\Omega}^{k-1}(u^k) - \sum_{\omega \in \Omega} p_{\omega} W_{\omega}(u^{\hat{k}}) + \underline{W}_{\Omega}^{k-1}(u^{\hat{k}}), &\forall \hat{k}, k. \end{aligned}$$

From the  $\alpha$ -Lipschitz continuity of each  $W_{\omega}$  and  $\underline{W}_{\Omega}^k$ , we have

$$\epsilon - \sum_{\omega \in \Omega} p_{\omega} W_{\omega}(u^{\hat{k}}) + \underline{W}_{\Omega}^{k-1}(u^{\hat{k}}) \leq 2\alpha \|u^k - u^{\hat{k}}\|, \forall \hat{k}, k.$$

By the same reasoning in Theorem 4.1,  $\sum_{\omega \in \Omega} p_{\omega} W_{\omega}(u^{\hat{k}}) - \underline{W}_{\Omega}^{k-1}(u^{\hat{k}}) = 0$ ,  $\forall \hat{k} < k$ , so

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

which is a contradiction of the compactness of  $\mathcal{U}$ . It must follow then, that no  $\epsilon > 0$  exists and  $w^* - \underline{w}^k \rightarrow 0$  as  $k \rightarrow \infty$ .  $\square$

Because the multi-cut method provides a stronger lower bound, we can expect that the same convergence result for the average-cut method above holds for the multi-cut method. It is largely a consensus of the stochastic programming community that average-cut and multi cut techniques differ negligibly in computational time. Although multi-cut bounds are tighter, the computation of  $u^k$  is slightly more expensive due to the presence of more cuts. In Birge and F. V. Louveaux (1988) a study is conducted which makes comparisons between the two methods.

### 4.2.3 Forming upper bounds

Although the above algorithms do converge in the limit, in practice, in order to terminate in a finite number of iterations with a guarantee of the optimality gap, we require knowledge of an upper bound on the optimal solution. In order to achieve this during an iteration of either of the above algorithms, we need to evaluate ‘how far away’ we are from the optimal objective value. One way that this can be achieved is by evaluating the true cost function  $W(u)$  with the current iterate  $u^k$ . This value

is certainly an upper bound on

$$w^* = \min_{u \in \mathcal{U}} W(u),$$

since, by definition,  $W(u) \geq w^*$ ,  $\forall u \in \mathcal{U}$ . This is formalised in the following corollary.

Corollary 4.1.

Under the same conditions as Theorem 5.1 we have

$$\lim_{k \rightarrow \infty} W(u^k) - w^* = 0.$$

*Proof.* In the proof of Theorem 5.1, we showed that  $W(u^k) - W^{k-1}(u^k)$  converged to zero. By noting that  $W^{k-1}(u^k) \leq w^*$ , this corollary's result is obtained.

□

Evaluating the objective function  $W(\cdot)$  at a particular iterate is a straight-forward task. In the case of Kelley's Cutting Plane method, this requires only a single function evaluation. However in the two-stage stochastic case, it requires  $|\Omega|$  function evaluations. The definition of the limit result established by Kelley's Theorem

$$\lim_{k \rightarrow \infty} w^* - \underline{w}^k = 0$$

means that for any bound gap  $\epsilon > 0$ , there exists some  $k \in \mathbb{N}$  for which  $w^* - \underline{w}^k < \epsilon$ ; that is, there exists some iteration for which we will be  $\epsilon$  away from the optimal solution.

## 4.3 Nested Benders and SDDP

Now that we have outlined the basic cutting plane algorithms for deterministic and stochastic problems, we shall extend these to the multistage setting. We seek to solve the problem by constructing lower bounds to the cost-to-go functions which come from dynamic programming. 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 (1985) and has seen a lot of attention from researchers since. An apt name for the broader collection of these methods is 'nested Benders decomposition'. At

each iteration of our algorithm, we compute a state and control trajectory using the solution to the stage problems which contain our best lower approximation of the cost-to-go function. These lower approximation functions are then updated using zeroth and first-order information from the subsequent stage problem.

### 4.3.1 The deterministic case

In order to develop an intuition for the stochastic version of the problem, we first present the simpler deterministic case. The dynamic programming equations that we are interested in here are given by

$$\begin{aligned} V_t(x_t) &= \min_{x_{t+1}, u} C_t(x_t, u_t) + V_{t+1}(x_{t+1}) \\ \text{s.t. } x_{t+1} &= f_t(x_t, u_t) \\ x_{t+1} &\in \mathcal{X}_{t+1} \\ u_t &\in \mathcal{U}_t(x_t), \end{aligned} \tag{4.13}$$

for  $t \in \{1, \dots, T-1\}$ , where we seek to compute the value of  $V_0(x_0)$ . The function  $V_T(\cdot)$  is called the *terminal cost-to-go function*. We present an algorithm below known as the dual dynamic programming algorithm, first seen in Pereira and Pinto (1991).

Algorithm 4.1 (The dual dynamic programming algorithm).

**Initialisation.** Define  $V_t^0 := -\infty$ ,  $\forall t < T$ . Because the final value function is given, we set  $V_T^k = V_T$ ,  $\forall k \in \mathbb{N}$ . Set the initial state  $x_0^k = x_0$ ,  $\forall k$ . Finally set  $k = 1$ .

**Iteration  $k$ .**

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

**Step 2.** Solve

$$\begin{aligned} \theta_t^k &= \min_{x_{t+1}, u_t} C_t(x_t^k, u_t) + V_{t+1}^{k-1}(x_{t+1}) \\ \text{s.t. } x_{t+1} &= f_t^x(x_t^k, u_t), \\ x_{t+1} &\in \mathcal{X}_{t+1}, \\ u_t &\in \mathcal{U}_t(x_t^k), \end{aligned} \tag{4.14}$$

storing  $(x_{t+1}^k, u_t^k)$  as the minimisers. Compute a subgradient  $\beta_t^k$  with respect to  $x_t^k$ .

**Step 3.** Update the lower bound value function as

$$V_t^k(x) := \max\{V_t^{k-1}(x), \underline{\theta}_t^k + \langle \beta_t^k, x - x_t^k \rangle\}. \quad (4.15)$$

**Step 4.** Set  $t = t + 1$ . If  $t = T$ , move on to iteration  $k + 1$ . Otherwise, continue from **Step 2**.  $\bowtie$

Similar to Kelly's method, Algorithm 4.1 iteratively updates lower bound cost-to-go functions in areas of interest. However, in the multistage case, we do this recursively until the final value function is met.

**Remark 4.1.**

It is possible to delay the update of the lower bound function during the algorithm until the future vertices in the graph have been updated. In doing so, we could compute dominant cuts which would improve the speed of convergence. We have refrained from outlining such a procedure here, as it is not necessary for the convergence of our algorithm and may obfuscate the conceptual understanding of the algorithm. In some sense, the convergence result is more general as we are using weakest set of cuts possible.

The following lemma plays a critical role in proving the convergence on several algorithms in this thesis. It gives a conditional convergence result that, coupled with an induction and a base case, is the main device of our eventual convergence proof. This lemma has been formed in a manner such that it can be appealed to in a variety of circumstances; its generality and technicality require thoughtful consideration.

**Lemma 4.1.**

Consider a collection (indexed by the set  $N$ ) of sequences of  $\alpha$ -Lipschitz functions  $\bar{H}_n^k$  and  $\underline{H}_n^k$  on  $\mathcal{Z}_n \subset \mathbb{R}^{d_n}$  where  $\mathcal{Z}_n$  is compact for all  $n \in N$  and  $N$  is finite. 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  $\Upsilon : \mathbb{N} \mapsto N$ , we have

$$\begin{aligned} \lim_{k \rightarrow \infty} (\bar{H}_{\Upsilon(k)}^k(z_{\Upsilon(k)}^k) - \underline{H}_{\Upsilon(k)}^k(z_{\Upsilon(k)}^k)) = 0 &\implies \\ \lim_{k \rightarrow \infty} (\bar{H}_{\Upsilon(k)}^{k-1}(z_{\Upsilon(k)}^k) - \underline{H}_{\Upsilon(k)}^{k-1}(z_{\Upsilon(k)}^k)) = 0. \end{aligned}$$

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

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

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

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

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

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

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

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

From our lemma's hypothesis, we have that

$$\lim_{k \in \mathbb{N} \rightarrow \infty} (\bar{H}_{Y(k)}^k(z_{Y(k)}^k) - \underline{H}_{Y(k)}^k(z_{Y(k)}^k)) = 0,$$

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

$$\lim_{k \in \mathbb{N}^* \rightarrow \infty} (\bar{H}_{n^*}^k(z_{n^*}^k) - \underline{H}_{n^*}^k(z_{n^*}^k)) = 0.$$

This limit property, combined with the monotonicity of  $(\bar{H}_n^k(z) - \underline{H}_n^k(z))$  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  $(\bar{H}_{n^*}^{k-1}(z_{n^*}^{\hat{k}}) - \underline{H}_{n^*}^{k-1}(z_{n^*}^{\hat{k}})) \leq \frac{\epsilon}{2}$ ,  $\forall \tilde{k} \leq k$ ,  $\tilde{k} \leq \hat{k} < k$ , and both  $k, \hat{k}$  are in  $\mathbb{N}^*$ . So

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

Dividing through, we have

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



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

$$\epsilon \leq \bar{H}_{\Gamma(k)}^{k-1}(z_{\Gamma(k)}^k) - \underline{H}_{\Gamma(k)}^{k-1}(z_{\Gamma(k)}^k), \quad \forall k,$$

concluding the proof.  $\square$

We call this result a *conditional convergence* result. If bounding functions converge at a sequence, then the bounding functions which lag the sequence will converge at the same sequence. It is the key to the following theorem which proves that Algorithm 4.1 converges.

#### Theorem 4.3.

Consider the sequence of functions  $\underline{V}_t^k$  and state and control trajectories  $(x_t^k, u_t^k)$  generated by Algorithm 4.1. We have

$$\lim_{k \rightarrow \infty} (V_t(x_t^k) - \underline{V}_t^k(x_t^k)) = 0, \quad \forall t \leq T.$$

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

$$\lim_{k \rightarrow \infty} (V_{t+1}(x_{t+1}^k) - \underline{V}_{t+1}^k(x_{t+1}^k)) = 0, \quad \forall t < T.$$

Obviously this is true for last stage by definition. Now we want to show

$$\lim_{k \rightarrow \infty} (V_t(x_t^k) - \underline{V}_t^k(x_t^k)) = 0, \quad \forall t \leq T,$$

assuming this induction hypothesis. From the definition of our bounding functions in Step 2 and Step 3 in Algorithm 4.1 we have

$$\begin{aligned} \underline{V}_t^k(x_t^k) &\geq \underline{\theta}_t^k \geq \min_{\substack{u_t \in \mathcal{U}_t(x_t^k), \\ f_t(x_t^k, u_t) \in \mathcal{X}_{t+1}}} C_t(x_t^k, u_t) + \underline{V}_t^{k-1}(f_t(x_t^k, u_t)) \\ &\geq C_t(x_t^k, u_t^k) + \underline{V}_t^{k-1}(x_{t+1}^k), \quad \forall t < T. \end{aligned}$$

Now for any control  $u$  we must have

$$V_t(x_t^k) \leq C_t(x_t^k, u) + V_{t+1}(f_t(x_t^k, u)), \quad \forall t < T,$$

and so in particular, we have

$$V_t(x_t^k) \leq C_t(x_t^k, u_t^k) + V_{t+1}(x_{t+1}^k), \forall t < T.$$

Subtracting these, we have

$$V_t(x_t^k) - \underline{V}_t^k(x_t^k) \leq V_{t+1}(x_{t+1}^k) - \underline{V}_{t+1}^{k-1}(x_{t+1}^k), \forall t < T. \quad (4.16)$$

Now we invoke Lemma 4.1. In this instance  $N = \{t + 1\}$ . Our finite (i.e. 1) collection of function sequences is now  $V_{t+1}, \underline{V}_{t+1}^k$ , our finite collection of sequences of iterates on their respective compact sets are given by  $x_{t+1}^k$ , and as  $|N| = 1$ , our function  $\Upsilon : \mathbb{N} \mapsto N$  is given by  $\Upsilon(k) = t + 1, \forall k \in \mathbb{N}$ . This has the right-hand side of (4.16) approaching 0 as  $k$  tends to infinity. This proves the induction hypothesis. Coupled with our base case, this concludes the proof.

□

### 4.3.2 The stochastic case

In this section, we extend the problem formulation from deterministic to stochastic. By a small modification of the deterministic dynamic programming equations, the associated dynamic programming equations for the stochastic case are given by

$$\begin{aligned} V_n(x_n) &= \min_{x_m, u_n} C_n(x_n, u_n) + \sum_{m \in R(n)} p(n, m) V_m(x_m) \\ \text{s.t. } x_m &= f_m(x_n, u_n), \forall m \in R(n), \\ x_m &\in \mathcal{X}_m, \forall m \in R(n), \\ u_n &\in \mathcal{U}_n(x_n). \end{aligned} \quad (4.17)$$

In the stochastic case in the previous section, we formed a lower bound by sampling every child  $\omega \in \Omega$  and formed either an average-cut or multi-cut lower bound approximation function. This required ‘enumeration’: we had to evaluate and obtain a subgradient for every function, one per outcome on the probability space. Extending this to the multistage case is a valid, but impractical approach to solving multistage stochastic optimisation problems. This would require  $|\mathcal{N}|$  (the number of vertices that make up our scenario tree) cost function evaluations; typically a very large number for even small problem instances.

A clever means of circumventing the enumeration of the tree during every iteration, is to only update our bounding functions along only one path in our

scenario tree per iteration. While this does not reduce the amount of ‘work’ required to solve the problem in its entirety, it at least reduces the amount of ‘work’ required for a single iteration. This method begs the question: “how should an algorithm determine which path in the scenario to update bounding functions on?”

We will now focus on a popular algorithm for solving these dynamic programming equations which solves the problem of determining a path to update bounding functions along. The algorithm is known as the stochastic dual dynamic programming algorithm and was first seen in Pereira and Pinto (1991). Since then many authors have studied these problems and developed different versions of these algorithms. Philpott and Guan (2008) developed the Dynamic Outer Approximation Sampling Algorithm or DOASA, while Chen and W. Powell (1999) developed an algorithm called Cutting Plane and Partial Sampling Algorithm or CUPPS. A common feature linking these algorithms together is that they are random in their nature; they avoid enumeration by a randomly sampling the scenario tree. We present a generic *multi-cut* version of the algorithm below.

Algorithm 4.2 (A multicut SDDP algorithm).

**Initialisation.** Define  $\underline{V}_n^0 := -\infty$ ,  $\forall n \in \mathcal{N} \setminus \mathcal{L}$ . Because the leaf value function are given, we set  $\underline{V}_n^k = V_n$ ,  $\forall n \in \mathcal{L}$ . Set the initial state  $x_0^k = x_0$ ,  $\forall k$ . Finally set  $k = 1$ .

**Iteration  $k$ .**

**Step 1.** Randomly sample a path  $\omega^k(\cdot)$  *independently* from previous sample paths.

**Step 2.** Set  $t = 0$  and  $n = \omega^k(0)$ .

**Step 3.** Solve

$$\begin{aligned}
 \underline{\theta}_n^k &= \min_{x_m, u_n} C_n(x_n^k, u_n) + \sum_{m \in R(n)} p(n, m) \underline{V}_m^{k-1}(x_m) \\
 \text{s.t. } & x_m = f_m(x_n^k, u_n), \quad \forall m \in R(n) \\
 & x_m \in \mathcal{X}_m, \quad \forall m \in R(n) \\
 & u_n \in \mathcal{U}_n(x_n).
 \end{aligned} \tag{4.18}$$

storing  $(x_m^k, u_n^k)$  as the minimisers. Compute a subgradient  $\beta_n^k$  with respect to  $x_n^k$ .

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

$$V_n^k(x) := \max\{V_n^{k-1}(x), \theta_n^k + \langle \beta_n^k, x - x_n^k \rangle\}. \quad (4.19)$$

**Step 5.** Set  $t = t + 1$  and  $n = \omega^k(t)$ . If  $t = T$ , move on to iteration  $k + 1$ .

Otherwise, continue from **Step 3**. ▷

This algorithm works in a very similar way to the deterministic algorithm; a series of samples and updates is made on the vertices of the tree. In the stochastic case however, at each iteration only a random path in the scenario tree is updated. The random sampling in Algorithm 4.2 avoids enumeration of the tree, but our convergence result is weakened; Girardeau et al. (2014) prove that Algorithm 4.2 converges in the limit almost surely i.e.

$$\mathbb{P}\left(\lim_{k \rightarrow \infty} V_n(x_n^k) - V_n^k(x_n^k) = 0\right) = 1.$$

Philpott and Guan (2008) prove a slightly stronger result for multistage stochastic optimisation problems with linear costs and polyhedral state and control sets: a similar algorithm will (almost surely) converge in a finite number of iterations. Both of these convergence results rely on well-known results in probability; in Girardeau et al. (2014), they rely on the Law of Large Numbers, while in Philpott and Guan (2008), the authors rely on the Borel-Cantelli Lemma (see Grimmett et al. (2001)). The intuition behind their proofs is clear; if one randomly samples the scenario tree independently enough times, and the lower bound function can never decrease, then the value functions at every vertex in the tree ought to converge.

### 4.3.3 Assumptions regarding uncertainty

Any multistage optimisation problem of significant number of stages  $T$  and scenarios  $\Omega$ , grows to be exponentially large. For instance, consider the probability space generated by a scenario tree of depth  $T$ , and where each non-leaf vertex has  $N$  children. The number of vertices in such a tree is given by

$$\sum_{t=0, \dots, T-1} N^t \geq N^{T-1}.$$

So for  $N = 6$  and  $T = 10$ , the number of vertices in the tree is at least  $6^9 \approx 10,000,000$ ; one could not hope to solve, nor even encode numerically, an optimisation problem of this size.

However, under an independence assumption, we are able to greatly reduce the size of the problem. Consider two probability spaces  $(\Omega^a, \mathcal{F}^a, \mathbb{P}^a)$ ,  $(\Omega^b, \mathcal{F}^b, \mathbb{P}^b)$  and their product probability space  $(\Omega^a \times \Omega^b, \mathcal{F}^a \otimes \mathcal{F}^b, \mathbb{P}^a \otimes \mathbb{P}^b)$ . The  $\sigma$ -algebra given by  $\mathcal{F}_1 = \{\{\}, \Omega^a \times \Omega^b\}$ ,  $\mathcal{F}_2 = \mathcal{F}^a \otimes \{\{\}, \Omega^b\}$  and  $\mathcal{F}_3 = \mathcal{F}^a \otimes \mathcal{F}^b$  defines a valid filtration on the product space (implying that experiment  $a$  is performed first). We say that random variables which come from their respective spaces are independent. So any random variable  $X$  (not necessarily real-valued) originating from the space  $(\Omega^b, \mathcal{F}^b, \mathbb{P}^b)$  defined on the product space  $(\Omega^a \times \Omega^b, \mathcal{F}^a \otimes \mathcal{F}^b, \mathbb{P}^a \otimes \mathbb{P}^b)$  has the following feature:

$$X((\omega_1^a, \omega^b)) = X((\omega_2^a, \omega^b)), \forall \omega_1^a, \omega_2^a \in \Omega^a, \forall \omega^b \in \Omega^b.$$

After observing an event in  $\Omega^a$ , for the random variable  $X$  the next event  $\omega^b$  is not impacted by our earlier observation; the ‘future looks the same’ no matter which outcome in  $\Omega^a$  occurs. Thus by identifying vertices on the scenario tree whose ‘futures looks the same’, we can form a directed acyclic graph from our scenario tree; we have effectively *compressed* the scenario tree. Figure 4.2 illustrates this concept. By taking advantage of this compression, we can greatly reduce the size of the

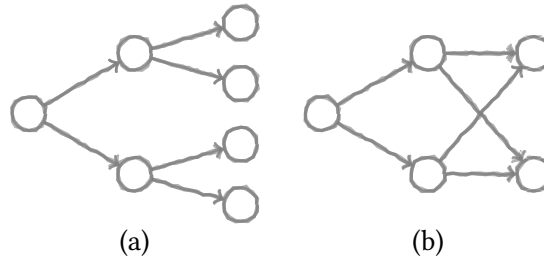


Figure 4.2: Compression of scenario tree

graph which encodes the problem. For our earlier example,  $N = 6$  and  $T = 10$ , we only require  $6 \times (10 - 1) + 1 = 55$  vertices to encode the problem. In practice, the assumption of independence (in the noises) is all but required to solve any large instance of a multistage stochastic optimisation problem.

Algorithm 4.2 in general does not 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. If we have independence, in any one iteration, the information we learn at a particular vertex could be valid for many vertices in our tree. This idea is commonly referred to as *cut-sharing* and is discussed extensively in Infanger and Morton (1996).

#### 4.3.4 Forming upper bounds in the stochastic case

In the deterministic case, we can form an upper bound on the objective value at the first stage in a manner similar to Kelley's Cutting plane method; we simply evaluate a given policy  $u_t^k$  on each of their cost function as in

$$\sum_{t=0, \dots, T-1} C_t(x_t^k, u_t^k) + V_T(x_T^k).$$

However, doing the same in the multistage stochastic i.e. evaluating

$$\sum_{n \in \mathcal{N} \setminus \mathcal{L}} p(n_0, n) C_n(x_n^k, u_n^k) + \sum_{n \in \mathcal{L}} V_n(x_n^k),$$

requires  $|\mathcal{N}|$  function evaluations (a full enumeration of the scenario tree) to evaluate the candidate policy; exactly what we sought to avoid. This problem is duly noted in Pereira and Pinto (1991), and it was then suggested that Monte Carlo simulation could be used to perform an estimate of such an evaluation. While these techniques vary in sophistication, an estimate of the cost of the candidate policy is evaluated (along with a confidence interval), then some statistical test is performed in order to determine whether or not the policy has converged. Shapiro (2011) suggests that a sensible stochastic convergence criterion is the following:

“A meaningful criterion would be to stop the procedure if the difference between the upper confidence bound and the true lower bound is less than a prescribed accuracy level  $\epsilon$ . This will suggest, with confidence of about  $(1 - \alpha)$ , that the SAA problem is solved with accuracy  $\epsilon$ .”

As we shall see in our computational study at the end of this chapter, performing this Monte Carlo can be an expensive exercise.

### 4.4 A deterministic algorithm

In this section, we present an algorithm for solving multistage stochastic optimisation problems which seeks to address the pitfalls of the SDDP algorithm: its probabilistic

convergence, and the expensive Monte Carlo simulation required in order to obtain an upper bound.

#### 4.4.1 An upper bound function

We begin by presenting our novel upper bound function. Consider an  $\alpha$ -Lipschitz continuous convex function  $W : \mathcal{U} \mapsto \mathbb{R}$ , 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  $S$  and  $u_s \in \mathcal{U}$ . We denote these ensuing upper and lower bounding functions  $\bar{W}$  and  $\underline{W}$ , respectively. Recall that the lower bound function can be written in *epigraph* form as

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

which can be solved as a linear programme. This lower bound function is a key to the algorithms presented so far. Complimentary to the lower bound function, we present the following definition of an upper bound convex function.

**Definition 4.2.**

For a finite, non-empty set of sample points  $S$ , let

$$\begin{aligned} \bar{W}^S(u) = \max_{\mu \in \mathbb{R}, \lambda \in \mathbb{R}^n} \quad & \mu + \langle \lambda, u \rangle \\ \text{s.t.} \quad & \mu + \langle \lambda, u_s \rangle \leq W(u_s), \quad \forall s \in S, \\ & \|\lambda\|_* \leq \alpha. \end{aligned} \tag{4.20}$$

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  $W(u_s)$  and have a bounded gradient. Upper bounds similar to this have been discussed in several previous works. We relate our upper bound function to Philpott, De Matos, and Finardi (2013) 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

$$\begin{aligned}
 \bar{W}^S(u) = \min_{\sigma_s} \quad & \sum_{s \in S} W(u_s) \sigma_s \\
 \text{s.t.} \quad & \sum_{s \in S} \sigma_s = 1, \\
 & \sum_{s \in S} \sigma_s u_s = u, \\
 & \sigma_s \geq 0, \forall s \in S.
 \end{aligned} \tag{4.21}$$

One of the drawbacks 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.  $\bar{W}(u) = \infty$ . So if  $\text{Conv}(S) \neq \mathcal{U}$ , then  $\bar{W}(u)$  is not Lipschitz on  $\mathcal{U}$ ; a property required by our ensuing algorithm. This idea can be traced back to Madansky (1959) where upper bounds on convex functions are also sought. Madansky correctly notes that in order to attain a bound where  $\bar{W}^S(u) < \infty$  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 dimension of  $\mathcal{U}$ . We avoid this issue by making use of the Lipschitz-continuity of our function of interest  $W$ . We arrive at Definition 4.2 by taking the linear programming dual of (4.21) and further constraining the gradient coefficients  $\lambda$ . Our upper bound definition has our desired Lipschitz continuity property directly encoded within its convex programming formulation.

**Lemma 4.2.**

The optimisation problem defining  $\bar{W}^S(u)$  for any non-empty  $S$  is bounded and feasible.

*Proof.* The point in  $(\mu, \lambda)$  space

$$\mu = \min_{s \in S} W(u_s), \text{ and } \lambda = \mathbf{0},$$

is always feasible for any  $S$ . Secondly, the vector  $\lambda$  is bounded above and below, while  $\mu$  is bounded from above in a maximisation problem, so clearly the optimisation problem defining  $\bar{W}^S(u)$  is bounded.  $\square$

We proceed by proving several important properties of our upper bound function which are critical for our deterministic algorithm.



Lemma 4.3.

If  $W(u)$  is a convex function and  $\alpha$ -Lipschitz on  $\mathcal{U}$  under the  $\|\cdot\|_p$ -norm, and  $u' \in S$ , then  $\bar{W}^S(u') = W(u')$ .

*Proof.* For a given  $S$ , the pair  $(\mu, \lambda)$  in the feasible region of (4.20) represent the parameters of a supporting hyperplane of the points  $(u_s, W(u_s))$ ,  $\forall s \in S$  where  $\mu$  is the intercept and  $\lambda$  is the gradient of the hyperplane. Recall from Lemma 5.3 that the feasible region is always non-empty. Because the function  $C(u)$  is convex in  $u$  and  $\alpha$ -Lipschitz, there exists a supporting hyperplane for which

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

where  $\|\tilde{d}\|_q \leq \alpha$  and  $\tilde{d}^u$  is in the subdifferential set of  $W(u)$  at  $u'$ . So a candidate solution for  $\bar{W}^S(u')$  is  $\tilde{\mu} = W(u') - \langle \tilde{d}, u' \rangle$  and  $\tilde{\lambda} = \tilde{d}$  which gives  $\bar{W}^S(u') = W(u')$ . If another candidate  $(\mu, \lambda)$  gives  $\mu + \langle \lambda, u' \rangle < W(u')$ , then the pair is not optimal because of the existence of  $(\tilde{\mu}, \tilde{\lambda})$ . If a further candidate  $(\mu, \lambda)$  gives  $\mu + \langle \lambda, u' \rangle > W(u')$ , then the pair does not support the point  $(u', W(u'))$ , making it infeasible. This completes the proof.  $\square$

Lemma 4.4.

If  $S^1 \subseteq S^2$ , then  $\bar{W}^{S^1}(u) \geq \bar{W}^{S^2}(u)$ .

*Proof.* The function  $\bar{W}^S(u, v)$  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.  $\square$

Lemma 4.5.

If  $W(u)$  is a convex function and  $\alpha$ -Lipschitz on  $\mathcal{U}$  under the  $\|\cdot\|_p$ -norm, then  $\bar{W}^S(u) \geq W(u)$ , for all  $u \in \mathcal{U}$ .

*Proof.* For the sake of contradiction, suppose there exists a  $u' \in \mathcal{U}$  for which  $\bar{W}^S(u') < W(u')$ . By defining  $S^* = S \cup u'$ , and by Lemma 5.5, we have  $W(u') = \bar{W}^{S^*}(u')$ . So  $\bar{W}^S(u') < \bar{W}^{S^*}(u')$ , which is precluded by Lemma 5.6, completing the proof.  $\square$

With the upper bound function  $\bar{W}(u)$  and its properties understood, we will show how it can be leveraged in a new algorithm for solving multistage deterministic optimisation problems.

#### 4.4.2 The deterministic algorithm

A feature of our algorithm is the ability to achieve deterministic convergence, even in the case where our problem 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 vertex in the scenario tree that gives the greatest bound closure for the given state. The work of Georghiou et al. (2017) 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 problem (called the *problem-child* selection problem) is simply an evaluation of the difference of the bounds at the next  $m$  future states and choosing the maximum. We present our algorithm below.

Algorithm 4.3.

**Initialisation.** Define  $\bar{V}_n^0 := \infty, \underline{V}_n^0 := -\infty, \forall n \in \mathcal{N} \setminus \mathcal{L}$ . Because the leaf value function are given, we set  $\bar{V}_n^k = \underline{V}_n^k = V_n, \forall n \in \mathcal{L}$ . Set the initial state  $x_0^k = x_0, \forall k$ . Finally set  $k = 1$ .

**Iteration  $k$ .**

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

**Step 2.** Solve

$$\begin{aligned} \underline{\theta}_n^k &= \min_{x_m, u_n} C_n(x_n^k, u_n) + \sum_{m \in R(n)} p(n, m) \underline{V}_m^{k-1}(x_m) \\ \text{s.t. } x_m &= f_m(x_n^k, u_n), \forall m \in R(n), \\ x_m &\in \mathcal{X}_m, \forall m \in R(n), \\ u_n &\in \mathcal{U}_n(x_n), \end{aligned} \tag{4.22}$$

storing  $u_n^k$  as the minimiser. Compute a subgradient  $\beta_n^k$  with respect to  $x_n^k$ .

**Step 3.** Solve

$$\begin{aligned}
 \bar{\theta}_n^k &= \min_{x_m, u_n} C_n(x_n^k, u_n) + \sum_{m \in R(n)} p(n, m) \bar{V}_m^{k-1}(x_m) \\
 \text{s.t. } x_m &= f_m(x_n^k, u_n), \forall m \in R(n), \\
 x_m &\in \mathcal{X}_m, \forall m \in R(n), \\
 u_n &\in \mathcal{U}_n(x_n).
 \end{aligned} \tag{4.23}$$

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

$$\underline{V}_n^k(x) := \max\{\underline{V}_n^{k-1}(x), \underline{\theta}_n^k + \langle \beta_n^k, x - x_n^k \rangle\}.$$

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

$$\begin{aligned}
 \bar{V}_n^k(x) &= \max_{\mu, \lambda} \mu + \langle \lambda, x \rangle \\
 \text{s.t. } \mu + \langle \lambda, x_n^{\hat{k}} \rangle &\leq \bar{\theta}_n^{\hat{k}}, \forall \hat{k} \leq k, \\
 ||\lambda|| &\leq \alpha_x.
 \end{aligned} \tag{4.24}$$

**Step 6.** Update  $\Phi_n^k$  as

$$\Phi_n^k = \arg \max_{m \in R(n)} p(n, m) (\bar{V}_m^{k-1}(f_m(x_n^k, u_n^k)) - \underline{V}_m^{k-1}(f_m(x_n^k, u_n^k))). \tag{4.25}$$

If  $\Phi_n^k$  is not unique, then select one arbitrarily. Further, update  $x_{\Phi_n^k}^k$ , as  $f_{\Phi_n^k}^x(x_n^k, u_n^k)$ . 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.  $x_n^{k+1} = x_n^k$ .  $\bowtie$

Contrary to the previous Algorithm [4.2](#), Algorithm [4.3](#) 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. This information is captured in the object  $\Phi_n^k$ , which keeps track of the child vertex of vertex  $n$  with the maximal difference in bounds at iteration  $k$  during the algorithm; keeping track of these vertices 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. We now state and prove our

deterministic convergence theorem.

Theorem 4.4.

Consider the sequence of functions  $\bar{V}_n^k$  and  $\underline{V}_n^k$ , state and control trajectories  $x_n^k, u_n^k$ , and problem-children  $\Phi_n^k$  generated by Algorithm 4.3. For all  $n \in \mathcal{N}$ , we have

$$\lim_{k \rightarrow \infty} (\bar{V}_n^k(x_n^k) - \underline{V}_n^k(x_n^k)) = 0.$$

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

$$\lim_{k \rightarrow \infty} (\bar{V}_{\Phi_n^k}^k(x_{\Phi_n^k}^k) - \underline{V}_{\Phi_n^k}^k(x_{\Phi_n^k}^k)) = 0.$$

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

$$\lim_{k \rightarrow \infty} (\bar{V}_n^k(x_n^k) - \underline{V}_n^k(x_n^k)) = 0, \quad \forall n \in \mathcal{N} \setminus \mathcal{L},$$

assuming the induction hypothesis. By arguments similar to those in Lemma 4.3, we have

$$\bar{V}_n^k(x_n^k) \leq C_n(x_n^k, u_n^k) + \sum_{m \in R(n)} p(n, m) \bar{V}_m^{k-1}(f_m^x(x_n^k, u_n^k)), \quad \forall n \in \mathcal{N} \setminus \mathcal{L}, \quad \forall k,$$

and symmetrically

$$\underline{V}_n^k(x_n^k) \geq C_n(x_n^k, u_n^k) + \sum_{m \in R(n)} p(n, m) \underline{V}_m^{k-1}(f_m^x(x_n^k, u_n^k)), \quad \forall n \in \mathcal{N} \setminus \mathcal{L}, \quad \forall k.$$

Subtracting these, we have

$$\begin{aligned} \bar{V}_n^k(x_n^k) - \underline{V}_n^k(x_n^k) &\leq \\ &\sum_{m \in R(n)} p(n, m) \left[ \bar{V}_m^{k-1}(f_m^x(x_n^k, u_n^k)) - \underline{V}_m^{k-1}(f_m^x(x_n^k, u_n^k)) \right], \quad \forall n \in \mathcal{N} \setminus \mathcal{L}, \quad \forall k. \end{aligned}$$

But by the problem-child selection principle (Step 6 of Algorithm 4.3), we must have

$$\bar{V}_n^k(x_n^k) - V_n^k(x_n^k) \leq |R(n)|p(n, \Phi_n^k) \left[ \bar{V}_{\Phi_n^k}^{k-1}(x_{\Phi_n^k}^k) - V_{\Phi_n^k}^{k-1}(x_{\Phi_n^k}^k) \right], \forall n \in \mathcal{N} \setminus \mathcal{L}, \forall k.$$

Similar to the proof of Theorem 4.3, we invoke Lemma 4.1 to conclude the proof; our finite collection of function sequences is now  $\bar{V}_m^k, V_m^k, \forall m \in R(n)$ , our finite collection of sequences of iterates on their respective compact sets are given by  $x_m^k, \forall m \in R(n)$ , and finally our map from  $\mathbb{N} \mapsto R(n)$  is given by  $\Phi_n^k$ . This proves the induction hypothesis. Coupled with our base case, this concludes the proof.  $\square$

The algorithm detailed above is deterministic, it requires no random sampling. Furthermore, there is no requirement to form an upper bound through policy simulation. The proof outline agrees with a certain intuition: that if a path which is guaranteed to have ‘disagreement’ in its bounds is sampled and improved, then the bound gaps over the whole tree must converge in the limit.

## 4.5 A computational study

In this section, we present the results of several computational experiments to demonstrate and compare the different algorithms explored in this chapter. We present a scalable test problem used to benchmark the implementations of Algorithms 4.2 and 4.3. Our implementations are written in Julia, making use of the JuMP library (see Dunning et al. 2015) to interface with Gurobi (see Gurobi Optimization 2016) to compute cut coefficients.

### 4.5.1 A test problem

In line with SDDP tradition, we present a multistage hydro-thermal scheduling problem to benchmark the algorithms. The problem considered consists of a three-reservoir, two-chain hydro system meeting demand at two nodes with a transmission line. The reservoirs receive stage-wise independent inflows. Demand can be met by hydro generation or by a mix of thermal generation whose fuel costs are deterministic

but may differ between stages. Many other hydro-thermal models with varying degrees of complexity exist – see Pereira and Pinto (1991) and Philpott and Pritchard (2013) as examples.

The structure of the hydro-thermal scheduling problem is best observed through its dynamic programming equations in (4.26) where:  $l_h$  is the reservoir level,  $r_h$  and  $s_h$  are the release and spill from reservoir  $h$  respectively;  $a_{f,i}$  is the supply of type  $f$  at node  $i$  which has a stage-dependent cost  $c_{f,i}^t$  per unit;  $\mathcal{U}(l)$  represents the reservoir dynamics and reservoir capacities; and  $f_m(r_h, s_h, l_h^t)$  are the linear dynamics of the system (containing a random inflow term).

$$\begin{aligned}
 V_n(l_h^t) = & \min_{a, l_{h,m}^{t+1}, r_h, s_h} \sum_{i \in \mathcal{I}} \sum_{f \in \mathcal{F}} c_{f,i}^t a_{f,i} + \mathbb{E}[V_m(l_{h,m}^{t+1})] \\
 \text{s.t. } & 0 \leq a_{f,i} \leq K_{f,i} \\
 & \sum_{f \in \mathcal{F}} a_{f,1} + \sum_{h \in \mathcal{H}} r_{h,1} + t = d_1^t \\
 & \sum_{f \in \mathcal{F}} a_{f,2} + \sum_{h \in \mathcal{H}} r_{h,2} - t = d_2^t \\
 & -T \leq t \leq T \\
 & (r_h, s_h) \in \mathcal{U}_n(l_h^t) \\
 & l_{h,m}^{t+1} = f_m(r_h, s_h, l_h^t), \quad \forall m \in R(n).
 \end{aligned} \tag{4.26}$$

Because of the stagewise independent structure of the uncertainty, we are able to share value function approximations between vertices in our scenario tree. The norm considered for this problem (which appears in Step 5 of Algorithm 4.3) is the  $\|\cdot\|_\infty$ -norm; this allows the upper bound function to be computed as a linear programme.

## 4.5.2 Results

We consider a 20-stage, 9-scenarios per stage instance of the problem above. Table 4.1 shows the results of several executions of the deterministic algorithm with a decreasing relative  $\epsilon$ , where

$$\epsilon^k = \frac{\bar{V}_0^k(x_0) - \underline{V}_0^k(x_0)}{\bar{V}_0^k(x_0)}.$$

As expected, in relatively few iterations, the Algorithm 4.3 is able produce a ‘good’ policy; similar to policies generated by the random sampling approach of 4.2

Table 4.1: Results for Problem 1

Relative $\epsilon$	Lower	Upper	Time (s)	Iterations	LP Solves
20.0%	406.86	506.89	48.02	295	28025
10.0%	413.10	457.63	137.26	533	50605
5.0%	416.25	438.06	396.39	929	88180
1.0%	419.01	423.24	7461.57	3653	346080

Much more computational effort is required to reduce the relative bound gap as the gap begins to close. We will discuss these ideas further in relation to Figures 4.3 and 4.4. It is worth mentioning here that without our method, a policy evaluation (in order to form a *deterministic* upper bound) for this problem would require 1,519,708,182,382,116,100 ( $\approx 1.52 \times 10^{18}$ ) linear programme evaluations.

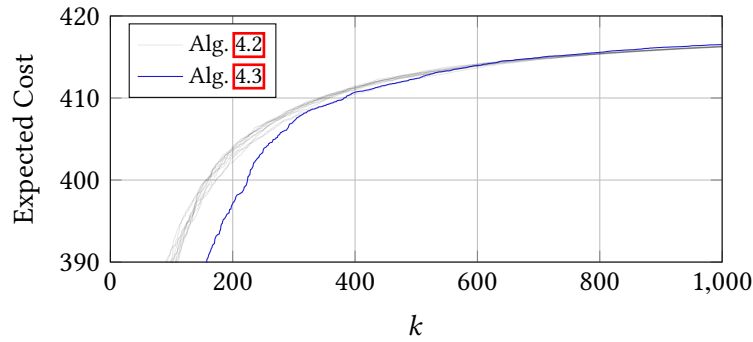


Figure 4.3: Lower bounds on the Expected Cost for the two algorithms.

Comparing results between our method and other methods requires careful consideration. We will first discuss the results shown in Figure 4.3. This figure plots the cost of the lower bound at each iteration i.e.  $\underline{V}_0^k(x_0)$ . Note that because Algorithm 4.2 uses random sampling, we have plotted the results of several runs of the algorithm on this problem, each with different random seeds. Here we can see that initially, the lower bound generated by Algorithm 4.3 lags behind the lower bounds generated by Algorithm 4.2. However, at iteration 600, we can see the lower bounds begin to match. Beyond this, our Algorithm 4.3 delivers a higher lower bound, for the same number of iterations.

We conjecture that the early poor performance of the lower bound generated by Algorithm 4.3 is due to the fact that directed sampling is not useful when the

bounding functions directing the sampling are inaccurate. As the bounding functions begin to converge, directed sampling provides more benefit. We conclude here that a random sampling scheme under stage-wise independence provides a coarse lower bound estimate quite quickly, but struggles to make the marginal gains towards the later portion of the algorithm.

Further, we also believe that good early performance of the lower bound generated by 4.2 is the relatively simple uncertainty structure (stage-wise independence) in our example. If our stochastic process is not atleast markovian, then we lose the ability to share value functions approximations between vertices in the scenario tree, making directed sampling more effective. To illustrate this concept, consider a multistage stochastic optimisation problem with a stochastic process with no independence or Markovian structure. As value function information cannot be shared between the vertices of the scenario tree, it becomes more likely that after several iterations of a random sampling algorithm, the bound gap in the value functions at each stage are likely to be far larger than in a problem where we have independence in our stochastic process. As such, being able to deterministically sample the path with the largest bound gap (rather than random sampling), becomes not only useful, but critical. When considering these types of optimisation problems, we assert that our deterministic sampling algorithm would perform significantly better than a random sampling method.

Moreover, the reason the random sampling method cannot close the bound gap towards iteration 1000, is that these may be associated with specific sample paths that have not been sampled, whereas our method identifies a path to search which guarantees an improvement at each iteration.

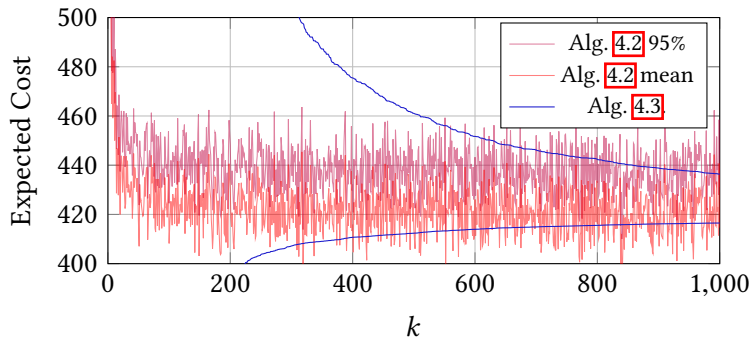


Figure 4.4: Convergence plot with Monte Carlo estimates for an upper bound.



### 4.5.3 Solution verification

Figure 4.4 shows both the upper bound and lower bound of the deterministic method, and an 800 sample Monte Carlo estimate of the expected cost of a policy generated from random sampling<sup>1</sup>. The purple line is the upper level of a 95% confidence interval constructed about the mean i.e. we are 95% sure that the expected cost of the current policy lies below this line and above the 95% confidence interval lower bound (not shown in the plot).

Table 4.2: Monte Carlo estimates required for convergence tests

Gap	Confidence Level			
	95%	97.5%	99%	99.9%
10%	440	574	762	1238
5%	1832	2393	3174	5161
1%	48690	63595	84366	137189

The construction of these confidence intervals (which is required for many statistical convergence tests) can be computationally expensive and requires the user to determine how often this check is performed, and to what confidence level. We contrast this with our deterministic algorithm: for each iteration of our algorithm, we have a bound on the maximum distance the current policy is from optimality.

In Table 4.2 above, we investigate the computational expense required to build confidence intervals to various tolerance levels (% gap) assuming the variance given by the policy in Figure 4.4 at iteration 1000. Specifically, we compute the number of Monte Carlo samples required to deliver a confidence interval with the same width as 1%, 5% or 10% of the best lower bound for the objective function. For example, almost 140,000 Monte Carlo samples would have to be carried out in order to form a 99.9% confidence interval with a width equal to the difference of the deterministic upper and lower bound at 1%. Because each sample requires the evaluation of 20 stage problems, this number of samples requires more than six times the number of linear programming evaluations (simply to generate a confidence interval) than our method would take to give a solution with a guaranteed 1% gap.

Finally, we note that convergence of a stochastic upper bound is sensitive to the variance of the distribution of costs given by a particular policy. The higher the

<sup>1</sup>I would like to acknowledge Oscar Dowson for kindly providing the code which performed the Monte Carlo simulation from a policy generated from an implementation of Algorithm 4.2.

policy's variance, the wider the confidence interval will be for a given sample-size and confidence level. This means that random sampling algorithms may find no evidence that the policy has not converged, even when the policy is far from optimal. However, since Algorithm 4.3 is deterministic, we do not need to simulate to estimate an upper bound and so never encounter cases where a convergence test gives either false positives or false negatives.

## Chapter 5

# Risk averse multistage stochastic programmes

This chapter forms the main core of the thesis. In this chapter we aim to combine the theory of coherent risk measures developed in the earlier chapters with the multistage stochastic programming models in Chapter 4.

First, we introduce a general risk averse multistage stochastic optimisation formulation, and develop dynamic programming equations. Secondly, we review the current methods for solving these problems and discuss their shortcomings. We then introduce a general multistage stochastic minimax dynamic programming theory, which attempts to remedy some of these shortcomings. To conclude, we show how our risk averse multistage stochastic optimisation models can be cast in a form which can leverage our newly developed minimax dynamic programming theory, and produce a numerical example. The work presented here is based in part upon Baucke, Downward, et al. (2018).

### 5.1 Problem class and current techniques

In this section, we will consider a class of problems which are almost identical to those considered in Chapter 4; however, rather than solving these optimisation problems in expectation, we wish to include risk aversion into the objective function in some way. For this, the conditional risk measures developed in Chapter 3 will take the place of the expectation operator in the objective functions in order to map random variables to the reals. It is by no means settled as to which is the best way to incorporate risk into multistage optimisation problems. The work of Homem-

De-Mello and Pagnoncelli (2016) showcases a class of dynamic risk measures called *expected conditional risk measures*. The authors of this work show how multistage stochastic optimisation with the expected conditional risk measure as the objective function, can be written in a risk neutral form with the addition of several auxiliary variables. We have chosen not to focus on these types of risk measures in this section.

We do note here that rather than minimising convex costs, like in Chapter 4, in this chapter our problems will be posed in terms maximising *concave* rewards; this avoids several notational problems.

### 5.1.1 Problem formulation

Given some discrete probability space  $(\Omega, \mathcal{F}, \mathbb{P})$  and filtration  $\{\mathcal{F}_t\}_{t=0}^T$  on the probability space, we can describe a *risk averse* multistage stochastic optimisation problem as  $V_0(y_0(\omega)) =$

$$\begin{aligned} \max_{v_t} \quad & v_0^T [C_0(y_0(\omega), v_0(\omega), \omega) + \dots + C_{T-1}(y_{T-1}(\omega), v_{T-1}(\omega), \omega) + V_T(y_T(\omega), \omega)] \\ \text{s.t.} \quad & v_t(\omega) \in \mathcal{V}_t(y_t(\omega), \omega), \quad \forall t < T, \forall \omega \in \Omega, \\ & y_t(\omega) \in \mathcal{Y}_t(\omega), \quad \forall t \leq T, \forall \omega \in \Omega, \\ & y_{t+1}(\omega) = f_{t+1}(y_t(\omega), v_t(\omega), \omega), \quad \forall t < T, \forall \omega \in \Omega, \\ & v_t(\omega) \text{ is } \mathcal{F}_t \text{ measureable}, \forall t < T. \end{aligned} \tag{5.1}$$

Here  $y$  represents the state of the system, while  $v$  represents the system control. The remaining problem data  $(\mathcal{Y}_t(\omega), \mathcal{V}_t(y, \omega), \text{etc.})$  take the same form as their risk neutral counterparts from Chapter 4 i.e. convexity in the sets, concavity in the reward functions, all with the appropriate measurability. The function  $v_0^T$  must be a conditionally coherent acceptance measure which has a decomposition i.e.

$$v_0^T = v^1 \circ \dots \circ v_{T-1}^T.$$

From here, and in the same the same fashion as Chapter 4, we can produce dynamic programming equations, which are required by our decomposition algorithms. By

defining

$$\begin{aligned}
 V_1(y_1(\omega), \omega) &= \max_{v_t} v_1^T [C_1(y_1(\omega), v_1(\omega), \omega) + \dots + V_T(y_T(\omega), \omega)](\omega) \\
 \text{s.t. } v_t(\omega) &\in \mathcal{V}_t(y_t(\omega), \omega), \forall 1 \leq t < T, \forall \omega \in \Omega, \\
 y_t(\omega) &\in \mathcal{Y}_t(\omega), \forall 1 \leq t \leq T, \forall \omega \in \Omega, \\
 y_{t+1}(\omega) &= f_{t+1}(y_t(\omega), v_t(\omega), \omega), \forall 1 \leq t < T, \forall \omega \in \Omega, \\
 v_t(\omega) &\text{ is } \mathcal{F}_t \text{ measvrealable}, \forall 1 \leq t < T,
 \end{aligned}$$

we obtain

$$\begin{aligned}
 V_0(y_0(\omega)) &= \max_{v_0} C_0(y_0(\omega), v_0(\omega), \omega) + v_0^1 [V_1(y_1(\omega), \omega)] \\
 \text{s.t. } v_0(\omega) &\in \mathcal{V}_0(y_0(\omega), \omega), \forall \omega \in \Omega, \\
 y_1(\omega) &\in \mathcal{Y}_1(\omega), \forall \omega \in \Omega, \\
 y_1(\omega) &= f_1(y_0(\omega), v_0(\omega), \omega), \forall \omega \in \Omega, \\
 v_0(\omega) &\text{ is } \mathcal{F}_0 \text{ measureable.}
 \end{aligned}$$

By recursively performing this decomposition for all  $t < T$ , we obtain

$$\begin{aligned}
 V_t(x_t(\omega), \omega) &= \max_{v_t} C_t(y_t(\omega), v_t(\omega), \omega) + v_t^{t+1} [V_{t+1}(y_{t+1}(\omega), \omega)](\omega) \\
 \text{s.t. } v_t(\omega) &\in \mathcal{V}_t(y_t(\omega), \omega), \forall \omega \in \Omega, \\
 y_{t+1}(\omega) &\in \mathcal{Y}_{t+1}(\omega), \forall \omega \in \Omega, \\
 y_{t+1}(\omega) &= f_{t+1}(y_t(\omega), v_t(\omega), \omega), \forall \omega \in \Omega, \\
 v_t(\omega) &\text{ is } \mathcal{F}_t \text{ measureable.}
 \end{aligned}$$

By making use of our scenario tree notation (and with a slight abuse of notation regarding  $v_n$ ), we write the dynamic programming equations as

$$\begin{aligned}
 V_n(y_n) &= \max_{v_n} C_n(y_n, v_n) + v_n [V_m(y_m)]_{m \in R(n)} \\
 \text{s.t. } v_n &\in \mathcal{V}_n(y_n), \\
 y_m &\in \mathcal{Y}_m, \forall m \in R(n), \\
 y_m &= f_m(y_n, v_n), \forall m \in R(n),
 \end{aligned} \tag{5.2}$$

for all  $n \in \mathcal{N} \setminus \mathcal{L}$ . By  $v_n$ , we mean a conditional coherent acceptance measure which maps the value of the children vertices to their parent. By making use of the dual representation of risk measures, where  $v_n$  induces a conditional risk set  $\mathcal{Q}_{v_n}$ , we can

write the above as

$$\begin{aligned}
 V_n(y_n) &= \max_{v_n} C_n(y_n, v_n) + \min_{\mu_m \in Q_{\rho n}} \sum_{m \in R(n)} \mu_m \times V_m(y_m) \\
 \text{s.t. } v_n &\in \mathcal{V}_n(y_n), \\
 y_m &\in \mathcal{Y}_m, \forall m \in R(n). \\
 y_m &= f_m(y_n, v_n), \forall m \in R(n),
 \end{aligned} \tag{5.3}$$

The minimax structure of the above formulation hints at the main theme of this chapter. We will now focus our attention on existing methods which solve problems of the form of (5.3).

### 5.1.2 Average-cut reweighting

Current methods of solving the formulations above are based upon slight differences in the algorithm for traditional sddp type algorithm. In this section, we outline the work of Philpott and De Matos (2012), and show how their algorithm can solve a problem.

In order to provide the proper context for their algorithm, we will first outline the *risk neutral* version of their algorithm. This algorithm is very similar to Algorithm 4.2, however rather than forming bounds on individual bounding functions throughout the algorithm, bounds are instead formed on the *average* value/cost-to-go functions. We refer to this as the average-cut algorithm, and its description is provided below. We remind the reader here that, as we are now maximising concave rewards, value functions are now concave and we form *upper* bounds on the value functions using linear cuts.

Algorithm 5.1 (An average-cut SDDP algorithm).

**Initialisation.** Define  $\mathcal{V}_n^0 := \infty$ ,  $\forall n \in \mathcal{N} \setminus \mathcal{L}$ . Set the initial state  $y_0^k = y_0$ ,  $\forall k$ .  
Finally set  $k = 1$ .

**Iteration  $k$ .**

**Step 1.** Randomly sample a path  $\omega^k(\cdot)$  *independently* from previous sample paths.

**Step 2.** Set  $n = n_0$  and  $t = 0$ .

**Step 3.** Solve

$$\begin{aligned} \max_{v_n} \quad & C_n(y_n^k, v_n) + \mathcal{V}_n^{k-1}(y_n^k, v_n) \\ \text{s.t.} \quad & y_m = f_m(y_n^k, v_n), \forall m \in R(n), \\ & y_m \in \mathcal{Y}_m, \forall m \in R(n), \\ & v_n \in \mathcal{V}_n(y_n^k). \end{aligned} \tag{5.4}$$

storing  $(y_m^k, v_n^k)$  as the maximisers.

**Step 4.** For all  $m \in R(n)$ , if  $m \in \mathcal{L}$  set

$$\bar{\theta}_m^k = V_m(y_m^k), \tag{5.5}$$

and compute a subgradient  $\gamma_m^k$  of  $V_m(y_m^k)$  with respect to  $y_m^k$ . If  $m \notin \mathcal{L}$ , then solve

$$\begin{aligned} \bar{\theta}_m^k = \max_{v_m} \quad & C_m(y_m^k, v_m) + \mathcal{V}_m^{k-1}(y_m^k, v_m) \\ \text{s.t.} \quad & y_{m'} = f_{m'}(y_m^k, v_m), \forall m' \in R(m), \\ & y_{m'} \in \mathcal{Y}_{m'}, \forall m' \in R(m), \\ & v_m \in \mathcal{V}_m(y_m), \end{aligned} \tag{5.6}$$

and compute a subgradient  $\gamma_m^k$  of (5.9) with respect to  $y_m^k$ .

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

$$\mathcal{V}_n^k(y, v) := \min\{\mathcal{V}_n^{k-1}(y, v), \sum_{m \in R(n)} p(n, m) [\bar{\theta}_m^k + \langle \gamma_m^k, f_m(y, v) - f_m(y_n^k, v_n^k) \rangle]\}. \tag{5.7}$$

**Step 6.** Set  $t = t + 1$  and  $n = \omega^k(t)$ . If  $t = T$ , move on to iteration  $k + 1$ .

Otherwise, continue from **Step 3**.  $\bowtie$

It is a matter of taste as to whether one employs Algorithm 4.2 (multi-cut) or Algorithm 5.1 (average-cut). However, the average-cut version has an advantage in that it can easily be adapted to a risk averse version with one slight modification. Here we introduce the algorithm described in Philpott and De Matos (2012), we call this the reweighting algorithm.

Algorithm 5.2 (An average-cut reweighting SDDP algorithm).

**Initialisation.** Define  $\mathcal{V}_n^0 := \infty, \forall n \in \mathcal{N} \setminus \mathcal{L}$ . Set the initial state  $y_0^k = y_0, \forall k$ .

Finally set  $k = 1$ .

**Iteration  $k$ .**

**Step 1.** Randomly sample a path  $\omega^k(\cdot)$  *independently* from previous sample paths.

**Step 2.** Set  $n = n_0$  and  $t = 0$ .

**Step 3.** Solve

$$\begin{aligned} \max_{v_n} \quad & C_n(y_n^k, v_n) + \mathcal{V}_n^{k-1}(y_n^k, v_n) \\ \text{s.t.} \quad & y_m = f_m(y_n^k, v_n), \forall m \in R(n), \\ & y_m \in \mathcal{Y}_m, \forall m \in R(n), \\ & v_n \in \mathcal{V}_n(y_n^k). \end{aligned} \tag{5.8}$$

storing  $(y_m^k, v_n^k)$  as the maximisers.

**Step 4.** For all  $m \in R(n)$ , if  $m \in \mathcal{L}$  set

$$\bar{\theta}_m^k = V_m(y_m^k), \tag{5.9}$$

and compute a subgradient  $\gamma_m^k$  of  $V_m(y_m^k)$  with respect to  $y_m^k$ . If  $m \notin \mathcal{L}$ , then solve

$$\begin{aligned} \bar{\theta}_m^k = \max_{v_m} \quad & C_m(y_m^k, v_m) + \mathcal{V}_m^{k-1}(y_m^k, v_m) \\ \text{s.t.} \quad & y_{m'} = f_{m'}(y_m^k, v_m), \forall m' \in R(m), \\ & y_{m'} \in \mathcal{Y}_{m'}, \forall m' \in R(m), \\ & v_m \in \mathcal{V}_m(y_m), \end{aligned} \tag{5.10}$$

and compute a subgradient  $\gamma_m^k$  of (5.9) with respect to  $y_m^k$ .

**Step 5.** Compute risk-adjusted probabilities  $\dot{\mu}_m$  that solve

$$\min_{\mu_m \in Q_{v_n}} \sum_{m \in R(n)} \mu_m \times \bar{\theta}_m^k. \tag{5.11}$$

**Step 6.** Update the upper bound value function as

$$\mathcal{V}_n^k(y, v) := \min\{\mathcal{V}_n^{k-1}(y, v), \sum_{m \in R(n)} \dot{\mu}_m [\bar{\theta}_m^k + \langle \gamma_m^k, f_m(y, v) - f_m(y_n^k, v_n^k) \rangle]\}. \tag{5.12}$$



**Step 7.** Set  $t = t + 1$  and  $n = \omega^k(t)$ . If  $t = T$ , move on to iteration  $k + 1$ .  
Otherwise, continue from **Step 3**.  $\bowtie$

The main difference between the two algorithms presented above is the forming of the *average cut*. In Algorithm 5.1 the cut coefficients are weighted according to their conditional probability  $p(n, m)$ . This is in contrast to Algorithm 5.2 where instead the average cut is *risk adjusted*; the cut coefficients are weighted according to their value under the worst-case conditional probability measure in the conditional risk set (governed by  $Q_{v_n}$ ). We do not provide a convergence proof here, but note that this ought to converge owing to the fact that reweighing every cut, will be a lower bound on  $v_n[V_m(y, v)]_{m \in R(n)}$ , and at optimality, the cut will be exactly  $v_n[V_m(y, v)]_{m \in R(n)}$ .

### 5.1.3 The case for a saddle algorithm

It is noted that in Philpott and De Matos (2012) and Shapiro, Dentcheva, et al. (2009), that forming an upper bound (in the convex/min case, lower bound in the concave/max case) on the policy can be a difficult undertaking in the risk averse case, even a stochastic bound i.e. by Monte Carlo simulation. This makes it difficult to check convergence of the method, thereby forfeiting any guarantee on the quality of the policy, stochastic or otherwise. A respite against this has been in the work of Philpott, De Matos, and Finardi (2013) where they develop a method for determining an upper bound on the value of the risk adjusted cost-to-go functions. However, they note that their upper bound procedure can be weak.

Another downfall of the above is the fact that the conditional risk set  $Q_{v_n}$  must be fixed at all vertices in the tree. The formulation has no way of parameterising the risk set and forming lower bounding cuts conditional of some *state* which can parameterise the risk set in some way. As discussed in Chapter 3, these types of formulations are susceptible to our law invariance issue, meaning that there is no guarantee that policies computed with these objective function have a stochastic dominance property.

This motivates the following algorithm for solving multistage stochastic minimax problems, which the formulations discussed in this section can be cast in. The key difference with this algorithm is that cost-to-go functions keep both a minimising and maximising state.

## 5.2 An algorithm for minimax problems

This section outlines the preliminaries required for the outline of our minimax saddle optimisation algorithm. It is the intention of this section to outline a sufficiently general theory in order to capture the risk averse stochastic programming problems outlined in (5.1) and beyond. Although these risk averse optimisation problems provide us with motivation for our theory, we will postpone the discussion of the connection between the two until the next section.

### 5.2.1 Saddle functions

In this section, we discuss several preliminary concepts which are essential for the description of our algorithm. We remind the reader of a saddle function, saddle points, and discuss an extended Lipschitz-continuity concept on the Cartesian product of two vector spaces. We begin by reviewing definitions and notation of various elementary concepts in order to keep this work self-contained.

#### Definition 5.1.

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 5.2.

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

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

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 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 5.1.

If  $g : \mathcal{U} \times \mathcal{V} \mapsto \mathbb{R}$  is a saddle function, as given in Definition (5.1) and  $\mathcal{U}$  and

$\mathcal{V}$  are both convex and compact, then the set of saddle points is given by

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

*Proof.* From the minimax theorem of Von Neumann (1928), 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

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

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.  $\square$

Following from Lemma 5.1 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  $\epsilon$ -saddle point. These points extend the idea of a saddle point and will aid in describing the convergent properties of the proposed algorithm.

### Definition 5.3.

A point  $(u, v)$  is an  $\epsilon$ -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  $\epsilon$ -saddle points as

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

It immediately follows that  $\mathcal{G}(0) = \mathcal{G}$ ; that is, the most strict set of  $\epsilon$ -saddle points is the set of ‘true’ saddle points  $\mathcal{G}$ . As  $\epsilon$  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  $\mathcal{G}$ , it is easy to see that  $\mathcal{G}(\epsilon)$  is a convex set for all  $\epsilon \geq 0$ . Figure 5.1 provides a graphical representation of a saddle function and two different sets of  $\epsilon$ -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

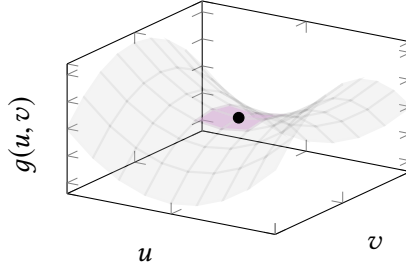


Figure 5.1: A saddle function  $g(u, v)$  with two sets of  $\epsilon$ -saddle points.

definition of the subdifferential set of a convex function  $f(u)$  is given by

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

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

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

This multifunction is a mapping from a point 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

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

If  $g(u, v)$  is differentiable everywhere, then we have that  $\partial^u g(u, v)$  is a singleton and  $\partial^u g(u, v) = D^u g(u, v)$ ; this being a standard notation for a partial derivative.

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 it in a natural way to allow for different Lipschitz constants in their respective vector spaces.

#### Definition 5.4.

Recall that a Lipschitz saddle function (under the  $p$ -norm) is a saddle function

$g(u, v)$  such that there exists some  $\alpha \in \mathbb{R}$  for which

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

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

**Definition 5.5.**

A  $(\alpha_u, \alpha_v)$ -Lipschitz saddle function (under the  $p$ -norm) is a saddle function  $g(u, v)$  for which there exists some  $\alpha_u, \alpha_v \in \mathbb{R}$  where

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

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

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

**Remark 5.1.**

We remark here that it can be shown that these two Lipschitz properties are equivalent; that is, a saddle function satisfying Definition 5.4 also satisfies Definition 5.5 (albeit for different values of their associated constants) and vice-versa. Nevertheless, it is useful to make a distinction between the particular constants  $\alpha$ ,  $\alpha_u$  and  $\alpha_v$  relating to their corresponding vector spaces.

### 5.2.2 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 in Chapter 4. In order to motivate this section, we present the following saddle point optimisation problem where the bounding functions we will develop become useful: for a given saddle function  $g(u, v)$  and tolerance  $\epsilon$ , we wish to compute a point  $(\hat{u}, \hat{v})$  such that

$$(\hat{u}, \hat{v}) \in \mathcal{G}_g(\epsilon) \subseteq \mathcal{U} \times \mathcal{V}, \quad (5.13)$$

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  $\epsilon$ -saddle points for the saddle function  $g(u, v)$ . Further, we require that the

saddle function is Lipschitz-continuous on  $\mathcal{U} \times \mathcal{V}$ . 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 5.6.**

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

$$\begin{aligned} \bar{G}^S(u, v) = \max_{\mu, \lambda} \quad & \mu + \langle \lambda, u \rangle \\ \text{s.t.} \quad & \mu + \langle \lambda, u_s \rangle \leq g(u_s, v_s) + \langle d_s^v, v - v_s \rangle, \quad \forall s \in S, \\ & \|\lambda\|_* \leq \alpha_u. \end{aligned} \quad (5.14)$$

Let us first develop an intuition of the upper bound function  $\bar{G}^S$  before laying out its properties. Consider Definition 5.6 when there is no  $v$  dependence. Then problem (5.14) reduces to (4.20). Alternatively, when the  $u$  dependence is dropped, we obtain the concave version of the epigraph formulation  $\bar{W}$ , given in the beginning of Section (4.4.1), which is an upper bound, in this case. 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 programme results,  $\bar{G}^S(u, v)$  is a piecewise bilinear function of  $u$  and  $v$ . We formalise this in the following lemma.

**Lemma 5.2.**

Consider the function  $\bar{G}^S(u, v)$  where the norm considered is the  $\|\cdot\|_\infty$ -norm. Further, assume that the resulting 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  $u$ .

*Proof.* It is well known that, in any parametric linear programme 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  $i \in \{1, \dots, I\}$ , 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 Martin [1975] and references therein, particularly Dinkelbach [1969]). Let us denote the basis corresponding to critical region  $i$  as  $B_i$ , for  $i \in \{1, \dots, I\}$ , 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 programme's optimal objective as function of its right-hand side; a well known result utilised widely in stochastic programming (see e.g. Birge and F. Louveaux [2011]) and the concavity in  $u$  is a simple analogue.  $\square$

Further intuition of the upper bound function in the univariate case can be obtained by studying Figure 5.2. We proceed to prove several properties of  $\bar{G}^S$ , which are utilised in the convergence proofs for the proposed saddle point algorithm. This collection of lemmas are similar to those laid out in Chapter 4.

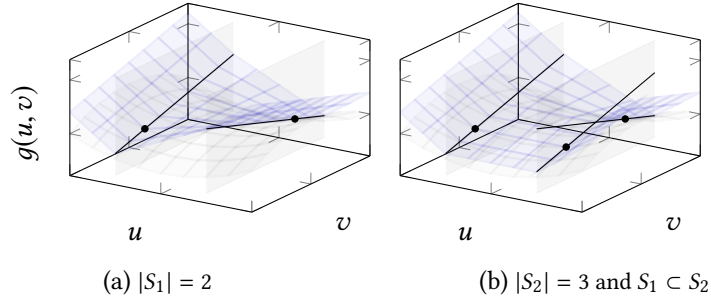


Figure 5.2: For the saddle function  $g(u, v)$  in grey, the blue surfaces are the upper bounding function  $\bar{G}(u, v)$ .

**Lemma 5.3.**

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

*Proof.* The point in  $(\mu, \lambda)$  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  $\lambda$  is bounded above and below,

while  $\mu$  is bounded from above in a maximisation problem, so clearly the optimisation problem defining  $\bar{G}^S(u, v)$  is bounded.  $\square$

**Lemma 5.4.**

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

*Proof.* For all  $v \in \mathcal{V}$ , the function  $\bar{G}^S(u, v)$  is clearly  $\alpha_u$ -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  $\alpha_v$ -Lipschitz, so its subgradients are uniformly bounded (under the  $\|\cdot\|_q$ -norm, where  $\frac{1}{p} + \frac{1}{q} = 1, \forall p \in (1, \infty)$ ). So  $\|d^v(u, v)\|_q \leq \alpha_v, \forall (u, v) \in \mathcal{U} \times \mathcal{V}$ . This gives the result.  $\square$

**Lemma 5.5.**

If  $g(u, v)$  is a saddle-function and  $(\alpha_u, \alpha_v)$ -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  $(\mu, \lambda)$  in the feasible region of (5.14) represent the coefficients of a supporting hyperplane of the points  $(u_s, g(u_s, v_s) + \langle d_s^v, v' - v_s \rangle), \forall s \in S$  where  $\mu$  is the intercept and  $\lambda$  is the gradient of the hyperplane. Recall from Lemma 5.3 that the feasible region is always non-empty. Because the function  $g(u, v)$  is convex in  $u$  and  $(\alpha_u, \alpha_v)$ -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  $(\mu, \lambda)$  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  $(\mu, \lambda)$  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.  $\square$

Lemma 5.6.

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.  $\square$

Lemma 5.7.

If  $g(u, v)$  is a saddle function and  $\alpha$ -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 5.5, 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 5.6, completing the proof.  $\square$

Remark 5.2.

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 4.2) as a special case of a saddle function upper bound i.e. with no  $v$  dependence, we can see that  $\bar{C}(u)$  does indeed have the properties claimed previously.

With these bounding functions introduced and their properties understood, we are now in a position to solve the motivating problem of this section. Before we outline our procedure, we establish a pair of useful bounding results with respect to the bounding functions' saddle points.

Lemma 5.8.

Under the conditions of Definition 5.6, we have

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

*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 5.7 completing the proof.  $\square$

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

Corollary 5.1.

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

The above corollary 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 5.8. The procedure we present below can be considered the saddle point analogue of Kelly's Cutting Plane algorithm; although novel, it is not the main result of this Chapter, 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 (5.13) on  $g(u, v)$  and  $\mathcal{U}$  and  $\mathcal{V}$  for the following theorem.

Theorem 5.1.

Consider the sequences

$$v^k = \arg \max_{v \in \mathcal{V}} \min_{u \in \mathcal{U}} \bar{G}^{S_{k-1}}(u, v), \quad (5.15)$$

$$u^k = \arg \min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \underline{G}^{S_{k-1}}(u, v), \quad (5.16)$$

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

with

$$\bar{g}^k = \min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \bar{G}^{S_{k-1}}(u, v), \quad \underline{g}^k = \min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \underline{G}^{S_{k-1}}(u, v). \quad (5.17)$$

We have that

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

To ease our notational burden, we will use  $\bar{G}^k(u, v)$  as a shorthand for  $\bar{G}^{S_k}(u, v)$ ; the same apply to the lower bound function  $\underline{G}$ . We proceed with the proof of Theorem 5.1.

*Proof.* From (5.15) and (5.17), we have

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

For  $\underline{g}^k$ , from (5.16) and (5.17), we have

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

It follows then that

$$\bar{g}^k - \underline{g}^k \leq \bar{G}^{k-1}(u^k, v^k) - \underline{G}^{k-1}(u^k, v^k), \forall k. \quad (5.18)$$

We will now show that the right-hand side of (5.18) converges to 0 as  $k$  tends to infinity – this will complete the proof since Corollary 5.1 bounds the difference of the saddle point values of these functions from below at 0. 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

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

From Lemma 5.1, there exists a constant  $\alpha$  for which both  $\underline{G}^k$  and  $\bar{G}^k$  are  $\alpha$ -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 5.5,  $\bar{G}^{k-1}(u^{\hat{k}}, v^{\hat{k}}) - \underline{G}^{k-1}(u^{\hat{k}}, v^{\hat{k}}) = 0$ ,  $\forall \hat{k} < k$ , since  $(u^{\hat{k}}, v^{\hat{k}})$  must be included in  $S^{k-1}$  for  $\hat{k} < k$ . 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 it must contain a convergent subsequence. It must follow then, that no  $\epsilon > 0$  exists and  $\bar{g}^k - \underline{g}^k \rightarrow 0$  as  $k \rightarrow \infty$ .  $\square$

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 given in Definition 5.2. Convergence of value 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 5.3. The following two lemmas give the desired result.

**Lemma 5.9.**

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 5.3, we have

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

where  $(\dot{u}, \dot{v})$  is a saddle point of  $g(\cdot, \cdot)$ . 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.  $\square$

**Lemma 5.10.**

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

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

*Proof.* From Lemma 5.9, for a point  $(u, v)$  to be a member of  $\mathcal{G}_g(\epsilon)$ , for some  $\epsilon \geq 0$ , we require that  $\epsilon$  satisfies

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

From Corollary 5.1, we have that

$$\bar{g}^k = \min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \bar{G}^{S_{k-1}}(u, v) \geq g_v(u'), \quad \forall u' \in \arg \min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \bar{G}^{k-1}(u, v),$$

and

$$\underline{g}^k = \min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \underline{G}^{S_{k-1}}(u, v) \leq g_u(v'), \quad \forall v' \in \arg \max_{v \in \mathcal{V}} \min_{u \in \mathcal{U}} \underline{G}^{k-1}(u, v).$$

So by taking the difference of these inequalities, we obtain

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

completing the proof. Further, from Lemma 5.1, we have that  $\bar{g}^k - \underline{g}^k$  approaches 0, so  $\mathcal{G}_g^k$  approaches the set of true saddle points.  $\square$

In review of this section, we have developed convex programming formulations for a pair of bounding functions that utilise zeroth and first order information. We have proved several properties of these functions and shown how these properties can be leveraged in an algorithm for computing an  $\epsilon$ -saddle point of a saddle function.

### 5.2.3 The stochastic setting

With the intuition of the saddle bounding functions developed in the previous section, we now turn our attention to the multistage stochastic case. In exactly the same fashion as the purely convex case, we leverage our scenario tree notation to form the dynamic programming equations as

$$\begin{aligned} G_n(x_n, y_n) &= \min_{x_m, u_n} \max_{y_m, v_n} C_n(x_n, y_n, u_n, v_n) + \sum_{m \in R(n)} p(n, m) G_m(x_m, y_m) \\ \text{s.t. } x_m &= f_m^x(x_n, u_n), \quad \forall m \in R(n), \\ y_m &= f_m^y(y_n, v_n), \quad \forall m \in R(n), \\ (x_m, y_m) &\in \mathcal{X}_m \times \mathcal{Y}_m, \quad \forall m \in R(n), \\ (u_n, v_n) &\in \mathcal{U}_n(x_n) \times \mathcal{V}_n(y_n), \end{aligned} \tag{5.19}$$

for all  $n \in \mathcal{N} \setminus \mathcal{L}$ , where  $R(n)$  is the set of immediate children of vertex  $n$ , and the final value function  $G_n(x_n, y_n)$ ,  $\forall n \in \mathcal{L}$ , is given. We require the following technical conditions in order for our algorithm to converge:

1. for all  $n \in \mathcal{N}$ ,  $\mathcal{X}_n$  and  $\mathcal{Y}_n$  are non-empty subsets of  $\mathbb{R}^{d_x}$  and  $\mathbb{R}^{d_y}$  respectively;
2. for all  $n \in \mathcal{N} \setminus \mathcal{L}$ , multifunctions  $\mathcal{U}_n(x)$  and  $\mathcal{V}_n(y)$  are convex and non-empty and compact valued;
3. the final value functions  $G_n(x_n, y_n)$ ,  $\forall n \in \mathcal{L}$  are saddle functions. The cost functions  $C_n(x, y, u, v)$ ,  $\forall n \in \mathcal{N} \setminus \mathcal{L}$  are convex in  $x$  and  $u$  and concave in  $y$  and  $v$ ;
4. for all  $n \in \mathcal{N} \setminus \mathcal{L}$ ,  $m \in R(n)$ , the functions  $f_m^x(x, u)$  and  $f_m^y(y, v)$  are affine in  $u, x$  and  $v, y$  respectively;
5. the final value functions  $G_n(x_n, y_n)$ ,  $\forall n \in \mathcal{L}$  are finite-valued and Lipschitz-continuous on  $\mathcal{X}_n \times \mathcal{Y}_n$ ;
6. for all  $n \in \mathcal{N} \setminus \mathcal{L}$ , there exists  $\delta_n^x > 0$  and  $\delta_n^y > 0$ , defining  $\mathcal{X}'_n := \mathcal{X}_n + B_n^x(\delta_n^x)$  and  $\mathcal{Y}'_n := \mathcal{Y}_n + B_n^y(\delta_n^y)$ , where

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

such that

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

$$f_m^x(x, \mathcal{U}_n(x)) \cap \mathcal{X}_m \text{ is non-empty, } \forall m \in R(n),$$

and

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

$$f_m^y(y, \mathcal{V}_n(y)) \cap \mathcal{Y}_m \text{ is non-empty, } \forall m \in R(n).$$

There is no particular motivation for the form of the conditions, only that they are as general as possible. Conditions 1-5 ensure that (5.19) 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. (2014), 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  $\delta^x$  and  $\delta^y$  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 a similar analysis can be performed to construct a particular  $(\alpha_u, \alpha_v)$  pair. Our eventual convergence 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 in Chapter 4.

We present our algorithm for solving minimax dynamic programmes of the above form. We encourage the reader to note the similarities between this algorithm and Algorithm 4.3 in Chapter 4.

Algorithm 5.3 (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  $\underline{G}_m^k(x, y) = G_m(x, y), \bar{G}_m^k(x, y) = G_m(x, y), \forall m \in \mathcal{L}, \forall k$ . 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

$$\begin{aligned} \underline{\theta}_n^k &= \min_{x_m, u_n} \max_{y_m, v_n} C_n(x_n^k, y_n^k, u_n, v_n) + \sum_{m \in R(n)} p(n, m) \underline{G}_m^{k-1}(x_m, y_m) \\ \text{s.t. } x_m &= f_m^x(x_n^k, u_n), \forall m \in R(n), \\ y_m &= f_m^y(y_n^k, v_n), \forall m \in R(n), \\ (x_m, y_m) &\in \mathcal{X}_m \times \mathcal{Y}_m, \forall m \in R(n), \\ (u_n, v_n) &\in \mathcal{U}_n(x_n^k) \times \mathcal{V}_n(y_n^k), \end{aligned} \tag{5.20}$$

storing  $u_n^k$  as the minimiser. Compute a subgradient  $\beta_n^k$  with respect to  $x_n^k$ .

**Step 3.** Solve

$$\begin{aligned}
 \bar{\theta}_n^k &= \max_{y_n, v_n} \min_{x_n, u_n} C_n(x_n^k, y_n^k, u_n, v_n) + \sum_{m \in R(n)} p(n, m) \bar{G}_m^{k-1}(x_m, y_m) \\
 \text{s.t. } x_m &= f_m^x(x_n^k, u_n), \quad \forall m \in R(n), \\
 y_m &= f_m^y(y_n^k, v_n), \quad \forall m \in R(n), \\
 (x_m, y_m) &\in \mathcal{X}_m \times \mathcal{Y}_m, \quad \forall m \in R(n), \\
 (u_n, v_n) &\in \mathcal{U}_n(x_n^k) \times \mathcal{V}_n(y_n^k),
 \end{aligned} \tag{5.21}$$

storing  $v_n^k$  as the maximiser. Compute a subgradient  $\gamma_n^k$  with respect to  $y_n^k$ .

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

$$\begin{aligned}
 \underline{G}_n^k(x, y) &= \min_{\mu, \lambda} \mu + \langle \lambda, y \rangle \\
 \text{s.t. } \mu + \langle \lambda, y_n^{\hat{k}} \rangle &\geq \underline{\theta}_n^{\hat{k}} + \langle \beta_n^{\hat{k}}, x - x_n^{\hat{k}} \rangle, \quad \forall \hat{k} \leq k, \\
 \|\lambda\|_* &\leq \alpha_y.
 \end{aligned} \tag{5.22}$$

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

$$\begin{aligned}
 \bar{G}_n^k(x, y) &= \max_{\mu, \lambda} \mu + \langle \lambda, x \rangle \\
 \text{s.t. } \mu + \langle \lambda, x_n^{\hat{k}} \rangle &\leq \bar{\theta}_n^{\hat{k}} + \langle \gamma_n^{\hat{k}}, y - y_n^{\hat{k}} \rangle, \quad \forall \hat{k} \leq k, \\
 \|\lambda\|_* &\leq \alpha_x.
 \end{aligned} \tag{5.23}$$

**Step 6.** Update the problem-child  $\Phi_n^k$  as

$$\begin{aligned}
 \Phi_n^k &= \arg \max_{m \in R(n)} p(n, m) (\bar{G}_m^{k-1}(f_m^x(x_n^k, u_n^k), f_m^y(y_n^k, v_n^k)) \\
 &\quad - \underline{G}_m^{k-1}(f_m^x(x_n^k, u_n^k), f_m^y(y_n^k, v_n^k))). \tag{5.24}
 \end{aligned}$$

If  $\Phi_n^k$  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.  $(x_n^{k+1}, y_n^{k+1}) = (x_n^k, y_n^k)$ .  $\bowtie$



**Remark 5.3.**

We remark that in the special case where the cost functions are solely convex (concave) and only a minimisation (maximisation) is considered, Algorithm 5.3 provides a deterministic guarantee of convergence for these problems also. For the convex case, our formulation (5.19) is identical to that presented in Chapter 4.

Before the statement and proof of our convergence theorem, we first develop a crucial inequality generated by Algorithm 5.3 that will be leveraged by our subsequent convergence proof.

**Lemma 5.11.**

Consider the sequences of functions  $\bar{G}_n^k$  and  $\underline{G}_n^k$ , and state trajectories  $(x_n^k, y_n^k)$  generated by the algorithm. For all  $t = 0, \dots, T - 1$ , we have

$$\begin{aligned} \bar{G}_n^k(x_n^k, y_n^k) &\leq C_n(x_n^k, y_n^k, u_n, v_n^k) \\ &\quad + \sum_{m \in R(n)} p(n, m) \bar{G}_m^{k-1}(f_m^x(x_n^k, u_n), y_m^k), \quad \forall u_n \in \mathcal{U}_n(x_n^k). \end{aligned}$$

*Proof.* From Step 5 in Algorithm 5.3 we have

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

So  $\bar{G}_n^k(x_n^k, y_n^k) \leq \bar{\theta}_n^k$ . From Step 3, we have

$$\begin{aligned} \bar{\theta}_n^k &= \min_{u_n \in \tilde{\mathcal{U}}(x_n^k)} \max_{v_n \in \tilde{\mathcal{V}}(y_n^k)} C_n(x_n^k, y_n^k, u_n, v_n) + \bar{G}_{t+1}^{k-1}(f_m^x(x_n^k, u_n), f_m^y(y_n^k, v_n)), \\ &= \min_{u_n \in \tilde{\mathcal{U}}(x_n^k)} C_n(x_n^k, y_n^k, u_n, v_n^k) + \sum_{m \in R(n)} \bar{G}_m^{k-1}(f_m^x(x_n^k, u_n), y_m^k), \\ &\leq C_n(x_n^k, y_n^k, u_n, v_n^k) + \sum_{m \in R(n)} \bar{G}_m^{k-1}(f_m^x(x_n^k, u_n), y_m^k), \quad \forall u_n \in \mathcal{U}_n(x_n^k). \end{aligned}$$

This concludes the proof.  $\square$

We now state and prove our deterministic convergence theorem for Algorithm 5.3

**Theorem 5.2.**

Consider the sequence of functions  $\bar{G}_n^k$  and  $G_n^k$ , and state and control trajectories  $((x_n^k, y_n^k), (u_n^k, v_n^k))$ , and problem-children  $\Phi_n^k$  generated by Algorithm 5.3. For all  $n \in \mathcal{N}$ , we have

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

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

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

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

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

assuming the induction hypothesis. By Lemma 5.11 we have

$$\begin{aligned} \bar{G}_n^k(x_n^k, y_n^k) &\leq C_n(x_n^k, y_n^k, u_n^k, v_n^k) \\ &+ \sum_{m \in R(n)} p(n, m) \bar{G}_m^{k-1}(f_m^x(x_n^k, u_n^k), f_m^y(y_n^k, v_n^k)), \quad \forall n \in \mathcal{N} \setminus \mathcal{L}, \quad \forall k, \end{aligned}$$

and symmetrically

$$\begin{aligned} G_n^k(x_n^k, y_n^k) &\geq C_n(x_n^k, y_n^k, u_n^k, v_n^k) \\ &+ \sum_{m \in R(n)} p(n, m) G_m^{k-1}(f_m^x(x_n^k, u_n^k), f_m^y(y_n^k, v_n^k)), \quad \forall n \in \mathcal{N} \setminus \mathcal{L}, \quad \forall k. \end{aligned}$$

Subtracting these, we have

$$\begin{aligned} \bar{G}_n^k(x_n^k, y_n^k) - G_n^k(x_n^k, y_n^k) &\leq \\ &\sum_{m \in R(n)} p(n, m) [\bar{G}_m^{k-1}(f_m^x(x_n^k, u_n^k), f_m^y(y_n^k, v_n^k)) \\ &\quad - G_m^{k-1}(f_m^x(x_n^k, u_n^k), f_m^y(y_n^k, v_n^k))], \quad \forall n \in \mathcal{N} \setminus \mathcal{L}, \quad \forall k. \end{aligned}$$

But by the problem-child selection principle (Step 6 of Algorithm 5.3), we must have

$$\begin{aligned} \bar{G}_n^k(x_n^k, y_n^k) - \underline{G}_n^k(x_n^k, y_n^k) \leq \\ |R(n)|p(n, \Phi_n^k) [\bar{G}_{\Phi_n^k}^{k-1}(x_{\Phi_n^k}^k, y_{\Phi_n^k}^k) - \underline{G}_{\Phi_n^k}^{k-1}(x_{\Phi_n^k}^k, y_{\Phi_n^k}^k)], \quad \forall n \in \mathcal{N} \setminus \mathcal{L}, \quad \forall k. \end{aligned}$$

Similar to the proof of Theorem 5.2, we invoke Lemma 4.1 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  $\Phi_n^k$ . This proves the induction hypothesis. Coupled with our base case, this concludes the proof.  $\square$

The convergence proof for Algorithm 5.3 and indeed the algorithm itself follow a very similar format to Algorithm 4.3 and Theorem 5.2; both algorithms leverage the *problem children*  $\Phi_n^k$  in order to direct the algorithm through the sampling tree. Both algorithms are essentially the same in philosophy; they iteratively refine bounding functions in ‘useful’ places. The theory in this section was dedicated to developing saddle bounding functions with several favourable properties (i.e. Lemmas 5.3 to 5.7) and an algorithm which refines these saddle bounding functions in the ‘useful’ places.

We wish to highlight here again that, although the initial motivation for the theory developed in this section was the solution of risk averse multistage stochastic optimisation problems, we have opted to develop the theory for an appropriately general class of formulations. Indeed this theory has already found usage in other areas of stochastic programming; in Downward et al. (2018), the authors leverage the saddle function approximations to bound cost-to-go functions which are saddle, but are not ‘minimax-ed’ over.

We expect these results to extend into the field of multistage robust optimisation. For instance, in the work of Iyengar (2005), the author notes that:

“when the set of measures associated with a policy satisfy a certain ‘rectangularity’ property, the following results extend to natural robust counterparts: the Bellman recursion, the optimality of deterministic policies, the contraction property of the value iteration operator, and the policy iteration algorithm. Rectangularity is a sort

of independence assumptions and is a minimal requirement for these results to hold."

The author notes (at the time of his work), that solving non-rectangular minimax formulations in the context of robust optimisation is an open problem. Budget constrained optimisation is an example of such formulations: here the 'minimiser', perhaps nature, has a budget over how to spoil the rewards of the 'maximiser'. This budget is carried across the stages of the optimisation problem, forming a non-rectangular uncertainty set. In the next section, we explore some of these ideas; in particular, focussing on risk averse formulations.

### 5.3 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 (5.19). 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. We choose to focus here on the CVaR risk measure here; as discussed in Chapter 2, this class of risk measure forms a 'basis' in the space of all coherent and law invariant risk measures. Recall from Chapter 2 that when considering a finite probability space  $(\Omega, \mathcal{F}, \mathbb{P})$  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

$$\begin{aligned} \text{CVaR}_\beta(Z(\omega)) &= \min_{\eta(\omega)} \sum_{\omega \in \Omega} \eta(\omega) \mathbb{P}(\omega) Z(\omega) \\ \text{s.t.} \quad &0 \leq \eta(\omega) \leq \frac{1}{\beta}, \forall \omega \in \Omega, \\ &\sum_{\omega \in \Omega} \eta(\omega) \mathbb{P}(\omega) = 1. \end{aligned} \tag{5.25}$$

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  $\mathbb{E}_{\mathbb{Q}}[Z(\omega)] = \text{CVaR}(Z(\omega))$ , where  $\dot{\eta}(\omega)$  is the optimal  $\eta(\omega)$ . 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:

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

If  $Z(v, \omega)$  is concave in  $v, \forall \omega \in \Omega$ , then the objective is a saddle function, since the entire objective is concave in  $v$  and convex (and linear) in  $\eta(\omega)$ . Consequently this formulation complies with our general minimax formulation presented in (4.13); our control in the maximisation vector space, is the vector  $v$ , while our control in the minimisation vector space is the vector of *probability scalars*  $\eta(\omega)$ . 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 2012). Secondly, we present a novel dynamic programming formulation which computes an optimal policy where risk is single CVaR evaluation of accumulated profits over the entire scenario tree; we call this *end-of-horizon* risk. This second formulation relies on the parametric conditional risk set theory developed in Chapter 3.

### 5.3.1 Rectangular risk

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

$$\begin{aligned} V_n(y_n) &= \max_{y_m, v_n} C_n(y_n, v_n) + \text{CVaR}_{\beta_n} [V_m(y_m)]_{m \in R(n)} \\ \text{s.t. } y_m &= f_m(y_n, v_n), \forall m \in R(n), \\ y_m &\in \mathcal{Y}_m, \forall m \in R(n), \\ v_n &\in \mathcal{V}_n(y_n). \end{aligned} \tag{5.26}$$

Here,  $\beta_n \in (0, 1]$  are predefined quantiles; one for each parent vertex in the tree. From now on, we will drop the  $v_n[\cdot]_{m \in R(n)}$  notation for  $v_n[\cdot]$ , as the context will be

clear. These formulations are termed rectangular because the risk sets generated by the conditional CVaR measures are independent of each other. This key observation is critical for the algorithm presented in Philpott and De Matos (2012) to converge. For a notational convenience, we will forgo the dependence of our formulation on the variables concerned with maximisation (i.e.  $y$  and  $v$ ) throughout the remainder of the chapter. This allows us to then write the above as

$$V_n = C_n + \text{CVaR}_{\beta_n}^+[V_m].$$

Figure 5.3 gives a diagrammatic representation of how such an objective function might be visualised; here the nested structure of formulation is made apparent.

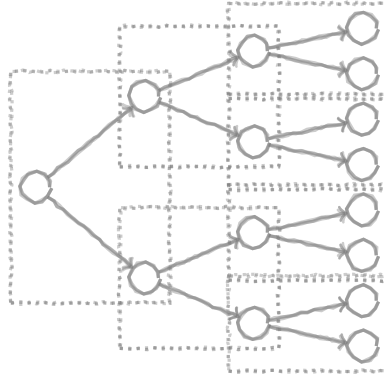


Figure 5.3: A nested risk measure on a scenario tree.

It is our goal now to cast this formulation into a form that complies with (5.19). Unfortunately, we cannot directly invoke the dual representation of  $\text{CVaR}^+$  to produce dynamic programming equations as in

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

our general problem formulation (i.e. in (5.19)) requires that we take the conditional expectation of the children value-to-go functions, rather than their risk adjustment. To circumvent this, we move the ‘risk adjustment’ to the cost functions of the children vertices; we can write dynamic programming equations that conform to our minimax

problem form as

$$\begin{aligned}
 G_n^N(\eta_n) = \min_{\eta_m} \quad & C_n \eta_n + \sum_{m \in R(n)} p(n, m) [G_m^N(\eta_m)] \\
 \text{s.t.} \quad & 0 \leq \eta_m \leq \frac{\eta_n}{\beta_n}, \quad \forall m \in R(n), \quad (5.27.1) \\
 & \sum_{m \in R(n)} p(n, m) \eta_m = \eta_n.
 \end{aligned} \tag{5.27}$$

Here,  $\eta_m$ , the probability scalars, are considered to be both the control and the states; the value of a particular  $\eta_n$  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  $\eta_n$  to  $\eta(\omega)$  in (5.25). The following lemma will be useful for our theorem of equivalence between (5.26) and (5.27).

Lemma 5.12.

Under the conditions required for (5.25), for  $\eta_0 \geq 0$ , we have

$$\begin{aligned}
 \text{CVaR}_\beta(Z(\omega)) \times \eta_0 = \min_{\eta(\omega)} \quad & \sum_{\omega \in \Omega} \eta(\omega) \mathbb{P}(\omega) Z(\omega) \\
 \text{s.t.} \quad & 0 \leq \eta(\omega) \leq \frac{\eta_0}{\beta}, \quad \forall \omega \in \Omega, \\
 & \sum_{\omega \in \Omega} \eta(\omega) \mathbb{P}(\omega) = \eta_0.
 \end{aligned} \tag{5.28}$$

*Proof.* The proof is simple when observing a re-scaling of  $\eta(\omega)$  in (5.28) as  $\eta(\omega)/\eta_0$ . For the case where  $\eta_0 = 0$ , the equality holds trivially.  $\square$

The following theorem shows that the dynamic programming formulation above coincides with the solution to (5.26).

Theorem 5.3.

Consider the two formulations given in (5.26) and (5.27). We have

$$G_0^N(1) = V_0.$$

*Proof.* For all  $n \in P(m)$ ,  $\forall m \in \mathcal{L}$ , i.e. the parents of leaf vertices, we have

$$\begin{aligned} G_n^N(\eta_n) &= \min_{\eta_m} C_n \eta_n + \sum_{m \in R(n)} p(n, m) [C_m \eta_m] \\ \text{s.t. } &0 \leq \eta_m \leq \frac{\eta_n}{\beta_n}, \forall m \in R(n), \\ &\sum_{m \in R(n)} p(n, m) \eta_m = \eta_n, \forall m \in R(n), \end{aligned}$$

which by Lemma 5.12 gives

$$G_n^N(\eta_n) = C_n \eta_n + \eta_n \times \text{CVaR}_{\beta_n}[C_m] = \eta_n [C_n + \text{CVaR}_{\beta_n}[C_m]].$$

So  $\forall n \in P(m)$ ,  $\forall m \in P(\dot{m})$ ,  $\forall \dot{m} \in \mathcal{L}$ , i.e. the parents of the parents of leaf vertices, we have

$$\begin{aligned} G_n^N(\eta_n) &= \min_{\eta_m} C_n \eta_n + \sum_{m \in R(n)} p(n, m) \eta_n [C_n + \text{CVaR}_{\beta_n}[C_m]] \\ \text{s.t. } &0 \leq \eta_m \leq \frac{\eta_n}{\beta_n}, \forall m \in R(n), \\ &\sum_{m \in R(n)} p(n, m) \eta_m = \eta_n, \forall m \in R(n), \end{aligned}$$

which, again, by Lemma 5.12 gives

$$\begin{aligned} G_n^N(\eta_n) &= C_n \eta_n + \eta_n \times \text{CVaR}_{\beta_n}[C_m + \text{CVaR}_{\beta_m}[C_{\dot{m}}]] \\ &= \eta_n \left[ C_n + \text{CVaR}_{\beta_n}[C_m + \text{CVaR}_{\beta_m}[C_{\dot{m}}]] \right]. \end{aligned}$$

By the recursive application of the above, we obtain

$$G_0^N(1) = C_0 + \text{CVaR}_{\beta_0}[C_m + \text{CVaR}_{\beta_m}[C_{\dot{m}} + \dots]],$$

giving the result.  $\square$

This result outlines how one can form the risk averse multistage stochastic formulations considered in Philpott and De Matos (2012) in a form that complies with our minimax formulation. In doing so, we can now achieve deterministic convergence for these optimisation problems.



### 5.3.2 End-of-horizon risk

In Chapter 3, we developed the theoretical basis for multistage risk measures that were *parametric*; these risk measures have the feature that their composition need not fail to have the stochastic dominance property. In this section, we develop a parameterised decomposition whose composition yields CVaR over the whole probability space i.e.

$$v_0^T = v_0^1 \circ \dots \circ v_{T-1}^T = \text{CVaR}_{\beta_0}^+. \quad (5.29)$$

Lemma 3.2 in Chapter 3 gives us a theoretical result that such decompositions exists, and that the parameterised conditional risk sets are *convex multifunctions*. This is significant here, because our algorithm requires convexity in both the maximising and minimising control sets with respect to their respective states. Without this condition, forming such decompositions would be an futile endeavour, as they could not be solved using our minimax dynamic programming framework.

In contrast to Figure 5.3, Figure 5.4 shows a diagrammatic representation of such an end-of-horizon risk measure.

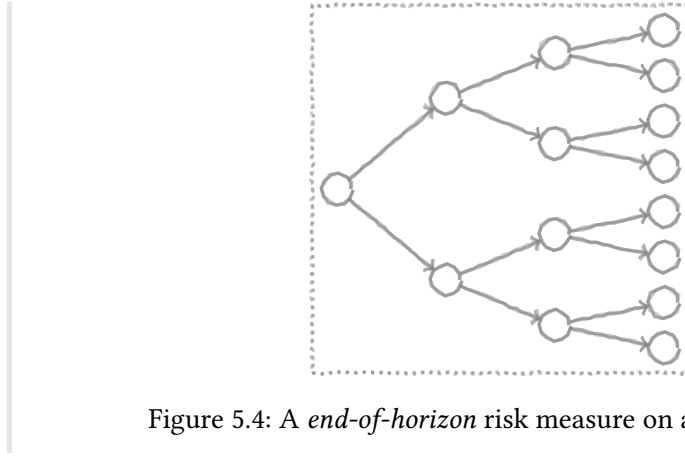


Figure 5.4: A *end-of-horizon* risk measure on a scenario tree.

In mathematical form, we seek

$$\text{CVaR}_{\beta_0}^+ \left[ \sum_{t=0, \dots, T} C_t(\omega) \right] \quad (5.30)$$

for some  $\beta_0 \in (0, 1]$ , and the rewards  $C_t$  are adapted to the filtration. In the above formulation,  $\omega \in \Omega$  is an element of the set of all paths in the scenario tree – cost are accumulated over each of the paths, and then CVaR evaluation is taken over these paths. In Pflug and Pichler (2016), 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 (5.30). Here we present, in our opinion, a more natural formulation for this problem, and one which complies with the structure of (5.19). The formulation below, along with Algorithm 5.3 allows for the tractable solution of problem (5.30); a major result of this thesis. The formulation is given by

$$\begin{aligned}
 G_n^E(\eta_n) &= \min_{\eta_m} \eta_n C_n + \sum_{m \in R(n)} p(n, m) [G_m^E(\eta_m)] \\
 \text{s.t.} \quad &0 \leq \eta_m \leq \frac{1}{\beta_0}, \forall m \in R(n), \quad (5.31.1) \\
 &\sum_{m \in R(n)} p(n, m) \eta_m = \eta_n,
 \end{aligned} \tag{5.31}$$

for all  $n \in \mathcal{N} \setminus \mathcal{L}$ . Once again, the probability scalers  $\eta_m$  have the same interpretation as (5.27), however, their feasible set differs slightly. Constraint (5.31.1) dictates that the probability scalers  $\eta_n$  at any vertex on the tree can not exceed the amount prescribed by the risk set determined at the the root vertex. We present the following theorem below, which shows that our dynamic programming formulation and (5.30) coincide.

**Theorem 5.4.**

Given the dynamic programming equations defined in (5.31), we have

$$G_0^E(1) = \text{CVaR}_{\beta_0} \left[ \sum_{t=0, \dots, T} C_t(\omega) \right].$$

*Proof.* For this proof, we will consider only a 3-stage instance of this problem; the intuition holds for any number of stages. By evaluating the recursion in

(5.31) and ‘merging on mins’, we obtain  $G_0^E(1) =$

$$\begin{aligned}
 \min_{\eta_m \in R(0)} \quad & C_0 + \sum_{m \in R(0)} p(0, m) \left[ \min_{\eta_{\dot{m}} \in R(m)} C_m \eta_m + \sum_{\dot{m} \in R(m)} p(m, \dot{m}) [C_{\dot{m}} \eta_{\dot{m}}] \right] \\
 \text{s.t.} \quad & 0 \leq \eta_m \leq \frac{1}{\beta_0}, \forall m \in R(0), \\
 & 0 \leq \eta_{\dot{m}} \leq \frac{1}{\beta_0}, \forall \dot{m} \in R(m), \forall m \in R(0), \\
 & \sum_{m \in R(0)} p(0, m) \eta_m = \eta_0, \forall m \in R(0), \quad (5.34.1) \\
 & \sum_{\dot{m} \in R(m)} p(m, \dot{m}) \eta_{\dot{m}} = \eta_m, \forall \dot{m} \in R(m), \forall m \in R(0). \quad (5.34.2)
 \end{aligned} \tag{5.32}$$

Note here that  $\eta_m$  is fully constrained. By substituting, (5.34.2) into (5.34.1), we obtain

$$\sum_{m \in R(0)} p(0, m) \sum_{\dot{m} \in R(m)} p(m, \dot{m}) \eta_{\dot{m}} = \eta_0, \forall m \in R(0), \forall \dot{m} \in R(m).$$

by definition, we have that  $p(m_1, m_2)p(m_2, m_3) = p(m_1, m_3)$ , so

$$\sum_{m \in R(0)} \sum_{\dot{m} \in R(m)} p(0, \dot{m}) \eta_{\dot{m}} = \eta_0, \forall m \in R(0), \forall \dot{m} \in R(m).$$

Similarly, substituting constraint (5.34.2) into the objective function of (5.34), we obtain

$$\begin{aligned}
 G_0^E(1) = \min_{\eta_m \in R(0)} \quad & C_0 + \sum_{m \in R(0)} p(0, m) \left[ \min_{\eta_{\dot{m}} \in R(m)} C_m \sum_{\dot{m} \in R(m)} p(m, \dot{m}) \eta_{\dot{m}} \right. \\
 & \left. + \sum_{\dot{m} \in R(m)} p(m, \dot{m}) [C_{\dot{m}} \eta_{\dot{m}}] \right]
 \end{aligned}$$

Simplifying yields

$$G_0^E(1) = \min_{\eta_{\dot{m}} \in R(R(0))} \sum_{m \in R(0)} \left[ \sum_{\dot{m} \in R(m)} p(0, \dot{m}) \eta_{\dot{m}} [C_0 + C_m + C_{\dot{m}}] \right].$$

We obtain an optimisation problem that reads as

$$\begin{aligned}
 G_0^E(1) = \min_{\eta_{\dot{m}} \in R(R(0))} \quad & \sum_{m \in R(0)} \left[ \sum_{\dot{m} \in R(m)} p(0, \dot{m}) \eta_{\dot{m}} [C_0 + C_m + C_{\dot{m}}] \right] \\
 \text{s.t.} \quad & 0 \leq \eta_{\dot{m}} \leq \frac{1}{\beta_0}, \quad \forall \dot{m} \in R(m), \forall m \in R(0), \\
 & \sum_{m \in R(0)} \sum_{\dot{m} \in R(m)} p(0, \dot{m}) \eta_{\dot{m}} = \eta_0, \quad \forall m \in R(0), \forall \dot{m} \in R(m).
 \end{aligned} \tag{5.33}$$

The summation

$$\sum_{m \in R(0)} \sum_{\dot{m} \in R(m)}$$

represents a sum over all vertices in a scenario tree; owing to equivalence between a scenario tree and its equivalent filtered probability space, we have

$$\begin{aligned}
 G_0^E(1) = \min_{\eta(\omega)} \quad & \mathbb{P}(\omega) \eta(\omega) [C_0(\omega) + C_1(\omega) + C_2(\omega)] \\
 \text{s.t.} \quad & 0 \leq \eta(\omega) \leq \frac{1}{\beta_0}, \quad \forall \omega \in \Omega, \\
 & \eta(\omega) \mathbb{P}(\omega) = 1, \quad \forall \omega \in \Omega,
 \end{aligned} \tag{5.34}$$

which gives our result. This analysis can be extended further to any number of stages.  $\square$

The key result from this theorem is that we can parameterise the conditional risk set at a given vertex in one state variable  $\eta_n$ . This result is surprising; In Chapter 3, we showed that it was possible that a conditional risk set parameterisation could be a function of the entire history of conditional probability selections. If this were the case, we could not hope to encode this in a reasonably sized state space. This result above shows that this is not case; we only require a single state variable to encode the conditional risk set at any one vertex in the scenario tree, making this approach very attractive.

### 5.3.3 Rectangularisation

Given the similar structure of the two minimax dynamic programming formulations presented earlier, a natural question to consider is the following: is it possible to compute a ‘rectangularisation’ – that is, to compute a set of CVaR quantiles  $\beta_n$  to ascribe to the rectangular formulation such that the optimal solution to  $G_0^E(1)$  is

also optimal for  $G_0^N(1)$ ? We present a heuristic proof of this conjecture for the case where  $\dot{\eta}_n > 0$ ,  $\forall n \in \mathcal{N}$ ; where we denote the optimal  $\eta_n$  of  $G_0^E(1)$  as  $\dot{\eta}_n$ . Note that the two dynamic programming formulations differ only in the right-hand side of constraint (5.27)1). By setting

$$\beta_n = \beta_0 \times \dot{\eta}_n$$

in  $G_0^N(1)$ , the right-hand side of constraint (5.27)1) becomes

$$\frac{\eta_n}{\beta_0 \times \dot{\eta}_n}.$$

At  $\eta_n = \dot{\eta}_n$ , we obtain

$$\frac{\dot{\eta}_n}{\beta_0 \times \dot{\eta}_n} = \frac{1}{\beta_0},$$

so clearly,  $\dot{\eta}_n$  is feasible for  $G_0^N(1)$ . We must now show that this point is optimal. Denote the optimal dual vector of constraint (5.31)1) as  $\pi_n$ . At  $\eta_n = \dot{\eta}_n$ , the dual vector  $\pi_n$  give stationarity for (5.27) (as the objective functions between (5.31) and (5.27) do not differ). Finally, dual feasibility (and complementary slackness) is ensured at  $\eta_n = \dot{\eta}_n$ , as the constraints sets between the (5.31) and (5.27) at this point are identical. So  $\dot{\eta}_n$  is a KKT point of (5.27), completing the proof.

This rectangularisation process can be considered the reverse of the algorithm presented in Asamov and Ruszczyński (2015). In their paper, they outline a procedure which iteratively estimates risk sets whose composition gives a risk set which contains the relevant probability distribution in the risk set corresponding to a end-of-horizon risk measure. Cast in the language of this paper, their algorithm exogenously adjusts  $\beta_n$  on the vertices of the tree in order to estimate a one-time risk evaluation, while our algorithm (combined with the end-of-horizon risk formulation) allows this process to occur endogenously.

### 5.3.4 Multi-quantile structure

Further to the discussion of representation of coherent risk measures, an important class of coherent risk measures are those called *spectral* risk measures. Developed in the work of Acerbi (2002), their main representation result is the following: any risk measure  $\nu(\cdot)$  for which

$$\nu(Z) = \int_0^1 \text{CVaR}_\beta(Z) d\beta(\omega)$$

where  $\beta$  is a probability measure on the closed unit interval, is said to be spectral. For the purposes of approximating these risk measures, here we consider a finite set of pairs  $(\theta_b, \beta^b) \in \mathcal{B}$ , which define an end-of-horizon spectral risk measure where

$$\sum_{b \in \mathcal{B}} \theta_b = 1, \text{ and } \theta_b \in [0, 1], \forall b \in \mathcal{B},$$

and  $\beta_0^b \in (0, 1], \forall b \in \mathcal{B}$ . Rather than seeking the solution to a single CVaR evaluation of accumulated profits (as in (5.30)), suppose we seek the solution to

$$\sum_{b \in \mathcal{B}} \theta_b \text{CVaR}_{\beta_0^b} \left[ \sum_{t=0, \dots, T} C_t(\omega) \right]$$

The full set of dynamic programming equations can be formulated as

$$\begin{aligned} \sum_{b \in \mathcal{B}} \theta_b G_n^E(\eta_n^b) &= \min_{\eta_m^b} \sum_{b \in \mathcal{B}} \theta_b \left[ \eta_n^b C_n + \sum_{m \in R(n)} p(n, m) G_m^E(\eta_m^b) \right] \\ \text{s.t. } 0 &\leq \eta_m^b \leq \frac{1}{\beta_0^b}, \forall m \in R(n), \forall b \in \mathcal{B}, \\ \sum_{m \in R(n)} \frac{p_m}{p_n} \eta_m^b &= \eta_n^b, \forall b \in \mathcal{B}. \end{aligned}$$

In light of the formulations presented in this section, we can adapt our algorithm outlined in this paper to exploit some of their structure. At the optimal solution of (5.3.4) we have

$$G_n^E(\eta_n^b) = \eta_n^b C_n + \sum_{m \in R(n)} p(n, m) G_m^E(\eta_m^b), \forall b \in \mathcal{B}.$$

When computing subgradients and the intercept for the cutting plane for  $\sum_{b \in \mathcal{B}} \theta_b G_n(\eta_n^b)$  during our algorithm, we can instead compute  $|\mathcal{B}|$  cutting planes, one for each term in the sum. The sum of the cuts will be the correct cut for (5.3.4), however the disaggregation of cuts allows for information to be shared between the quantiles, providing a dominant set of cuts. By taking advantage of the structure, we can essentially reduce the state space dimensionality from  $|\mathcal{B}|$  to unity, greatly increasing the computational effort required to solve the problem.

## 5.4 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 implementation of our algorithm written in Julia, which interfaces with Gurobi as a linear programme solver (see Dunning et al. [2015] and Gurobi Optimization [2016] 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 [5.2] 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. [5.20] and [6.2]).

In many instances, it may be difficult to determine an appropriate bound on the magnitude of the gradient  $\lambda$  inside our bounding function definitions (i.e. the Lipschitz constant). Indeed, a more accurate constant  $\alpha$  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 [5.22] and [5.23]. We proceed by introducing a portfolio optimisation problem to illustrate our algorithm and end-of-horizon risk concept.

#### 5.4.1 A portfolio optimisation problem

Consider a collection of  $S$  securities  $\mathcal{S} = \{s_1, s_2, \dots, s_S\}$ . A portfolio is a vector  $x \in \mathbb{R}^S$  which represents the value of each of the holdings. The control vector  $u \in \mathbb{R}^S$  represents the transactions made on the portfolio – these may have an associated immediate cost, often referred to as a transaction cost. The portfolio may be constrained in some way, such as a minimum cash holding requirement or a long-only portfolio. 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.

In our numerical demonstration, we have  $S = 3$ ,  $|R(n)| = 6$ , require a long-only portfolio, and investments made *into* the most risky investment  $s_3$  can only be withdrawn at maturity. The model has 12 stages, and the randomness is a stage-wise independent process (allowing the use of cut-sharing between the vertices).

Table 5.1: Multivariate normal distribution parameters.

$\mathcal{S}$	Monthly		Annual	
	$\mu$	$\sigma$	$\mu$	$\sigma$
$s_1$	0.00%	0	0.00%	0
$s_2$	0.247%	0.281%	3.00%	1.00%
$s_3$	0.327%	0.7%	4.00%	2.51%

The portfolio constraints at a given vertex  $n$  can be captured by

$$\begin{aligned}
y_{ms} &= \Psi_{ms} \times (y_{ns} + v_{ns}), \quad \forall m \in R(n), \quad \forall s \in \mathcal{S}, \\
\sum_{s \in \mathcal{S}} v_{ns} &= 0, \\
v_{ns_3} &\geq 0, \\
y_{ns} + v_{ns} &\geq 0, \quad \forall s \in \mathcal{S},
\end{aligned} \tag{5.35}$$

where  $\Psi_{ms}$  represents the fluctuation of the value of security  $s$  in child node  $m$ ; these fluctuations are sampled from a multivariate normal distribution with parameters shown in Table 5.1. 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  $\Sigma = 0.3$ .

With an end-of-horizon risk measure given by  $\text{CVaR}_{\beta_0}^\dagger$ , we obtain the following optimisation problem

$$\begin{aligned}
\max_{y, v} \quad & \text{CVaR}_{\beta_0}^\dagger \left[ \sum_{n \in \omega} C_n(y_n, v_n) \right] \\
\text{s.t.} \quad & y_{ms} = \Psi_{ms} \times (y_{ns} + v_{ns}), \quad \forall m \in R(n), \quad \forall s \in \mathcal{S}, \quad \forall n \in \mathcal{N} \setminus \mathcal{L}, \\
& \sum_{s \in \mathcal{S}} v_{ns} = 0, \quad \forall n \in \mathcal{N} \setminus \mathcal{L}, \\
& v_{ns_3} \geq 0, \quad \forall n \in \mathcal{N} \setminus \mathcal{L}, \\
& y_{ns} + v_{ns} \geq 0, \quad \forall s \in \mathcal{S}, \quad \forall n \in \mathcal{N} \setminus \mathcal{L}.
\end{aligned} \tag{5.36}$$

By Theorem 5.4 we are able to formulate a set of dynamic programming equations to supply to our solver. Figure 5.5 shows the convergence toward the optimal saddle point value for a  $\text{CVaR}$  quantile of  $\beta_0 = 0.4$ . Note here that both the upper and lower bounds are *exact* for the optimal end-of-horizon risk adjusted value of the portfolio. Our convergence profile is similar to many cutting plane algorithms. As



discussed in Nesterov (1998), even the traditional Kelley Cutting Plane method can yield arbitrarily poor convergence.

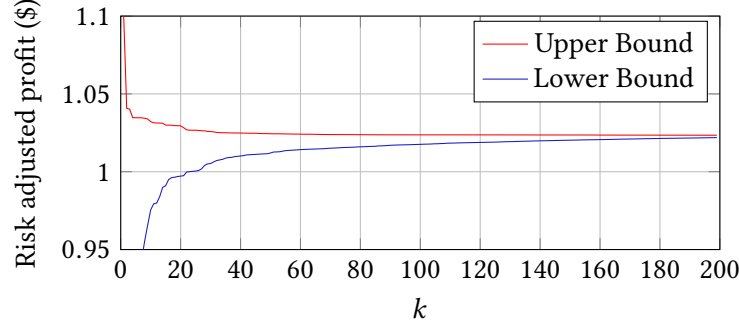


Figure 5.5: Convergence plot for portfolio optimisation example with  $\beta_0 = 0.4$ .

#### 5.4.2 Solution interpretation

Because of the complexity of our class of problems, careful consideration is required when reflecting on their solutions. Figures 5.5 and 5.6 pertain to our portfolio optimisation example where  $\beta_0 = 0.4$  and the portfolio manager begins with a unit of  $s_1$ . Figure 5.5 shows that the optimal policy obtains a risk-adjusted payoff over the planning horizon of at least \$1.0200 but no more than \$1.0215 (an annual increase between 2.0% and 2.15%). From the 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

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

where  $\eta_n$  are the optimal leaf probability scalars. Figure 5.6 plots a histogram of a 2000 sample Monte Carlo simulation which estimates the distribution of accumulated costs for two different optimal 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

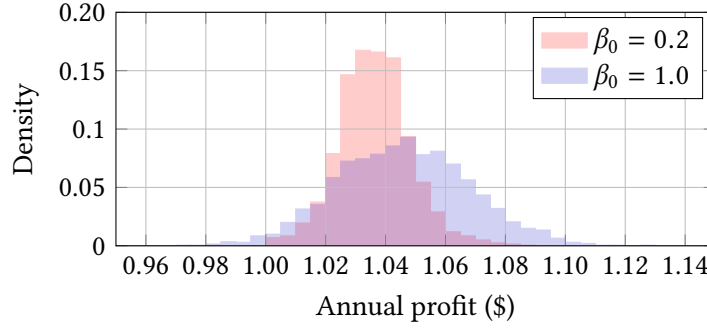


Figure 5.6: Profit distribution for different policies

Table 5.2: Monte Carlo estimation of CVaR quantiles for several policies.

CVaR $_{\beta}$ 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%

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  $\beta_0$  proportion of outcomes. This is in contrast to the traditional nested formulation where risk aversion does not depend on history.

Table 5.2 contains the results of investigation in which our portfolio optimisation problem is parameterised in the CVaR quantile parameter  $\beta_0$ . In this study, we use our algorithm to develop an optimal policy for five different levels of risk aversion,  $\beta_0 = 2$  through to  $\beta_0 = 1$ . For each policy, we then estimate (using Monte Carlo simulation) five conditional expectations of accumulated profits ranging from the worst %20 through to %100. Each estimate is computed from 2000 Monte Carlo samples.

Table 5.2 is to be understood as follows: for the policy optimised for  $\beta_0 = 0.6$ , the results of the five Monte Carlo simulations can be found in the third column.

The conditional expectation of the worst 20% of samples (or similarly the  $\text{CVaR}_{0.2}^+$  estimation), yields a return of %1.30. For the policy generated by  $\text{CVaR}_{0.2}^+$ , we obtain a yield of %1.52 in the  $\text{CVaR}_{0.2}^+$  estimation. This is exactly how we expect each policy to perform; that they give the best performance in their respective quantiles, perhaps at the expense of other quantiles. From our results, it can be observed that the optimal policy performs 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.



## Chapter 6

### Risk in Markov decision processes

All of our work so far has been concerned with problems whose scenario trees are finite. Markov decision processes are a modelling framework that extend the finite horizon problems we have been dealing with so far to infinite horizon problems.

Two main types of objective functions are used in these problems: the infinite horizon discount model and the average cost model. We will focus on the discount model and adapt our results finite horizon to this setting. This chapter contains, in part, work from Baucke (2018) and is motivated by the work of Ruszczyński (2010). The author of this work develops a theory of risk aversion in Markov decision processes, but is limited to ‘rectangular’ risk sets.

Developed alongside dynamic programming, Markov decision processes are a popular framework for decision making. Very technically, a Markov decision process has several key components: a state space, an action space, a probability transition function, and a reward function. The reward and transition probabilities can be functions of both state and action. The agent then seeks a policy which minimises some objective; these policies have the property of being time-stationary, that is they only depend on the state of the system and not the time in the system. formulations are useful for modelling problems which have a natural steady-state nature: for instance inventory problems or machine maintenance problems, where a control is based on the state of the system regardless of its age or history. Unfortunately, the field of Markov decision processes is too large to consider their risk averse counterparts in general; instead we shall focus on a special subset of Markov decision processes which are related to the finite horizon problems that we have been studying throughout this thesis.

## 6.1 The deterministic model

The main result of this work is a convergence theorem for our ensuing algorithm; as has been the case throughout this thesis, we first examine the deterministic problem in order to ‘get to the heart’ of why such an algorithm might converge. We will then carry this intuition over to the stochastic case.

### 6.1.1 Problem formulation

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

$$\begin{aligned} V(x^0) = \min_{u^l} \quad & \sum_{l=0}^{\infty} \delta^l C(x^l, u^l) \\ \text{s.t.} \quad & x^{l+1} = f(x^l, u^l), \quad \forall l \in \mathbb{N}, \\ & x^l \in \mathcal{X}, \quad \forall l \in \mathbb{N}, \\ & u^l \in \mathcal{U}(x^l), \quad \forall l \in \mathbb{N}. \end{aligned} \tag{6.1}$$

We call  $x^l$  the state of the system, while  $u^l$  is called a control. Our objective is to minimise the discounted infinite sum of costs, given an initial state  $x^0$ . We require the functions and multifunctions that make up the above optimisation problem to take a specific form – these conditions ensure that the optimisation problem is convex, feasible, and that subgradients are bounded. This formulation is similar to the ones studied in Warrington et al. (2017) and Nannicini et al. (2017). The work of Warrington et al. (2017) provides an interesting convergence result; their algorithm generates lower bounds that need not be the maximum of linear functions but from larger classes of functions (i.e. perhaps quadratic). Our work differs in several key ways: we study a general non-linear (but convex) version of the problem, where cost functions and constraint sets need not be linear. Furthermore, similar to algorithms presented in Chapters 4 and 5, in the stochastic case we prove that our algorithm converges deterministically. This makes any requirement to estimate an upper bound on the best policy using Monte Carlo simulation redundant.

For the following analysis, we will use

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

and  $\text{Aff}(\mathcal{X})$  to denote the affine hull of  $\mathcal{X}$ . We assume that the optimisation problem (6.1) has the following properties:

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

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

such that

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

These conditions are the same as those considered in Chapter 4. Under these conditions, we will layout several properties of the *cost-to-go* functions induced by (6.1). Noting the self-similar nature of the formulation above, we define the following *cost-to-go* function  $V(x) : \text{Aff}(\mathcal{X}) \mapsto \bar{\mathbb{R}}$  as

$$V(x) = \begin{cases} \min_{u \in \mathcal{U}(x)} C(x, u) + \delta V(f(x, u)), & x \in \mathcal{X}, \\ +\infty, & \text{otherwise.} \end{cases} \quad (6.2)$$

We can pictorially represent the problem in (6.2) as that in Figure 6.1. Here the vertex represents the cost function and the arc represents the transition to the new state. Diagrams like in Figure 6.1 are a useful method for describing more complicated Markov stochastic processes. Given this problem formulation, we will explore the



Figure 6.1: A graph representation of our motivating problem

properties of the induced cost-to-go functions.

### 6.1.2 Properties of value functions

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

$$\tilde{V}(x) = \inf_{u \in \tilde{\mathcal{U}}(x)} C(x, u) + \delta \tilde{V}(f(x, u)). \quad (6.3)$$

Lemma 6.1.

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

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

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

Since  $C(x, u)$  is lower semicontinuous and  $\tilde{\mathcal{U}}(x)$  is non-empty and compact-valued, the (infinitely many) minima above are attained and therefore are equal to the infima in (6.3).  $\square$

Lemma 6.2.

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

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

Lemma 6.3.

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

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

$$-\infty < \frac{1}{1-\delta} \underline{M}_C \leq \tilde{V}(x) \leq \frac{1}{1-\delta} \bar{M}_C < \infty, \quad \forall x \in \mathcal{X}', \quad (6.4)$$



concluding the proof.  $\square$

**Lemma 6.4.**

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

*Proof.* From Lemma 6.3,  $\tilde{V}(x)$  is bounded on  $\mathcal{X}'$ . So  $\tilde{V}(x)$  is Lipschitz on  $\mathcal{X}$ , since from Lemma 6.2,  $\tilde{V}(x)$  is convex, and bounded on  $\mathcal{X}'$ , and  $\mathcal{X}$  is a compact subset of the relative interior of its domain. From Lemma 6.1 we have  $\tilde{V}(x) = V(x)$ ,  $\forall x \in \mathcal{X}$ , so  $V(x)$  is convex and Lipschitz on  $\mathcal{X}$ .  $\square$

Because  $V(x)$  is convex, we can use the convex bounding functions introduced in Chapter 4 to bound our value function  $V(x)$ . Furthermore, because  $V(x)$  is Lipschitz-continuous, we can be sure that our bounding functions are Lipschitz-continuous also.

### 6.1.3 An algorithm and its convergence

Consider an iteration of Algorithm 4.1 from Chapter 4; a state and control trajectory is generated, and value functions are updated until a terminal value function is reached. Since there is no ‘end point’ in an infinite horizon problem, this approach would never be able to complete an iteration. With this considered, a tempting approach to solving (6.1), would be to approximate the objective function with a sufficiently lengthy finite sum and then apply Algorithm 4.1. This approach suffers from two problems: it is difficult (in general) to determine how long a horizon would be required in order to solve the problem to a given accuracy, and secondly, such an approach would neglect the self-similar nature of the value functions, resulting in significantly weaker bounding functions at a given iteration.

When considering the infinite horizon problem, a new approach must be taken in order to achieve convergence. The general idea of the new approach is to allow the length of the forward and backward pass grow larger as the algorithm progresses. Throughout this work we refer to this as ‘looking ahead’. The work of Nannicini et al. (2017) is of particular relevance here; they develop an algorithm which looks ahead further one stage each iteration. Further, we prove that under a certain assumption, the look-ahead process need not be monotonically increasing and only requires that its limit superior tends to infinity.

An important element in our algorithm which addresses the aforementioned pitfall is the ‘look-ahead’ function  $L : \mathbb{N} \mapsto \mathbb{N}$ . For the  $k^{\text{th}}$  iteration, the algorithm generates a control trajectory of length  $L(k)$ , and updates the cost-to-go function at the resulting states. This allows each iteration to contain a finite amount of computation. We present the following algorithm which solves problems in the form of (6.1).

**Algorithm 6.1** (The singleton infinite horizon algorithm).

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

**Iteration  $k$ .**

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

**Step 2.** Solve

$$\begin{aligned} \bar{\theta}^{k,l} &= \min_{x^{l+1}, u^l} C(x^{k,l}, u^l) + \delta \bar{V}^{k-1}(x^{l+1}) \\ \text{s.t. } &x^{l+1} = f^x(x^{k,l}, u^l) \\ &x^{l+1} \in \mathcal{X} \\ &u^l \in \mathcal{U}(x^{k,l}). \end{aligned} \quad (6.5)$$

**Step 3.** Solve

$$\begin{aligned} \underline{\theta}^{k,l} &= \min_{x^{l+1}, u^l} C(x^{k,l}, u^l) + \delta \underline{V}^{k-1}(x^{l+1}) \\ \text{s.t. } &x^{l+1} = f^x(x^{k,l}, u^l) \\ &x^{l+1} \in \mathcal{X} \\ &u^l \in \mathcal{U}(x^{k,l}), \end{aligned} \quad (6.6)$$

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

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

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

$$\begin{aligned} \bar{V}^k(x) &= \max_{\mu \in \mathbb{R}, \lambda \in \mathbb{R}^n} \mu + \langle \lambda, x \rangle \\ \text{s.t. } &\mu + \langle \lambda, x^{k',l} \rangle \leq \bar{\theta}^{k',l}, \forall l \leq L(k'), \forall k' \leq k, \\ &\|\lambda\|_* \leq \alpha. \end{aligned} \quad (6.7)$$

**Step 6.** Update the lower bound as

$$\begin{aligned} \underline{V}^k(x) &= \min_{\mu \in \mathbb{R}} \mu \\ \text{s.t. } \mu &\geq \underline{\theta}^{k',l} + \langle \beta^{k',l}, x - x^{k',l} \rangle, \forall l \leq L(k'), \forall k' \leq k. \end{aligned} \quad (6.8)$$

**Step 7.** Move on to iteration  $k + 1$ .  $\bowtie$

**Theorem 6.1.**

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

$$\lim_{k \rightarrow \infty} (\bar{V}^k(x^{k,l}) - \underline{V}^k(x^{k,l})) = 0, \forall l \in \mathbb{N}. \quad (6.9)$$

For a notational convenience we denote the iteration counter  $k$  as  $k_0$  and let  $\bar{V} = \bar{V} - \underline{V}$ . We will first prove the following technical lemmas which will be useful for the proof of Theorem [6.1](#).

**Lemma 6.5.**

Consider a ‘look-ahead’ function  $L : \mathbb{N} \mapsto \mathbb{N}$  with  $\limsup_{k \in \mathbb{N} \rightarrow \infty} L(k) = \infty$ .

The set

$$\mathbb{N}_L(n) := \{k \in \mathbb{N} \mid L(k) \geq n\},$$

is countably infinite for any  $n \in \mathbb{N}$ .

*Proof.* This comes directly from the limit property of  $L$ .  $\square$

An example of a look-ahead function and its induced set  $\mathbb{N}_L(n)$  is given in Figure [6.2](#). Lemma [6.6](#) constructs a key inequality which will be exploited by our eventual algorithm.

**Lemma 6.6.**

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

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

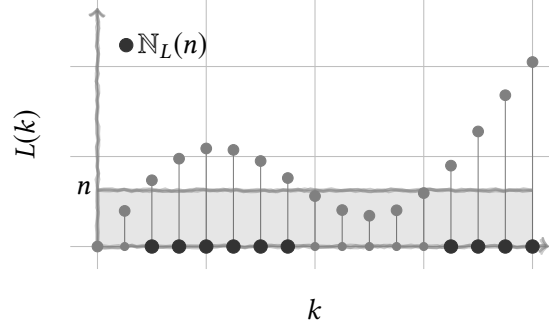


Figure 6.2: The look-ahead function  $L$  and induced set  $\mathbb{N}_L(n)$ .

*Proof.* From (6.6) and (6.8) in Algorithm 4.1, we obtain the following inequality:

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

Similarly for the upper bound, from (6.5) and (6.7) we obtain

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

By taking the difference of these inequalities, we obtain our desired result.  $\square$

**Remark 6.1.**

Note that the inequality developed in Lemma 6.6 holds not only for

$$\forall k_0 \in \mathbb{N}, \forall l \leq L(k_0)$$

but also (by design) for

$$\forall k_0 \in \mathbb{N}_L(l), \forall l \in \mathbb{N}.$$

We proceed with the proof of Theorem 6.1. The proof utilises a fact that agrees with a certain intuition: our bounding functions should eventually converge as long as we look far enough into the future often enough. If  $\limsup_{k \in \mathbb{N} \rightarrow \infty} L(k) = \infty$ , then we have our desired ‘look-ahead’ property.

*Proof of Theorem 6.1* Suppose for the sake of contradiction that this theorem’s hypothesis is false, i.e. there exists an  $\epsilon > 0$  and  $l^* \in \mathbb{N}$  for which

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

From Lemma 6.6 and the monotonicity of  $\bar{V}^k(x)$  with respect to  $k$  for all  $x \in \mathcal{X}$  from Proposition 1.3, we have

$$\epsilon \leq \bar{V}^{k_0}(x^{k_0, l^*}) \leq \delta \bar{V}^{k_1}(x^{k_0, l^*+1}), \quad \forall k_0 > k_1 \in \mathbb{N}_L(l^*). \quad (6.10)$$

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

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

By combining the two previous inequalities, we obtain

$$\epsilon \leq \delta [\bar{V}^{k_1}(x^{k_1, l^*+1}) + 2\alpha \|x^{k_1, l^*+1} - x^{k_0, l^*+1}\|], \quad \forall k_0 > k_1 \in \mathbb{N}_L(l^*). \quad (6.11)$$

We can form a similar inequality by the same process as above – from Lemma 6.6 we have

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

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

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

By merging the two inequalities above, we obtain

$$\bar{V}^{k_1}(x^{k_1, l^*+1}) \leq \delta [\bar{V}^{k_2}(x^{k_2, l^*+2}) + 2\alpha \|x^{k_2, l^*+2} - x^{k_1, l^*+2}\|], \\ \forall k_1 > k_2 \in \mathbb{N}_L(l^* + 1). \quad (6.12)$$

By substituting (6.12) into (6.11), we have

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

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

$$\epsilon \leq \delta^s \bar{V}^{k_s}(x^{k_s, l^*+s}) + 2\alpha \sum_{s'=1, \dots, s} \delta^{s'} \|x^{k_{s'}, l^*+s'} - x^{k_{s'-1}, l^*+s'}\|,$$

$$\forall k_0 > k_1 > \dots > k_s \in \mathbb{N}_L(l^* + s - 1).$$

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

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

$$\forall k_0 > k_1 > \dots > k_{\tilde{s}} \in \mathbb{N}_L(l^* + \tilde{s} - 1).$$

By Lemma [6.5](#), the set  $\mathbb{N}_L(l^* + \tilde{s} - 1)$  is countably infinite, so the above is a contradiction of the compactness of  $\mathcal{X}$  since at least one of the sequences in  $\{x^{k, l^*+1}, \dots, x^{k, l^*+\tilde{s}}\}$ ,  $\forall k \in \mathbb{N}_L(l^* + \tilde{s} - 1)$  contains a non-convergent subsequence. So there exists no  $\epsilon > 0$  and no  $l^* \in \mathbb{N}$  for which

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

concluding the proof.  $\square$

#### Remark 6.2.

In this section, we considered the the ‘convex’ formulation. We saw earlier in this thesis that the minimax formulations in Chapter 5 extended the convex formulations in Chapter 4. By the same token, we could have also considered an infinite horizon minimax formulation here, generated same inequalities and relied upon the same compactness arguments in order to give a similar the convergence result.

With this convergence result understood, we will now consider the stochastic formulation and discuss how risk aversion can be incorporated into these models.

## 6.2 The stochastic model

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

$$\begin{aligned} V_{n_0}^{\text{Ex}}(x_{n_0}^0) &= \min_{u_{\omega(l)}} \mathbb{E}_{\mathbb{P}} \left[ \sum_{l=0}^{\infty} \delta^l C_{\omega(l)}(x_{\omega(l)}, u_{\omega(l)}) \right] \\ \text{s.t. } x_{\omega(l+1)} &= f_{\omega(l+1)}(x_{\omega(l)}, u_{\omega(l)}), \quad \forall l \in \mathbb{N}, \quad \omega \in \Omega, \\ x_{\omega(l+1)} &\in \mathcal{X}_{\omega(l+1)}, \quad \forall l \in \mathbb{N}, \quad \omega \in \Omega, \\ u_{\omega(l)} &\in \mathcal{U}_{\omega(l)}(x_{\omega(l)}), \quad \forall l \in \mathbb{N}, \quad \omega \in \Omega. \end{aligned} \quad (6.13)$$

The set  $\Omega$  is the set of all paths of a countably infinite scenario tree (which encodes the non-anticipativity), and  $\omega(l)$  is the  $l^{\text{th}}$  vertex along path  $\omega$ . This implies that  $n_0 = \omega(0), \forall \omega \in \Omega$  is the root vertex. Here we seek to minimise the expectation over all paths possible of the infinite discounted accumulated costs through that particular path. In the Markov case, rather than associating our probability space with an infinite scenario tree, we can ‘compress’ the tree and describe the problem’s uncertainty with a connected di-graph with every vertex having an in-degree and out-degree of at least one. Denote the vertices on the graph  $\mathcal{N}$  with  $N$  vertices as  $\{n_0, \dots, n_N\}$ , with the set of child vertices of vertex  $n$  as  $R(n)$ . Figure 6.3 depicts an example of such a graph.

Rather than stating the Markov problem in extensive form i.e. (6.13), it is quite natural to express the problem formulation directly through its dynamic programming equations. The infinite horizon dynamic programming equation for an arbitrary vertex  $n \in \mathcal{N}$  is given by

$$\begin{aligned} V_n(x_n) &= \min_{x_m, u_n} C_n(x_n, u_n) + \delta \sum_{m \in R(n)} p_{n,m} V_m(x_m) \\ \text{s.t. } x_m &= f_{n,m}(x_n, u_n), \quad \forall m \in R(n), \\ x_m &\in \mathcal{X}_m, \quad \forall m \in R(n), \\ u_n &\in \mathcal{U}_n(x_n). \end{aligned} \quad (6.14)$$

where  $p_{n,m}$  is the *conditional probability* of reaching vertex  $m$  from vertex  $n$ . Although we do not show it here, we have that the extensive formulation and dynamic programming formulation coincide i.e.  $V_{n_0}^{\text{Ex}}(x_{n_0}^0) = V_{n_0}(x_{n_0}^0)$ . Once again, we restrict

the cost functions and state and control sets to take the form of those considered in the single vertex case. It is also worth mentioning here that a stochastic process of independent and identically distributed random variables can be considered a special case of a Markov process.

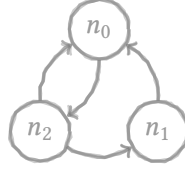


Figure 6.3: A graph representation of our motivating problem

### 6.2.1 Algorithm

The algorithm presented here to solve (6.14) is similar to Algorithm 6.1 except we make use of the *problem-child* criterion outlined in Chapter 4; this is the key ingredient to achieving deterministic convergence. As we ‘look ahead’ during an iteration of our algorithm, we record the child vertex with the largest bound difference for the new state. We label this vertex  $\phi_n^{k,l}$ , the problem-child of vertex  $n$ . Similar to the analysis in Lemma 6.3, it is possible to show that the value functions are convex and Lipschitz-continuous and we can compute initial bounds on the cost-to-go functions ( $\bar{M}_n, \underline{M}_n$ ) at each vertex. Our algorithm for computing  $V_{n_0}(x_{n_0}^0)$  is outlined below.

Algorithm 6.2 (The Markov infinite horizon algorithm).

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

Finally, set  $k = 1$ .

**Iteration  $k$ .**

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



**Step 2.** Solve

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

**Step 3.** Solve

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

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

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

$$\Phi_n^{k,l} = \arg \max_{m \in R(n)} p_{n,m} (\bar{V}_m^{k-1}(f_{n,m}(x_n^{k,l}, u_n^{k,l})) - \underline{V}_m^{k-1}(f_{n,m}(x_n^{k,l}, u_n^{k,l}))). \quad (6.15)$$

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

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

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

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

**Step 7.** For all  $n \in \mathcal{N}$ , update the lower bound functions as

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

**Step 8.** If a particular object has not recieved an update in this iteration, give that object the null update i.e.  $x_n^{k,l} = x_n^{k-1,l}$ . Move on to iteration  $k + 1$ .  $\bowtie$

This Algorithm borrows the *look-ahead* aspect of Algorithm 6.1 and the *problem-child* feature of Algorithm 5.3. We do not rigorously prove its convergence here, but rather appeal to the arguments in Theorem 6.1, Lemma 4.1, and Theorem 5.2, to give our result.

## 6.2.2 Risk averse formulations

Similar to Chapter 6, we now turn to a discussion about risk averse formulations. In the same way that we form risk averse formulations for the finite horizon case, we can form risk averse formulations for the infinite horizon case. Under our Markov assumption of uncertainty, we can write in an extensive form a *risk averse Markov decision process* as

$$\begin{aligned} V_{n_0}(y_{n_0}^0) &= \max_{v_{\omega(l)}} v \left[ \sum_{l=0}^{\infty} \delta^l C_{\omega(l)}(y_{\omega(l)}, v_{\omega(l)}) \right] \\ \text{s.t. } y_{\omega(l+1)} &= f_{\omega(l+1)}(y_{\omega(l)}, u_{\omega(l)}), \forall l \in \mathbb{N}, \omega \in \Omega, \\ y_{\omega(l+1)} &\in \mathcal{Y}_{\omega(l+1)}, \forall l \in \mathbb{N}, \omega \in \Omega, \\ v_{\omega(l)} &\in \mathcal{V}_{\omega(l)}(y_{\omega(l)}), \forall l \in \mathbb{N}, \omega \in \Omega. \end{aligned} \quad (6.16)$$

In order to form dynamic programming equations for the above formulation, we require that  $v$  has the decomposability property. In particular for the infinite horizon case, we require that

$$v = v_0^1 \circ v_1^2 \circ \dots$$

In Chapter 5, we discussed the *rectangular* decomposition for finite horizon stochastic optimisation problems. Once again, we will discuss these ideas in terms of acceptance measures rather than risk measures and will forgo denoting the ‘max’ variables to ease the notational burden. So, without loss of generality, we seek to compute

$$v \left[ \sum_{t=0, \dots, T} C_t(\omega) \right], \quad (6.17)$$

where  $v$  is a *decomposable* acceptance measure, and the rewards  $C_t$  are adapted to the filtration. The rectangular CVaR formulation for the infinite horizon case is given

by

$$\begin{aligned}
 G_n^N(\eta_n) &= \min_{\eta_m} \quad \eta_n C_n + \delta \sum_{m \in R(n)} p(n, m) [G_m^N(\eta_m)] \\
 \text{s.t.} \quad &0 \leq \eta_m \leq \frac{\eta_n}{\beta_n}, \quad \forall m \in R(n), \\
 &\sum_{m \in R(n)} p(n, m) \eta_m = \eta_n,
 \end{aligned} \tag{6.18}$$

for all  $n \in \mathcal{N}$ . This model of risk in the infinite horizon case is identical to the one considered by Ruszczyński (2010). In their paper, a critical observation is made: that not only is a stationary Markov policy generated by the maximiser, but also a *stationary probability measure* generated by nature. The author correctly notes the connection between risk aversion and minimax models.

Of particular interest is the formulation where  $v$  in (6.17) is the CVaR acceptance measure. In the context of the comments of Ruszczyński (2010), we not only generate a Markov policy for the maximiser, but also a stationary probability measure, which now can be a function of previously selected probability measures. The CVaR end-of-horizon dynamic programming formulation is given as

$$\begin{aligned}
 G_n^E(\eta_n) &= \min_{\eta_m} \quad \eta_n C_n + \delta \sum_{m \in R(n)} p(n, m) [G_m^E(\eta_m)] \\
 \text{s.t.} \quad &0 \leq \eta_m \leq \frac{1}{\beta_0}, \quad \forall m \in R(n), \quad (6.19.1) \\
 &\sum_{m \in R(n)} p(n, m) \eta_m = \eta_n,
 \end{aligned} \tag{6.19}$$

for all  $n \in \mathcal{N}$ . This formulation has an inherent paradox embedded within: we seek to compute the end-of-horizon risk adjusted value of a random variable whose horizon is not bounded. This, of course, is perfectly reconcilable if one notes that the infinite horizon sum is bounded. Nevertheless, this provides an interesting philosophical focal point for discussion about how risk is modelled in multistage optimisation problems.

This formulation, along with Algorithm 6.2 skirts what would seem an unavoidable problem when computing CVaR of a geometrically discounted infinite sum of random variables. Naïvely, one would think it necessary to have to simulate an extraordinarily large number of samples in order to compute CVaR. This need not be the case as our formulation shows. We merely need to compute the value functions given in (6.19), a far easier task. We contend that this formulation contributes one of the main results of this thesis.

In conclusion of this section, we have presented a general infinite horizon Markov decision process model; one which extends the formulations in Chapters 4 and 5. We have noted that these formulations require a new algorithm to solve, due to their infinite nature. We have presented a new algorithm to solve these models and proved its convergence under a natural assumption of the look ahead function  $L : \mathbb{N} \mapsto \mathbb{N}$ . Finally, we presented to minimax formulations which are equivalent to two risk averse Markov decision process models. The first mimics the formulations studied in Ruszczyński (2010), while the second is a end-of-horizon, infinite horizon risk model. Both of these formulations can be solved by Algorithm 6.2; convergence is deterministic and there is no need to use Monte Carlo estimation to obtain an upper bound on the optimal solution.

## Chapter 7

### Conclusions and further research

Over this thesis, we have discussed topics ranging from probability, to optimisation, algorithms, and convergence analysis. This section seeks to provide a short summary of this thesis, and to discuss what in particular this thesis contributes to fields of stochastic programming, and operations research, more generally.

#### 7.1 How this research contributes

Chapter 2 provided largely a introduction to the concepts of coherent risk measures, which are all but ubiquitous through stochastic programming today. We demonstrated several key results of stochastic dominance and law invariance, and not least of all the *dual representation* of risk measures, which is where the connection between risk aversion and minimax optimisation is first made.

Chapter 3 discussed how these risk measures can be extended into the multi-stage setting in a way which ensures that the measures maintain several desirable properties; these were the concept of time-consistency and stochastic dominance. We declared that an optimisation problem which has a dynamic programming decomposition is equivalent to a sequence of optimisation problems which obey time-consistency. Using this time-consistency concept, we showed how conditional risk measures (be they parametric or not), when nested, form a temporally logical risk evaluation of a random variable.

Chapter 4 pivoted somewhat to a more algorithmic theme, where we introduced a convex class of *risk neutral* multistage stochastic optimisation problems. From here, we first review the classic result of Kelley (1960) and built up towards an algorithm for solving multistage stochastic optimisation problems (SDDP). We then developed the

*upper bound* function and showed how this, along with the *problem child* criterion, could be used in an algorithm for solving these problems deterministically. This is a key finding from this thesis; the deterministic convergence of our algorithm means that: we forgo the need for Monte Carlo estimation and a statistical test of convergence; and that we produce tighter lower bounds, owing to the fact that the problem child mechanism directs sampling to the ‘useful’ parts of the scenario tree.

Chapter 5 introduced a general framework for minimax stochastic multistage optimisation problems, an extension of the risk neutral version of the optimisation problem. We developed upper and lower bounds for saddle functions and showed how these saddle functions could be refined in an algorithm similar to SDDP, in order to solve these minimax stochastic dynamic programmes. We then showed how we were able to produce minimax formulations which were equivalent to common risk averse formulations for multistage stochastic optimisation problems.

Chapter 6 took from Chapters 4 and 5 and extended these insights to the infinite horizon paradigm. We provided a new convergence result for both the standard convex and minimax formulations of these problems, when the objective function is a geometric sum of discounted random variables. From here, we developed risk averse formulations, similar to those in Chapter 5. Of particular note here is the *end-of-horizon* risk averse formulation, where we paradoxically seek to compute an optimal policy which maximises the risk-adjusted reward over the end of the whole (infinite) horizon.

In the thesis, we have highlighted further the connections between risk averse optimisation and robust optimisation; especially when placed in a dynamic setting. In particular, we have built a case for using end-of-horizon risk measures in these optimisation problems, and have developed a suite of new and remarkable results, along with an algorithm in order to solve these problems. This saddle function theory has found application in other areas of stochastic optimisation (see Downward et al. 2018).

### 7.1.1 Challenges

No research endeavour is without its challenges. Researchers in mathematical programming and operations research should, and rightly so, always maintain an eye for the practical application of their methods. The popularity of multistage stochastic optimisation models in energy and finance applications, as well as the desire to form

risk averse policies in these applications, gives a motivation and validation for the work in this thesis.

However, this thesis is largely a theoretical work; the computational examples provided in this thesis are used more of a ‘proof-of-concept’ of a given algorithm rather than a demonstration of the efficacy of the particular implementation of the algorithm. While linear programming and mixed-integer programming solvers have enjoyed a sharp rise in utilisation in the last decade; designing robust (in the non-technical sense of the word) solvers for more complex optimisation problems still remains a challenge. The designer must trade off against code complexity, which can introduce many bugs, and the ability for the solver to accept a diverse range of inputs.

Often, computational models cannot be underestimated as a means of communicating an underlying theoretical concept; we believe there is still much to be explored in terms of demonstrating the subtleties in risk averse decision making, comparing different models of risk, and highlighting the strengths and weaknesses of the different models of risk. To this end, the development of a reliable and robust minimax saddle point solvers for use in the algorithms developed in this work, would be a worthwhile endeavour.

## 7.2 Research questions borne from this thesis

During the course of this PhD, distributionally robust optimisation has risen in popularity within the stochastic programming community (see Hanasusanto and Kuhn [2016](#); Mohajerin Esfahani and Kuhn [2017](#)). In Chapter 5, we showed how we could construct parametric conditional risk sets whose composition gave a risk set equal to CVaR. Of particular popularity is the Wasserstein uncertainty set (see Mohajerin Esfahani and Kuhn [2017](#)). It would be interesting to investigate other conditional uncertainty sets whose composition give other uncertainty sets such as a Wasserstein ball. This idea can of course be extended to multistage robust optimisation in general; robust optimisation with parametric uncertainty sets are often referred to as *budget constrained robust optimisation problems*. Solution techniques for these optimisation problems are still in their infancy; it is hoped that the research in this thesis contributes to their development.

Chapter 4 introduced the *problem child* concept, where we were able to achieve deterministic convergence for multistage stochastic programming problems on finite probability spaces. It is hoped that this concept will provide a catalyst for research dedicated to an algorithm which obtains deterministic convergence for stochastic optimisation problems on continuous probability spaces. The benefits of such an approach over the sample average approximation methods are twofold: fewer samples may be required in order to characterise the optimal solution (meaning faster computational times); and the method provides a deterministic guarantee of solution quality, leading to more robust policies.

Common in many SDDP implementations is the the concept of *cut selection* (see Bandarra and Guigues [2017](#)). This can be thought of a subroutine that is periodically called during the algorithm. For a given value function approximation  $V(x)$ , this subroutine seeks to remove constraints from the definition of the value function; the result being a reduction in the size of the linear programming representation of the value function. It is the function of the subroutine to identify ‘cuts’ which will never become binding; that is, the identified cut is dominated everywhere by other cuts. Such an endeavour is difficult, even in the convex case of SDDP. The Algorithm developed in Chapter 5 would also benefit from a *cut selection* subroutine for saddle function representations, however it is not clear how such a subroutine would work. A possible avenue for future research would be to investigate different cut selection methodologies for *saddle function* upper and lower bounds outlined in this thesis.

That being said, we hope that the results developed in this thesis provide a satisfying resolution to some of the apprehension surrounding the current models of risk in multistage optimisation problems, and help the continual ‘bridging of the gap’ between risk aversion and robust optimisation.



## References

- Acerbi, C. (2002). “Spectral measures of risk: A coherent representation of subjective risk aversion”. In: *Journal of Banking and Finance* 26.7, pp. 1505–1518.
- Artzner, P., Delbaen, F., Eber, J.-M., and Heath, D. (1999). “Coherent Measures of Risk”. In: *Mathematical Finance* 9.3, pp. 203–228.
- Artzner, P., Delbaen, F., Eber, J.-M., Heath, D., and Ku, H. (2002). “Coherent multi-period risk measurement”. In: *Manuscript, ETH Zurich*. Pp. 1–13.
- Asamov, T. and Ruszczyński, A. (2015). “Time-consistent approximations of risk-averse multistage stochastic optimization problems”. In: *Mathematical Programming* 153.2, pp. 459–493.
- Bandarra, M. and Guigues, V. (2017). “Multicut decomposition methods with cut selection for multistage stochastic programs”. In: *Optimization Online*. arXiv: [1705.08977](https://arxiv.org/abs/1705.08977).
- Baucke, R. (2018). “An algorithm for solving infinite horizon Markov dynamic programmes”. In: *Optimization Online*.
- Baucke, R., Downward, A., and Zakeri, G. (2017). “A deterministic algorithm for solving multistage stochastic programming problems”.
- (2018). “A deterministic algorithm for solving multistage stochastic minimax dynamic programmes”. In: *Optimization Online*.
- Bellman, R. (1954). “The theory of dynamic programming”. In: *Bulletin of the American Mathematical Society*. Vol. 60. 6, pp. 503–515.
- Benders, J. F. (1962). “Partitioning procedures for solving mixed-variables programming problems”. In: *Numerische Mathematik* 4.1, pp. 238–252.
- Billingsley, P. (1995). *Probability and Measure*. Wiley Series in Probability and Statistics. Wiley.
- Birge, J. R. (1985). “Decomposition and Partitioning Methods for Multistage Stochastic Linear Programs”. In: *Operations Research* 33.5, pp. 989–1007.

- Birge, J. R. and Louveaux, F. (2011). “Two-Stage Recourse Problems”. In: *Introduction to Stochastic Programming*. New York, NY: Springer New York, pp. 181–263.
- Birge, J. R. and Louveaux, F. V. (1988). “A multicut algorithm for two-stage stochastic linear programs”. In: *European Journal of Operational Research* 34.3, pp. 384–392.
- Carpentier, P., Chancelier, J.-P., Cohen, G., De Lara, M., and Girardeau, P. (2012). “Dynamic consistency for stochastic optimal control problems”. In: *Annals of Operations Research* 200.1, pp. 247–263.
- Chen, Z. and Powell, W. (1999). “Convergent Cutting-Plane and Partial-Sampling Algorithm for Multistage Stochastic Linear Programs with Recourse 1”. In: *Optimization* 102.3, pp. 497–524.
- Dinkelbach, W. (1969). “Parametrische lineare Programmierung”. In: *Sensitivitätsanalysen und parametrische Programmierung*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 90–149.
- Downward, A., Dowson, O., and Baucke, R. (2018). “Stochastic dual dynamic programming with stagewise dependent objective uncertainty”. In: *Optimization Online*.
- Dunning, I., Huchette, J., and Lubin, M. (2015). “JuMP: A Modeling Language for Mathematical Optimization”. In: *arXiv:1508.01982 [math.OC]*, pp. 1–25. arXiv: [1508.01982](https://arxiv.org/abs/1508.01982)
- Georghiou, A., Wiesemann, W., and Tsoukalas, A. (2017). “Robust Dual Dynamic Programming”. In: *Optimization Online* 1.1.
- Girardeau, P., Leclere, V., and Philpott, A. (2014). “On the Convergence of Decomposition Methods for Multistage Stochastic Convex Programs”. In: *Mathematics of Operations Research* 40.1, pp. 130–145.
- Grimmett, G., Grimmett, P., Stirzaker, D., Stirzaker, M., and Grimmett, S. (2001). *Probability and Random Processes*. Probability and Random Processes. OUP Oxford.
- Gurobi Optimization, I. (2016). *Gurobi Optimizer Reference Manual*.
- Hanasusanto, G. A. and Kuhn, D. (2016). “Conic Programming Reformulations of Two-Stage Distributionally Robust Linear Programs over Wasserstein Balls”. In: *Operations Research* August. arXiv: [1609.07505](https://arxiv.org/abs/1609.07505).
- Homem-De-Mello, T. and Pagnoncelli, B. K. (2016). “Risk aversion in multistage stochastic programming: A modeling and algorithmic perspective”. In: *European Journal of Operational Research* 249.1, pp. 188–199.

- Infanger, G. and Morton, D. P. (1996). "Cut Sharing for Multistage Stochastic Linear Programs with Interstage Dependency". In: *Mathematical Programming* 75.241-256, pp. 241–256.
- Iyengar, G. N. G. (2005). "Robust Dynamic Programming". In: *Mathematics of Operations Research* 30.2, pp. 1–29.
- Kelley, J. (1960). "The cutting-plane method". In: *Journal of the Society for Industrial and Applied Mathematics* 8.4, pp. 703–712.
- Kupper, M. and Schachermayer, W. (2009). "Representation results for law invariant time consistent functions". In: *Mathematics and Financial Economics* 2.3, pp. 189–210.
- Kusuoka, S. (2001). "On law invariant coherent risk measures". In: *Advances in Mathematical Economics*. Ed. by S. Kusuoka and T. Maruyama. Tokyo: Springer Japan, pp. 83–95.
- Lindvall, T. (1999). "On strassen's theorem on stochastic domination". In: *Electronic Communications in Probability* 4, pp. 51–59.
- Madansky, A. (1959). "Bounds on the Expectation of a Convex Function of a Multivariate Random Variable". In: *The Annals of Mathematical Statistics* 30.3, pp. 743–746.
- Markowitz, H. (1952). "Portfolio Selection". In: *The Journal of Finance* 7.1, pp. 77–91. arXiv: [arXiv:1011.1669v3](https://arxiv.org/abs/1011.1669v3).
- Martin, D. H. (1975). "On the continuity of the maximum in parametric linear programming". In: *Journal of Optimization Theory and Applications* 17.3-4, pp. 205–210.
- Mohajerin Esfahani, P. and Kuhn, D. (2017). *Data-driven distributionally robust optimization using the Wasserstein metric: performance guarantees and tractable reformulations*. Springer Berlin Heidelberg, pp. 1–52. arXiv: [1505.05116](https://arxiv.org/abs/1505.05116).
- Nannicini, G., Traversi, E., and Calvo, R. W. (2017). "A Benders squared (B2) framework for infinite-horizon stochastic linear programs". In: Umr 7538.
- Nesterov, Y. (1998). "Introductory Lectures on Convex Programming Volume I: Basic course". In: *Lecture Notes I*, pp. 1–211.
- Pereira, M. V. F. and Pinto, L. M. V. G. (1991). "Multi-stage stochastic optimization applied to energy planning". In: *Mathematical Programming* 52.1-3, pp. 359–375.

- Pflug, G. (2000). "Some Remarks on the Value-at-Risk and the Conditional Value-at-Risk". In: *Probabilistic Constrained Optimization. Nonconvex Optimization and Its Applications* 49. Ed. by S. Uryasev, pp. 272–281.
- Pflug, G. and Pichler, A. (2016). "Time-inconsistent multistage stochastic programs: Martingale bounds". In: *European Journal of Operational Research* 249.1, pp. 155–163.
- Philpott, A. and De Matos, V. (2012). "Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion". In: *European Journal of Operational Research* 218.2, pp. 470–483.
- Philpott, A., De Matos, V., and Finardi, E. (2013). "On Solving Multistage Stochastic Programs with Coherent Risk Measures". In: *Operations Research* 51.4, pp. 957–970.
- Philpott, A. and Guan, Z. (2008). "On the convergence of stochastic dual dynamic programming and related methods". In: *Operations Research Letters* 36.4, pp. 450–455.
- Philpott, A. and Pritchard, G. (2013). "EMI-DOASA". In: Powell, W. B. (2011). *Approximate Dynamic Programming: Solving the Curses of Dimensionality: Second Edition*. arXiv: [arXiv: 1011.1669v3](https://arxiv.org/abs/1011.1669v3).
- Riedel, F. (2004). "Dynamic coherent risk measures". In: *Stochastic Processes and their Applications* 112.2, pp. 185–200.
- Rockafellar, R. T. and Uryasev, S. (2000). "Optimization of conditional value-at-risk". In: *Journal of Risk* 2, pp. 21–41. arXiv: [arXiv: 1011.1669v3](https://arxiv.org/abs/1011.1669v3).
- Rockafellar, R. T., Uryasev, S. P., and Zabarankin, M. (2002). *Deviation Measures in Risk Analysis and Optimization*. Tech. rep., pp. 1–27.
- Ruszczynski, A. (2010). "Risk-averse dynamic programming for Markov decision processes". In: *Mathematical Programming* 125.2, pp. 235–261.
- Ruszczynski, A. and Shapiro, A. (2006a). "Conditional Risk Mappings". In: *Mathematics of Operations Research* 31.3, pp. 544–561.
- (2006b). "Optimization of Convex Risk Functions". In: *Mathematics of Operations Research* 31.3, pp. 433–452. arXiv: [1111.0194v1](https://arxiv.org/abs/1111.0194v1).
- Shapiro, A. (May 2009). "On a time consistency concept in risk averse multistage stochastic programming". In: *Operations Research Letters* 37.3, pp. 143–147.
- (2011). "Analysis of stochastic dual dynamic programming method". In: *European Journal of Operational Research* 209.1, pp. 63–72.

- (Nov. 2012). “Time consistency of dynamic risk measures”. In: *Operations Research Letters* 40.6, pp. 436–439.
- Shapiro, A., Dentcheva, D., and Ruszczyński, A. (2009). “Lectures on stochastic programming: modeling and theory”. In: *Technology*, p. 447.
- Shapiro, A. and Homem-de-Mello, T. (2000). “On the Rate of Convergence of Optimal Solutions of Monte Carlo Approximations of Stochastic Programs”. In: *SIAM Journal on Optimization*.
- Shapiro, A. and Nemirovski, A. (2005). “On Complexity of Stochastic Programming Problems”. In: *Continuous Optimization: Current Trends and Modern Applications*. Ed. by V. Jeyakumar and A. Rubinov. Boston, MA: Springer US, pp. 111–146.
- Von Neumann, J. (1928). “Zur Theorie der Gesellschaftsspiele”. In: *Mathematische Annalen* 100.1, pp. 295–320.
- Warrington, J., Beuchat, P. N., and Lygeros, J. (2017). “Generalized Dual Dynamic Programming for Infinite Horizon Problems in Continuous State and Action Spaces”. In: arXiv: [1711.07222](https://arxiv.org/abs/1711.07222).