River Optimisation : short-term hydro-bidding under uncertainty

Faisal Wahid

NNT : 2017SACLX030

THESE DE DOCTORAT
DE
L'UNIVERSITE PARIS-SACLAY
THESE UNIVERSITE PREPAREE A
L'ÉCOLE POLYTECHNIQUE

ECOLE DOCTORALE N°574
ÉCOLE DOCTORALE DE MATHEMATIQUES HADAMARD
SPECIALITE DE DOCTORAT : MATHEMATIQUES APPLIQUEES

Par

M. Faisal Wahid

Optimisation de la rivière : enchères à court terme de l'hydroélectricité sous incertitude

Thèse présentée et soutenue au centre Inria-Saclay, le 2 juin 2017.

Composition du jury:

Mme. Claudia d'Ambrosio	CNRS & LIX, École Polytechnique	Président
M. Stein-Erik Fleten	Norwegian University of Science and Technology, Trondheim	Rapporteur
M. Michel de Lara	Cermics, ENPC Paris Tech	Rapporteur
M. Danny Ralph	Cambridge Judge Business School, Cambridge University	Examinateur
M. Anthony Downward	Department of Engineering Science, University of Auckland	Examinateur
M. Frédéric Bonnans	Inria et CMAP, École Polytechnique	Directeur de thèse
M. Andrew Philpott	Department of Engineering Science, University of Auckland	Directeur de these

Résume

L’énergie hydroélectrique est la forme la plus répandue d’énergie renouvelable, avec une production de 16 400 TWh par an. Il fournit non seulement une électricité propre et renouvelable, mais il atténue également la volatilité d'autres sources de production renouvelables, telles que l'éolien et solaire en agissant comme un tampon pour équilibrer l’alimentation avec la charge. En outre, l’énergie hydroélectrique contribue de manière significative à la fiabilité d'un réseau électrique. En raison du démarrage court temps, les centrales hydroélectriques sont très efficaces en tant que primaire et secondaire source d'énergie de réserve, ainsi qu'un support de tension idéal. Dans cette thèse, nous étudions l’optimisation de l’hydroélectricité dans un contexte de marché libéralisé, dans deux pays, la Nouvelle-Zélande et la France.

Marchés de l'électricité en Nouvelle-Zélande et en France :

Le marché de l'électricité en Nouvelle-Zélande fonctionne comme un marché intra-journalier, où les participants fournissent une offre en cinq tranches toutes les demi-heures à le 'ISO (« Transmission System Operator ») pour les 72 prochaines périodes d'échange. En utilisant ces offres, le TSO calcule les prix au comptant provisoires, qui sont ensuite rapprochés basé sur la consommation réelle. Afin d'aider les acteurs du marché comprendre leur position sur le marché, le 'ISO prévoit des prix au comptant toutes les deux heures pour chaque période d'échange jusqu'à 36 heures à l'avance. Le 'ISO fournit également des prix indicatifs de cinq minutes. Ceux-ci sont calculés à la fin de chaque période de cinq minutes et tenir compte de l'évolution des conditions du système d'alimentation. Cependant, les cinq minutes indicatives les prix ne coïncident pas avec les nouvelles offres, mais sont basés sur les modifications du système d'alimentation proche du temps réel. De l'autre Par contre, les prévisions de prix sur deux heures correspondent aux nouvelles offres sont alignés sur les délais de soumission des offres.

Le marché français de l’électricité fait partie du vaste réseau régional de l’électricité initiatives ou « Electricity Regional Initiative » (ERI) initialement mises en vigueur en 1996 par la directive 96/92 / CE qui a finalement évolué vers le centre couplage de marché ouest-européen couvrant la France, la Belgique, Pays-Bas, Allemagne, Luxembourg et Autriche. L’ERI a pour objectif d’avoir un marché paneuropéen de l’électricité, appelé le marché intégré de l’électricité ou « Integrated Electricity Market » (IEM), qui consiste à coupler différents marchés de l’énergie à travers l’Europe en volume et / ou en prix.

L'IEM est divisé en plusieurs niveaux, le premier niveau est le contrôle zones des différents gestionnaires de réseau de transport qui, dans le cas de la France, c'est le long de la frontière nationale. Les marchés situés à l'intérieur des zones de contrôle sont constitués d'un marché à terme, d'un marché journalier intérieur ou « Day-Ahead Market » (DAM), un marché intra-journalier ou « Intra-Day Market » (IDM) et un marché d'équilibrage ou « Balancing Market ». Le marché à terme est un marché financier qui négocie des contrats bilatéraux et contrats en vente libre pour les charges de base et de pointe. Ce commerce est basé sur les prévisions de charge à long terme des zones de contrôle.

Le probleme Hydro-bidding:

Le problème de hydro-bidding concerne le calcul des strategies d'offre optimale afin de maximiser le bénéfice attendu d'un producteur hydroélectrique participer à ces marchés de l'électricité. Il combine la décision processus de fabrication à la fois du commerçant et de l'hydro-régulateur en un problème d'optimisation stochastique. Hydro-bidding modèles à prise de prix, qui supposent que les agents n’ont pas de pouvoir de marché, visent à arbitrer entre les différents prix, à différentes périodes, afin de maximiser les profits. Ces modèles sont généralement formulés en temps discret, horizon fini, problème de contrôle optimal stochastique.

Le problème hydro-bidding peut faire l'expérience par :

- Grand espace d'incertitude de la modélisation de la nature stochastique des prix, des flux entrants et de la consommation,
- Fonctions de valeur non concaves en raison de la présence de variables entières et de fonctions

de production d'énergie non concaves,
- Coûteux en termes de calcul à résoudre en présence de grands systèmes hydroélectriques avec nombreux réservoirs interconnectés.

Cela a attiré l'utilisation de diverses techniques pour résoudre différentes variantes. Les principales méthodes utilisées sont les suivantes :

- **Stochastic Dynamic Programming (SDP)** : SDP peuvent être formulés en tant que programme dynamique stochastique à horizon fini, utilisant l'équation bien connue de valeurs. Le problème majeur avec SDP est le « curse of dimensionality ». Depuis SDP utilise la récursion en arrière pour calculer la stratégie optimale, de gros problèmes avec de nombreuses étapes deviennent rapidement insolubles en raison du nombre astronomique d'étapes de calcul nécessaires pour résoudre ces modèles.
- **Stochastic Mixed-integer Programming (SMIP)** : En construisant une approximation finie et discrète de la distribution des variables stochastiques, il est possible de formuler un modèle équivalent déterministe complet. Dans de tels modèles, toutes les réalisations des variables stochastiques dans le temps sont représentées sous forme d'arborescence de scénarios basée sur l'approximation discrète. La limitation de cette approche est taille de modelé. Comme SDP, il est presque impossible de construire et de résoudre de gros problèmes avec de nombreuses périodes d'échange en raison de la croissance exponentielle de l'arbre de scénario avec un nombre croissant de périodes d'échange et de scénarios.
- **Stochastic Dual Dynamic Programming (SDDP)**: SDDP est un algorithme multi-étage basé sur le Benders Decomposition, qui approxime la fonction de valeur avec d'hyper-plans. En supposant un échantillonnage indépendant, SDDP garantit une convergence presque sûre lorsque chacun des sous-problèmes est convexe, mais pas autrement. Le problème peut par être non convexe parce que de la modélisation de la fonction de production des puissances et de l'engagement de l'unité. Par conséquent, l'utilisation de SDDP pour résoudre le problème d'enchères hydrauliques non convexes ne garantit pas la convergence du SDDP.
- **Aproximate Dynamic Programming (ADP)** : Une nouvelle classe d'algorithmes pour la dynamic programs peut réduire la malédiction de la dimensionnalité en se rapprochant de la fonction des valeurs. ADP résout des problèmes comportant des fonctions des valeurs monotoniques non concaves. Cependant, le défi avec ADP est d'approcher avec précision la fonction de Valeurs. De plus, ces algorithmes ne garantissent pas des stratégies optimales. Ce n'est que lorsque l'on simule la politique que l'on peut évaluer ses performances. À cet égard, SDDP présentent un avantage, car ils calculent également une limite supérieure et une limite inférieure dans lesquelles la stratégie candidate est délimitée. Cela donne à l'utilisateur une bonne indication de la proximité entre l'optimalité et la stratégie actuelle.

Sur la base de ces méthodes existantes, nous observons qu'il existe un nombre limité d'algorithmes qui traitent de la non convexité des hydro-bidding modèles découlant de variables entières, de contraintes non linéaires ou d'objectifs. Dans cette thèse, nous étudions des variantes du problème hydro-bidding non convexes et développons de nouvelles techniques pour les résoudre.

Notre contribution :

Dans cette thèse, nous proposons une nouvelle méthode d'optimisation stochastique appelée le Mixed-Integer Dynamic Approximation Scheme (MIDAS). MIDAS résout des programmes stochastiques non convexe avec des fonctions des valeurs monotones. Il fonctionne de manière similaire à la Stochastic Dual Dynamic Programming (SDDP), mais au lieu utiliser des hyperplans, il utilise des fonctions d'étape pour créer une approximation externe de la fonction de valeur. MIDAS converge « presque sûrement » à $2T\varepsilon$ solution optimale des première étage décisions.

Nous utilisons MIDAS pour résoudre trois types de problèmes hydro-bidding non convexes. Le premier modèle d'hydro-bidding que nous résolvons a des variables d'état entier car les productions sont discrètes. Dans ce modèle, nous démontrons que MIDAS est meilleur que SDDP. Le modèle suivant d'hydro-bidding utilise des processus de prix autorégressifs au lieu d'une chaîne de Markov. Le dernier modèle d'hydro-bidding intègre les effets de hauteur d'eau, où la fonction de production d'énergie dépend du niveau de stockage du réservoir et du débit d'eau de la turbine. Dans tous ces modèles, nous démontrons que la convergence de MIDAS en un nombre fini d'itération.

I dedicate this thesis to the three most influential adults, my mum Hosna Ara Khanam, my dad Mohammed Shahjahan Miah, and my highschool maths teacher Johanna McHardy. They taught me to live life to the fullest with empathy, kindness, passion, perseverance, and most importantly, love. I will forever be in your debt.

Acknowledgements

If this thesis were a biography, then I would merely be its ghost writer. The ideas, however well or terribly written, are the result of collaboration and support from passionate experts, across New Zealand and France. I was very lucky to have worked with them.

The first group of experts that I am eternally grateful to are my supervisors Professor Andy Philpott and Professor Frédéric Bonnans. Andy, thank you for all the help in the past four years. I will always take with me the memorable moments, like when we formulated the HERBS model at Geneva airport, or when we came up with the basic idea of MIDAS while sipping coffee at cafe in Paris. I am grateful from your example on how to bring out the best in people. Frédéric, thank you for welcoming me to the CMAP family, the continuous support on the many administrative processes, and for making me a Francophile. Thank you for patiently listening to my impractical ideas, and for meticulously reviewing my mathematical formulations and drafts of papers and presentations.

The second group of experts I would like to thank are Anes Dallagi, Cédric Gouvernet, and Sandrine Charousset from EDF and PGMO, Andrew Kerr and Grant Telfar from Meridian Energy. Anes, Cédric and Sandrine, thank you for welcoming me to the EDF Clamart office, and for the financial support while I was based in France. Andrew and Grant, thank you for both the financial support from Meridian, and for your continous advocacy for this research project.

For nearly a decade, I have been fortunate to call the Department of Engineering Science my home. To my fellow colleagues in the Electricity and Power Optimization Center (EPOC), the supervisors Tony Downward, Golbon Zakeri and Geoff Pritchard, and to the PhD students Corey Kok, Mahbubeh Habibian, Keith Ruddell, Oscar Dowson and Regan Baucke. It has been a privilege to work with you. To those doing their PhD, good luck. I would like to also the acknowledge administrative staff of the Department of Engineering Science and CMAP in École Polytechnique: Barbara, Jessica and Naaséra. Without you we would be all running like headless chickens.

The quality of this thesis has been uplifted by Alif Wahid, Ben Booker, Kathleen Gilbert, Oscar Dowson, and Merlin the dog. No amount of thanks is enough for you all taking the time to read my writing and provide valuable feedback. To my big brother Alif, thank you for all the constructive criticism and words of motivation. Ben, thank you for the detail corrections of spelling, or lack of. I hope reading the drafts helped you with your insomnia. Oscar, thank you for picking out the details in my math formulations, and analysis. Kat, thank you for all the advice on how to structure the thesis and the coffee sessions. Those sessions profoundly helped me through a dark period of my PhD. And Merlin, thank you for the engmatic feedback.

This past four years would have been a very lonely journey if it was not for the support of my family and friends. There are not enough pages in this thesis to describe my gratitude so if you are not explicitly named, then please know that your support is not forgotten. I would like to begin with thanking my ex-flatmates and friends, Aidan Sharples, Daniel Xu, the two Bens, Pieter Ubels, and Markus Haller, thank you for great conversations full of laughs. To my big sister Sharmeen Jahan, thank you for your immense support, the free lunches and dinners, and for being a quirky big sister. I hope you are proud of your little brother. Lastly, I would like to thank a love story between a cat and a hedgehog for teaching me to stress less and enjoy more.

“You never really understand a person until you consider things from his point of view ... until you climb into his skin and walk around in it”

— Atticus Finch from *To Kill a Mockingbird.*

"You deny them hope. Any man in this world, Atticus, any man who has a head and arms and legs, was born with hope in his heart. "

— Jean-Louise Finch from *Go Set a Watchman.*

These two quotes are for the protectionists, the conservative voters of people like Donald Trump, Marine Le Pen, Nigel Farage, and Winston Peters. For the the Pieter Thiels of society. If by a serendipitous coincidence you come across this thesis, and this page, stop and reflect. Reflect on your actions. Reflect on your reasons. Ask yourself if you have walked in our shoes. Us, who you are happy to de-humanize. Us, who you are happy to call *'bad hombres'*, and the *'undesirables'*. Because we, the 'undesirables', walk in your shoes everyday. Like you, we have loved and have been loved, we have triumphed and we have lost. Like you, we dream and we regret. Like you, we rise in the morning with hope, and close our eyes each night with gratitude. And dream *bigly* of a better tomorrow.

Contents

List of Figures

List of Tables

THE UNIVERSITY OF AUCKLAND
SCHOOL OF GRADUATE STUDIES

Graduate Centre
The ClockTower – East Wing
22 Princes Street, Auckland
Phone: +64 9 373 7599 ext 81321
Fax: +64 9 373 7610
Email: postgraduate@auckland.ac.nz
www.postgrad.auckland.ac.nz

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. Research presented in Chapter 4 Sections 4.3 and 4.4 (pages 97 to 119), and Chapter 6 Section 6.1 (pages 150 to 173) were co-authored paper, titled "MIDAS: Mixed-integer dynamic approximation scheme", which was submitted to the Springer's Mathematical Programming Series A and B.

Nature of contribution by PhD candidate	Preparation of the draft paper; developing the convergence proof for MIDAS and the computational results.
Extent of contribution by PhD candidate (%)	65

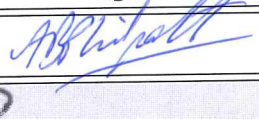
CO-AUTHORS

Name	Nature of Contribution
Professor Andy Philpott	Contribution to the framework for convergence proof, and writing the some of the paper.
Professor Frédéric Bonnans	Editorial work for the proof and the paper.

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
Professor Andy Philpott		27/10/2016
Professor Frédéric Bonnans		27/10/2016

Chapter 1

Introduction

Hydro power is the most common form of renewable energy, with a global production of $16,400$ TWh per year [30]. It not only provides clean, renewable electricity, but it also mitigates the volatility of other renewable generation sources, such as wind and photo-voltaic, by balancing the supply with the load [4]. Furthermore, hydro power provides significant contributions to ensuring the reliability of an electricity grid. Due to the short starting time, hydro power plants are highly effective as a primary and secondary source of reserve energy, as well as an ideal voltage supporter [35]. Such capabilities and benefits of hydro power have made it a proven renewable energy technology that is utilized to produce the majority of the electricity in many countries (see Table 2.1) around the world. In this thesis we study hydroelectricity optimization in two countries: New Zealand and France.

Share of power	**Countries**
100%	Albania, DR of Congo, Mozambique, Nepal, Paraguay, Tajikstan, Zambia.
>90%	Norway.
>80%	Brazil, Ethiopia, Georgia, Kyrgyzstan, Namibia.
>70%	Angola, Columbia, Costa Rica, Ghana, Myanmar, Venezuela.
>60%	Austria, Cameroon, Canada, Congo, Iceland, Latvia, Peru, Tanzania, Togo.
>50%	Croatia, Ecuador, Gabon, DPR of Korea, New Zealand, Switzerland, Uruguay, Zimbabwe.

Table 1.1: List of countries where majority of the electricity is generated by hydropower (source: [4]).

Electricity production in New Zealand is hydro dominated, with up to 55% of the total production being generated by hydro power plants [49]. As listed in Table 2.2, New Zealand does not have nuclear power. The majority of the country's installed capacity is from renewable resources (72%), such as hydro, geothermal, and wind. There is no commercial photo-voltaic capacity in New Zealand. All photo-voltaics are installed locally, and are considered as small-scale distributed generation [49].

Generation Source	Installed Capacity (GW)	Percentage
Hydro	5.34	55%
Gas	1.50	15%
Geothermal	0.97	10%
Wind	0.68	7%
Coal/Gas	0.50	5%
Co-generation Gas	0.33	3%
Diesel	0.16	3%
Other Co-generation	0.24	2%

Table 1.2: Installed capacity of New Zealand electricity recorded as of 2014 (source:[49]).

There are two main hydro schemes in New Zealand, one in the North Island called the Waikato scheme (capacity of 1052 MWs), and one in the South Island called the Waitaki scheme (capacity of 1738 MWs). Figure 2.1 illustrates the topology of each of the respective hydro schemes. The topology of the Waikato scheme is different to the topology of the Waitaki scheme. The Waikato scheme connects eight dams in series along the Waikato River, whereas the Waitaki scheme has two main branches of waterflow that converge to a central branch. As illustrated in a waterflow network diagram on the right of Figure 2.1, lakes Tekapo and Pukaki converge under one branch with lake Ohau, which then connects in series with lakes Benmore, Aviemore and Waitaki.

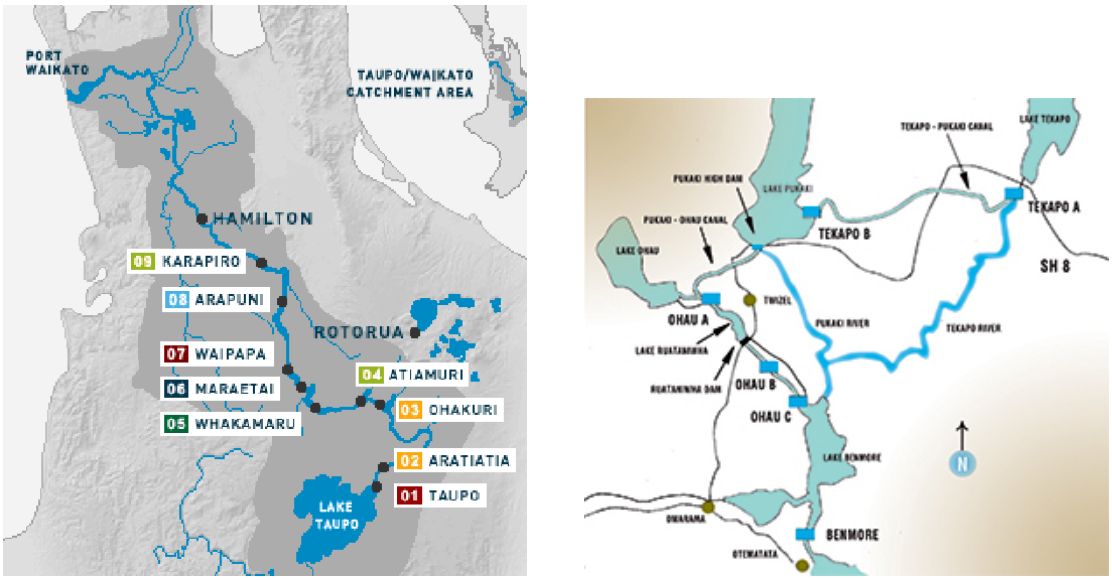


Figure 1.1: The topology of the Waikato (left) and Waitaki (right) hydro scheme in New Zealand (sources: [63**?**]).

While hydro power dominates the New Zealand electricity landscape, nuclear power dominates the French power, which has the second-largest fleet of nuclear stations among the OECD countries [3]. The second dominating generation source is hydropower with 25.2 GW (21%) of the total generation capacity reported in 2014 [3]. Table 2.3 breaks down the generation capacity of France by fuel types.

Generation Source	Installed Capacity (GW)	Percentage
Nuclear	63.1	53%
Hydro	25.2	21%
Wind	8.1	7%
Oil-fired and combustion turbines	7.0	6%
Combined-cycle gas turbines	5.3	4%
Coal	5.0	4%
Photovoltaics	4.3	4%
Other renewables	1.1	1%

Table 1.3: Installed capacity of French electricity recorded as of 2014 (source: [66]).

1.1 Electricity markets in New Zealand and France

Over the past two decades, electricity sectors have been deregulated in places such as New Zealand, parts of North America, and Europe [24]. Many of these sectors have adopted a decentralized, bid-based, competitive, market structure to determine the wholesale price (also known as the spot price) of electricity. A bid-based market operates by having power producers submit offers at regular intervals. These offers are aggregated by the market operator, also known as the ISO or power exchange, who computes the wholesale price of electricity by dispatching the cheapest generators for production until demand is met. The wholesale price, at which the market clears, is the offer of the marginal generator when supply meets demand at the minimal cost of dispatch, while ensuring that the physical and operational constraints of the electricity grid are met [76, 82]. An example of how the wholesale price is computed is illustrated in Figure 2.2 below, where the green spot price is computed to be the point where the supply equals the demand based on the aggregated offers from all producers.

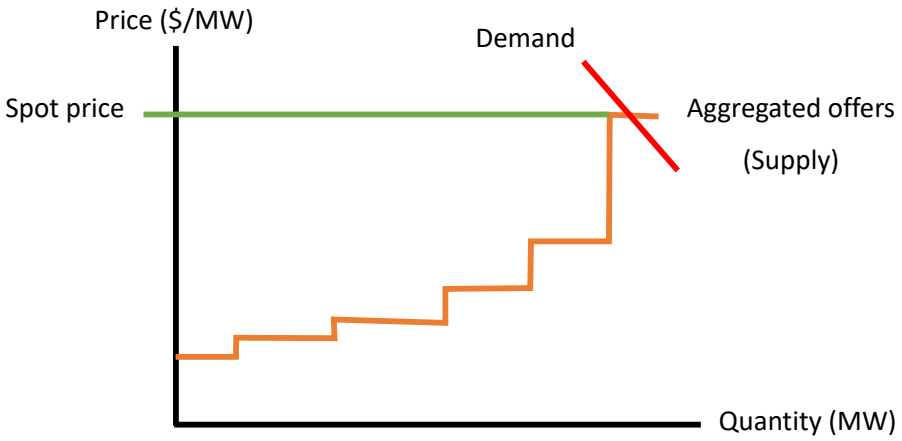


Figure 1.2: Example of market clearing price based on aggregated offers of all participating generators.

The electricity market in New Zealand operates as an intra-day market, where participants provide a five-tranche offerstack half hourly to the ISO for the next 72 trading periods. Using these offers, the ISO computes the provisional spot prices, which are later reconciled based on the actual consumption. In order to help market participants understand their market position, the ISO forecasts spot prices every two hours for each trading period up to 36 hours ahead of time. The ISO also provides five-minute indicative prices. These are calculated at the end of each five-minute period and take into account the changing conditions of the power system. However, the five-minute indicative prices do not coincide with the new offers, but are instead based on the changes to the power system close to real time. On the other hand, the two-hourly price forecasts correspond to new offers as they are in alignment with the offer submission deadlines.

The French electricity market is part of the wider Electricity Regional Initiatives (ERI) initially put into force in 1996 by Directive 96/92/EC [5], which eventually evolved into the Central West European (CWE) market coupling that covered France, Belgium,

Netherlands, Germany, Luxembourg, and Austria [48]. The goal of the ERI is to have a Pan-European electricity market, called the Integrated Electricity Market (IEM), which involves coupling different power markets across Europe via volume and/or price [41]. Figure 2.3 shows the key Electricity Regional Initiatives in Europe.

Figure 1.3: Electricity Regional Initiatives across Europe for the Integrated Electricity Market (source: [41]).

The IEM is divided into several tiers, the first tier is the control zones of the different Transmission System Operators (TSOs) which, in the case of France, is along the national border [48]. Markets within the control zones consist of a futures market, a day-ahead market (DAM), an intra-day market (IDM), and a balancing market [48]. The futures market is a financial market that trades bilateral contracts and over-the-counter contracts for base and peak load. This trading is based on the long-term load forecasts of the control zones [48].

The DAM is run as a day-ahead spot market. Offers are collected from market participants until a set time, and then the market clearing algorithm is run to compute the generation schedules for the next day [48]. Since the DAM is based on the load forecast

for the next day, the actual load conditions can be different from what was predicted. This requires the presence of the IDM and the balancing market [48]. Trading is continuous in the IDM where offers are executed immediately as the appropriate market clearing price becomes available [48]. In the balancing market, individual generators submit two types of offers to the TSO, incremental and decremental bids [79]. An incremental offer, consisting of a single price-quantity pair, represents the price that the generator gets paid by the TSO to increase production by the offered quantity on top of their dispatched schedule [79]. The decremental bid on the other hand, represents the price the generator is willing to compensate the TSO in order to reduce their supply from their dispatched schedule [79]. It is discussed in greater detail in Chapter 4. Figure 2.4 illustrates the chronological operation of each market within a control zone. In Figure 2.4, the futures market is first cleared before $D-1$, where D stands for day of realtime production, followed by DAM a day before the actual production, and then the IDM and the Balancing Market on the day in real time.

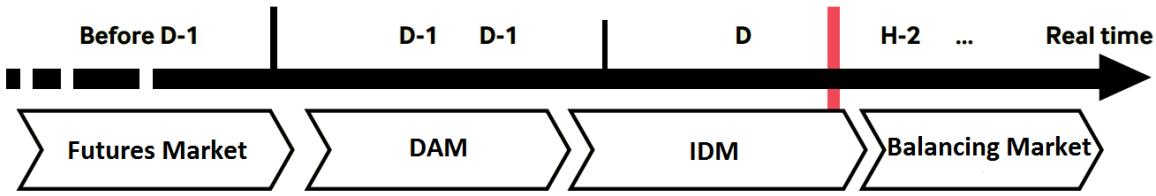


Figure 1.4: Integrated Electricity Market (IEM) Structure over time within a control zone (source: [66]).

1.2 The hydro-bidding problem

Since New Zealand's electricity production is hydro-dominated, NZEM prices are correlated with droughts where high electricity prices are observed during periods of low lake levels. For electricity generation companies, such as Meridian, Genesis and Mercury, this poses a great problem in managing water levels for reservoirs with small

storage. The generation potential of hydro stations depends on the height of the stored water. The water level of reservoirs with small storage capacity is thus heavily dependent on changes in the volume of water. Therefore, hydro producers participating in the NZEM need to manage how to consume the water stock in order to produce electricity, especially when operating hydro schemes under a high risk of drought. Their challenge is to submit bids in the current trading periods while keeping their reservoir levels high for maximum generation in the future.

Unlike New Zealand, where hydro power is used as a major generation source, hydro power in France is predominantly utilized for balancing the supply with the load in real time. This is because nuclear stations are used to meet non fluctuating base load due to their inability to rapidly regulate the power output. Hydro stations, on the other hand, can regulate their generation and have greater flexibility. Therefore, hydro stations owned by generation companies, such as Electricité De France (EDF), are mostly utilized to participate in the intra-day and balancing markets.

A common management approach to producing hydro power in the intra-day markets, both for NZEM and the balancing market in France, is to separate the trading decisions from the production decisions. The usual operation, as illustrated in Figure 2.5, involves the traders first constructing and submitting offers into the intra-day market. After the intra-day market clears, the quantity of power allocated to the hydro producer (i.e. dispatch), is then sent to the hydro-dispatcher. The hydro-dispatcher decides on how to generate the power based on the hydrology of the hydro schemes and future hydrological information such as inflow. The production actions of the hydro-dispatchers set the future generation capacity of the hydro scheme, which is then used by the traders to construct offers for the next period.

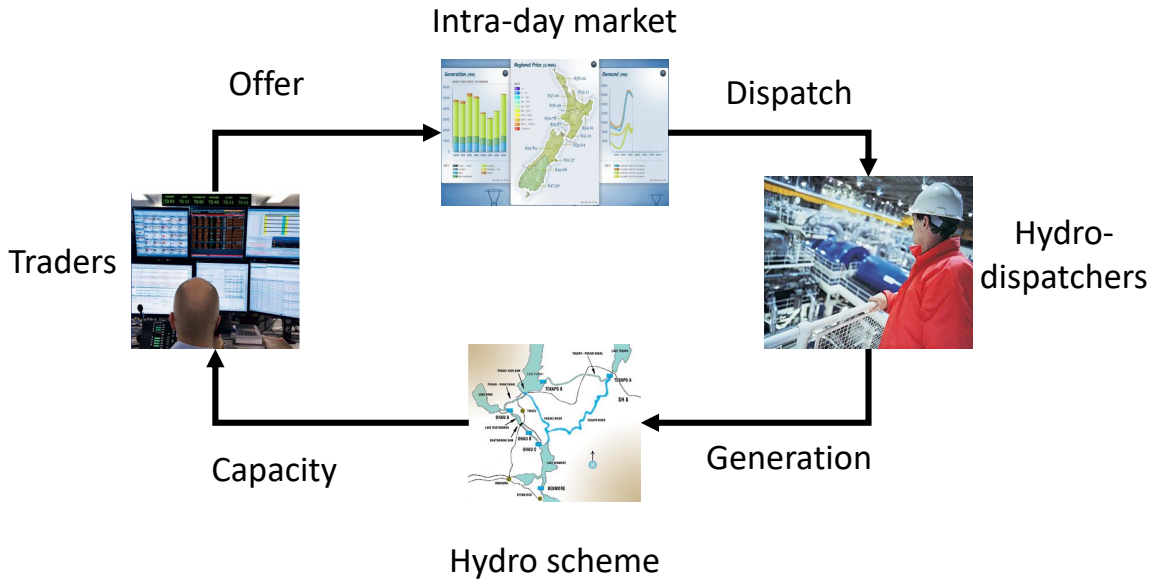


Figure 1.5: Hydro-production in an intra-day market environment.

The major limitation with this hydro management process is that it does not integrate the trading and generation decisions. A hydro producer, employing both the trader and hydro-dispatcher, can only offer quantities of power which they can feasibly generate based on the hydrological state of their hydro scheme. With the current process this is communicated verbally between the trader and the hydro-dispatcher, which might be sub-optimal. Furthermore, this process is mostly asynchronous, where the hydro-dispatcher has to make decisions based on the actions of the trader. This is also sub-optimal because in order to maximize profit, the best strategy would be to manage the hydrology of the river in order to offer energy during high prices, which requires optimizing both the offers and the production as one.

If the hydro producer were to simultaneously optimize their trading and generation decisions in these markets, they face a complex decision problem. Their offers must be feasible in the hydro scheme as well as maximize their profits, while taking into account uncertainty in information such as inflows and the market-clearing price [25, 44, 47,

51]. Furthermore, the stochastic information is observed gradually over time, which prevents them from computing their optimal decisions for all future periods. This is because in each period the hydro producer submits their offer based on the current state of their waterstock and their view of the future prices. Then, the hydro producer observes the spot price, from the clearing of the market, and determines which of their bids was cleared for dispatch. They then generate their dispatched quantity of energy. Afterwards, the hydro producer recomputes the state of their water stock based on their generation and arrival of new inflows to their reservoirs, and begins the process of offering into the next period. This type of decision making framework, as illustrated in Figure 2.6, is classified as a sequential decision-making problem [61, 73].

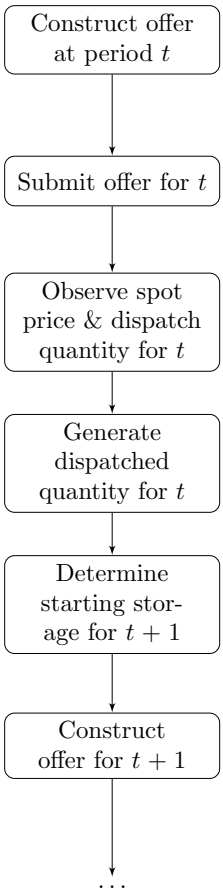


Figure 1.6: The sequential decision making process of a hydro producer bidding in an intra-day market.

In this type of problem, the maximum profit of a hydro producer is intertwined with the intrinsic value of the waterstock. Since hydro producers are able to store water (i.e. energy) in dams and reservoirs, they have the flexibility to choose when and how much to generate. Therefore, they have to make the trade-off between selling water in current periods and selling it in future periods when the wholesale price of electricity might be higher [75]. Such complexity in constructing the optimal offers gives rise to the *hydro-bidding problem*.

The hydro-bidding problem is about computing optimal offer policies in order to maximize the expected profit of a hydroelectric producer participating in these electricity markets. It combines the decision making process of both the trader and the hydro-dispatcher into one stochastic optimization problem. Price-taking hydro-bidding models, which assume that agents do not have market power [75], aim to arbitrage between the different prices, at different time periods, in order to maximize profits. These models are typically formulated as discrete-time, finite horizon, stochastic optimal control problems [14], where in each period the hydro producer is solving the following problem.

Objective: Maximize the expected profit from the current offerstack & the opportunity cost of the remaining water,

Subject to:

1. ensuring feasible generation of offered quantities of power if dispatched,

2. meeting the physical constraints of the hydro scheme, such as storage bounds and water-flow balance,

3. meeting the operation constraints of the turbines such as plant capacity and power generation function,

4. meeting ancillary requirements, such as primary and secondary reserve requirements.

Problem 1: Statement of a general hydro-bidding problem as a stochastic dynamic programming problem.

The hydro-bidding problem, as stated in Problem 1, can have:

1. Large uncertainty space from modelling the stochastic nature of prices, inflow and demand,

2. Non-concave value functions due the presence of integer variables and non-concave power generation functions,

3. Computationally expense to solve in the presence of large hydro schemes with many interconnected reservoirs.

This has attracted the use of various techniques to solve different variants of Problem 1. The key methods that have been used are:

Stochastic Dynamic Programming models (SDP): Problem 1 can be formulated as a finite horizon, stochastic dynamic program [82] using the well known Bellman equation. SDP-based hydro-bidding models have been developed by [51] and [65]. The major issue with SDP is the well-known *curse of dimensionality* [61]. Because SDP uses backward recursion to compute the optimum policy, large problems with many stages quickly become intractable due to the astronomical number of computational steps required to solve these models.

Stochastic Mixed-integer Programming (SMIP): By constructing a finite, discrete approximation of the distribution of the stochastic variables, Problem 1 can be formulated as a complete deterministic equivalent model. In such models all realizations of the stochastic variables across time are represented as a scenario tree based on the discrete approximation. Hydro-bidding models based on this approach have been developed by [1], [33], [44], [32], and [25]. The major limitation with this approach is scalability. Like SDP, it is nearly impossible to construct and solve large problems with many trading periods due to the exponential growth of the scenario tree with an increasing number of trading periods, and scenarios.

Stochastic Dual Dynamic Programming (SDDP): Originally proposed by [55], SDDP is a multistage Benders decomposition based algorithm that approximates the value function through hyper-planes. Models based on SDDP have been developed by [37], [40], [38], [44], [57], and [47]. Assuming independent sampling, SDDP guarantees almost sure convergence [58] when each of the subproblems is convex, but not otherwise. Problem 1 can inherently be non-convex when

modelling the power generation function and unit commitment. Hence, using SDDP to solve Problem 1 when it is non-convex does not guarantee the convergence of SDDP. Recent such as [83] have applied a similar method like SDDP for stochastic mixed-integer programs with binary state variables.

Approximate Dynamic Programming (ADP): A new class of algorithms for dynamic programming can reduce the curse of dimensionality by approximating the value function [61]. Although there are no ADP-based hydro-bidding models, similar models have been developed such as [43], which is based on an ADP algorithm developed by [42]. It solves problems which have non-concave monotonic value functions. However, the challenge with ADP is accurately approximating the value function. Furthermore, these algorithms do not guarantee optimal policies. It is not until one simulates the policy that one can assess its performance. In this regard, SDDP-based models have an advantage because it also computes an upper and lower bound in which the candidate policy is bounded between. This gives the user a good indication of how close the current policy is optimality.

Based on these existing methods, we observe that there are a limited number of algorithms that address the non-convexity of hydro-bidding models that arise from integer variables or nonlinear constraints or objective. In this thesis we study variants of the non-convex hydro-bidding problem (i.e. Problem 1) and develop novel techniques to solve them.

1.3 Our Contributions

Our contributions to the field of multistage, stochastic programming, and hydro-bidding are as follows. First, we develop a hydro-bidding problem for the French balancing market called HERBS (Hydro-Electric Reservoir Bidding System). The purpose of this model is to help hydro producers to construct balancing bids that maximize their profit from the intra-day balancing market, while also minimizing their exposure to penalties for unmet dispatched load from their hydro schemes. HERBS is a two-stage, stochastic, mixed-integer programming (SMIP) model, which computes the optimal incremental offer and decremental bid based on a day-ahead schedule. It is difficult to solve HERBS for large problems due to it being a SMIP. Moreover, HERBS is not convex. This poses a major challenge to using methods like SDDP.

The challenge of solving HERBS has led to the development of a new decomposition method called Mixed Integer Dynamic Approximation Scheme (MIDAS). MIDAS is a sampling algorithm similar to SDDP that solves multistage stochastic programming models where the value function is semicontinuous and nonconcave. MIDAS creates an ε-outer approximation of the value function using step functions. It works similarly to SDDP, where it uses a forward pass to compute new states, and a backward pass to update the value-function approximation in these state values. We prove that MIDAS converges almost surely in a finite number of iterations to a $2T\varepsilon$-optimal first stage decision. We use MIDAS to solve Problem 1, based on the following three variants:

1. **Discrete production:** Solve Problem 1 with integer storage states arising from modelling the power generation function by a finite set of predefined, feasible generation quantities.

2. **Price uncertainty:** Solve Problem 1 with an autoregressive price process.

3. **Headwater effects:** Solve Problem 1 incorporating headwater effects, where

the power generation function is both dependent on the turbine-water flow and the water levels of the upstream reservoir.

All of these model variants are hard to solve by competing approaches. We demonstrate the effectiveness of MIDAS by solving these problems in a finite number of iterations, and constructing near-optimal policies.

Our final contribution is presenting two heuristics that help to improve the computational efficiency of MIDAS. Even though MIDAS is able to solve the non-convex hydro-bidding problems, it becomes slow when scaled to solve large hydro schemes. To address this limitation, we first introduce a heuristic that tightens the value function approximation by removing redundant step functions. We then introduce the second heuristic, which is a sub-problem solving method based on the optimal cuts and the integer L-shaped method of [45]. Applying these heuristics we demonstrate improvements to the MIDAS algorithm in solving these hydro-bidding problems, and extend MIDAS to solve a hydro-bidding problem consisting of 4 reservoirs and 4 stations, with integer state variables, over 4 time periods.

1.4 Thesis structure

The rest of this thesis is structured as follows:

Chapter 3 - Analysis of the hydro-bidding problem: Chapter 3 defines a hydro scheme and how optimizing it is mathematically represented as a multistage stochastic programming problem. A mathematical formulation of Problem 1 is presented as a general hydro-bidding model in Section 3.1. Within this model, various key features are discussed with reference to existing methods and formulations. The hydro-bidding model is then used as a basis to discuss in detail existing decomposition methods in Section 3.2. They are demonstrated using an example hydro-bidding problem to study their performance in Section 3.3. These methods are compared against each other on the optimality of their computed policies, and their computational efficiency when incorporating large hydro schemes.

Chapter 4 - Hydro-bidding in a balancing market: Chapter 4 introduces a hydro-bidding model, called HERBS, for a hydro producer competing in an intra-day balancing market. This hydro-bidding problem differs from Problem 1. In Section 4.3, we present a mathematical formulation of HERBS, which is a two-stage, stochastic, mixed-integer program (SMIP) with incremental and decremental bids. As is the case with SMIPs when extending to multistage problems, it explodes in size due to the exponential growth of the scenario tree with the number of periods. Hence, a rolling horizon solution approach is proposed to solve a multistage version of HERBS in Section 4.4, where the two-stage SMIP is solved iteratively across each time period. Two hydro schemes, owned by EDF, are used as case studies to analyse its performance. HERBS produces sub-optimal offer policies because it assumes observations of all future prices in

the second stage, which is discussed in Section 4.5.

Chapter 5 - Mixed Integer Dynamic Approximation Scheme: Chapter 5 introduces the novel approach to approximating the Bellman function, and the general MIDAS algorithm to solving multistage stochastic programming problems. In Section 5.2, a proof is presented for the convergence of MIDAS to a $2T\varepsilon$-optimal solution to a multistage deterministic optimization problem. Then, in Section 5.3 MIDAS is extended to solve the multistage stochastic optimization problems, where the random variables are modelled as a scenario tree with finite number of nodes. Using this new type of problem, the chapter shows that MIDAS will almost-surely converge to a $2T\varepsilon$-optimal first stage policy. It does this by presenting two versions of the MIDAS algorithm. The first version, called the *Full-tree MIDAS,* visits all the nodes of the scenario tree in each iteration. This version is presented in Section 5.3.1. In the other version of MIDAS, called the *Sampled MIDAS,* it samples the scenario tree similar to SDDP. The result for the Sampled MIDAS algorithm is presented in Section 5.3.2.

Chapter 6 - Solving the hydro-bidding problem with integer state variables: Chapter 6 introduces the hydro-bidding model with discrete production and integer state variables. We illustrate how the value function of this model is non-concave in Section 6.2. We then present a MIDAS-based approach to solving this hydro-bidding model in sections 6.3 and 6.4. We then compare the numerical performance of MIDAS with respect to the SDDP equivalent in Section 6.5. We observe that MIDAS produces better polices than SDDP, with a better approximation of the value function. However, we also observe that MIDAS is computationally expensive with significantly longer solution times compared to SDDP.

Chapter 7 - Solving the hydro-bidding problem with continuous state variables: Chapter 7 studies Problem 1 with an autoregressive price process, and secondly with a power function which incorporates headwater effects. In Section 7.1, we model the price process as a state variable inside the value function in MIDAS. We also develop an SDDP equivalent, in Section 7.1.2, by introducing a cut interpolation technique, which interpolates the value function based on two sets of cuts at two different price nodes. This enables us to represent the autoregressive price process inside SDDP. We then, introduce the third non-convex hydro-bidding model. This model approximates the power generation function with headwater effects as a difference of two quadratic functions linearised using piecewise linear approximations discussed in Section 7.2. For both of these models, the overall hydro-bidding problem is a stochastic mixed-integer program. Both of these models are solved using MIDAS and SDDP. We compare the policies of MIDAS and SDDP in order to analyse how well MIDAS approximates the value function, and if MIDAS constructs policies that are better performing than SDDP.

Chapter 8 - Improving the computation of MIDAS: Chapter 8 discusses some numerical results of the hydro-bidding problem solved both by SDDP and MIDAS. We observe that each of the sub-problems in MIDAS are mixed integer programs (MIP) arising from the use of step functions. The size of these MIP sub-problems increases with each iteration in the MIDAS algorithm. This is due to the addition of extra binary variables when introducing new step functions. As the size of the sub-problem starts to increase it takes longer to solve them, which in turn adds to the computation time. Hence, MIDAS becomes computationally expensive with large data sets as it uses a large number of step functions to approximate the value function. In order to alleviate an increasing

computation time, we introduce two heuristics. The first heuristic is a step function selection scheme, presented in Section 8.1, which tightens the value function approximation by removing redundant step functions. The second heuristic is a sub-problem solver heuristic based on the optimal cuts and the integer L-shaped method of [45], presented in Section 8.2. We apply these heuristics and demonstrate the improvements in the solution time of MIDAS. Using these two heuristics we then solve a hydro scheme consisting of 4 reservoirs and 4 stages.

Chapter 9 **- Concluding remarks:** The final chapter of this thesis reviews the notable findings in Section 9.1, summarizes the major contributions in Section 9.2, and discusses the future research directions in Section 9.3.

Chapter 2

Analysis of the hydro-bidding problem

Hydro stations can be built on a single water source, or can be part of a network of reservoirs and generation stations that are connected by rivers and tributaries. This is referred to as a hydro scheme or a river chain. Based on the topology of the hydro scheme, released water for generation from upstream reservoirs will flow downstream to add to the volume of water in downstream reservoirs. This in turn can be used for generation by downstream power plants. Figure 3.1 illustrates such a hydro scheme. The water discharged from the reservoir of hydro power plant 1 (HPP1) will flow into hydro power plant 2 (HPP2), and then flow into HPP3, being used to generate power in several locations along the chain.

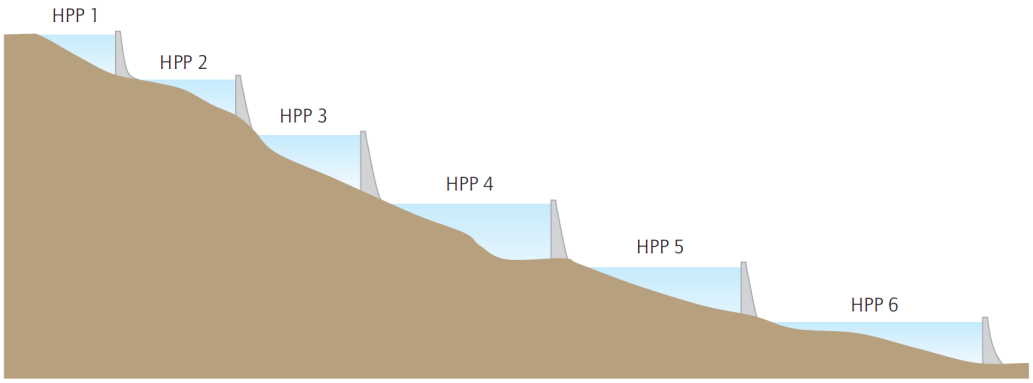


Figure 2.1: Topology of an example hydro scheme with cascading reservoirs and hydropower plants (HPPs) (source: [4]).

From the dammed lakes at these reservoirs and hydro stations, the energy potential of the water, stored at different altitudes, is converted into electricity [35]. A typical hydro station consists of a forebay, a tailrace, a spill gate, and for each turbine a penstock. The forebay houses the water and is synonymous with dammed natural lakes and man-made reservoirs. The penstock is the pipe that provides passage of the water to the turbine. The spill gate provides an alternative passage downstream in order to manage the reservoir storage, especially during situations where the reservoir storage is at maximum capacity and is at risk of overflowing. The tailrace is the basin where the water accumulates at the bottom of the station after travelling through the turbine. An illustration of a cross section of a hydro station is provided in Figure 3.2, which depicts how the stored water travels through the turbine converting its potential energy into kinetic energy, from the rotation of the turbine, and then into electrical energy through a generator.

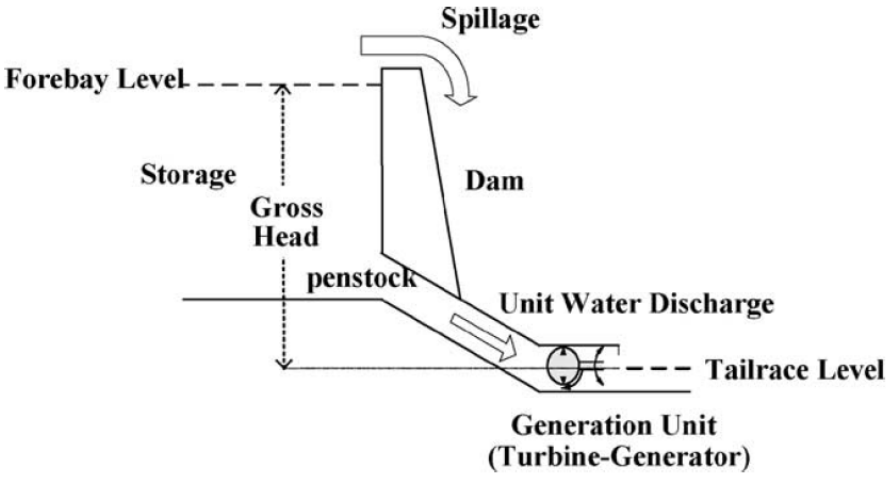


Figure 2.2: Cross section of a hydro station with a single turbine (source: [31]).

2.1 Modeling the hydro-bidding problem

As mentioned in Chapter 2, the hydro-bidding problem is a sequential decision making problem. With appropriate definitions of states and actions, sequential decision making problems can be formulated as discrete-time, finite horizon stochastic program [14]. These stochastic programs can be modeled as a stochastic dynamic program (SDP) using the Bellman Equation [62].

The following assumptions are made for the hydro-bidding models presented in this thesis. These are are the following:

Assumption 1. ***The only market uncertainty in the model is price.***

The founding assumption in this thesis is that hydroproducers cannot influence the market-clearing price. This assumption is valid if the producer does not have a significant portion of the total energy supply [75]. If a producer does have a significant

proportion of the energy supply, then they can alter the market clearing price through their bids. By withholding large quantities of their supply they can potentially increase the spot price by forcing the market to dispatch more expensive generators. Alternatively, they can inject excess capacity at low prices and potentially depress the price by under-bidding their competing generators [75]. The price taking assumption is made so that we can treat the spot price as an exogenous random variable and use models and methods from the stochastic programming literature [51].

Assumption 2. ***Natural inflows between trading periods are deterministic.***

In Model (3.1.1) the two sources of uncertainty are inflow (i.e. natural inflow from rainfall and snow melt) and price. Inflows exhibit seasonal variations throughout the year. Figure 3.3, as illustrated in [64], shows an example of the seasonal variation inflow for lake Benmore in the Waitaki hydro scheme. As illustrated, the highest inflow is during December and January. High inflows occur during these months mainly due to the melting of the snow caps when New Zealand enters the summer season.

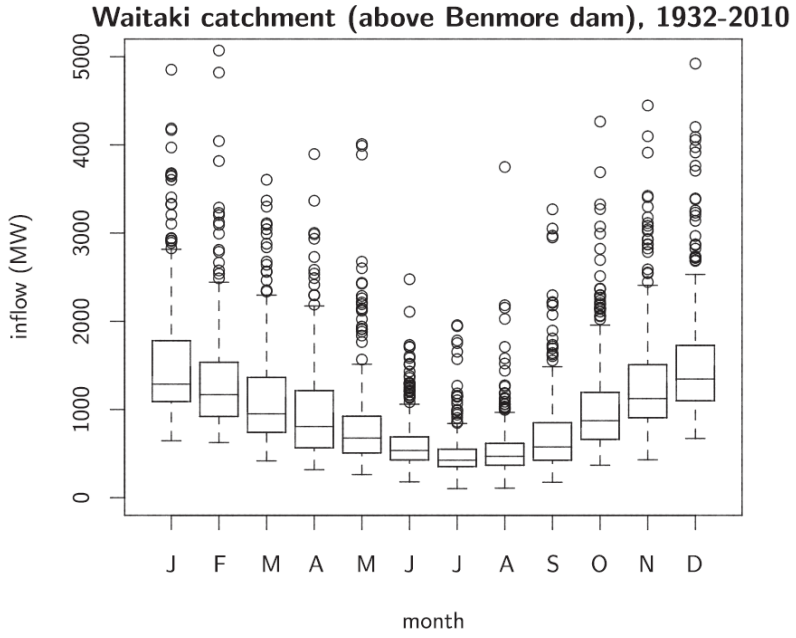


Figure 2.3: Raw inflow data for the waterflow going into lake Benmore in the Waitaki hydro scheme (source: [64]).

Assumption 3. ***There are no delays in flow between upstream and downstream reservoirs.***

Since hydro schemes are a network of lakes interconnected by rivers, they tend to cover large geographical areas. The flow of water between two connected reservoirs can have a delay from a few hours to nearly a day. Water released earlier in the day from upstream reservoirs may not arrive downstream till several hours later. This adds another layer of complexity for hydroproducers, as they have to carefully time their releases upstream so that they arrive downstream at the appropriate time when the prices are high. Flow delays also add another layer of complexity to the hydro-bidding models because the state transition equation, or the water-balance constraint, is now coupled with the storage levels in the previous period as well as the water discharged several periods earlier.

For SDP and SDDP based models, integer flow delays can be modeled by adding dummy nodes between an upstream and a downstream node of a river section with flow delays. The number of dummy nodes equals the number of time periods of flow delay, where each section of river between the dummy nodes is assumed to have unit delay. The SDP-based hydro-bidding model by [65] discusses how this can be implemented to model flow delays.

2.1.1 Hydro-bidding Model Formulation

Model (3.1.1) represents a generic hazard-decision based multistage stochastic program, through the dynamic programming principle, of a hydro-bidding problem for a hydro scheme consisting of a set of reservoirs $\mathcal{R}$, a set of stations $\mathcal{S}$, and an offer curve of M segments or tranches. At the beginning of period t the hydroproducer observes the price P_t. Then based on their starting storage x_t, they will compute their optimal policy, based on the decision variables $(o_{t,j}, u_{t,j}, l_{t,j})$ for each segment j from the set

of all feasible decisions $D_t(x_t)$.

$$\begin{cases} V_{t,i}(x_t) = \sum_{j=1}^{M} \rho_{i,j}(t) \max_{(o_{t,j},u_{t,j},l_{t,j})\in D_t(x_t)} \{r_t(o_{t,j},\pi_j) + V_{t+1,j}\left(f(x_t,u_{t,j},l_{t,j},\omega_t)\right)\} \\ \text{for } x_t \in X, \text{ for } t = 0,\ldots,T-1, \\ \\ V_{T,i}(x_T) = R_i(x_T), \text{ for } i = 1,\ldots,M. \end{cases} \tag{2.1.1}$$

where:

T	=	the number of stage in the planning horizon (i.e. $t = 1, 2, \ldots, T$),
$\mathcal{R}$	=	the set of Reservoir node labels,
$\mathcal{S}$	=	the set of Station node labels,
M	=	the number of tranches in the offer stack,
$D_t(x)$	=	the set of feasible actions $(o_{t,j}, u_{t,j}, l_{t,j})$ for each tranche j given a storage x,
P_t	=	the stochastic market price variable,
X	=	the set of feasible storage levels,
π_j	=	conditionally expected price for tranche j when $P_t \in [p_j, p_{j+1})$,
x_t	=	vector of starting storage levels belonging to set X at the beginning of stage t for each reservoir in the hydro scheme,

$u_{t,j}$	=	decision variable representing a vector of turbine water discharge (cubic meters per second) for each station in the hydro scheme for tranche j among M tranches,
ω_t	=	vector of inflows (cubic meters per second) at period t for each reservoir in the hydro scheme,
$l_{t,j}$	=	decision variable representing a vector of spill flows (cubic meters per second) for period t at each reservoir in the hydro scheme for tranche j among M tranches,
$o_{t,j}$	=	decision variable representing an offer quantity (MW) for tranche j, among M tranches, at time period t for the hydro scheme,
$x_{t+1,j}$	=	vector of storage levels belonging to the set X at the end of period t if tranche j is dispatched,
$r_t(o_{t,j}, \pi_j)$	=	expected profit from clearing tranche i based on the offer quantity $o_{t,i}$ when $P_t \in [p_i, p_{i+1})$,
$f(x_t, u_{t,j}, l_{t,j}, \omega_t)$	=	the state transition function, or the dynamics, is based on the starting storage levels x_t, turbine-water discharge $u_{t,j}$, spill $l_{t,j}$, and a deterministic inflow ω_t in tranche j (see Section 3.1.1.1 for the formulation of the dynamics),
$V_{t+1,j}(f(x_t, u_{t,j}, l_{t,j}, \omega_t))$	=	value function representing the contributions of future stages based on the dynamics for tranche j.

In this model the stochastic variable is price P_t. It is an exogenous random variable. We approximate P_t as a time inhomogeneous Markov chain. At period t, P_t is partitioned by M price intervals with $M+1$ price points as,

$$\{[p_1, p_2), [p_1, p_2), \ldots, [p_{M-1}, p_M), [p_M, p_{M+1})\}. \tag{2.1.2}$$

Each Markov state j represents a $P_t \in [p_j, p_{j+1})$ at time period t. Based on the realization of P_{t-1} in the previous period, P_t can transition to another interval. The probability (i.e. the transition probability) that the P_t is in interval j in the current period t given that it was in interval i in the previous period is

$$\rho_{i,j}(t) = \mathbb{P}\left[P_t \in [p_j, p_{j+1}) \mid P_{t-1} \in [p_i, p_{i+1})\right]. \tag{2.1.3}$$

We can then define the conditionally expected price as,

$$\pi_j = \mathbb{E}\left[P_t \mid P_t \in [p_j, p_{j+1})\right]. \tag{2.1.4}$$

In each time period t the hydroproducer submits an offers curve of M price-quantity pairs, defined as

$$\{(p_1, o_{t,1}), (p_2, o_{t,2}), \ldots, (p_{M-1}, o_{t,M-1}), (p_M, o_{t,M})\}. \tag{2.1.5}$$

After the market clears, he/she observes the market-clearing price P_t. If P_t is observed to be within some interval j then the hydroproducer is dispatched on their offered quantity $o_{t,j}$. In order to ensure that the hydroproducer will definitely be dispatched if P_t is in interval j, he/she offers $o_{t,j}$ amount of power at the lower price point p_j of the interval.

In price-taking hydro-bidding models, the hydroproducer is computing the optimal

quantities $(o_{t,1}, \ldots, o_{t,M})$ which will maximize their total expected profit $\sum_{j=1}^{M} r_t(o_{t,j}, \pi_j)$. For each tranche j, the hydroproducers expected profit is their offer quantity $o_{t,j}$ multiplied by the conditionally expected price π_j and the transition probability$\rho_{i,j}(t)$. This defines their reward function as,

$$r_t(o_{t,j}, \pi_j) = \rho_{i,j}(t)\pi_j o_{t,j}. \tag{2.1.6}$$

For example, consider the following four-tranche offerstack in Figure 3.4. This offerstack consists of four quantities $o_{t,1}$, $o_{t,2}$, $o_{t,3}$, and $o_{t,4}$, with the price p_1, p_2, p_3, and p_4. As illustrated, the observed market-clearing price is within the price interval of the 3rd tranche (i.e. $P_t \in [p_3, p_4)$), which dispatches the quantity $o_{t,3}$. The hydroproducer will be paid by the market at price P_t to generate power $o_{t,3}$. In the hydro-bidding models, the hydroproducer will be paid at π_3 on average.

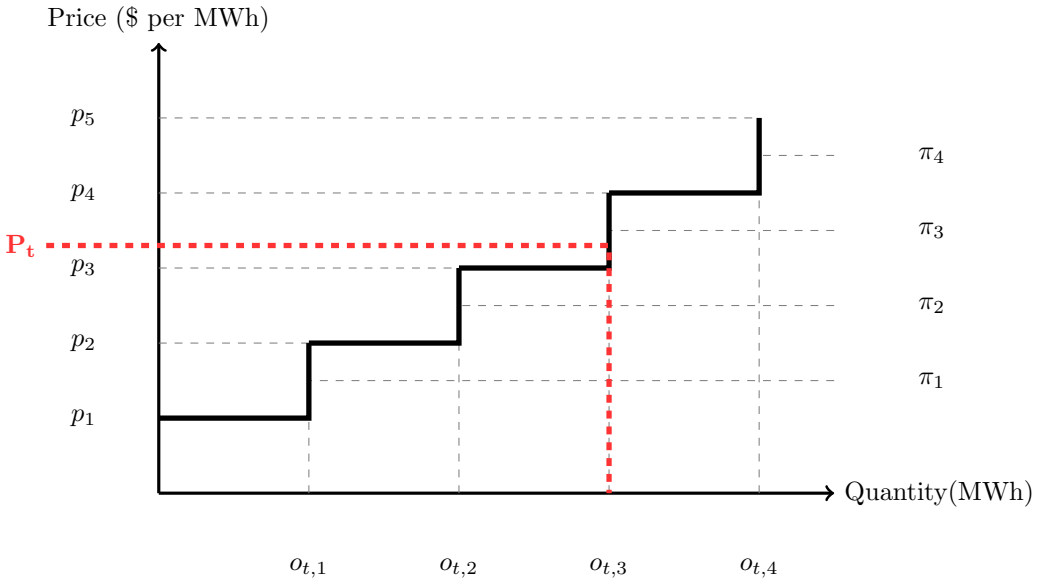


Figure 2.4: Example of an offerstack with four tranches.

The discharge variable $u_{t,j}$, for period t and tranche j, belongs to the feasible set $U(x) = \{u \in \mathbb{R}^{|\mathcal{S}|} : \ u_s \in [\underline{u}_s, \overline{u}_s] \ \forall s \in \mathcal{S}\}$, which is the allowable discharge of water

bounded between $[\underline{u}_s, \overline{u}_s]$ for station s in the hydro scheme. The set $L(x) = \{l \in \mathbb{R}^{|\mathcal{R}|} : 0 \leq l_r \leq \bar{l}_r \ \forall r \in \mathcal{R}\}$, where $l_{t,j} \in L(x)$ is the set of allowable spill bounded between 0 and $\min\{\bar{l}_r, x_r\}$ for each $r \in \mathcal{R}$. This, then makes the set of feasible decisions $D_t(x) = X \times U(x) \times L(x) \times \mathbb{R}^M$.

The state transition function, also known as the state equation, $f(x_t, u_{t,j}, l_{t,j}, \omega_t)$ represents the water-balance constraint, which enforces the conservation of the waterflow across the hydro scheme. It outputs the storage level at period $t+1$ based on the starting storage x_t, the station flows $u_{t,j}$, spill flows $l_{t,j}$, and the inflows ω_t for each tranche j. We refer the reader to Section 3.1.1.1 for a detailed formulation of the water balance constraint.

The hydroproducer has to be able to feasibly generate the quantities $(o_{t,1}, \ldots, o_{t,M})$ declared in their offerstack. In the model the feasible set of offers is represented by the set,

$$\begin{aligned} O(x_t, u_{t,j}, l_{t,j}) = \{ \quad & (o_{t,1}, \ldots, o_{t,M}) \in \mathbb{R}^M : \ o_j = \textstyle\sum_{s\in\mathcal{S}} g_s(x_t, u_{t,j,s}, l_{t,j,r}), \\ & o_j \leq o_{j+1}, \ \text{for } j = 1, \ldots, M\}. \end{aligned} \tag{2.1.7}$$

Monotonicity of the offer stack is enforced through imposing $o_j \leq o_{j+1}$ when $p_j \leq p_{j+1}$. The function $\sum_{s\in\mathcal{S}} g_s(x_t, u_{t,j,s}, l_{t,j,r})$, where $g : \ X \times U(x_t) \to \mathbb{R}$ is the power generation function for station s in the hydro scheme. The generation function is dependent on the initial storage level x_t, the water releases out of each station $u_{t,j,s}$, and the spill flows across the hydro scheme $l_{t,j,r}$. If the market-clearing price is between tranche i and tranche $i+1$, where $P_t \in [p_i, p_{i+1})$, then the hydroproducer will be dispatched at a total quantity of $\sum_{s\in\mathcal{S}} g_s(x_t, u_{t,j,s}, l_{t,j,r})$, which corresponds to the offer quantity $o_{t,j}$.

With sequential decision problems, the actions taken in early stages of the planning horizon will affect circumstances of future decisions [14]. This is represented as the future value function $V_{t+1,i}(f(x_t, u_{t,j}, l_{t,j}, \omega_t))$, where $V_{t+1,i} : X \to \mathbb{R}$, that is dependent

on the dynamic $f(x, u_{t,j}, l_{t,j}, \omega_t)$. It indicates the opportunity cost of future profits based on the remaining waterstock by state transition function $f(x_t, u_{t,j}, l_{t,j}, \omega_t)$ at beginning of period $t+1$. The value function $V_{T,i}(x_T)$ for the end period T, also called the terminal value function, in the model represents the future value of the remaining waterstock beyond the planning horizon for each $i = 1, \ldots, M$. It is introduced to prevent policies which drain the water from all reservoirs by the end of stage T.

Model (3.1.1) can be solved using various techniques in the Stochastic Programming literature [75]. Some of the key methods used to solve these types of models are presented in Section 3.2. However, before progressing to the solution methods, we first discuss in detail the water-balance constraint in Section 3.1.1.1, and then the effects of reservoir storage levels (i.e. headwater effects) on the generation function in Section 3.1.1.2.

2.1.1.1 State equation: the water-balance constraint

As described earlier in the introduction of Chapter 3, hydro schemes can be represented as a network of water flows (refer to the Waitaki hydro scheme in Figure 2.1), where the nodes are the reservoirs and stations, and the arcs are rivers, natural inflows, and tributaries. The state transition function $f(x_t, u_{t,j}, l_{t,j}, \omega_t)$, also commonly referred to as the water-balance constraint [22]. In stochastic programming models, the state transition function sets the storage level for the state variable $x_{t+1,j}$, where $x_{t+1,i} = f(x_t, u_{t,j}, l_{t,j}, \omega_t)$. It is defined as,

$$f(x_t, u_{t,j}, l_{t,j}, \omega_t) = x_t + \delta t \left(A u_{t,j} + B l_{t,j} + \omega_t\right), \; j = 1, \ldots, M. \tag{2.1.8}$$

Here, δt represents the length of each discrete time period, variables $u_{t,j}$ and $l_{t,j}$ the respective station and spill flow, x_t and ω_t the respective storage and inflow, and the

matrices A and B are the arc incidence matrices that represent the topology of the station and spill flows.

The water-balance constraint ensures that the flow of water follows the network topology of the hydro scheme, where the volume of water in a reservoir is conserved between each time period, based on the arrival of water from upstream reservoirs, and the release of water to downstream reservoirs. The state variables x_t and $x_{t+1,j}$ are vectors of reservoir storage levels in cubic meters. The flow variables $u_{t,j}$, $l_{t,j}$, and ω_t (cubic meters per second) represent the station flow, spill flow, and the stochastic inflow for each reservoir along the hydro scheme. They are all multiplied by δt, which represents the number of seconds in each time period t in order to convert them to cubic meters. The matrices A and B are incidence matrices with the respective dimensions $|\mathcal{R}| \times |\mathcal{S}|$ and $|\mathcal{R}| \times |\mathcal{V}|$. The set $\mathcal{V} \subseteq \mathcal{R} \bigcup \mathcal{S}$ represents the set of spillway nodes. It contains both station and reservoir nodes. The matrix A and B represent the topology of the network, by indicating how the water will flow between upstream and downstream reservoirs. Their elements $a_{r,s}$ and $b_{r,s}$ have the following definition,

$$a_{s,r} = \begin{cases} -1 & \text{if reservoir } r \text{ is upstream to station } s, \\ 1 & \text{if reservoir } r \text{ is downstream to station } s, \\ 0 & \text{otherwise,} \end{cases} \tag{2.1.9}$$

$$b_{r,v} = \begin{cases} -1 & \text{if reservoir } r \text{ is upstream to spillway } v, \\ 1 & \text{if reservoir } r \text{ is downstream to spillway } v, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1.10}$$

At the end of period t, for a particular tranche j, the storage level at each reservoir r either experiences a net accumulation or a net reduction in the volume of water.

This due to the arrival of upstream flows and inflows, and the departure of discharged water for generation and spill. This sets the starting storage level $x_{t+1,j}$ for the next period. An example is illustrated in Figure 3.5, where at an arbitrary reservoir r_2 there is a reduction in the volume of water at the beginning of period $t+1$. The total flow arriving at r_2 from upstream reservoir r_1 (i.e. $u_{t,j,r_1} + l_{t,j,r_1} + \omega_{t,r_2}$) is less than the total flow discharged from r_2 (i.e. $u_{t,r_2} + l_{t,r_2}$), which results in a net reduction.

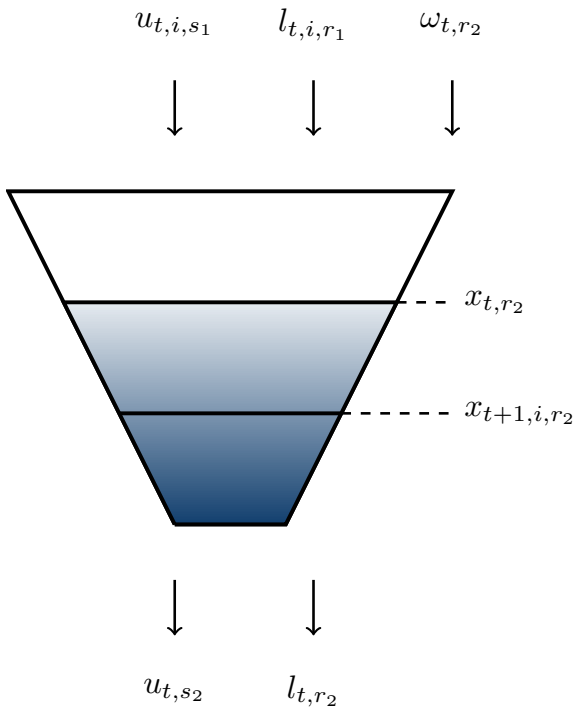


Figure 2.5: Example of the state transition (i.e. water-balance) for reservoir r_2 based on the upstream reservoir r_1, where there is a net outflow of water decreasing the storage level of r_2.

Hydro schemes can span large geographical distances. Therefore, the flow of water can take several hours to travel from upstream to down stream. In this description of the water-balance constraint, flow delays are omitted with the assumption that there are no flow delays.

2.1.1.2 Headwater effects

The power generation function $g_s(x_t, u_{t,j,s}, l_{t,j,v})$ is defined as,

$$g_s(x_t, u_{t,j,s}, l_{t,j,v}) = 9.81 \times 10^{-3} u_{t,j,s} \eta_s h_r(x_{t,r}, u_{t,j,s}, l_{t,j,v}), \tag{2.1.11}$$

where it is a function of the station flow $u_{t,j,s}$, the spill flow $l_{t,j,v}$, the net headwater level $h_r(x_{t,r}, u_{t,j,s}, l_{t,j,v})$ of the connecting reservoir, the efficiency factor η_s, and the constant 9.81×10^{-3} which is the acceleration due to gravity multiplied by the density of water [27]. The headwater level $h_r(x_{t,r}, u_{t,j,s}, l_{t,j,v})$ is a function of the storage $x_{t,r}$, the station flow $u_{t,j,s}$, and the spill flow $l_{t,j,v}$. It is the difference in the water level at the top of the reservoir (h_r^{forebay}) minus the water level at the tailrace ($h_r^{\text{tailrace}}(u_{t,j,s}, l_{t,j,v})$), and a constant headwater losses (h_r^{loss}) in the penstock. It can be defined as,

$$h_r(x_{t,r}, u_{t,j,s}, l_{t,j,v}) = h_r^{\text{forebay}}(x_{t,r}) - h_r^{\text{tailrace}}(u_{t,j,s}, l_{t,j,v}) - h_r^{\text{loss}}. \tag{2.1.12}$$

The efficiency factor η_s is represented by plant capability curves, also known as hill curves, of the turbines in station s. These curves are based on empirical measurements of the actual power [27]. They are constructed by measuring power output under various net headwater and releases (i.e. $\eta_s = \eta_s(h_r, u_{t,j,s})$) [6]. An example of a hill curve is illustrated in Figure 3.6. The most efficient points for generation, points which consume the least volume of water to produce the most energy, are within specific intervals in the hill curves.

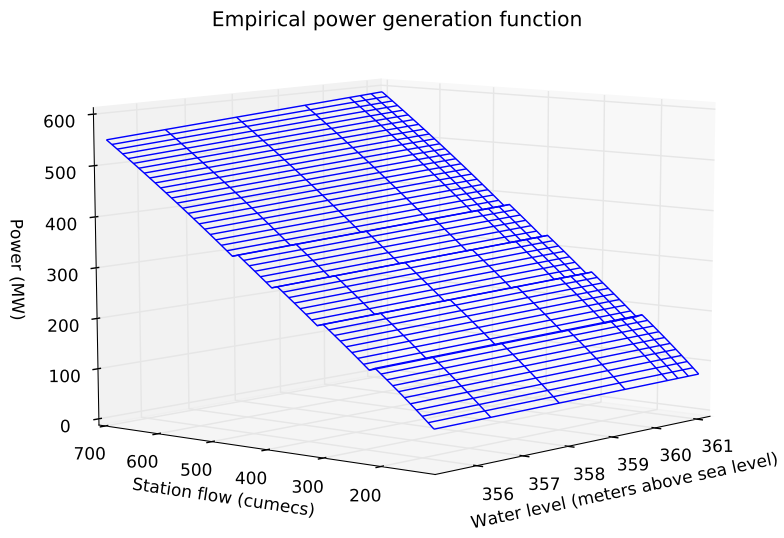


Figure 2.6: Single plant production function for a particular water level (meters above sea level).

In order to model headwater levels in the reservoirs, we define the net head variable $h_{t,r}^{\text{net}} = h_r(x_{t,r}, u_{t,j,s}, l_{t,j,v})$ to represent the headwater function. It is easy to incorporate the water level of the forebay as it is simply the function of reservoir storage level. On the other hand, incorporating the tail water level is complicated because it is a function of station and spill flow. In many cases there is insufficient data to fit empirical functions which model the headwater functions of forebays and tailraces. One assumption that we can make to simplify modeling the tailrace and the forebay is to take the net head from headwater level at the minimum reservoir storage level. Then, $h_{t,r}^{\text{net}} = h_r^{\text{forebay}}(x_{t,r}) - h_r^{\min}$ becomes a function of only reservoir volume. There are numerous ways to represent the function $h_r^{\text{forebay}}(x_{t,r})$. The easiest is dividing it with an average surface area constant K to get $h_r^{\text{forebay}}(x_{t,r}) = \dfrac{x_{t,r}}{K}$, and net head defined as,

$$h_{t,r}^{\text{net}} = \frac{x_{t,r}}{K} - h_r^{\min}. \tag{2.1.13}$$

This assumes that the reservoir basin is cylindrical in shape where the volume increases linearly with net head. Other methods involve using empirically-derived functions based on flow (converted into volume) and the water level of reservoirs (converted into net head) [6]. These empirical functions usually involve polynomials to reflect non linear reservoir basin shapes like cone shapes.

If the changes in net headwater $h_{t,r}^{\text{net}}$ have very little impact on the power generation, then it is safe to ignore the net head and define g_s as a function of $u_{t,j,s}$. Models such as [19, 29, 56, 67, 69] have taken this assumption for reservoirs with a large forebays. Other models such as [19, 29] approximate $g_s(x_t, u_{t,j,s}, l_{t,j,v})$ as piecewise linear functions, while [67, 69] approximates them as a quadratic concave function of $u_{t,j,s}$. For head dependent reservoirs there are several approaches that can be used to approximate the power generation function. The aforementioned piecewise functions can be extended for discrete levels of net head, as is the approach taken by [17, 20, 74], where a single piecewise function is constructed for each given net head value. In all of these approximations, additional integer variables are introduced in order to model the discrete head levels in each reservoir, which pose computational challenges when incorporating them into stochastic programming models.

A more efficient approach would be to consider head level as a continuous variable. The piecewise functions, indexed by discrete head levels, can be extended by interpolating between adjacent head levels, as modeled in [15, 34], which allows $h_{t,r}^{\text{net}}$ to be a continuous variable. In [10], the power generation function is a linear function of flow multiplied by a quadratic function of the net head. It is defined as,

$$g_s(h_{t,r}^{\text{net}}, u_{t,j,s}) = k u_{t,j,s}(\alpha(h_{t,r}^{\text{net}})^2 + \beta h_{t,r}^{\text{net}} + \gamma), \tag{2.1.14}$$

where α, β, and γ are empirically derived model parameters.

A similar approach is taken by [16], where the power generation function is approxi-

mated using polynomials of order 2 and 4 that are based on both the flow and head level. An alternative approximation, one which does not require integer variables nor uses nonlinear functions, is used to represent the power generation function as the convex hull of net head and flow [13]. The benefit of using this approximation is that it does not require the use of additional integer variables and thus improves the computational efficiency of the model. However, it does convexify the hill curves, which as illustrated already, are non-concave in nature. Furthermore, it is impossible to represent minimum flow conditions, where the flow has to be above a minimum threshold in order to generate non-zero power.

Based on our chosen approximation of the headwater function, we would need to include the defintion of net head inside the model. For example, if we take into account a cylindrical reservoir then $h_{t,r}^{\text{net}} = \frac{x_{t,r}}{K} - h_r^{\min}$, and add to Model (3.1.1) along with the generation function defined in Equation (3.1.11).

2.2 Methods of solving the hydro-bidding problem

There are several existing methods that solve Model (3.1.1). The most common methods are Stochastic Dynamic Programming (SDP), Stochastic Mixed Integer Programming (SMIP), Approximate Dynamic Programming (ADP), and Stochastic Dual Dynamic Programming (SDDP). In this section we present some of these methods of solving Model (3.1.1) and in Section 3.3 we compare their performance.

2.2.1 Stochastic Dynamic Programming based hydro-bidding model

Model (3.1.1) can be solved as a Stochastic Dynamic Program (SDP) using backward recursion [62, 73]. However, it requires a few adjustments. The first is to discretize

the sets $D_t(x)$, X, $U(x)$,and $L(x)$. In addition to discretizing the sets, we introduce the parameter $x_{t,i}$ for period t and Markov state i in order to store the state value (i.e. the storage level) [61]. The state variable stores the transitioned state either as an array or lookup table. Lastly, we define the parameter $\hat{V}_{t+1,j}(f(x_{t,i}, u_{t,j}, l_{t,j}, \omega_t))$. The approximate value function represents the value of $V_{t+1,j}(f(x_{t,i}, u_{t,j}, l_{t,j}, \omega_t))$ for each $x_{t,i} \in X$. In reality, $\hat{V}_{t+1,j}(f(x_{t,i}, u_{t,j}, l_{t,j}, \omega_t))$ is represented as an array or a lookup table, similar to $x_{t,i}$. This allows us to then define Model (3.2.1) for use in the backward recursion algorithm (see Algorithm 1). Solving Model (3.2.1) gives an optimal policy for every $x_{t,i} \in X$ for $t = 1, 2, \ldots, T$ and $i = 1, \ldots, M$ [61].

$$\begin{cases} V_{t,i}(x_t) = \sum_{j=1}^{M} \rho_{i,j}(t) \max\limits_{(o_{t,j}, u_{t,j}, l_{t,j}) \in D_t(x_t)} \left\{ r_t(o_{t,j}, \pi_j) + \hat{V}_{t+1,j}\left(f(x_t, u_{t,j}, l_{t,j}, \omega_t)\right) \right\} \\ \text{for } x_t \in X, \text{ for } t = 0, \ldots, T-1, \\ \\ V_{T,i}(x_T) = R_i(x_T), \text{ for } i = 1, \ldots, M. \end{cases} \tag{2.2.1}$$

Backward recursion involves iterating backwards in time and Markov states, and computing $V_{t,i}(x_{t,i})$ across all values of $x_{t,i} \in X$ for $t = 1, 2, \ldots, T$ and $i = 1, \ldots, M$. The optimal value function $V_{t+1,j}\left(f(x_{t,i}, u_{t,j}, l_{t,j}, \omega_t)\right)$ is stored by the parameter representing the future value function $\hat{V}_{t+1,j}(f(x_{t,i}, u_{t,j}, l_{t,j}, \omega_t))$. Algorithm 1 provides the pseudocode for computing the optimum policy. It enumerates across all the storage levels, station flows, offer quantities, and price states in order to compute the optimum value function $V_{t,i}(x)$ across each the planning horizon. The computed optimum policy of Model (3.2.1) is in the form of a decision tree. Once we observe which interval P_t is in and x_t, we can use the information to determine the optimal quantity $o_{t,j}$, station flow $u_{t,j}$, and spill flow $l_{t,j}$ for each Markov state j.

Algorithm 1 Backward recursion algorithm for solving Model (3.2.1).

1. Intialize $\mathcal{R}$, $\mathcal{S}$, M, X, L, $U(x)$, $D(x)$, $\rho_{i,j}(t)$, $\pi_{t,j}$, $R(x_{T,j})$.
2. Initialize $\hat{V}_{t,j}(x_{t,j}) = 0$ for $t = 1, \ldots, T$, for $j = 1, \ldots, M$ and across all $x_{t,j} \in X$.
3. For $t = T, \ldots, 2, 1$, $i = 1, 2, \ldots, M$ and $\forall x_{t,i} \in X$ do the following:
   (a) for $j = 1, \ldots, M$ and $(l_{t,j}, u_{t,j}, o_{t,j}) \in D_t(x_{t,i})$,
      i. compute $x_{t+1,j} = f(x_{t,i}, u_{t,j}, l_{t,j}, \omega_t)$,
      ii. compute $V_{t,i}(x_{t,i})$ in (3.2.1)
      iii. if $\hat{V}_{t,i}(x_{t,i}) < V_{t,i}(x_{t,i})$ then set $\hat{V}_{t,i}(x_{t,i}) =: V_{t,i}(x_{t,i})$

Like all Stochastic Dynamic Programming models, this hydro-bidding model runs into the curse of dimensionality. The computational time is heavily dependent on the size of X and $U(x)$ [62]. For instance, $|X| = K^{|\mathcal{R}|}$ where $|\mathcal{R}|$ is the number of reservoirs, and K is the number of storage levels for each reservoir. The size of X grows exponentially with increasing number of reservoirs (i.e. the dimension of x). Similarly, for the set of feasible controls $U(x)$ its cardinality is equal to $G^{|\mathcal{S}|}$, which is the number of feasible generation pairs G raised to the power $|\mathcal{S}|$. And like X, $U(x)$ exhibits the same exponential growth in size with the number of stations $|\mathcal{S}|$. Iterating over these sets in Algorithm 1 quickly becomes intractable when dealing with large hydro schemes that consist of numerous reservoirs and stations. For this reason, Model (3.2.1) is impractical for solving hydro-bidding problems with many reservoirs and stations.

2.2.2 Stochastic MIP based hydro-bidding model

As the Markov price process is discrete and finite, it can be represented by a scenario tree with a set $\mathcal{N}$ of nodes, and set $\mathcal{L}$ of leaf nodes with the following parameters:

$$
\begin{aligned}
d(n) &= \text{the depth of node } n \text{ with } 0 \leq d(n) \leq T, \\
n^- &= \text{the unique predecessor node for node } n \text{ with } |n^-| = 1, \\
n^+ &= \text{the set of child nodes for node } n \text{ with } |n^+| = M, \\
\rho_n &= \mathbb{P}\left[P_{d(n)} \in [p_j, p_{j+1})\right] \text{ for } 1 \leq j \leq M, \\
\pi_n &= \mathbb{E}\left[P_{d(n)} \mid P_{d(n)} \in [p_j, p_{j+1})\right] \text{ for } 1 \leq j \leq M.
\end{aligned}
$$

The scenario tree represents the exhaustive outcomes of the each Markov state across the planning horizon. Figure 3.7 illustrates an example of a scenario tree with two price segments ($M = 2$). Based on the π_n at each node n, the hydroproducer computes the offer curve for each node $m \in n^+$.

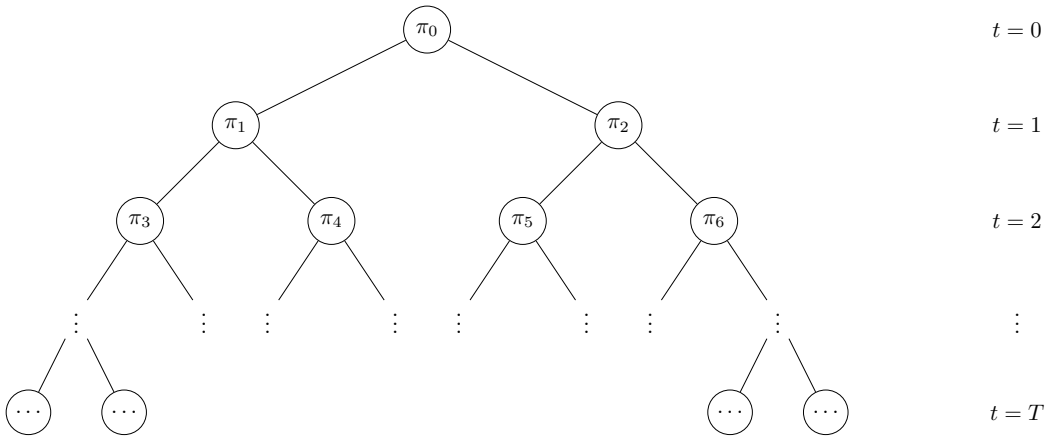


Figure 2.7: Example of a scenario tree with two price states.

A deterministic mixed-integer programming model (MIP) of Model (3.1.1) can be constructed using a scenario tree. Model (3.2.2) represents the stochastic mixed-integer linear program (SMILP) for the scenario tree. The objective function $V_{\mathcal{N}}(\tilde{x}, \tilde{\pi})$ represents the total expected profit across the scenario tree. It is dependent on an initial storage $\tilde{x}$ and conditionally expected price $\tilde{\pi}$ at the root node. The long-term value of water is defined by the terminal value function $R(x_k)$ for each leaf node $k \in \mathcal{L}$. In the objective $V_{\mathcal{N}}$, we sum across all the nodes $n \in \mathcal{N}$ with conditionally expected

profit $\rho_n \pi_n o_n$ at the node. Constraint (3.2.2a) represents the transition equation between node n and its child nodes $m \in n^+$. It is the water-balance constraint discussed in Section 3.1.1.1. Constraint (3.2.2b) defines the offer quantity variable o_n for each node $n \in \mathcal{N}$. It represents the total generation across the hydro scheme. The function $g_s(x_{n,r}, u_{m,s}, l_{m,r})$ represents the generation function for station s based on the station flow $u_{m,s}$, the spill flow $l_{m,r}$ at each child node m, and the storage $x_{n,r}$. Note that this function is represented as an arbitrary function because the power generation function of a hydro station can be modeled using any of the techniques mentioned in Section 3.1. Lastly, constraint (3.2.2c) enforces the monotonic condition of the offer-stack where each offer is increasing with price, while constraints (3.2.2d) to (3.2.2i) represent the station flow, spill flow and the reservoir storage bounds, and initializes the root node.

$$V_{\mathcal{N}}(\tilde{x}, \tilde{\pi}) = \quad \max \sum_{n \in \mathcal{N} \setminus \mathcal{L}} \rho_n \pi_n o_n + \sum_{k \in \mathcal{L}} \rho_k R(x_k) \tag{2.2.2}$$

subject to:

$$x_{m,r} = x_{n,r} + \delta t\,(A_r \cdot u_m + B_r \cdot l_m + \omega_r) \quad \text{for } m \in n^+, \quad \text{for r} \in \mathcal{R}, \quad \text{for } n \in \mathcal{N}, \tag{2.2.2a}$$

$$o_n = \sum_{s \in S} g_s(x_{n,r}, u_{n,s}, l_{n,r}) \quad \text{for r} \in \mathcal{R}, \quad \text{for } n \in \mathcal{N}, \tag{2.2.2b}$$

$$o_m \leq o_{m+1} \quad \text{for } m \in n^+, \quad \text{for } n \in \mathcal{N}, \tag{2.2.2c}$$

$$u_{n,r} \in [\underline{u}_s, \overline{u}_s] \quad \text{for s} \in \mathcal{S}, \quad \text{for } n \in \mathcal{N}, \tag{2.2.2d}$$

$$l_{n,r} \in \left[0, \min\left\{x_{n,r}, \bar{l}_r\right\}\right] \quad \text{for r} \in \mathcal{R}, \quad \text{for } n \in \mathcal{N}, \tag{2.2.2e}$$

$$x_{n,r} \in [\underline{x}_r, \overline{x}_r] \quad \text{for } r \in \mathcal{R}, \quad \text{for } n \in \mathcal{N}. \tag{2.2.2f}$$

(2.2.2g)

$$\pi_0 = \tilde{\pi}, \tag{2.2.2h}$$

$$x_0 = \tilde{x}. \tag{2.2.2i}$$

Solving Model (3.2.2) will output an optimal policy in the form of a decision tree. Based on the observed price at a particular node n, the hydroproducer will be able to determine their optimal offer curve for the next period by aggregating the offers of the child node $m \in n^+$, and then by observing the spot price traverse to the next node in

the decision tree and repeating the same process.

Model (3.2.2) can include additional features such as unit commitment, reserve energy allocation, non-convex generation functions and headwater effects. Adding such features to the model introduces integer variables, thus making Model (3.2.2) a mixed integer program. Scaling such a model for large hydro schemes, across many price states, and time periods quickly becomes computationally expensive to solve or even intractable due to the exponential growth in the number of variables and constraints with an expanding planning horizon [71, 73]. This is discussed in Section 3.3, which compares this model with the SDDP model presented in the Section 3.2.3 next.

2.2.3 SDDP based hydro-bidding model

Using an algorithmic approach similar to SDP and the scenario tree of the SMIP model, the hydro-bidding model can be attacked using a sampling-based method, such as Stochastic Dual Dynamic Programming which was originally proposed in [55]. When the value functions are known to be convex (if minimizing) or concave (if maximizing) then they can be approximated by hyper planes or cutting planes. This is what has popularized the SDDP algorithm. The algorithm creates a sequence of cutting-plane outer approximations to the value function at each stage of the Bellman Equation (see Model (3.1.1)) by evaluating independent sampled sequences of random outcomes from the scenario tree. This creates an upper bound, which through iterative sampling and adding cutting planes improves the approximation and reduces the upper bound. SDDP simulates the policy, that can be obtained from the iterative approximations, to compute a lower-bound. The optimal policy is bounded between the upper and lower bound. As the algorithm progresses the upper bound and the lower bound starts to converge ensuring a good policy. Variations of this idea have been proposed by [21, 28, 37, 39].

Since SDDP marries the scenario tree representation of the stochastic variables, and dynamic programming, it can sequentially sample the scenario tree without complete representation [72]. Furthermore, since it approximates the value function using Benders cuts the sub-problems, Model (3.1.1) becomes a linear programming problem, instead of a SDP, enabling it to solve much larger problems. The use of the Benders cuts allows SDDP to exploit the natural convexity of the value function in order to converge towards the optimum policy [72]. Due to these benefits, SDDP has been used to solve large convex hydro-scheduling, and hydro-bidding problems under uncertainty, with models developed by [37], [38], [46],[59], [18], and [57].

The first formal proof of almost-sure convergence for SDDP-based algorithms was provided by [21] for their CUPPS algorithm. Their proof however relied on a key unstated assumption identified by [58], who provided a new proof of almost-sure convergence for problems with polyhedral value functions without requiring this assumption. Another almost-sure convergence for general convex value functions was recently published in [36].

The stochastic dynamic programming model, Model (3.2.1) in Section 3.1.1, can be re-formulated into $T \cdot M$ linear programs each with the formulation defined by Model (3.2.4). In Model (3.2.4), the value function variable $\hat{V}_{t+1,j} \in \mathbb{R}$ is represented by a set of $K_{t+1,j}$ cutting planes for period $t+1$ and price state j. Each cut is described by the gradient parameter $\theta_{t+1,j,c} \in \mathbb{R}^{|\mathcal{R}|}$ which is calculated from the dual variable values of the water balance constraints (see Constraint (3.2.4a)), the parameter $\beta_{t+1,j,c} \in \mathbb{R}$ which is the intercept in the $\hat{V}_{t+1,j}$ dimension, and $\alpha_{t+1,j,c} \in X$ is the initial storage level for that cut. Then, a single cut can be defined as,

$$\hat{V}_{t+1,j} \leq \beta_{t,j,c} + \theta_{j,c}^{\top}(x_{t+1,j} - \alpha_{j,c}). \tag{2.2.3}$$

$$V_{t,i}(x) = \max \sum_{j=1}^{M} \rho_{i,j}(t) \left[\pi_{t,j} o_{t,j} + \hat{V}_{t+1,j}\right] \tag{2.2.4}$$

subject to:

$$x_{t+1,i,r} = x_{t,r} + \delta t \left(A_r^\top u_{t,i} + B_r^\top l_{t,i} + \omega_{t,r}\right) \quad \text{for all } j = 1, \ldots, M, \text{ and } r \in \mathcal{R}, \tag{2.2.4a}$$

$$x_{t,j} = x^0 \text{ for } j = 1, \ldots, M, \qquad [\theta_{j,r}], \tag{2.2.4b}$$

$$o_{t,j} = \sum_{s \in \mathcal{S}} g_s(x_{t,r}, u_{t,j,s}, l_{t,i,r}) \quad \text{for } j = 1, \ldots, M, \tag{2.2.4c}$$

$$o_{t,j} \leq o_{t,j+1} \quad \text{for } j = 1, \ldots, M-1, \tag{2.2.4d}$$

$$u_{t,j,s} \in [\underline{u}_s, \overline{u}_s] \quad \text{for } j = 1, \ldots, M, \text{ and } s \in \mathcal{S}, \tag{2.2.4e}$$

$$x_{t+1,j,r} \in [\underline{x}_r, \overline{x}_r] \quad \text{for } j = 1, \ldots M, \text{ and } r \in \mathcal{R}, \tag{2.2.4f}$$

(2.2.4g)

$$\hat{V}_{t+1,j} \leq \beta_{t+1,j,c} + \hat{\theta}_{t+1,j,c}^\top (x_{t+1,j} - \alpha_{t+1,j,c}) \quad \text{for } j = 1, \ldots M, \text{ and } c = 1, \ldots, K_{t+1,j}. \tag{2.2.4h}$$

Algorithm 2 describes the key steps for computing the optimal policy of the SDDP based hydro-bidding model. Like the standard SDDP algorithm, Algorithm 2 has four main phases which are the sampling phase, the forward pass, the convergence test, and the backward pass. Special attention should be focused towards Step 6(a)ii in the

backward pass, which outlines how the individual cuts are computed. The gradient of each of the cuts is computed using the dual variable values $\theta_{j,r}$ for each reservoir r and price state j. For $r \in \mathcal{R}$, the r'th component of $\hat{\theta}_{t+1,i,K_{t+1,i}+1}$ is $\sum_{j=1}^{M} \rho_{i,j}(t)\theta_{j,r}$ for $r \in R$. The parameter $\beta_{t,j,K_{t,j}+1} = V_{t,j}(x_t)$ is the objective function value of the sub-problem, and $\alpha_{t,j,K_{t,j}+1} = x_t$ is the state trajectory generated in the forward pass.

Algorithm 2 Solving the hydro-bidding problem using the SDDP algorithm.

---

1. Initialize $\mathcal{R}$, $\mathcal{S}$, M, $[\underline{x}, \overline{x}]$, $[\underline{u}, \overline{u}]$, $\rho_{i,j}(t)$, $\pi_{t,j}$, A, and B.

2. Set $K_{t,i} = 1$ and $\hat{V}_{t,i} \leq \overline{V}_{t,i}$ for $t = 1, \ldots, T$ for $j = 1, \ldots, M$, and $\hat{x}_0 = \tilde{x}$.

3. **Sampling:**

   (a) Sample price scenario $\{\hat{\pi}_1, \ldots, \hat{\pi}_T\}$ and its corresponding state index $\{\hat{i}_1, \ldots, \hat{i}_T\}$.

4. **Forward pass:**

   (a) For $t = 1, \ldots, T$ do,

      i. solve $V_{t,i}(\hat{x}_t)$ (Model (3.2.4)) and let $\hat{x}_{t+1} = x_{t+1,\hat{i}_{t+1}}$.

5. **Convergence test (at 95% confidence level):**

   (a) Let $\hat{z}^{\text{low}} = [0 \text{ for } n = 1, \ldots, N]$.

   (b) Solve $V_{1,\tilde{i}}(\tilde{x})$ (Model (3.2.4)) and let $z^{\text{up}} = V_{1,\tilde{i}}(\tilde{x})$.

   (c) For $n = 1, \ldots, N$, do the following:

      i. sample price scenario $\{\hat{\pi}_1, \ldots, \hat{\pi}_T\}$ and its corresponding state index $\{\hat{i}_1, \ldots, \hat{i}_T\}$,

      ii. for $t = 1, \ldots, T$ do,

         A. solve $V_{t,i}(\hat{x}_t)$ (Model (3.2.4)) and let $\hat{x}_{t+1} = x_{t+1,\hat{i}_{t+1}}$,

B. compute $\hat{z}_n^{\text{low}} = \hat{z}_n^{\text{low}} + \pi_{t,\hat{i}_{t+1}} q_{t,\hat{i}_{t+1}}$.

(d) Calculate $z^{\text{low}} = \frac{1}{N}\sum_{n=1}^{N} \hat{z}_n^{\text{low}}$ and $\sigma^{\text{low}} = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(z^{\text{low}} - \hat{z}_n^{\text{low}})^2}$.

(e) If $z^{\text{low}} - \frac{1.96}{\sqrt{N}}\sigma^{\text{low}} \leq z^{\text{up}} \leq z^{\text{low}} + \frac{1.96}{\sqrt{N}}\sigma^{\text{low}}$ then terminate.

6. **Backward pass:**

(a) For $t = T, \ldots, 2$ and $i = 1, \ldots, M$ do,

i. solve $V_{t,i}(\hat{x}_t)$ (Model (3.2.4)).

ii. let $\beta_{t,i,K_{t,i}+1} = V_{t,i}(\hat{x}_t)$, $\alpha_{t,i,K_{t,i}+1} =: \hat{x}_t$ and $\hat{\theta}_{t,i,K_{t,i}+1,r} = \sum_{j=1}^{M} \rho_{i,j}(t)\theta_{j,r}$ for $r \in \mathcal{R}$,

iii. set $K_{t,i} =: K_{t,i} + 1$

iv. go to Phase 3.

Unlike the previous versions of the hydro-bidding model, the optimal policy computed by SDDP is not a decision tree but is a set of cuts that approximate the value function. In order to compute the optimal offerstack, one simply has to solve the respective sub-problem based on the observed price and storage level. A major advantage of solving the hydro-bidding problem using SDDP is that the quality of the policy can be reviewed during its execution. In each iteration the policy is bounded between the lower and the upper bound, and since over time these two bounds get closer together, one can assess how close the current policy is to the optimum policy. Using this information, the algorithm can be terminated when the user is happy with the quality of the policy. However, [72] states that SDDP could terminate early when there is a large variation in the lower bound estimate. This can be remedied by increasing the sample size, and running SDDP for a minimum number of iterations before carrying out the convergence test.

The SDDP-based model is particularly effective when it comes to hydro-bidding. As mentioned earlier, the state variable $x_{t+1,j}$ in the hydro-bidding problem represents the waterstock of the individual reservoirs, and are coupled through the water-balance constraint (Constraint (3.2.4a)). This restricts the state variable to specific regions of the state space X. SDDP benefits from introducing more cuts to accurately approximate the value functions within these regions while having a rougher approximation in the other regions.

Computational improvements can be made to Algorithm 2. The backward and forward passes can be executed in parallel for Algorithm 2 when the price process is stage-wise independent. By running several scenarios in parallel at each iteration, multiple cuts can be generated to further improve the value function approximation [26]. In addition to parallelization of the forward and backward passes, cut selection schemes can be applied in order to reduce the size of the sub-problems [26]. With every iteration, each sub-problem is adding a cut, which is adding a constraint. This can slow down the algorithm in later stages of its execution. By removing cuts that are dominated, one can still retain the current accuracy of the value function approximation but with fewer constraints and a smaller sub-problem. This reduces the time it takes to solve the sub-problem.

2.3 Comparing hydro-bidding models

In order to illustrate how these models perform and scale with respect to increases in the dimension of the state variable and the number of stages, we compare SDDP and the scenario tree based hydro-bidding models. We assess their performance in three key areas:

1. The quality of their policies for the candidate problems;

2. the quality of their value function approximation;

3. the computational efficiency with respect to the increase in the dimension of the state space and the number of stages.

We construct hydro schemes of various sizes using the following elementary reservoir and station. This elementary reservoir has a storage capacity of 200 cubic meters. Its respective station has a turbine flow capacity of 70 cubic meters, and a generation capacity of 70 MWh. The turbine curve for this station is a piecewise linear function, and is illustrated by Figure 3.8.

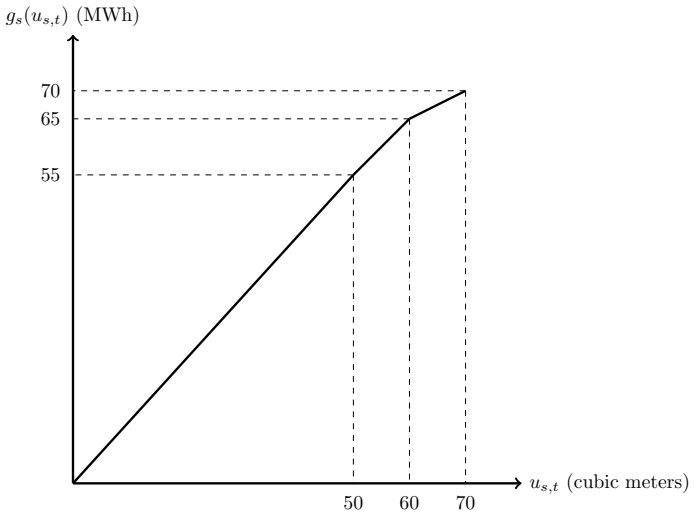


Figure 2.8: Power production curve of the elementary station.

Using a single elementary reservoir enables us to construct various hydro scheme topologies, under various price processes, and planning horizons. Using this contrived data, the models can be tested and easily analysed for various parameter settings. The price process used for these models is a 3 price state (i.e. $M = 3$), time-inhomogeneous Markov chain with the transition probabilities illustrated in Figure 3.9 below.

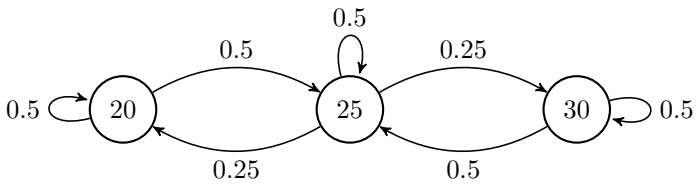


Figure 2.9: Three price state Markov chain used to represent the market clearing prices.

An optimal bidding policy was computed using the SDDP based hydro-bidding model (Model (3.2.4)) under a combination of initial storage levels for each case study. We used a desktop computer composed with an Intel i-5 CPU dual-core (with processing power of 3.4 GHz in each core) and 8 gigabytes of memory. The commercial mathematical program solver CPLEX [23] (version 12.5) was used to solve the hydro-bidding models in each method. The optimal policy of SDDP was simulated across the entire scenario tree. The performance of the SDDP policy was measured by calculating how close the objective of the SDDP policy was to the optimum policy, computed using the Scenario Tree based model, as a percentage. This was averaged across the set of initial storage levels. Table 3.3 summarizes how close the SDDP policy is to the optimum policy for cascading hydro schemes ranging from 2 reservoirs to 4 reservoirs. In all case studies, policies computed by SDDP were, on average, within 99% of the optimum policy.

		Scenario tree			**Optimality of SDDP Policy (%)**		
$\|\mathcal{R}\|$	$\|\mathcal{S}\|$	**Variables**	**Nodes**	**Scenarios**	**Mean**	**LQ**	**UQ**
2	2	605	120	81	99.561,	99.530	99.975
3	3	847	120	81	99.551	99.499	99.948
4	4	1089	120	81	99.641	99.615	99.955

Table 2.3: Evaluation of the SDDP policy with respect to the exact optimum policy computed by Model (3.2.2) with varying number of reservoirs $|\mathcal{R}|$ and stations $|\mathcal{S}|$.

The reason why SDDP is very effective can be illustrated by Figure 3.10, which depicts the SDDP approximation and the exact first stage value function for the 2 reservoir hydro scheme case study. The exact value function constructed from the solution of the scenario tree is concave. Since Model (3.2.4) is creating an outer approximation of the value function using cutting planes, it is able to accurately represent the shape of the value function and hence produce near-optimal policies (see Figure 3.11).

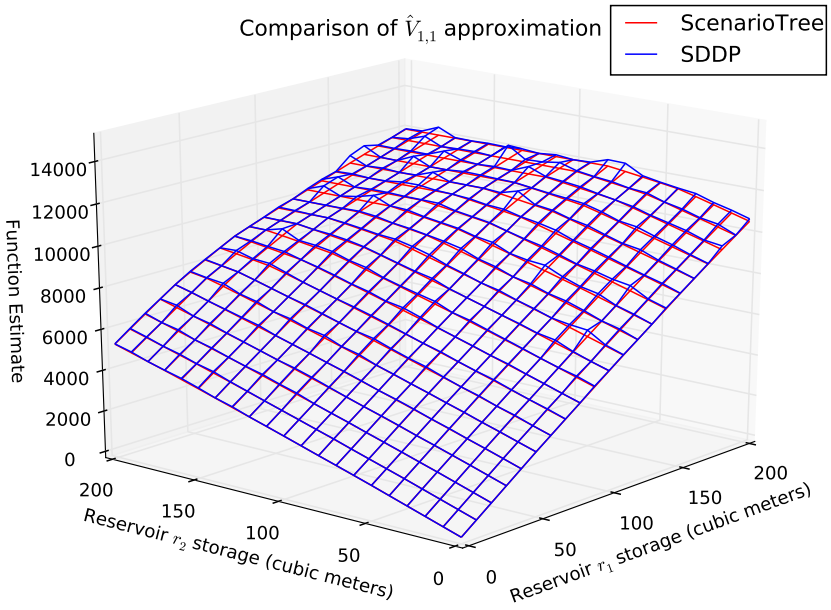


Figure 2.10: Value function approximation comparison between SDDP and the exact value function from the Scenario Tree.

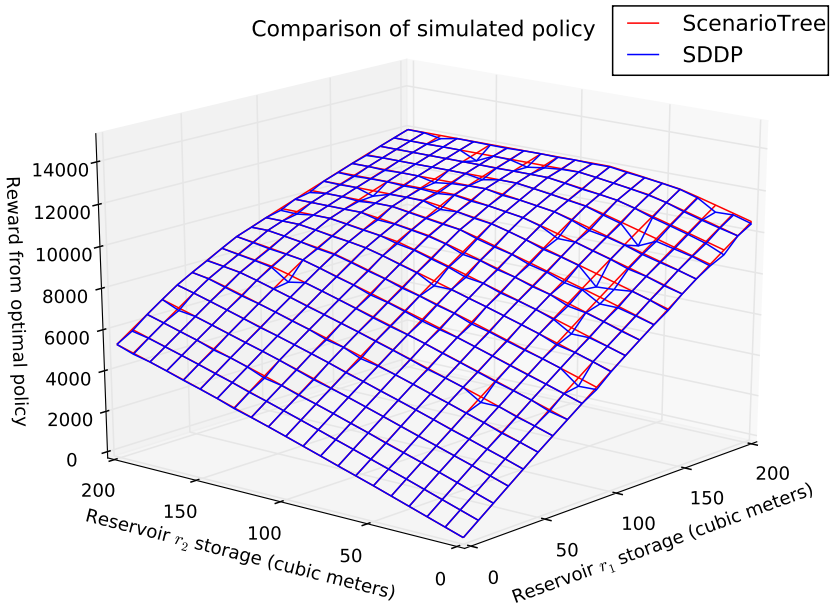


Figure 2.11: Policy simulation comparison between SDDP and the exact value function from the Scenario Tree.

Case studies used to compare the performance and quality of the SDDP based model with that of the scenario tree based model were relatively small in the number of variables. As shown in Table 3.3, the number of nodes and scenarios remains constant with the increase in the number of reservoirs. The number of variables increases linearly by $|\mathcal{N}|(|\mathcal{R}| + |\mathcal{S}| + 1)$ (note: $\mathcal{N}$ is the set of nodes in the scenario tree). Therefore, the case studies in Table 3.3 pose little challenge for industrial solvers such as CPLEX [23]. However, when expanding the planning horizon (i.e. increasing T) or the Markov chain of prices (i.e. increasing M), solving Model (3.2.2) quickly increases in computation time, as summarized in Table 3.4.

In order to obtain the results in Table 3.4, Model (3.2.2) was solved using CPLEX for the planning horizon T to be between 5 and 9 stages for a 2-reservoir hydro scheme with 3 tranche offerstack, and planning horizon of 5 and 6 periods for a 5-reservoir hydro scheme with 5 tranche offerstack (see Figure 3.12). The latter case study is more realistic of the NZEM, as the regulation requires that every generator participating in

the market provide a 5 tranche, monotonic increasing offerstack. However, due to the large number of Markov states, the scenario tree based models were not able to be solved beyond 6 period planning horizons. The solvers applied to these problems ran out of memory, a symptom of the curse of dimensionality, due to the large number of variables. These case studies were tested under initial storage levels of 100 cubic meters for all reservoirs.

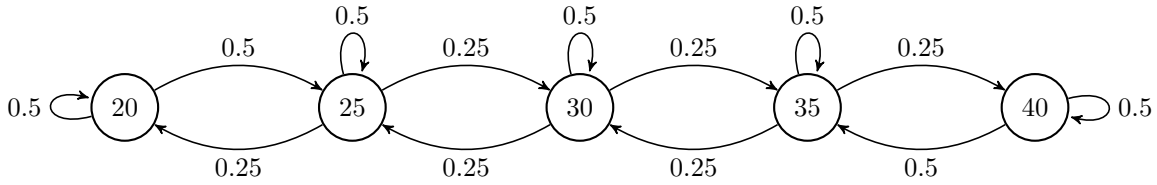


Figure 2.12: Five price state Markov chain to represent the market clearing prices.

				Scenario Tree			**SDDP**	
$\|\mathcal{R}\|$	$\|\mathcal{S}\|$	M	T	**Variables**	**Nodes**	**Mean Solution Time (sec)**	**Mean Solution Time (sec)**	**Mean optimality (%)**
2	2	3	5	1820	364	0.5356	4.712	99.552
2	2	3	6	5465	1093	2.207	5.641	98.342
2	2	3	7	16400	3280	13.654	7.431	98.189
2	2	3	8	49205	9841	98.138	10.385	99.489
2	2	3	9	147620	29524	909.156	12.192	99.673
5	5	5	5	42966	3906	236.154	5.541	98.944
5	5	5	6	214841	19531	5995.001	9.153	99.265

Table 2.4: Evaluation of the SDDP policy with respect to the exact optimum policy computed by Model (3.2.2) under combinations of $|\mathcal{R}|$ reservoirs, $|\mathcal{S}|$ stations, M tranche offerstacks, and T periods.

As is shown in Table 3.4, the number of variables, nodes, scenarios and the average

solution time increases exponentially for Model (3.2.2). As expected, the SDDP based hydro-bidding model, Model (3.2.4), is far superior in computation time, and produces optimal policies that are on average 99% optimal. Using an algorithm like SDDP provides great gains in computation power with a minimal decrease in the optimality of the computed policy.

Both Models (3.2.1) and (3.2.2) face the curse of dimensionality. For Model (3.2.1) the size of the discretized state and control set (i.e. X and $U(x)$) increases the computational complexities of the backward recursion algorithm. Model (3.2.2) runs into the curse of dimensionality because the number of nodes in its scenario tree is equivalent to $|\mathcal{N}| = \prod_{t=1}^{T} M$ and the number of scenarios is M^{T-1}, which means that the scenario tree grows exponentially with increasing time periods [71]. For example, when $M = 5$ and $T = 48$ we will have a total of 4.441×10^{31} nodes and 7.10543×10^{32} scenarios. For a hydro scheme with multiple reservoirs (i.e. $|\mathcal{R}| > 1$), Model (3.2.2) will have an astronomical number of variables, making it impractical to solve. Therefore, methods like SDDP are an effective way to solve large scale stochastic optimal control problems.

However, the major limitation for SDDP is the requirement for the value function to be continuous and convex in order to guarantee convergence. In the case for Model (3.2.4), the value function can become non-convex when the power generation function $g_s(x_t, u_{t,j,s}, l_{t,j,r})$ is bilinear or discrete.

Non-concavity of the value function creates an estimation gap between the cutting plane approximation and the true value function that cannot be reduced with the cutting plane approximation of SDDP. Figure 3.13 illustrates how the estimation gap can form between the true value function $V_{t+1,j}(f(x_t, u_{t,j}, l_{t,j}, \omega_t))$, shown by the grey line, and its single cut approximation $\hat{V}_{t+1,j} \leq \beta + \theta^T(x - \alpha)$ shown by the solid black line. When $x = \hat{x}$ the approximation gives an estimate of $\beta + \theta^T(\hat{x} - \alpha)$, which is strictly greater than the true value $V_{t+1,j}(\hat{x})$. This gap will never be reduced due to

the non-concave structure of the true value function.

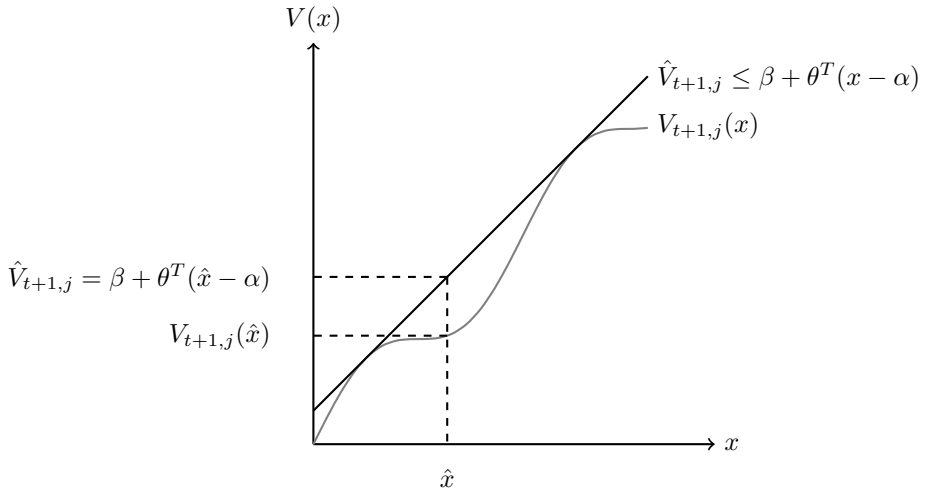


Figure 2.13: The estimation gap between the cutting plane approximation and the true value function in SDDP.

The strict requirement for concavity of the value function also makes it challenging to solve multistage stochastic mixed integer programming (SMIP) problems using SDDP. Due to the discrete variables that are present in the sub-problem, the value function becomes a piecewise polyhedral function (Figure 3.14) and in many cases discontinuous [68]. SDDP exploits the gradient of the cutting planes to explore new state trajectories, and it does this through the dual variables. Therefore, having a discontinuous value function makes either extracting the dual variables impossible or their values meaningless. One can relax the SMIP models to be linear programs, which will allow the dual variables to be extracted. However, this then creates the estimation gap (Figure 3.13) [52].

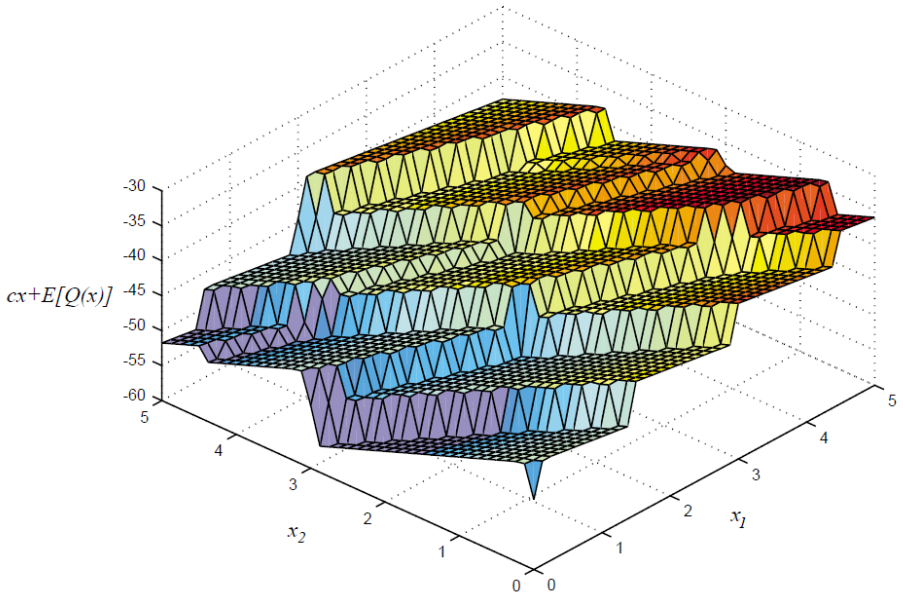


Figure 2.14: Example of a piecewise polyhedral value function (source: [68])

A number of authors have looked to extend SDDP methods to deal with non-convex stage problems. The first approach replaces the non-convex components of the problem with convex approximations as was done in [18] to convexify the hydro production functions. The second approach convexifies the value function, e.g. using Lagrangian relaxation techniques. A recent paper by [78] proposes a Lagrangian relaxation of the SDDP sub-problem, and then uses the Lagrange multipliers to produce valid Bender's cuts. A similar approach is adopted by [77]. The contribution of [2] introduces a heuristic to add locally valid cuts that enhances a convexified approximation of the value function. Even though convexification may make the overall problem fit into the SDDP framework, the computed policies may not be optimal with respect to the original, non-convex problem. If we are dealing with a highly non-convex problem, using these methods will still be subject to the estimation gap (illustrated in Figure 3.13) in which case convergence will never be achieved.

An altered version of the SDDP framework has been recently developed by [83]. In their work they use Benders cuts, and the binary optimal cuts proposed by [45] to ap-

proximate the value function with binary state variables. They use their algorithm to solve the multistage capacity expansion problem under uncertainty. Methods proposed by [83] and the contributions of this thesis, are early stages of extending algorithms like SDDP.

2.4 Summary

In this chapter we introduced the hydro-bidding model, Model (3.1.1). Then, we discuss, in detail, its various components such as the state transition constraint known as the water-balance constraint, which ensures conservation of the flow of water across the hydro scheme, and the power generation function, which represents the energy conversion of the energy potential of dammed water through the turbines in the hydropower stations. We observe that power generation functions are nonlinear (i.e. bilinear) functions that depend on the waterflow through the turbines and the net headwater level of the reservoir. We critique various modelling approaches taken to approximate this function when incorporating it inside the hydro-bidding problems. We observe that due to integer variables, the non-concave power generation function is difficult to incorporate inside Model (3.1.1). This is because non-concavity of the power generation function makes Model (3.1.1) non-concave and hard to solve.

Based on these characteristics, we progress to methods of solving Model (3.1.1). We discuss three fundamental methods in stochastic programming literature, stochastic dynamic programming (SDP), stochastic mixed-integer programming (SMIP) with scenario tree, and stochastic dual dynamic programming (SDDP). For each method, we present variants of Model (3.1.1) and discuss their advantages and limitations. We then apply these methods to various hydro schemes and under various parameter settings, and analyze their performance in approximating the value function, producing optimal

policies, and computational efficiency. When the hydro-bidding model is convex, we illustrate that SDDP produces policies that are 99% of the optimal policy (Table 3.4), but also accurately approximates the value function (Figure 3.10) all in reasonable time.

Due to the cutting planes used by SDDP to approximate the value function, it cannot guarantee convergence for non-convex SOCP. Non-concavity in the value function creates an estimation gap between the cutting plane approximation and the true value function (Figure 3.13) that cannot be reduced. This can prevent SDDP from meeting its convergence criteria. This is not an issue if the hydro-bidding model is inherently convex, however as we have already illustrated that it is not the case. Hence, our pursuit for solving non-convex hydro-bidding problems has led to the development of a new method called the Mixed-Integer Dynamic Approximation Scheme (MIDAS). MIDAS operates similarly to SDDP, but uses step functions to approximate the value function instead of hyperplanes. This allows MIDAS to approximate monotonic increasing value functions with continuous and integer state variables. In Chapter 5 we present MIDAS and prove its almost sure convergence.

Chapter 3

Hydro-bidding in the balancing market

In Chapter 3, we introduced the formulation for the hydro-bidding problem, and analysed various approaches to solving this multistage, stochastic optimization problem. We observed that the stochastic dual dynamic programming (SDDP) method was the best approach to solving convex hydro-bidding problems. SDDP produces near-optimal policies with less computational effort than the other methods. However, it cannot solve non-convex hydro-bidding problems. Since SDDP uses hyperplanes to approximate the value function, it can only guarantee almost-sure convergence when the value function is concave. However, when the hydro-bidding problem is nonconvex, either it contains nonlinear functions and/or discrete state variables, there is no guarantee that SDDP will converge.

The balancing market is an intra-day market that adjusts the generation schedule set by the day-ahead market to meet changes in the demand in near realtime (see Chapter 2). The purpose of the balancing markets is to economically adjust this total generation in order to meet the observed demand. In each period, the generators

participating in the balancing market, submit decremental bids and incremental offers as price-quantity pairs. These offers and bids are based on their reference schedule and maximum generation capacity. The incremental offer of a generator indicates how much they would like to be compensated for generating extra power, while the decremental bid indicates how much the generator is willing to compensate another generator to meet their reduction in generation. Figure 4.1 depicts the potential decremental bid and incremental offer a generator can provide based on their reference dispatch. The incremental offer, shown by the blue arrow (upwards), indicates the quantity of power the generator can submit into the balancing market. The decremental bid, illustrated by the green arrow (downwards), indicates the quantity that the generator can reduce from their reference schedule.

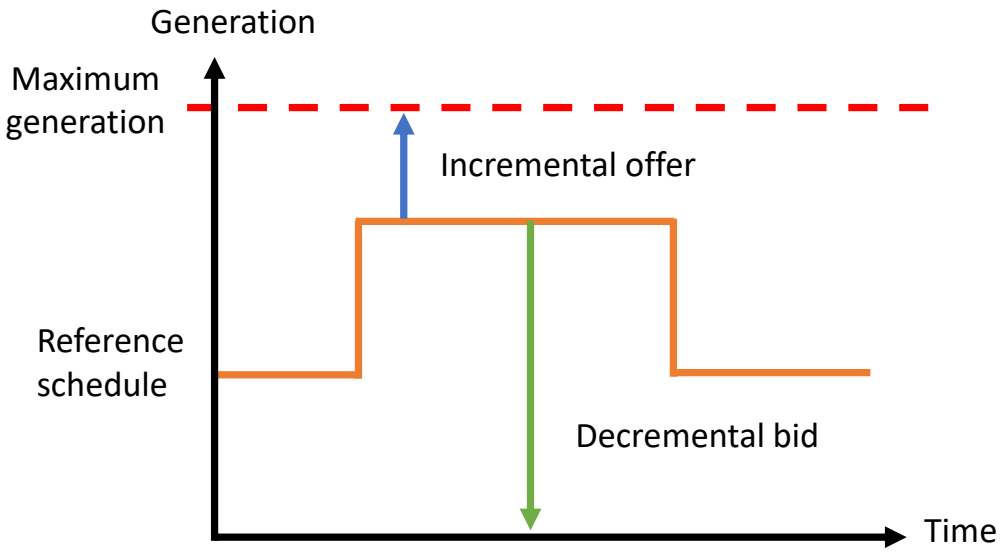


Figure 3.1: Illustration of how incremental offers and decremental bids are related to the reference schedule for a hydro hydro producer.

In this chapter, we present a two-stage stochastic programming model called the Hydro Electric River Bidding System (HERBS). HERBS is intended to compute balancing

offerstacks for hydro producers to bid as price-taking agents into an intra-day balancing market. We first describe the type of balancing market that we are studying in Section 4.1. Taking the proposed view of the balancing market, we describe the hydro-bidding problem in this market design in Section 4.2. In particular, we describe how this hydro-bidding problem is similar to the original hydro-bidding problem introduced in Chapter 2. In Section 4.3, we introduce two-stage stochastic program called HERBS, which stands for the Hydroelectric Reservoir Bidding System. We then use HERBS in a rolling horizon fashion in order to construct multi-staged bidding policies. We apply HERBS to two hydro schemes provided by EDF in Section 4.4 for the French balancing market being implemented in 2017. As the rules of the future balancing market in France have not yet been settled at the time of establishing HERBS, we propose the balancing bid computation process discussed in Section 4.1. In Section 4.5, we discuss how the multistage offer policies of HERBS are sub-optimal, and show that a multistage stochastic program is more appropriate. Furthermore, we discuss why it is difficult to solve a multistage HERBS due to the non-convexities identified in the model. Lastly, we conclude this chapter with a summary in Section 4.6.

3.1 Balancing Market Description

In this thesis,we look at a simplified balancing market with a central merit order, where the decremental bids and the incremental offers are cleared for given load (demand). This makes the balancing market similar to the intra-day electricity market described at the beginning of this thesis in Section 2.1 of Chapter 2.

In each period, the market operator aggregates the incremental offers and the decremental bids submitted by the generators. Then, balancing is managed around a reference dispatch of quantity Q at a reference price p_0. Agents will be regulated up for

offer price p bigger than p_0, and regulated down for offer price p less than p_0. Generators whose bids are accepted will be required to reduce their generation by their bid quantity from their reference schedule, and be required to pay for this reduction at the balancing price p. Generators whose incremental offers are accepted are required to increase their generation by their offer quantity from their reference schedule. They will be paid for their additional generation at the balancing price p.

Being regulated up is equivalent to selling extra power to the TSO at price p, where the generator is paid p_0 for their reference dispatch. For example, suppose that a generator submits an incremental offer of 10MW on top of their reference dispatch of 100MW, and that they are regulated up by their offer. Then, they will be paid at price p for the 10MW and p_0 for the 100MW. They will receive in total $100p_0 + 10p = 110p_0 + 10(p - p_0)$. They are paid p_0 for the 110MW generated plus a bonus for providing the regulation.

Being regulating down is equivalent to buying power back from the TSO at price p. Suppose that a generator, with a reference dispatch quantity of 100MW, submits a decremental bid of 10MW, and that they are regulated down by the TSO at their bid quantity. They will be paid p_0 for their reference dispatch quantity of 100MW and their decremental bid by 10MW at price p. They will receive in total $100p_0 - 10(p) = 90p_0 + 10(p_0 - p)$. They are paid p_0 for the 90MW generated plus a bonus for providing the regulation.

Whenever a generator is dispatched on their balancing offer or bid, their reference schedule is altered to the reference schedule. If they are dispatched at their bids or offers, then they may be limited in the future to meet their re-declared schedule. This creates a potential risk for the TSO to manage the grid in order to balance generation and consumption for future periods. In order to incentivize offers and bids which reduce this risk, a deviation penalty can be introduced. This deviation penalty

penalizes the generators for submitting offers or bids for which they cannot feasibly meet their re-declared schedule.

3.2 Hydro-bidding in a balancing market

Hydro generators, who manage a hydro scheme, competing in these markets have to make a similar trade-off, like their counterparts participating in an intra-day market. They have the same decision making framework, where they make the trade-off between maximizing their surplus they can earn from their balancing offers and bids in the current period with the opportunity cost from the balancing prices in the future. This makes the hydro-bidding problem described in Chapter 2 a good basis to study hydro-bidding problems in a balancing market environment.

In each period the hydro generator participating in the balancing market is trying to maximize their overall surplus, which is the profit from their current offers and bids and the opportunity cost of the remaining water, as well as minimizing the cost of deviation of their re-declared schedule from their reference schedule. They must be able to meet their re-declared schedule feasibly across their hydro scheme if their offers or bids are accepted. This leads to Problem 2. As we can see this problem is very similar to the hydro-bidding problem defined by Problem 1.

Objective: Maximize the expected profit from the current offers, and minimze the deviation penalty cost.

Subject to:

1. ensuring feasible generation of offered quantities of power if dispatched,
2. meeting the physical constraints of the hydro scheme, such as storage bounds and water-flow balance,
3. meeting the operation constraints of the turbines such as plant capacity and power generation function,
4. meeting ancillary requirements, such as primary and secondary reserve requirements.

Problem 2: Statement of a general hydro-bidding in an intra-day balancing market as a stochastic dynamic programming problem.

In the next section we introduce a two stage stochastic programming model called the Hydroelectric Reservoir Bidding System (HERBS). HERBS is a two-stage stochastic program, which computes the optimal supply curve for a generator with a hydro scheme under stochastic balancing prices. The supply combines both the incremental offers and the decremental bids as one supply curve. Incremental offers are represented as positive price-quantity pairs while decremental bids are negative price and negative quantity pairs. An illustration of a supply curve is provided in Figure 4.2.

3.3 HERBS model formulation

As mentioned earlier, HERBS is a two-stage stochastic programming model for computing optimum balancing supply curves under stochastic balancing price. It is formulated for a hydro scheme consisting of a set of reservoirs $\mathcal{R}$, and a set of stations $\mathcal{S}$, across T time periods. The random variable is the balancing price P_T. This is equivalent to p, which is used to describe the balancing market price in Section 4.1. It is partitioned by $2M + 1$ price intervals as,

$$\{[p_1, p_2), [p_1, p_2), , \ldots, [p_{M-1}, p_M), \ldots, [p_M, p_{M+1}), [p_{M+1}, p_{M+2}), \ldots, [p_{2M}, p_{2M+1})\}, \tag{3.3.1}$$

and modelled as a Markov price process as described in Section 3.1.1 of Chapter 4. Like in Section 3.1.1 $\pi_{t,j}$ represents the conditionally expected price, and $\rho_{ij}(t)$ represents the probability of being dispatched in tranche j, given the balancing price $P_t \in [p_i, p_{i+1})$, in time period t.

The hydro generators offer a monotonic supply curve to the balancing market. The curve contains $2M$ price-quantity pairs with positive prices, defined as,

$$\{(p_1, o_1), (p_2, o_2), \ldots, (p_M, o_M), \ldots, (p_{M+1}, o_{M+1}), (p_{M+2}, o_{M+2}), \ldots, (p_{2M}, o_{2M})\}. \tag{3.3.2}$$

Within the supply curve, there are M decremental bids and M incremental offers. The incremental offers of M tranches are represented by the price-quantity pairs, with the price being above the market-clearing price in the supply curve. This means that the generator is regulated up, where they generate additional power on top of their reference schedule. On the other hand, the M decremental bids are modelled as price-

quantity pairs where the prices are below the market-clearing price in the supply curve. The hydro generator in decremental bid is regulated down. This means that they will generate below their reference schedule, and thus will have to purchase energy from the market in order to meet their shortfall. An example supply curve is illustrated in Figure 4.2, which show the incremental offers that are above the market-clearing price p_0 (i.e. $(p_4, o_{t,4}) \geq (p_3, o_{t,3})$ and $p_3 \geq p_0$), and decremental bids that are below p_0 (i.e. $(p_1 o_{t,1}) \leq (p_2, o_{t,2})$ and $p_2 \leq p_0$). The supply curve needs to be monotonic in order to determine a unique market clearing price. To ensure monotonic supply curves, we enforce adjacent supply quantities to be monotonic increasing, where $o_{t,j} \leq o_{t,j+1}$ if $p_j < p_{j+1}$ for $j = 1, 2, \ldots, 2M$.

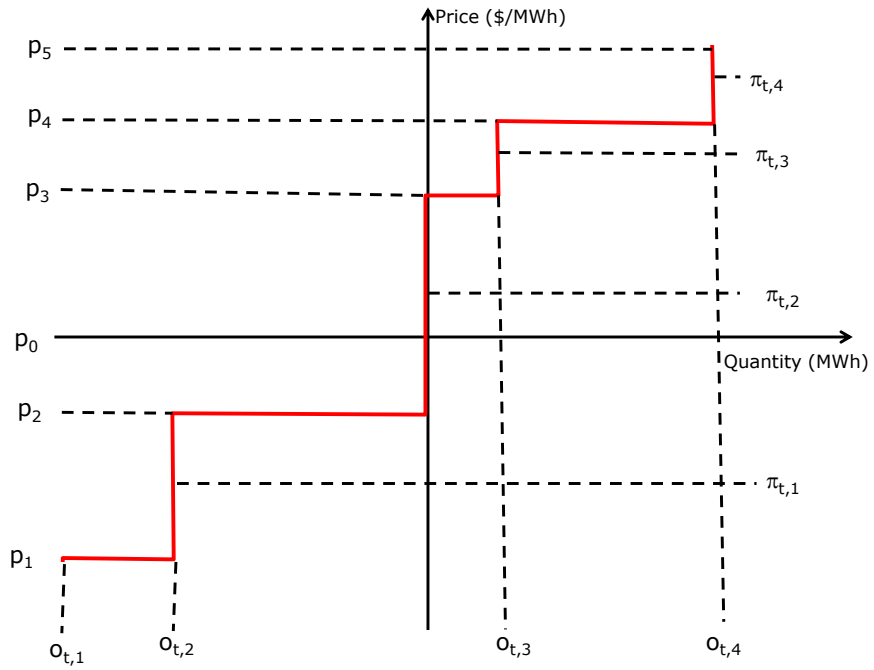


Figure 3.2: Illustration of the supply curve of a generator participating in the balancing market. Prices below the market-clearing price p_0 represent decremental bids, and prices above p_0 are incremental offers.

In this model, the hydro generator begins with a reference schedule q_t^a (MWs) from the day-ahead market. Whenever an offer j is accepted it changes the reference schedule

q_t^a to the re-declared dispatch,

$$y_{t,j} = q_t^a + o_{t,j}, \text{ for } t = 1, 2, \ldots, T. \tag{3.3.3}$$

Because of the physical constraints of the hydro scheme, the actual quantity $g_{t,j}$, called the realized dispatch, can be different from the $y_{t,j}$ schedule. The realized dispatch will satisfy the constraints of the hydro scheme such as the water-balance constraint, and will be as close as possible to the $y_{t,j}$. If the realized dispatch $g_{t,j}$ is different from the re-declared dispatch $y_{t,j}$ then the hydro generator will incur a penalty on the amount of negative deviation,

$$\delta_{t,j}^- = y_{t,j} - g_{t,j} \geq 0, \tag{3.3.4}$$

as well as the amount of positive deviation,

$$\delta_{t,j}^+ = g_{t,j} - y_{t,j} \geq 0. \tag{3.3.5}$$

Model (4.3.8) describes the two-stage, hazard-decision based, stochastic program called HERBS. In the first stage, denoted by the starting period index k, the generator computes their supply curve that maximizes their expected pay-off as,

$$\sum_{j=1}^{M} \rho_{i,j}(t)\pi_{k,j}o_{k,,j} \tag{3.3.6}$$

in period k across the M price scenarios.

Based on the observed cleared balancing price P_t in period $t = k$ and their re-declared dispatch, the hydro generator will try to generate close to their re-declared schedule

$y_{t,j}$ for periods $t = k+1, \ldots, T$, in order to minimize their expected deviation penalty,

$$\sum_{j=1}^{M} \rho_{i,j}(k) \left[\sum_{t=k+1}^{T} P_{t,j}(\delta_{t,j}^{+} + \delta_{t,j}^{-}) \right], \tag{3.3.7}$$

across the M price scenarios.This leads to the objective in the model which is to maximize the net profit in Equation (4.3.8).

Constraints (4.3.8a), (4.3.8b), (4.3.8l), and (4.3.8m) in HERBS represent the hydro scheme constraints, such as the water balance constraint (4.3.8a), the flow and storage bounds (4.3.8l) and (4.3.8m), and the initialization of the storage constraint (4.3.8b).These constraints have been discussed in detail in Chapter 3.

Constraints (4.3.8f) to (4.3.8g) define the offers and bids, and the re-declared schedules according to Equation (4.3.3). The set of constraints (4.3.8h), (4.3.8i), (4.3.8j), and (4.3.8k) define the positive and negative deviations according to equations (4.3.4) and (4.3.5). Two binary variables $v_{t,j}^{+}$ and $v_{t,j}^{-}$ have been added in order to select between the positive and negative deviations. When $v_{t,j}^{+} = 1$ it enforces either a 0 or positive deviation, where $g_{t,j} - y_{t,j} \geq 0$. The opposite occurs when $v_{t,j}^{-} = 1$, where it enforces either 0 or positive deviation, where $y_{t,j} - g_{t,j} \geq 0$.

In HERBS, the production is represented as a discrete set of $L_{i,s}$ production states. For each period $t = k, k+1, \ldots, T$ and station s there are $L_{i,s}$ pairs of $(\theta_{i,s,l}, \eta_{i,s,l})$ that represent the water discharge and the equivalent power generation pairs. The parameter $\theta_{i,s,l}$ in the pair represents the water discharge (in cubic meters) through the turbines and $\eta_{i,s,l}$ represents the respective power (in MWh's). These pairs are coupled, through constraints (4.3.8c) and (4.3.8d), by the binary variable $z_{i,j,s,l}$ that selects the optimal water discharge and power generation pairs for each period, and station (i.e. when $z_{i,j,s,l} = 1$, pair l with $(\theta_{i,s,l}, \eta_{i,s,l})$ is selected). At most, one discrete production pair can be selected for each station s, therefore constraint (4.3.8e) is added to ensure that at most one $z_{i,j,s,l}$ can be equal to 1.

Model (4.3.8) is executed in a rolling horizon fashion, where at the reference period k, $V_k(x)$ is solved. Then, the balancing market price P_t is computed, and the re-declared dispatch instructions for the j'th cleared offer are followed. Afterwards, HERBS iterates to the next bidding period $k+2$ and the process is carried out again.

$$V(x,i,k) = \max \sum_{j=1}^{M} \rho_{i,j}(k) \left[\pi_{k,j} o_{k,j} - \sum_{t=k+1}^{T} P_{t,j}(\delta_{t,j}^{+} + \delta_{t,j}^{-}) \right] \tag{3.3.8}$$

subject to:

$$x_{t+1,j,r} = x_{t,j,r} + \sum_{s\in\mathcal{S}} A_{r,s}u_{t,j,s} + \omega_{i,j,r} \quad \text{for } j = 1,\ldots,2M, \quad \text{for } r \in \mathcal{R}, \quad \text{and } t = k, k+1,\ldots,T, \tag{3.3.8a}$$

$$x_{k,j,r} = x_r \quad \text{for } j = 1,\ldots,2M, \quad \text{for } r \in \mathcal{R}, \tag{3.3.8b}$$

$$q_{t,j} = \sum_{s\in\mathcal{S}}\sum_{l=1}^{L_{t,s}} \eta_{t,s,l} z_{t,j,s,l} \quad \text{for } j = 1,\ldots,2M, \quad \text{and } t = k, k+1,\ldots,T, \tag{3.3.8c}$$

$$u_{t,j,s} = \sum_{l=1}^{L_{t,s}} \theta_{t,s,l} z_{t,j,s,l} \quad \text{for } j = 1,\ldots,2M, \quad \text{for } s \in \mathcal{S}, \quad \text{and } t = k, k+1,\ldots,T, \tag{3.3.8d}$$

$$\sum_{l=1}^{L_{t,s}} z_{t,j,s,l} \leq 1 \quad \text{for } j = 1,\ldots,2M, \quad \text{for } s \in \mathcal{S}, \quad \text{and } t = k, k+1,\ldots,T, \tag{3.3.8e}$$

$$o_{t,j} \leq o_{t,j+1} \quad \text{for } j = 1,\ldots,2M-1, \tag{3.3.8f}$$

$$o_{t,j} = y_{t,j} - q_t^a \quad \text{for } j = 1,\ldots,2M, \quad \text{and } t = k, k+1,\ldots,T, \tag{3.3.8g}$$

$$\delta_{i,j}^{-} \leq \bar{\delta} v_{t,j}^{-} \quad \text{for } j = 1, \ldots, 2M, \text{ and } t = k+1, \ldots, T, \tag{3.3.8h}$$

$$\delta_{i,j}^{+} \leq \bar{\delta} v_{t,j}^{+} \quad \text{for } j = 1, \ldots, 2M, \text{ and } t = k+1, \ldots, T, \tag{3.3.8i}$$

$$v_{i,j}^{+} + v_{i,j}^{-} = 1 \text{ for } j = 1, \ldots, 2M, \text{ and } t = k+1, \ldots, T, \tag{3.3.8j}$$

$$\delta_{i,j}^{+} - \delta_{i,j}^{-} = y_{i,j} - g_{t,j} \text{ for } j = 1, \ldots, 2M, \text{ and } t = k+1, \ldots, T, \tag{3.3.8k}$$

$$u_{i,j,s} \in [\underline{u}_s, \min\{x_{i,r}, \overline{u}_s\}] \quad \text{for } j = 1, \ldots, 2M, \text{ for } s \in \mathcal{S}, \text{ and } t = k, k+1, \ldots, T, \tag{3.3.8l}$$

$$x_{i,j,r} \in [\underline{x}_r, \overline{x}_r] \quad \text{for } j = 1, \ldots, 2M, \text{ for } r \in \mathcal{R}, \text{ and } i = t, t+1, \ldots, T. \tag{3.3.8m}$$

$$z_{i,j,s,l} \in \{0,1\} \quad \text{for } j = 1, \ldots, 2M, \text{ for } s \in \mathcal{S}, \text{ for } l = 1, \ldots, L_{i,s}, \text{ and } i = 1, t+1, \ldots, T, \tag{3.3.8n}$$

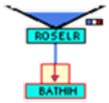
$$v_{t,j}^{+},\ v_{t,j}^{-} \in \{0,1\} \quad \text{for } j = 1, \ldots, 2M, \text{ and } t = k+1, \ldots, T, \tag{3.3.8o}$$

$$\delta_{i,j}^{+},\ \delta_{i,j}^{-} \geq 0 \quad \text{for } j = 1, \ldots, 2M, \text{ and } t = k+1, \ldots, T, \tag{3.3.8p}$$

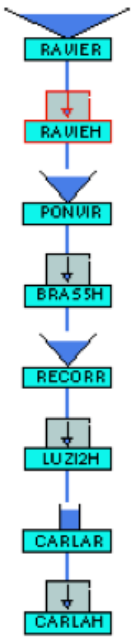
$$y_{i,j},\ g_{t,j} \geq 0 \text{ for } j = 1, \ldots, 2M, \text{ and } t = k+1, \ldots, T. \tag{3.3.8q}$$

3.4 Computing balancing bids using HERBS

HERBS was applied to two French hydro schemes, provided by EDF. These two hydro schemes are called Roselend and Agout. Roselend is a single reservoir, single station hydro scheme with the topology shown in Figure 4.3a. Agout, on the other hand, is a hydro scheme with 4 reservoirs and stations in cascade with its topology illustrated in Figure 4.3b.



(a) Roselend



(b) Agout

Figure 3.3: The topology of the Roselend (left) and Agout (right) hydro scheme in France.

A 96-period bidding horizon of the HERBS model was used to compute the supply curve for both of these hydro schemes. The daily price of a day in March 2013 from the French balancing market was used as the price realization. Due to the commercial sensitivity of the prices and the supply curves computed by HERBS, we regrettably

are not able to indicate the exact values of the price-quantity pairs and re-declared schedules. Figure 4.4 illustrates the supply curve for Roselend in period 65, with Figure 4.5 showing the dispatch following each price-quantity pair of either offer or bid. As we can see, in period 65 Roselend's supply curve had 5 unique tranches, none of which required the station to incur any deviation penalties from the realized dispatch. However, based on the offers, Roselend was willing to reduce its generation for period 65 to 0 for the minimum bid (i.e. decremental bid), and increase its generation well above the reference dispatch for the maximum generation quantity (i.e. incremental offer).

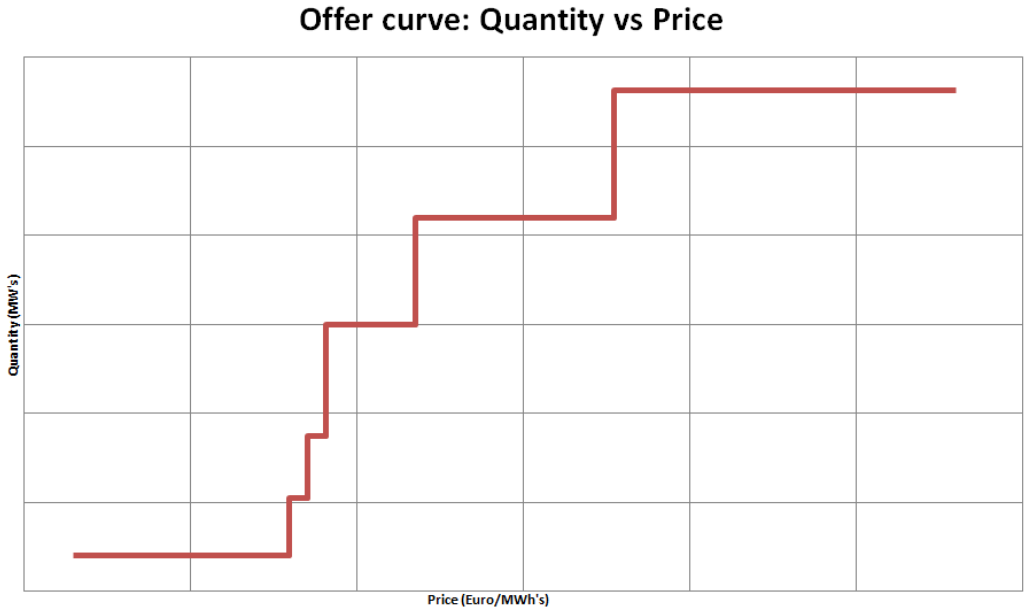


Figure 3.4: Supply curve for Roselend at period 65.

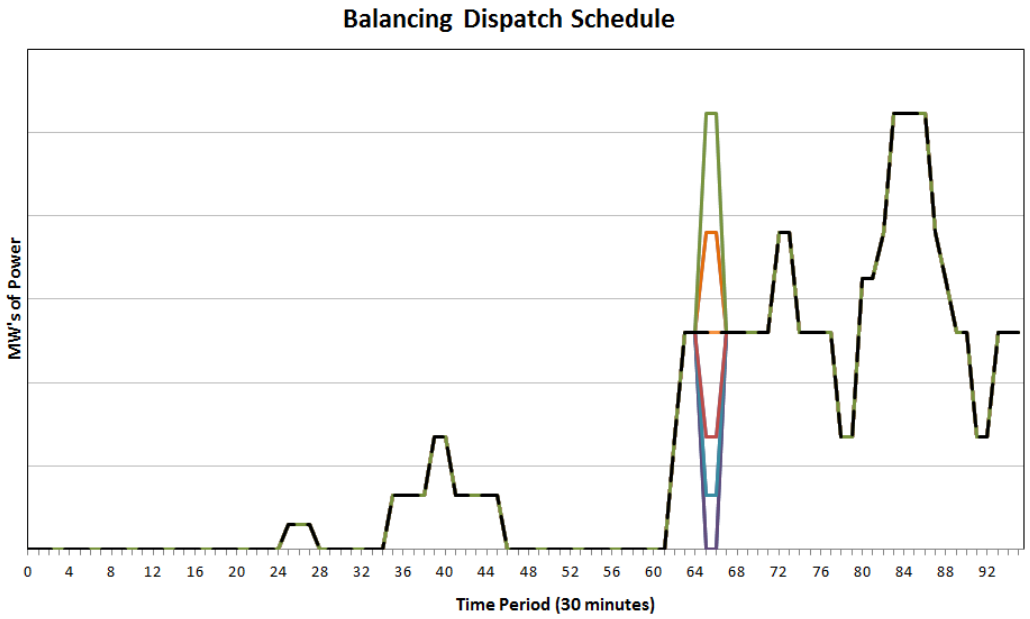


Figure 3.5: Dispatch following each price-quantity pairs in the supply curve for Roselend at period 65.

The supply curve for Agout only contains 3 unique price-quantity pairs (Figure 4.6), even though the model requires it to produce 5 price-quantity pairs. This is due to some of the price-quantity pairs being equal. However, for each pair, Agout is willing to incur deviation penalties in order to meet these quantities (Figure 4.7). This occurs when the revenue from the balancing price is higher than the cost of incurring a deviation penalty. Therefore, it becomes economical for the hydro generator, in this case EDF, to provide offers with deviation penalties.

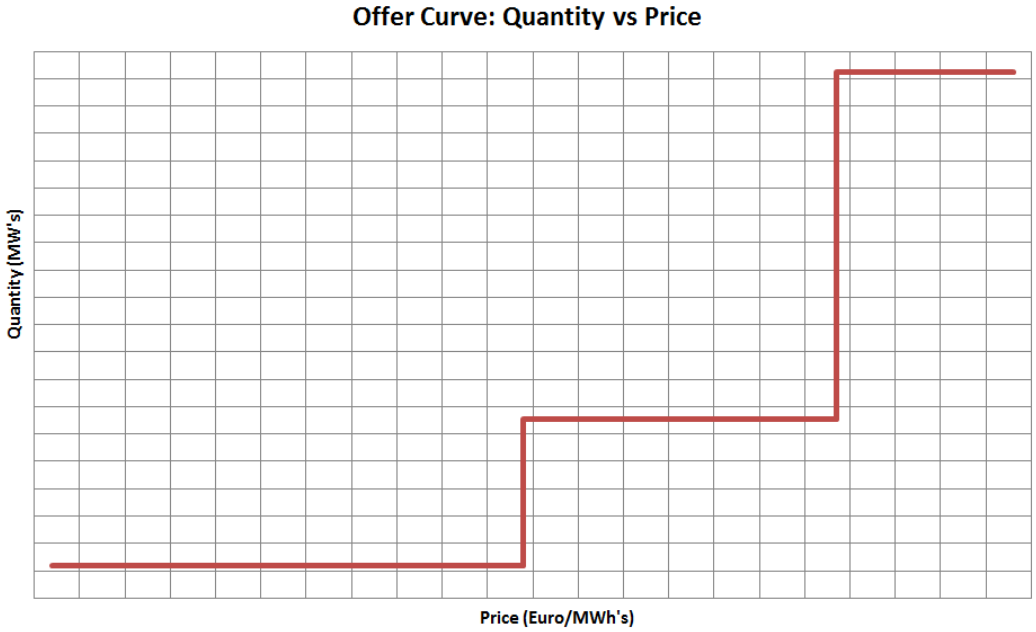


Figure 3.6: Offerstack for Agout at period 65.

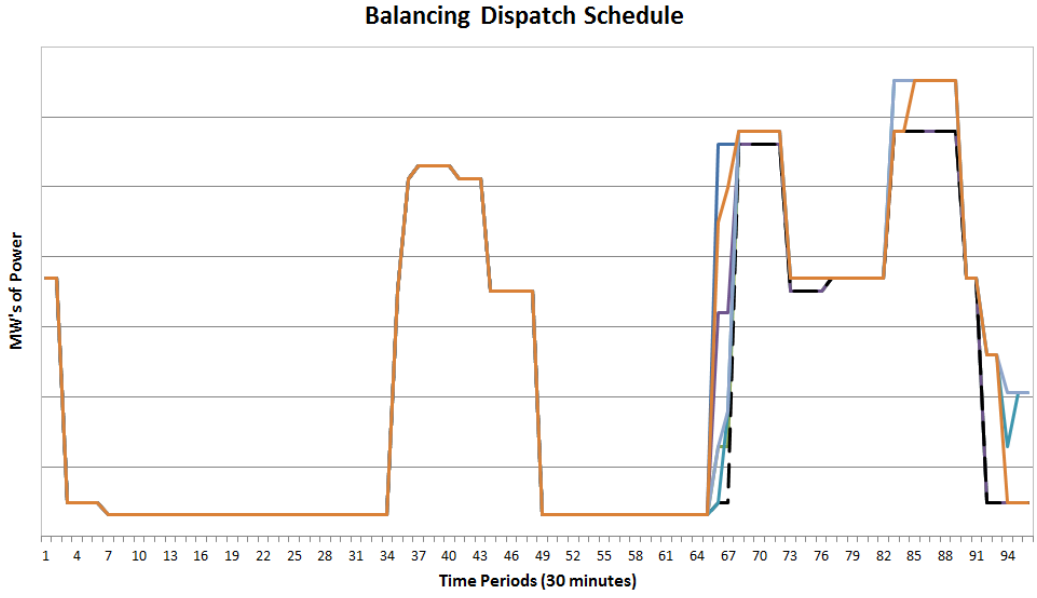


Figure 3.7: Dispatch following each bids for Agout at period 65.

The benefit of using a rolling horizon method for HERBS is that the majority of the policies will be based around the reference dispatch q_t^a as any deviations from $y_{t,j}$ are penalized. Unless the forecasted load in the day-ahead market is significantly different

from the actual load with high balancing prices, there is little incentive for any hydro generator to incur the penalty and deviate away from their re-declared schedule. Hence the balancing bids offered by the hydro generator will ensure that their re-declared dispatch does not incur any penalties in future periods. This is good for the TSO because they will not receive extreme offers that are infeasible for dispatching the power across the grid.

3.5 Limitations of HERBS

Since HERBS is a two-stage stochastic program, it assumes that all balancing price periods for $k+1$ to T will be observed in the second stage. In reality this is incorrect because the balancing market operates as an intra-day market, where market prices are observed sequentially over time. In this case, the future balancing price scenarios P_t will likely be different. Therefore, the policies that are produced by HERBS will be sub-optimal. It is more appropriate to model the balancing price using a scenario tree and solve HERBS as a multistage stochastic program.

The penalization term $\sum_{t=k+1}^{T} P_t(\delta_{t,j}^{+} + \delta_{t,j}^{-})$ is the sum of all deviations from periods $k+1$ to T. This can make the term greater than the profit earned from the current bid during initial periods, when there is a large number periods between $k+1$ and T, of the overall bidding horizon. Therefore, the hydro generator will incur large deviation penalties, which will prevent them from offering more quantities of balancing power early in the day. As the algorithm progresses towards the latter trading periods, the sum of the penalization terms becomes smaller making it more profitable to offer bids that incur deviation penalties. This is not an optimal policy, as ideally a hydro generator wants to have the waterstock in their hydro scheme at optimal levels during the day ready for future high prices. Furthermore, the penalty term discriminates

against offers in early trading periods, where the hydro generator may have greater flexibility to adapt to changes in their schedules as opposed to later in the day.

In order to produce better bidding policies, we need to model the overall participation in the balancing market as a multistage stochastic programming problem, like the hydro-bidding models presented in Chapter 3. This will ensure that the hydro generator constructs policies that arbitrage between different bidding periods while incurring minimum deviation penalties. However, formulating HERBS as a multistage stochastic programming problem runs into the curse of dimensionality [61], particularly when solving large hydro schemes with many reservoirs. Due to discrete production levels makes the multistage version of HERBS non-convex. This means that methods such as SDDP may not produce optimal policies and guarantee convergence. When the stage problems are convex, we might apply SDDP to these models. However, since we are dealing with discrete production levels, the HERBS stage problems are not convex.

3.6 Summary

In this chapter we presented a hydro-bidding model called Hydro Electric River Bidding System (HERBS). HERBS computes supply curves for hydro generators bidding as price-taking agents into a balancing market. A balancing market economically adjusts the total generation in order to meet the changes in the demand in real time. Hydro generators in a balancing market begin with a reference schedule set by the day-ahead market. They then submit a supply curve, which has both positive price-quantity pairs (i.e. incremental offers) and negative price-quantity pairs (i.e. decremental bids). The incremental offers indicate how much additional energy, on top of the reference schedule, the hydro generator is willing to generate at a particular

price. The decremental bids indicate how much energy the hydro generator is willing to compensate the market for allowing it to reduce its generation from the reference schedule. Based on these offers a hydro generator may be dispatched by the TSO on either its offer or bid. Whenever a hydro generator is dispatched on non-zero balancing bid or offer their reference schedule changes to a re-declared schedule, which takes into account their new generation requirement. If a hydro generator is unable to match its re-declared schedule it incurs a deviation penalty. The objective of HERBS, is to compute a bidding policy that maximizes the profit from incremental offers and decremental bids, and minimizes the cost of deviating away from the re-declared schedule.

HERBS was applied to two hydro schemes, called Roselend and Agout (figures 4.3a and 4.3b). It was observed that Roselend offered 5 price-quantity pairs, none of which incurred any deviational penalties. The supply curve of Agout on the other hand incurred deviation penalties in order to clear its offers. This indicates that it might be profitable to incur deviation penalties in order to obtain a better profit. The benefit of using a rolling horizon method for this problem is that a majority of the policies will be based around the reference dispatch as any deviation from the re-declared dispatch is penalized. Unless the forecasted load in the day-ahead market is significantly different from the actual load with high balancing prices, there is little incentive for any hydro generators to incur the penalty and deviate away from their re-declared schedule.

As HERBS is a two-stage stochastic program, executed in a rolling horizon, its offer policies are sub-optimal. The hydro generator does not arbitrage between different offers in different time periods. This is because at the beginning iterations of the algorithm, when it is cycling through the first few periods, the penalization term is greater than the profit that can be earned from the current bid. Hence the generator will not produce bids that will incur any penalties. But, as the algorithm progresses towards the latter trading periods the sum of the penalization terms become smaller,

and so it becomes more economical to offer bids that incur deviation penalties.

The second stage of HERBS assumes that all prices are observed for future periods. As the balancing market is an intra-day market, the price is observed sequentially. In order to produce multistage offer policies that take into account sequential observations of the price, a multistage HERBS model would be more appropriate, with the price modeled as a random process.

In order to re-formulate HERBS as a multistage stochastic programming problem we run into the curse of dimensionality. This is due to the discrete production levels of HERBS, which make it a non-convex hydro-bidding problem. Our pursuit to solve a non-convex version of HERBS has led us to the development of a new decomposition method MIDAS (Mixed Integer Dynamic Approximation Scheme). In the next chapter, Chapter 5, we introduce MIDAS and prove its almost sure convergence for integer and continuous state variables.

Chapter 4

The Mixed Integer Dynamic Approximation Scheme

In Chapter 2, we stated the key challenges to solving hydro-bidding models. These are: dealing with non-convex hydro-bidding models, incorporating price uncertainty, and modeling effects of reservoir head levels inside the power generation function. In Chapter 3, we formulated the hydro-bidding problem as a multi-stage stochastic programming problem. Taking the base formulation we analyzed various stochastic programming solution methods and showed that the Stochastic Dual Dynamic Programming (SDDP) method has an advantage in solving the hydro-bidding problem compared with other methods. We also discussed its requirement for concavity in the value function. In Chapter 4, we introduced the model HERBS (Hydro-Electric Reservoir Bidding System) a 2-stage stochastic mixed integer program, hydro-bidding model for an intra-day balancing market. We solved HERBS in a rolling horizon fashion across a 48-period planning horizon.

Since each stage problem in HERBS is not convex, due to the integer variables, we cannot guarantee that using SDDP to solve HERBS will converge to an optimal policy.

We can convexify HERBS by relaxing the discrete variables or using other decomposition methods like Lagrangian relaxations, but this yields at best an approximate solution, which is limited by the lack of the integer variables. HERBS can also be formulated as a dynamic programming problem, but will quickly run into the curse of dimensionality if we increase the number of price scenarios and reservoirs in the hydro-scheme.

In order to solve models like HERBS, we have developed a new algorithm called the Mixed-Integer Dynamic Approximation Scheme (MIDAS). MIDAS is a sampling algorithm similar to SDDP that solves multi-stage stochastic programming models where the value function is nonconcave. It works similarly to SDDP, where it uses a forward simulation (i.e. the forward pass) to compute new states, and a backward recursion step (i.e. the backward pass) at the visited states in order to update the value-function approximation. We refer the reader to Section 3.2.3 of Chapter 3 for greater detail in how SDDP can be used to solve hydro-bidding models that are convex. In contrast to SDDP, which uses cutting planes, MIDAS uses step functions to approximate the value function. By using step functions, we can approximate the nonconcave nature of the value function.

In this chapter, we describe MIDAS, and prove its almost-sure convergence. We begin by outlining the approximation of $V_t(x)$ that is used by MIDAS. In Section 5.2, we prove the convergence of MIDAS to a $2T\varepsilon$-optimal solution to a multistage deterministic optimization problem. Lastly, in Section 5.3, we extend our proof to a sampling-based algorithm applied to the multistage stochastic optimization problem, and demonstrate its almost-sure convergence.

4.1 Solving a static problem using MIDAS

Consider first the case of a static problem: $T = 1$, in a deterministic setting. The problem can then be written in the form

$$\max_{u \in U} r(u) + R(f(u)). \tag{4.1.1}$$

Here the decision set U is a compact (possibly discrete) subset of $\mathbb{R}^M$. With $u \in U$ is associated with the "state" $f(u)$, $f : U \to X$, where X is the set of feasible states, a compact subset of $\mathbb{R}^N$. The cost functions are $r : U \to \mathbb{R}$ and $R : X \to \mathbb{R}$, with r and $R \circ f$ upper semicontinuous, Denote by $\mathbf{1}$ a vector with all components equal to 1. We assume the existence of $\delta > 0$, $\varepsilon \geq 0$, such that

$$R(x) \leq R(y) + \varepsilon, \text{ if } x \leq y + \delta\mathbf{1}, \quad \text{for all } x, y \text{ in } X. \tag{4.1.2}$$

Remark 1. (i) We do not assume that R is nondecreasing, but in that case ε may have large values. So we can call the above condition an *approximate nondecreasing condition.*

(ii) If U (and hence X) is a discrete set, and R is nondecreasing, then (5.1.2) is satisfied by taking δ less than the distance between two distinct points of X, and $\varepsilon = 0$.

(iii) Since X is compact, if R is continuous and nondecreasing, since R is uniformly continuous, for any $\varepsilon > 0$, we can find $\delta > 0$ such that (5.1.2) holds.

Before stating an algorithm, let us explain how to compute an approximation of the function $R(x)$. Let $\bar{R} \geq \max\{R(x);\ x \in X\}$. Given a sequence x^k in X, with $k = 1$ to H, set

$$q^k := R(x^k). \tag{4.1.3}$$

Define the set

$$\Omega^k = \{\Big(x^h, q^h) : h = 1, 2, \ldots, k-1\Big\}. \tag{4.1.4}$$

The set of *supporting indices* (at step k) of $x \in X$ is defined as

$$\mathcal{H}^k(x) := \{h : 1 \leq h < k,\ x_i < x_i^h + \delta,\ i = 1, \ldots, N\}. \tag{4.1.5}$$

Consider also the following function, that may be viewed as an approximation of the function R (see the lemma below):

$$Q^k(x) = \begin{cases} \bar{R} & \text{if } \mathcal{H}^k(x) \text{ is empty,} \\ \min\Big\{q^h : h \in \mathcal{H}^k(x)\Big\} & \text{otherwise.} \end{cases} \tag{4.1.6}$$

Note the strict inequality in (5.1.5); by virtue of this, the function $Q^k(x)$ is upper semicontinuous.

Lemma 1. *(i) The function Q^k is nondecreasing and upper semicontinuous in x, and nonincreasing in k:*

$$Q^{k+1}(x) \leq Q^k(x), \quad x \in X. \tag{4.1.7}$$

(ii) We have that

$$R(x) \leq Q^k(x) + \varepsilon, \quad \textit{for any } x \in X, \tag{4.1.8}$$

$$k \in \mathcal{H}^{k+1}(x^k), \quad \textit{for any } k, \tag{4.1.9}$$

$$Q^{k+1}(x^k) \leq q^k, \quad \textit{for any } 1 \leq k \leq H. \tag{4.1.10}$$

Proof. (i) That Q^k is nondecreasing as a function of x, follows from the fact that $\mathcal{H}^k(x)$ is itself nonincreasing as a function of x. That Q^k is nonincreasing as a function of

k follows from the fact that $\mathcal{H}^k(x)$ is itself nondecreasing as a function of k. Upper semicontinuity of Q^k follows directly from its definition.

(ii) If $\mathcal{H}^k(x)$ is empty, (5.1.8) obviously holds, since then $Q^k(x) \geq R(x)$. Assume now that $\mathcal{H}^k(x)$ is not empty. Let $x \in X$ and $h \in \mathcal{H}^k(x)$. By (5.1.2) and the definition of $\mathcal{H}^k(x)$, we get

$$R(x) - \varepsilon \leq R(x^h) = q^h. \tag{4.1.11}$$

Minimizing over $k \in \mathcal{H}^k(x)$ and using (5.1.6) yields (5.1.8). Since trivially $x_i^k < x_i^k + \delta$, for $i = 1$ to N, we have (5.1.9), and applying (5.1.6) then gives (5.1.10). ■

Two examples of the approximation of $Q(x)$ by $Q^k(x)$ are given in Figure 5.1 and Figure 5.2.

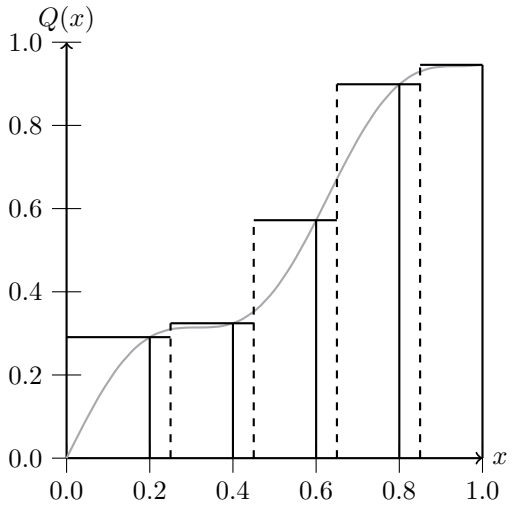


Figure 4.1: Approximation of $Q(x) = x + 0.1\sin(10x)$ shown in grey, by piecewise constant $Q^k(x)$, shown in black. Here $\delta = 0.05$ and $x^h = 0.2, 0.4, 0.6, 0.8$.

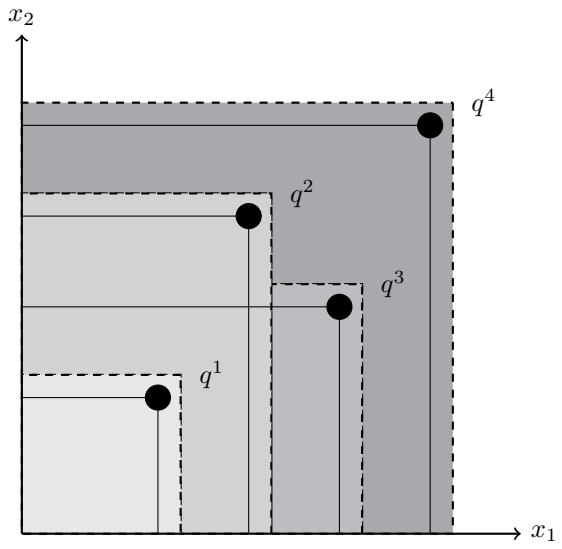


Figure 4.2: Contour plot of $Q^k(x)$ when $k = 4$. The circled points are x^h, $h = 1, 2, 3, 4$. The darker shading indicates increasing values of $Q^k(x)$ (which equals $Q(x^h)$ in the region containing x^h.)

The definition of Q^k in terms of supporting indices is made for notational convenience. In practice $Q^k(x)$ is computed using mixed integer programming (hence the name MIDAS). We propose one possible formulation for doing this in Section 5.5.

We propose the following algorithm:

Algorithm 3 Static MIDAS

1. Set $k = 1$, $\Omega^k = \emptyset$, and $Q^k(x) = \bar{R}$.

2. Forward pass:

   (a) Solve $\max_{u \in U} \left\{ r(u) + Q^k(f(u)) \right\}$ to give u^k;

   (b) If $\left\| f(u^k) - x^h \right\|_\infty < \delta$ for some $1 < h < k$, then set $x^k := x^h$, else set $x^k := f(u^k)$.

3. Backward pass: $\Omega^{k+1} := \Omega^k \cup \left\{ (x^k, q^k) \right\}$, where $q^k := R(x^k)$.

4. Stopping test: Set $\varepsilon_k := Q^k(f(u^k)) - Q^{k+1}(f(u^k))$. If $\varepsilon_k \leq \bar{\varepsilon}$, set $H := k$, and stop.

5. Increase k by 1 and go to step 2.

Remark 2. (i) The maximum in step 2(a) is attained if U is a discrete set, and more generally if f is continuous, since r and Q^k are upper semicontinuous, the latter as a consequence of lemma 1.

(ii) Note that $\varepsilon_k \geq 0$.

Theorem 1. *(i) At any iteration k, u^k is a $(2\varepsilon + \varepsilon_k)$-solution of problem* (5.1.1).

(ii) The stopping test is activated after finitely many iterations.

Proof. (i) Denote by V the value of problem (5.1.1). Since $\|x^k - f(u^k)\|_\infty < \delta$, we have that

$$k \in \mathcal{H}^{k+1}(f(u^k)). \tag{4.1.12}$$

Therefore

$$\begin{aligned}
V &= \max_{u\in U} (r(u) + R(f(u))) && \text{definition of } V \\
&\leq \max_{u\in U} \Big(r(u) + Q^k(f(u))\Big) + \varepsilon && \text{by (5.1.8)} \\
&= r(u^k) + Q^k(f(u^k)) + \varepsilon && \text{definition of } u^k \\
&= r(u^k) + Q^{k+1}(f(u^k)) + \varepsilon + \varepsilon_k && \text{definition of } \varepsilon_k \text{ in the algorithm} \\
&\leq r(u^k) + q^k + \varepsilon + \varepsilon_k && \text{consequence of (5.1.12) and (5.1.6)} \\
&\leq r(u^k) + R(f(u^k)) + 2\varepsilon + \varepsilon_k && \text{consequence of (5.1.2) and } \|x^k - f(u^k)\|_\infty < \delta.
\end{aligned}$$

(ii) Since the algorithm generates finitely many points, after finitely many iterations, $Q^{k+1} = Q^k$, and consequently $\varepsilon_k = 0$. The conclusion follows. ■

Remark 3. (i) Once convergence of the algorithm is obtained, so that a $2\varepsilon+\varepsilon_H$-solution has been found, one can choose to continue the computations with a smaller value of ε. This may be efficient for avoiding small steps that would occur when taking a small value of ε at first.

(i) If the set X is discrete and if R is nondecreasing, taking $\delta > 0$ less than the ℓ^∞ distance between two distinct points in X, we can take $\varepsilon = 0$. The output of the above algorithm is then a solution of (5.1.1).

4.2 Multistage optimization problems

In this Section we describe the MIDAS algorithm for a deterministic multistage optimization problem, and prove its convergence. Given an initial state $\bar{x}$, we consider a deterministic optimization problem of the form:

$$\begin{array}{lll} \text{MP:} \quad \max\limits_{x,u} & \sum\limits_{t=\tau}^{T} r_t(x_t, u_t) \;+\; R(x_{T+1}) & \\ \text{s.t.} & x_{t+1} = f_t(x_t, u_t), & t = \tau, \ldots, T, \\ & x_1 = \bar{x}, & \\ & u_t \in U_t(x_t), \;\; x_t \in X_t, & t = \tau, \ldots, T. \end{array}$$

Here $1 \leq \tau \leq T+1$. We have the state constraint $x_t \in X_t$, where X_t is a compact subset of $\mathbb{R}^N$, and for each $x_t \in X_t$, $U_t(x)$ is a compact subset of $\mathbb{R}^M$. The multimapping from X_t to the set of subsets of $\mathbb{R}^m$, $x \mapsto U_t(x)$ is assumed to be compatible with the state constraint, in the sense that $f_t(x_t, u_t) \in X_{t+1}$, whenever $x_t \in X_t$ and $u_t \in U_t(x_t)$. We denote by $V_\tau(x_\tau)$ the value of the above problem with initial value of the state equal to x_τ. It is well known that V_t satisfies the dynamic programming principle

$$\left\{ \begin{array}{lcll} V_t(x) & = & \sup_{u \in U_t(x)} \{ r_t(x,u) + V_{t+1}(f_t(x,u)) \}, & t = 1, \ldots, T, \\ V_{T+1}(x) & = & R(x). & \end{array} \right. \tag{4.2.1}$$

Let $\bar{V}$ be an upper bound on $V_t(x)$, $t = 1, 2, \ldots, T+1$. We assume that there exist

$\delta > 0$ and $\varepsilon \geq 0$ such that the following approximate nondecreasing condition holds:

$$V_t(x) \leq V_t(y) + \varepsilon, \text{ if } x \leq y + \delta \mathbf{1}, \quad \text{for all } x, y \text{ in } X_t, t = 2 \text{ to } T+1. \tag{4.2.2}$$

Note that for $t = 1$ we need no such condition, since we need to estimate V_1 only at the point $\bar{x}$. In the multistage case, we approximate each stage value function $V_t(x)$ with a piecewise constant function $Q_t^k(x)$. At each stage t we have a sequence $x_t^k \in X_t$, $k = 1, \ldots, H$. We will define the associated numbers q_t^k later, and set

$$\Omega_t^k = \{(x_t^h, q_t^h) : h = 1, \ldots, k-1\}, \quad \text{for all } t = 1, \ldots, T+1.$$

Given these we now define the set of supporting indices

$$\mathcal{H}_t^k(x) := \{1 \leq h < k;\ x_{ti} < x_{ti}^h + \delta,\ i = 1, \ldots, N\}, \tag{4.2.3}$$

as well as the following approximations of the Bellman functions:

$$Q_t^k(x) = \begin{cases} \bar{V} & \text{if } \mathcal{H}_t^k(x) \text{ is empty,} \\ \min\left\{q_t^h : h \in \mathcal{H}_t^k(x)\right\} & \text{otherwise.} \end{cases} \tag{4.2.4}$$

We will make use of the following simple result.

Lemma 2. *If $h < k$ and $\left\|x - x_t^h\right\|_\infty < \delta$ then $Q_t^k(x) \leq q_t^h$.*

Proof. We have $x_i < x_{ti}^h + \delta,\ i = 1, \ldots, N$, so $h \in \mathcal{H}_t^k(x)$. Thus $Q_t^k(x) \leq q_t^h$ as required. ■

The deterministic MIDAS algorithm (Algorithm 4) generates a sequence of functions $Q_t^k(x)$, $t = 1, \ldots, T+1$. For each k we define:

forward-step decision:

$$u_t^k \in \arg\max_{u \in U_t(x_t^k)} \left\{ r_t(x_t^k, u) + Q_{t+1}^k(f_t(x_t^k, u)) \right\}; \tag{4.2.5}$$

the *backward-step* decision:

$$\hat{u}_t^k \in \arg\max_{u \in U_t(x_t^k)} \left\{ r_t(x_t^k, u) + Q_{t+1}^{k+1}(f_t(x_t^k, u)) \right\}; \tag{4.2.6}$$

the *optimal decision*:

$$u_{t-1}^* \in \arg\max_{u \in U_{t-1}(x_{t-1}^k)} \{ r_{t-1}(x_{t-1}^k, u) + V_t(f_{t-1}(x_{t-1}^k, u)) \}; \tag{4.2.7}$$

the *stage error*:

$$\varepsilon_{k,t} := Q_{t+1}^k(f_t(x_t^k, u_t^k)) - Q_{t+1}^{k+1}(f_t(x_t^k, u_t^k)), \quad t = 1, \ldots, T. \tag{4.2.8}$$

Note that $\varepsilon_{k,t} \geq 0$. Given $\bar{\varepsilon} \geq 0$, the stopping test is

$$\sum_{t=1}^{T} \varepsilon_{k,t} \leq \bar{\varepsilon}. \tag{4.2.9}$$

We can now write down the steps of the algorithm as follows.

Algorithm 4 Deterministic MIDAS

1. Choose $\bar{\varepsilon} \geq 0$, and set $k = 1$, $\Omega_t^k = \emptyset$, and $Q_t^k(x) = \bar{V}$ an upper bound on $V_t(x)$, $t = 1, 2, \ldots, T+1$.

2. Forward pass: Set $x_1^k = \bar{x}$. For $t = 1$ to T,

   (a) Solve $\max\limits_{u \in U_t(x_t^k)} \left\{ r_t(x_t^k, u) + Q_{t+1}^k(f_t(x_t^k, u)) \right\}$ to give u_t^k;

   (b) If $\left\| f_t(x_t^k, u_t^k) - x_{t+1}^h \right\|_\infty < \delta$ for $h < k$ then set $x_{t+1}^k = x_{t+1}^h$, else set $x_{t+1}^k = f_t(x_t^k, u_t^k)$.

3. Backward pass: Set $\Omega_{T+1}^{k+1} = \Omega_{T+1}^k \cup \left\{ (x_{T+1}^k, q_{T+1}^k) \right\}$ where $q_{T+1}^k := R(x_{T+1}^k)$. For every $t = T$ down to 1,

   (a) Compute $q_t^k = \max\limits_{u \in U_t(x_t^k)} \left\{ r_t(x_t^k, u) + Q_{t+1}^{k+1}(f_t(x_t^k, u)) \right\}$;

   (b) Set $\Omega_t^{k+1} = \Omega_t^k \cup \left\{ (x_t^k, q_t^k) \right\}$.

4. Stopping test: if (5.2.9) holds, set $H := k$, and stop.

5. Increase k by 1 and go to step 2.

Lemma 3. *For all iterations k,*

$$\begin{cases} \text{(i)} \quad Q_t^k(x) \geq V_t(x) - (T + 2 - t)\varepsilon, \ \textit{for all } x \in X_t, \ t = 2, \ldots, T+1, \\ \text{(ii)} \quad q_1^k \geq V_1(\bar{x}) - T\varepsilon. \end{cases} \tag{4.2.10}$$

Proof. (i) We prove (5.2.10)(i) by backward induction. For $t = T+1$, the result follows from Lemma 1 since $V_{T+1}(x) = R(x)$ for all $x \in X_{T+1}$. Now let (5.2.10) hold for some

$2 < t \leq T+1$. It follows that for every k

$$\begin{aligned} q_{t-1}^k &= \max_{u \in U_{t-1}(x_{t-1}^k)} \left\{ r_{t-1}(x_{t-1}^k, u) + Q_t^{k+1}(f_{t-1}(x_{t-1}^k, u)) \right\} && \text{definition of } q_{t-1}^k \\ &\geq r_{t-1}(x_{t-1}^k, u_{t-1}^*) + Q_t^{k+1}(f_{t-1}(x_{t-1}^k, u_{t-1}^*)) && \text{definition of a maximum} \\ &\geq r_{t-1}(x_{t-1}^k, u_{t-1}^*) + V_t(f_{t-1}(x_{t-1}^k, u_{t-1}^*)) - (T+2-t)\varepsilon && \text{induction hypothesis} \\ &= V_{t-1}(x_{t-1}^k) - (T+2-t)\varepsilon. && \text{definition of } u_{t-1}^* \end{aligned} \tag{4.2.11}$$

Now let $x \in X_{t-1}$, and $h \in \mathcal{H}_{t-1}^k(x)$. Then $x_i < x_{t-1,i}^h + \delta$, $i = 1$ to N. If $t \geq 3$, by (5.2.2) and (5.2.11), we get that

$$V_{t-1}(x) \leq V_{t-1}(x_{t-1}^k) + \varepsilon \leq q_{t-1}^h + (T+3-t)\varepsilon. \tag{4.2.12}$$

Minimizing over $h \in \mathcal{H}_{t-1}^k(x)$ and using (5.1.6) we obtain that (5.2.10) holds for $t-1$. Point (i) follows.

(ii) In step (a) of this proof, we have obtained (5.2.11) also when $t = 2$, which upon setting $x_1^k = \bar{x}$ becomes

$$q_1^k \geq V_1(\bar{x}) - T\varepsilon, \tag{4.2.13}$$

so (5.2.10)(ii) follows. ■

Lemma 4. *(i) For each iteration k,*

$$V_1(\bar{x}) \leq \sum_{t=1}^{T} r_t(x_t^k, u_t^k) + R(x_{t+1}^k) + T\varepsilon + \sum_{t=1}^{T} \varepsilon_{k,t}. \tag{4.2.14}$$

(ii) The algorithm terminates after finitely many iterations at iteration $k = H$, and the resulting pseudo trajectory satisfies

$$V_1(\bar{x}) \leq \sum_{t=1}^{T} r_t(x_t^H, u_t^H) + R(x_{t+1}^H) + T\varepsilon + \bar{\varepsilon}. \tag{4.2.15}$$

Proof. (i) After step 2(b) of iteration k we have, for $t = 1$ to T:

$$\left\| x_{t+1}^k - f_t(x_t^k, u_t^k) \right\|_\infty < \delta.$$

So, $f_t(x_t^k, u_t^k) < x_{t+1}^k + \delta \mathbf{1}$. This implies

$$k \in \mathcal{H}_t^{k+1}(f_t(x_t^k, u_t^k)). \tag{4.2.16}$$

We also have that, for $t = 1$ to T:

$$\begin{aligned}
q_t^k &= r_t(x_t^k, \hat{u}_t^k) + Q_{t+1}^{k+1}(f_t(x_t^k, \hat{u}_t^k)) && \text{by (5.2.6) and the definition of } q_t^k \\
&\leq r_t(x_t^k, \hat{u}_t^k) + Q_{t+1}^{k}(f_t(x_t^k, \hat{u}_t^k)) && \text{by the monotonicity of } k \mapsto Q_{t+1}^k \\
&\leq r_t(x_t^k, u_t^k) + Q_{t+1}^{k}(f_t(x_t^k, u_t^k)) && \text{by the definition of } u_t^k \text{ in (5.2.5)} \\
&= r_t(x_t^k, u_t^k) + Q_{t+1}^{k+1}(f_t(x_t^k, u_t^k)) + \varepsilon_{k,t} && \text{by the definition of } \varepsilon_{k,t} \\
&\leq r_t(x_t^k, u_t^k) + q_{t+1}^k + \varepsilon_{k,t} && \text{by (5.2.16).}
\end{aligned}$$

Also,

$$q_{T+1}^k = R(x_{T+1}^k). \tag{4.2.17}$$

Summing the previous inequalities and equality, we get that

$$q_1^k \leq \sum_{t=1}^{T} r_t(x_t^k, u_t^k) + R(x_{T+1}^k) + \sum_{t=1}^{T} \varepsilon_{k,t}. \tag{4.2.18}$$

By Lemma 3 for $t = 1$, we have that

$$V_1(\bar{x}) \leq q_1^k + T\varepsilon. \tag{4.2.19}$$

The result follows by combining (5.2.18) and (5.2.19).

(ii) The algorithm visits finitely many points. So, by backward induction over t, we

get that the pairs (x_t^k, q_t^k) also take finitely many values. So, for large enough k, if the algorithm does not stop, the functions Q_t^k do not depend on k, and therefore all $\varepsilon_{k,t} = 0$. But this contradicts the fact that the stopping test is never satisfied. ■

The previous result applies to the sequence (x_t^H, u_t^H) which satisfies

$$\|x_{t+1} - f_t(x_t, u_t)\|_\infty < \delta, \quad t = 1, \ldots, T. \tag{4.2.20}$$

Assume that with any such pseudo trajectory, we can associate a true trajectory by increasing the cost by some nonnegative amount c_δ. Denote by $(\bar{x}_t^H, \bar{u}_t^H)$, for $t = 1$ to $T+1$, such a trajectory associated with (x^H, u^H). We get then the following corollary:

Corollary 1. *Set* $\hat{\varepsilon}_H := T\varepsilon + \sum_{t=1}^T \varepsilon_{H,t} + c_\delta$. *We have that, at each iteration* $1 \le k \le M$*: of the algorithm:*

$$V_1(\bar{x}) \le \sum_{t=1}^{T} r_t(\bar{x}_t^H, \bar{u}_t^H) + R(\bar{x}_{t+1}^H) + \hat{\varepsilon}_H. \tag{4.2.21}$$

In particular, the output trajectory $(\bar{x}^H, \bar{u}^H)$ *is* $T\varepsilon + \bar{\varepsilon} + c_\delta$ *optimal.*

The previous result is rather weak in that it gives no bounds of the size of c_δ. If we set $\bar{\varepsilon} = 0$, then we can obtain a bound on optimality for the trajectory obtained when the algorithm terminates. This will occur at iteration $k = H$ when for every $t = 1, 2, \ldots, T$,

$$Q_{t+1}^H(f_t(x_t^H, u_t^H)) - Q_{t+1}^{H+1}(f_t(x_t^H, u_t^H)) = 0, \quad t = 1, \ldots, T. \tag{4.2.22}$$

This means that the forward pass in iteration $H+1$ will visit the same points as in iteration H, so

$$(x_t^{H+1}, u_t^{H+1}) = (x_t^H, u_t^H),\ t = 1, 2, \ldots, T+1,$$

and so the functions Q_{t+1}^H and Q_{t+1}^{H+1} will be identical for all $t = 1, 2, \ldots, T+1$. It follows

that in the forward pass, u_t^H will maximize $r_t(x_t^H, u_t^H) + Q_{t+1}^{H+1}(f_t(x_t^H, u_t^H))$ giving

$$r_t(x_t^H, u_t^H) + Q_{t+1}^{H+1}(f_t(x_t^H, u_t^H)) \geq r_t(x_t^H, \hat{u}_t^H) + Q_{t+1}^{H+1}(f_t(x_t^H, \hat{u}_t^H)),$$

and (5.2.6) implies

$$r_t(x_t^H, u_t^H) + Q_{t+1}^{H+1}(f_t(x_t^H, u_t^H)) \leq r_t(x_t^H, \hat{u}_t^H) + Q_{t+1}^{H+1}(f_t(x_t^H, \hat{u}_t^H)),$$

so

$$r_t(x_t^H, u_t^H) + Q_{t+1}^{H}(f_t(x_t^H, u_t^H)) = r_t(x_t^H, \hat{u}_t^H) + Q_{t+1}^{H+1}(f_t(x_t^H, \hat{u}_t^H)) \tag{4.2.23}$$

We now can establish the following result.

Lemma 5. *For every* $t = 1, 2, \ldots, T$,

$$Q_{t+1}^{H+1}(f_t(x_t^H, u_t^H)) \leq V_{t+1}(f_t(x_t^H, u_t^H)) + (T - t + 1)\varepsilon.$$

Proof. We proceed by backwards induction on t. When $t = T$, after step 2(b) of iteration H we have for some $h \leq H$

$$\left\| x_{T+1}^h - f_T(x_T^H, u_T^H) \right\|_\infty < \delta$$

so by Lemma 2

$$\begin{aligned} Q_{T+1}^{H+1}(f_T(x_T^H, u_T^H)) &\leq q_{T+1}^h \\ &= V_{T+1}(x_{T+1}^h) \\ &\leq V_{T+1}(f(x_T^H, u_T^H)) + \varepsilon, \end{aligned}$$

by (5.2.2).

Assume as an inductive hypothesis that

$$Q_{t+1}^{H+1}(f_t(x_t^H, u_t^H)) \leq V_{t+1}(f(x_t^H, u_t^H)) + (T-t+1)\varepsilon$$

We show that

$$Q_t^{H+1}(f_{t-1}(x_{t-1}^H, u_{t-1}^H)) \leq V_t(f_{t-1}(x_{t-1}^H, u_{t-1}^H)) + (T-t+2)\varepsilon.$$

Since at termination of the algorithm

$$\|x_t^H - f_{t-1}(x_{t-1}^H, u_{t-1}^H)\|_\infty < \delta,$$

it follows that

$$\begin{aligned}
Q_t^{H+1}(f_{t-1}(x_{t-1}^H, u_{t-1}^H)) &\leq q_t^H && \text{by Lemma 2} \\
&= r(x_t^H, \hat{u}_t^H) + Q_{t+1}^{H+1}(f_t(x_t^H, \hat{u}_t^H)) && \text{by (5.2.6)} \\
&= r(x_t^H, u_t^H) + Q_{t+1}^{H}(f_t(x_t^H, u_t^H)) && \text{by (5.2.23)} \\
&\leq r(x_t^H, u_t^H) + V_{t+1}(f_t(x_t^H, u_t^H)) + (T-t+1)\varepsilon && \text{induction hypothesis} \\
&\leq r(x_t^H, u_t^*) + V_{t+1}(f_t(x_t^H, u_t^*)) + (T-t+1)\varepsilon && \text{optimality of } u^* \\
&= V_t(x_t^H) + (T-t+1)\varepsilon && \text{definition of } V_t \\
&\leq V_t(f_{t-1}(x_{t-1}^H, u_{t-1}^H)) + (T-t+2)\varepsilon
\end{aligned}$$

where the last inequality follows from $\|x_t^H - f_{t-1}(x_{t-1}^H, u_{t-1}^H)\|_\infty < \delta$ and (5.2.2). This establishes the result for $t-1$ and hence all t by induction. ■

Using Lemma 5, we can show that the first-stage decision u_1^H obtained when Algorithm 2 terminates (at iteration H where $\varepsilon^H = 0$) is $2T\varepsilon$-optimal.

Theorem 2. *Suppose $\bar{\varepsilon} = 0$. Upon termination of the algorithm the first-stage decision*

u_1^H *satisfies*

$$r_1(\bar{x}, u_1^H) + V_2(f_1(\bar{x}, u_1^H)) \geq V_1(\bar{x}) - 2T\varepsilon.$$

Proof. For the optimal first stage decision u_1^*, Lemma 3 gives

$$r_1(\bar{x}, u_1^*) + Q_2^H(f_1(\bar{x}, u_1^*)) \geq r_1(\bar{x}, u_1^*) + V_2(f_1(\bar{x}, u_1^*)) - T\varepsilon$$

and by (5.2.5)

$$r_1(\bar{x}, u_1^H) + Q_2^H(f_1(\bar{x}, u_1^H)) \geq r_1(\bar{x}, u_1^*) + Q_2^H(f_1(\bar{x}, u_1^*))$$

so

$$r_1(\bar{x}, u_1^H) + Q_2^H(f_1(\bar{x}, u_1^H)) \geq r_1(\bar{x}, u_1^*) + V_2(f_1(\bar{x}, u_1^*)) - T\varepsilon. \tag{4.2.24}$$

Now upon termination of the algorithm (5.2.23) gives

$$Q_2^H(f_1(\bar{x}, u_1^H)) = Q_2^{H+1}(f_1(\bar{x}, u_1^H)), \tag{4.2.25}$$

and Lemma 5 implies

$$Q_2^{H+1}(f_1(\bar{x}, u_1^H)) \leq V_2(f_1(\bar{x}, u_1^H)) + T\varepsilon, \tag{4.2.26}$$

so (5.2.24), (5.3.22), and (5.2.26) yield

$$r_1(\bar{x}, u_1^H) + V_2(f_1(\bar{x}, u_1^H)) \geq r_1(\bar{x}, u_1^*) + V_2(f_1(\bar{x}, u_1^*)) - 2T\varepsilon,$$

giving the result. ■

4.3 Multistage stochastic optimization problems

We extend model MP, described in Section 5.2, to include random noise on the state transition function. We model the realizations of the noise in the form of a *scenario tree* with nodes $n \in \mathcal{N}$ and leaves in $\mathcal{L}$, where n represents different future states of the world. Each node n has a probability $p(n)$. By convention we number the root node $n = 0$ (with $p(0) = 1$). The unique predecessor of node $n \neq 0$ is denoted by $n-$. We denote the set of children of node $n \in \mathcal{N} \setminus \mathcal{L}$ by $n+$, and let $M_n = |(n+)|$. The depth $d(n)$ of node n is the number of nodes on the path from node n to node 0, so $d(0) = 1$ and we assume that every leaf node has the same depth, say $d_{\mathcal{L}}$. The depth of a node can be interpreted as a time index, so we can identify $d_{\mathcal{L}}$ with time $T + 1$ as defined in Section 5.2. Set $\mathcal{N}_* := \mathcal{N} \setminus \{0\}$. The formulation of MSP in the scenario tree is

$$\begin{aligned}
\text{MSPT:} \quad \max \quad & \sum_{n \in \mathcal{N}_* \setminus \mathcal{L}} p(n) r_n(x_{n-}, u_n) + \sum_{n \in \mathcal{L}} p(n) R(x_n) & \\
\text{s.t.} \quad & x_n = f_n(x_{n-}, u_n), & n \in \mathcal{N}_*, \\
& x_0 = \overline{x}, & \\
& u_n \in U_n(x_{n-}), & n \in \mathcal{N}_*, \\
& x_n \in X_n, & n \in \mathcal{N}_*.
\end{aligned}$$

The probabilities must satisfy

$$p(n) \geq 0,\ n \in \mathcal{N}; \quad \sum_{n \in \mathcal{L}} p(n) = 1; \quad p(n) = \sum_{m \in n+} p(m), \quad n \in \mathcal{N} \setminus \mathcal{L}. \tag{4.3.1}$$

Observe in MSPT that we have a choice between hazard-decision and decision-hazard formulations that was not relevant in the deterministic problem. To be consistent with most implementations of SDDP, we have chosen a hazard-decision setting. This means that u is chosen in node n after the information from node n is revealed. The

dynamic programming principle for MSPT can be expressed as

$$\begin{cases} V_n(x_n) &= \sum\limits_{m\in n+} \frac{p(m)}{p(n)} \max_{u\in U_m(x_n)} \left\{ r_m(x_n,u) + V_m\left(f_m(x_n,u)\right)\right\}, & n\in\mathcal{N}\setminus\mathcal{L}, \\ V_n(x_n) &= R(x_n), & n\in\mathcal{L}. \end{cases}$$

We seek a policy that maximizes $V_0(\bar{x})$. Below we give two different extensions of the MIDAS algorithm of the deterministic setting; *Full-tree MIDAS* goes over every node at each iteration; and *Sampled MIDAS* computes only one pseudo trajectory at each iteration.

Now given a scenario tree, we approximate each node value function $V_n(x)$, $n\in\mathcal{N}_*$, with a piecewise constant function $Q_n^k(x)$, where $k = 1,2,\ldots,H$ is the algorithm iteration. We have a sequence x_n^k in X_n. Associated with each x_n^k is a value q_n^k. For leaf nodes we have that

$$q_n^k := R(x_n^k), \quad n\in\mathcal{L}. \tag{4.3.2}$$

As before we define

$$\Omega_n^k = \{(x_n^k, q_n^k) : h = 1,\ldots,k-1\}, \quad \text{for all } n\in\mathcal{N}_*\setminus\mathcal{L}$$

We now define the *supporting indices* of $x\in X_n$ as

$$\mathcal{H}_n^k(x) := \{1\le h<k;\ x_i < x_{n,i}^k + \delta,\ i=1,\ldots,N\}, \tag{4.3.3}$$

as well as the following function:

$$Q_n^k(x) = \begin{cases} \bar{V} & \text{if } \mathcal{H}_n^k(x) \text{ is empty,} \\ \min\left\{q_n^k : h\in\mathcal{H}_n^k(x)\right\} & \text{otherwise.} \end{cases} \tag{4.3.4}$$

and as before we suppress the dependence of $\mathcal{H}_n^k(x)$ and $Q_n^k(x)$ on the parameter δ. The following result is immediate.

Lemma 6. *If $h < k$ and $\left\|x - x_n^h\right\|_\infty < \delta$ then $Q_n^k(x) \leq q_n^h$.*

4.3.1 Full-tree MIDAS algorithm

The stopping test is based on the following amounts:

$$\begin{cases} \varepsilon_{k,m} & := \quad Q_m^k(f_m(x_n^k, u_m^k)) - Q_m^{k+1}(f_m(x_n^k, u_m^k)), \quad \text{for all } n \in \mathcal{N} \setminus \mathcal{L} \text{ and } m \in n+, \\ \varepsilon_k : & = \quad \sum_{m \in \mathcal{N}_*} p(m)\varepsilon_{k,m}. \end{cases} \tag{4.3.5}$$

Algorithm 5 Full tree MIDAS

Set $k = 1$, and $\Omega_n^k = \emptyset$, for all $n \in \mathcal{N}$.

1. Forward pass: Set $x_0^k = \overline{x}$, and $n = 0$. While $n \notin \mathcal{L}$, for each $m \in n+$:

   (a) Solve $\max_{u \in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^k(f_m(x_n^k, u)) \right\}$ to give u_m^k;

   (b) If $\left\| f_m(x_n^k, u_m^k) - x_m^k \right\|_\infty < \delta$ for some $h < k$ then set $x_m^k = x_m^k$.
   Otherwise, set $x_m^k = f_m(x_n^k, u_m^k)$.

   (c) Set $n = m$.

2. Backward pass:

   (a) For every $n \in \mathcal{L}$, set

   $$\Omega_n^{k+1} := \Omega_n^k \cup \{(x_n^k, q_n^k)\}, \quad \text{where } q_n^k = R(x_n^k). \tag{4.3.6}$$

   (b) 'In order of decreasing depth', for each node n

i. Compute

$$q_n^k = \sum_{m \in n+} \frac{p(m)}{p(n)} \left[\max_{u \in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^{k+1}(f_m(x_n^k, u)) \right\} \right]. \quad (4.3.7)$$

with maxima in computation of q_n^k attained at $\hat{u}_m^k$, for each $m \in n+$.

ii. Set $\Omega_n^{k+1} := \Omega_n^k \cup \{(x_n^k, q_n^k)\}$.

3. If ε_k, defined in (5.3.5), satisfies $\varepsilon_k \leq \bar{\varepsilon}$, set $H := k$ and stop.

4. Increase k by 1 and go to step 1.

We assume that there exist $\delta > 0$ and $\varepsilon > 0$ such that, for any $n \in \mathcal{N}_*$, the following approximate nondecreasing condition holds:

$$V_n(x) \leq V_n(y) + \varepsilon, \text{ if } x \leq y + \delta \mathbf{1}, \quad \text{for all } x, y \text{ in } X_n. \quad (4.3.8)$$

Lemma 7. *For every $n \in \mathcal{N}_*$, and for all iterations k*

$$Q_n^k(x) \geq V_n(x) - (d_{\mathcal{L}} + 1 - d(n))\varepsilon. \quad (4.3.9)$$

Proof. For a leaf node m in $\mathcal{L}$, with depth $d(m) = d_{\mathcal{L}}$, (5.3.9) follows from Lemma 1. Now let $n \in \mathcal{N}_* \setminus \mathcal{L}$ have depth $d(n)$, and suppose as an inductive hypothesis that (5.3.9) holds for all nodes with depth greater than $d(n)$. Let $x \in X_n$, and ($h \in \mathcal{H}_n^k(x)$, for some $h < k$. Let

$$u_m^* \in \arg \max_{u \in U_m(x_n^h)} \left\{ r_m(x_n^h, u) + V_m \left(f_m(x_m^h, u) \right) \right\}, \quad \text{for any } m \in n+.$$

Then

$$\begin{aligned} q_n^h &= \sum_{m\in n+} \frac{p(m)}{p(n)} \max_{u\in U_m(x_n^h)} \left\{ r_m(x_n^h, u) + Q_m^{h+1}\left(f_m(x_n^h, u)\right)\right\} \\ &\geq \sum_{m\in n+} \frac{p(m)}{p(n)} \left(r_m(x_n^h, u_m^*) + Q_m^{h+1}\left(f_m(x_n^h, u_m^*)\right)\right). \end{aligned}$$

Applying (5.3.9) now yields

$$\begin{aligned} q_n^h &\geq \sum_{m\in n+} \frac{p(m)}{p(n)} (r_m(x_n^h, u_m^*) + V_m\left(f_m(x_n^h, u_m^*)\right) - (d_{\mathcal{L}} + 1 - d(m))\varepsilon) \\ &= V_n(x_n^h) - (d_{\mathcal{L}} + 1 - d(m))\varepsilon, \\ &= V_n(x_n^h) - (d_{\mathcal{L}} - d(n))\varepsilon, \end{aligned}$$

since $d(n) = d(m) - 1$. Now for $h \in \mathcal{H}_n^k(x)$, $x_i < x_{ni}^k + \delta$, $i = 1, 2, \ldots, N$. By (5.3.8) and the above inequality, we get that

$$V_n(x) \leq V_n(x_n^h) + \varepsilon \leq q_n^h + (d_{\mathcal{L}} + 1 - d(n))\varepsilon. \tag{4.3.10}$$

The conclusion follows by minimizing over $h \in \mathcal{H}_n^k(x)$. ■

We next analyze the convergence of the Full-tree MIDAS algorithm. As in the deterministic case, we define

$$u_m^k \in \arg \max_{u\in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^k(f_m(x_n^k, u))\right\}, \quad m \in n+, \tag{4.3.11}$$

$$\hat{u}_m^k \in \arg \max_{u\in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^{k+1}(f_m(x_n^k, u))\right\}, \quad m \in n+. \tag{4.3.12}$$

Theorem 3. *i) We have, at each iteration k of the algorithm:*

$$V_0(\bar{x}) \leq \sum_{n\in\mathcal{N}_*\setminus\mathcal{L}} p(n) r_n(x^k_{n-}, u^k_n) + \sum_{n\in\mathcal{L}} p(n) R\left(x^k_n\right) + (d_{\mathcal{L}} - 1)\varepsilon + \varepsilon_k. \tag{4.3.13}$$

ii) The stopping test is satisfied after a finite value number of steps (when $k = H$), and then, the current policy is such that

$$V_0(\bar{x}) \leq \sum_{n\in\mathcal{N}_*\setminus\mathcal{L}} p(n) r_n(x^H_{n-}, u^H_n) + \sum_{n\in\mathcal{L}} p(n) R\left(x^H_n\right) + (d_{\mathcal{L}} - 1)\varepsilon + \bar{\varepsilon}. \tag{4.3.14}$$

Proof. (i) Adapting the arguments for the deterministic case, we can write

$$\begin{aligned}
p(n)q^k_n &= \sum_{m\in n+} p(m)\left(r_m(x^k_n, \hat{u}^k_m) + Q^{k+1}_m(f_m(x^k_n, \hat{u}^k_m))\right) \\
&\leq \sum_{m\in n+} p(m)\left(r_m(x^k_n, \hat{u}^k_m) + Q^{k}_m(f_m(x^k_n, \hat{u}^k_m))\right) \\
&\leq \sum_{m\in n+} p(m)\left(r_m(x^k_n, u^k_m) + Q^{k}_m(f_m(x^k_n, u^k_m))\right) \\
&= \sum_{m\in n+} p(m)\left(r_m(x^k_n, u^k_m) + Q^{k+1}_m(f_m(x^k_n, u^k_m)) + \varepsilon_{k,m}\right) \\
&\leq \sum_{m\in n+} p(m)\left(r_m(x^k_n, u^k_m) + q^k_m + \varepsilon_{k,m}\right).
\end{aligned} \tag{4.3.15}$$

As a special case

$$q^k_0 \leq \sum_{m\in 0+} p(m)\left(r_m(\bar{x}, u^k_m) + q^k_m + \varepsilon_{k,m}\right),$$

so substituting for $p(m)q^k_m$ recursively throughout the tree and using (5.3.15) and

$$p(n)q^k_n = p(n)R(x^k_n), \quad \text{for all } n \in \mathcal{L}, \tag{4.3.16}$$

yields

$$q^k_0 \leq \sum_{n\in\mathcal{N}_*} p(n) r_n(x_{n-}, u^k_n) + \sum_{n\in\mathcal{L}} p(n) R\left(x^k_n\right) + \sum_{n\in\mathcal{N}_*} p(n)\varepsilon_{k,n} \tag{4.3.17}$$

On the other hand, Lemma 7 gives

$$\begin{aligned} V_0(\bar{x}) &= \sum_{m\in 0+} p(m)\left(r_m(\bar{x},u_m^*)+V_m(f_m(\bar{x},u_m^*))\right) \\ &\le \sum_{m\in 0+} p(m)\left(r_m(\bar{x},u_m^*)+Q_m^{k+1}(f_m(\bar{x},u_m^*))\right)+(d_{\mathcal{L}}-1)\varepsilon \\ &\le \sum_{m\in 0+} p(m)\left(r_m(\bar{x},\hat{u}_m^k)+Q_m^{k+1}(f_m(\bar{x},\hat{u}_m^k))\right)+(d_{\mathcal{L}}-1)\varepsilon \\ &= q_0^k+(d_{\mathcal{L}}-1)\varepsilon. \end{aligned} \tag{4.3.18}$$

The result follows from combining (5.3.17) and (5.3.18).

(ii) The result is a consequence of (i) and the fact that, as in the deterministic case, after finitely many iterations, we will have $\varepsilon_{k,n}=0$ for each node n. ■

As before, if we set $\bar{\varepsilon}=0$, then at some iteration $k=H$ the forward pass in every iteration $k>H$ will visit the same points x_n^H as in iteration H, so for every $t=1,2,\ldots,T$,

$$Q_n^H(f_n(x_{n-}^H,u_n^H))=Q_n^{H+1}(f_n(x_{n-}^H,u_n^H)) \tag{4.3.19}$$

This means that the functions Q_n^H and Q_n^{H+1} will be identical for all $n\in\mathcal{N}_*$. It follows that

$$r_n(x_{n-},u_n^H)+Q_n^H(f_n(x_{n-},u_n^H))=r_n(x_{n-}^H,\hat{u}_n^H)+Q_n^{H+1}(f_n(x_{n-},\hat{u}_n^H)) \tag{4.3.20}$$

We now can establish the following result.

Lemma 8. *For every $n\in\mathcal{N}_*$,*

$$Q_n^{H+1}(f_n(x_{n-}^H,u_n^H))\le V_n(f_n(x_{n-}^H,u_n^H))+(d_{\mathcal{L}}-d(n)+1)\varepsilon.$$

Proof. We proceed by induction on n. When $n\in\mathcal{L}$, after step 2(b) of iteration H we

have for some $h < H$

$$\left\| x_n^h - f_n(x_{n-}^H, u_n^H) \right\|_\infty < \delta$$

so by Lemma 6

$$\begin{aligned} Q_n^{H+1}(f_n(x_{n-}^H, u_n^H)) &\leq q_n^h \\ &= V_n(x_n^h) \\ &\leq Q_n^{H+1}(f_n(x_{n-}^H, u_n^H)) + \varepsilon. \end{aligned}$$

by (5.3.8).

Assume as an inductive hypothesis that for every m with $d_m = d$ we have

$$Q_m^{H+1}(f_m(x_{m-}^H, u_m^H)) \leq Q_m^{H+1}(f_m(x_{m-}^H, u_m^H)) + (d_{\mathcal{L}} - d + 1)\varepsilon.$$

We show that for every node n with $d_n = d - 1$

$$Q_n^{H+1}(f_n(x_{n-}^H, u_n^H)) \leq Q_n^{H+1}(f_n(x_{n-}^H, u_n^H)) + (d_{\mathcal{L}} - d + 2)\varepsilon.$$

Since at termination of the algorithm

$$\|x_n^H - f_n(x_{n-}^H, u_n^H)\|_\infty < \delta,$$

it follows that

$$
\begin{aligned}
Q_n^{H+1}(f_n(x_{n-}^H, u_n^H)) &\leq q_n^H \\
&\text{by Lemma 6} \\
&= \sum_{m \in n+} p(m) \left(r_m(x_n^H, \hat{u}_m^H) + Q_m^{H+1}(f_m(x_n^H, \hat{u}_m^H) \right) \\
&\text{by (5.3.11)} \\
&= \sum_{m \in n+} p(m) \left(r_m(x_n^H, u_m^H) + Q_m^{H}(f_m(x_n^H, u_m^H) \right) \\
&\text{by (5.3.20)} \\
&\leq \sum_{m \in n+} p(m) \left(r_m(x_n^H, u_m^H) + V_m(f_m(x_n^H, u_m^H) + (d_{\mathcal{L}} - d_m + 1)\varepsilon \right) \\
&\text{induction hypothesis} \\
&\leq \sum_{m \in n+} p(m) \left(r_m(x_n^H, u_m^*) + V_m(f_m(x_n^H, u_m^*) \right) + (d_{\mathcal{L}} - d + 1)\varepsilon \\
&\text{optimality of } u^* \\
&= V_n(x_n^H) + (d_{\mathcal{L}} - d + 1)\varepsilon \\
&\text{definition of } V_n \\
&\leq V_n(f_n(x_{n-}^H, u_n^H)) + (d_{\mathcal{L}} - d + 2)\varepsilon
\end{aligned}
$$

where the last inequality follows from $\|x_n^H - f_n(x_{n-}^H, u_n^H)\|_\infty < \delta$ and (5.3.8). This establishes the result for nodes with depth $d-1$ and hence all $n \in \mathcal{N}_*$ by induction. ■

Theorem 4. *Suppose $\bar{\varepsilon} = 0$. Upon termination of the algorithm, the first-stage decisions u_m^H satisfy*

$$
\sum_{m \in 0+} p(m) \left(r_m(\bar{x}, u_m^H) + V_m(f_m(\bar{x}, u_m^H)) \right) \geq V_0(\bar{x}) - 2(d_{\mathcal{L}} - 1)\varepsilon.
$$

Proof. For every $m \in 0+$, $d(m) = 2$. So for the optimal first stage decisions u_m^*,

Lemma 7 gives

$$r_m(\bar{x}, u_m^*) + Q_m^H(f_m(\bar{x}, u_m^*)) \geq r_m(\bar{x}, u_m^*) + V_m(f_m(\bar{x}, u_m^*)) - (d_{\mathcal{L}} - 1)\varepsilon,$$

and by (5.3.11)

$$r_m(\bar{x}, u_m^H) + Q_m^H(f_m(\bar{x}, u_m^H)) \geq r_M(\bar{x}, u_m^*) + Q_m^H(f_m(\bar{x}, u_m^*)),$$

$$r_m(\bar{x}, u_m^H) + Q_m^H(f_m(\bar{x}, u_m^H)) \geq r_m(\bar{x}, u_m^*) + V_m(f_m(\bar{x}, u_m^*)) - (d_{\mathcal{L}} - 1)\varepsilon. \tag{4.3.21}$$

Now upon termination of the algorithm (5.3.19) gives

$$Q_m^H(f_m(\bar{x}, u_m^H) = Q_m^{H+1}(f_m(\bar{x}, u_m^H), \tag{4.3.22}$$

and Lemma 8 implies

$$Q_m^{H+1}(f_m(\bar{x}, u_m^H) \leq V_m(f_m(\bar{x}, u_m^H)) + (d_{\mathcal{L}} - 1)\varepsilon \tag{4.3.23}$$

so (5.3.21), (5.3.22), and (5.3.23) yield

$$\begin{aligned} & r_m(\bar{x}, u_m^H) + V_m(f_m(\bar{x}, u_m^H)) + (d_{\mathcal{L}} - 1)\varepsilon \\ \geq\ & r_m(\bar{x}, u_m^*) + V_m(f_m(\bar{x}, u_m^*)) - (d_{\mathcal{L}} - 1)\varepsilon \end{aligned}$$

so

$$\begin{aligned} & \sum_{m\in 0+} p(m)(r_m(\bar{x}, u_m^H) + V_m(f_m(\bar{x}, u_m^H))) \\ \geq & \sum_{m\in 0+} p(m)(r_m(\bar{x}, u_m^*) + V_m(f_m(\bar{x}, u_m^*))) - 2(d_{\mathcal{L}} - 1)\varepsilon \end{aligned}$$

giving the result. ■

4.3.2 Sampled MIDAS algorithm

We consider next a variant where, in the forward step, only one trajectory is explored, and no stopping test is given.

Algorithm 6 Sampled MIDAS

Set $k = 1$, and $\Omega_n^k = \emptyset$, for all $n \in \mathcal{N}$.

1. Forward pass: Set $x_0^k = \overline{x}$, and $n = 0$. While $n \notin \mathcal{L}$:

   (a) Sample $m \in n+$;

   (b) Solve $\max_{u\in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^k(f_m(x_n^k, u)) \right\}$ to give u_m^k;

   (c) If $\left\| f_m(x_n^k, u_m^k) - x_m^h \right\|_\infty < \delta$ for some $h < k$ then set $x_m^k = x_m^h$. Otherwise, set $x_m^k = f_m(x_n^k, u_m^k)$.

   (d) Set $n = m$.

2. Leaf node update: Set $\Omega_n^{k+1} := \Omega_n^k$, for all $n \in \mathcal{N}$.

   For the particular leaf node $n \in \mathcal{L}$ at the end of step 1:

   (a) Set $q_n^k = R(x_n^k)$ and $\Omega_n^{k+1} := \Omega_n^{k+1} \cup \{(x_n^k, q_n^k)\}$.

3. Backward pass: While $n > 0$:

   (a) Set $n = n-$;

(b) Compute

$$q_n^k = \sum_{m \in n+} \frac{p(m)}{p(n)} \left[\max_{u \in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^{k+1}(f_m(x_n^k, u)) \right\} \right]$$

attained at

$$\hat{u}_m \in \arg \max_{u \in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^{k+1}(f_m(x_n^k, u)) \right\}.$$

(c) Set $\Omega_n^{k+1} := \Omega_n^{k+1} \cup \{(x_n^k, q_n^k)\}$.

4. Increase k by 1 and go to step 1.

Following [58] we assume that sample paths satisfy the Forward Pass Sampling Property.

Forward Pass Sampling Property (FPSP):

Each node is visited infinitely many times with probability 1. (4.3.24)

There are many sampling methods satisfying this property. For example, one method is to select a child node at each node n in the forward pass, by independently sampling with a positive probability for each outcome $m \in n+$. This meets FPSP by the Borel-Cantelli lemma. Another sampling method that satisfies FPSP is to repeat an exhaustive enumeration of each scenario in the forward pass.

Recall that $d(n)$ is the depth of node n. The following result ensures that Q_n^k is a $(d_{\mathcal{L}} + 1 - d(n))\varepsilon$-upper bound on V_n. Its proof is identical to the one in the case of the full tree algorithm.

Lemma 9. *For every $n \in \mathcal{N}$, and for all iterations k,* (5.3.9) *holds.*

Theorem 5. *Almost surely, after finitely many (say H) iterations, the functions Q_n^k remain constant. Then, any policy is such that*

$$V_0(\bar{x}) \leq \sum_{n \in \mathcal{N}_* \setminus \mathcal{L}} p(n) r_n(x_{n-}^H, u_n^H) + \sum_{n \in \mathcal{L}} p(n) R\left(x_n^H\right) + d_{\mathcal{L}} \varepsilon. \tag{4.3.25}$$

and the first-stage decisions satisfy

$$\sum_{m \in 0+} p(m) \left(r_m(\bar{x}, u_m^H) + V_m(f_m(\bar{x}, u_m^H)) \right) \geq V_0(\bar{x}) - 2(d_{\mathcal{L}} - 1)\varepsilon.$$

Proof. Since the algorithm generates finitely many points, it is clear that the functions Q_n^k remain constant after finitely many iterations. Since the FPSP is satisfied, each node is a.s. visited infinitely many times. We easily see that the estimates of the full tree algorithm are then valid. The conclusion follows. ■

In practical implementations of MIDAS, we choose to stop after a fixed number H_{max} of iterations. We also remark that Algorithm 6 simplifies when the random variables are stagewise independent. In this case, the points (x_n^k, q_n^k) can be shared across all nodes having the same depth.

This means that there is a single approximation Q_n^k shared by all these nodes and updated once for all in each backward pass. The almost-sure convergence result for MIDAS applies in this special case, but one might expect the number of iterations needed to decrease dramatically in comparison with the general case.

4.4 Integer State Variables

MIDAS can be extended to solve multistage stochastic mixed-integer programs (SMIPs), when the dynamic, or the state equation $f_m(x_n^k, u_m^k)$, contains only integer variables. Based on the scenario tree from Section 5.3, we approximate each node value function

$V_n(x)$, $n \in \mathcal{N}_*$, with a piecewise constant function $Q_n^k(x)$, where $k = 1, 2, \ldots, H$ is the algorithm iteration. We have a sequence x_n^k in $X_n \subseteq \mathbb{Z}$. Associated with each x_n^k is a value q_n^k. For leaf nodes we have that

$$q_n^k := R(x_n^k), \quad n \in \mathcal{L}. \tag{4.4.1}$$

As before we define

$$\Omega_n^k = \{(x_n^k, q_n^k) : h = 1, \ldots, k-1\}, \quad \text{for all } n \in \mathcal{N}_* \setminus \mathcal{L}$$

We now define the *supporting indices* of $x \in X_n$ as

$$\mathcal{H}_n^k(x) := \{1 \leq h < k;\ x_i < x_{n,i}^k,\ i = 1, \ldots, N\}, \tag{4.4.2}$$

as well as the following function:

$$Q_n^k(x) = \begin{cases} \bar{V} & \text{if } \mathcal{H}_n^k(x) \text{ is empty,} \\ \min\left\{q_n^k : h \in \mathcal{H}_n^k(x)\right\} & \text{otherwise.} \end{cases} \tag{4.4.3}$$

In this formulation, we no longer need a δ. This is because when the dynamic contains all integers, MIDAS will produce feasible state trajectory. There must exist a feasible action between two integer state points for t and $t+1$. Otherwise the problem will become infeasible. By removing the δ, Algorithm 6 can be simplified into Algorithm 7. The main difference between the two algorithms is that Algorithm 7 does not have the additional step which sets state points that are within δ distance.

Algorithm 7 Sampled MIDAS with Integer State Variables

Set $k = 1$, and $\Omega_n^k = \emptyset$, for all $n \in \mathcal{N}$.

1. Forward pass: Set $x_0^k = \overline{x}$, and $n = 0$. While $n \notin \mathcal{L}$:

(a) Sample $m \in n+$;

(b) Solve $\max_{u \in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^k(f_m(x_n^k, u)) \right\}$ to give u_m^k;
Otherwise, set $x_m^k = f_m(x_n^k, u_m^k)$.

(c) Set $n = m$.

2. Leaf node update: Set $\Omega_n^{k+1} := \Omega_n^k$, for all $n \in \mathcal{N}$.

For the particular leaf node $n \in \mathcal{L}$ at the end of step 1:

(a) Set $q_n^k = R(x_n^k)$ and $\Omega_n^{k+1} := \Omega_n^{k+1} \cup \{(x_n^k, q_n^k)\}$.

3. Backward pass: While $n > 0$:

(a) Set $n = n-$;

(b) Compute

$$q_n^k = \sum_{m \in n+} \frac{p(m)}{p(n)} \left[\max_{u \in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^{k+1}(f_m(x_n^k, u)) \right\} \right]$$

attained at

$$\hat{u}_m \in \arg \max_{u \in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^{k+1}(f_m(x_n^k, u)) \right\}.$$

(c) Set $\Omega_n^{k+1} := \Omega_n^{k+1} \cup \{(x_n^k, q_n^k)\}$.

4. Increase k by 1 and go to step 1.

Algorithm 7 will converge to the optimal policy by iteratively exploring state trajectories. The algorithm will terminate when it has re-visited an existing state-trajectory for all nodes in the scenario tree (see Lemma 10). This can make the algorithm a brute force-based algorithm, where in the worst case it iterates through all state trajectories until it reaches the optimal policy.

Lemma 10. *If* $h < k$ *and* $\left\| x - x_n^h \right\|_\infty <= 0$ *then* $Q_n^k(x) \leq q_n^h$.

4.5 A MIP representation of $Q^H(x)$

Assume that $X = \{x : 0 \leq x_i \leq K_i,\ i = 1, 2, \ldots, N\}$, and let $V(x)$ be any upper semi-continuous function defined on X. Suppose for some points x^h, $h = 1, 2, \ldots, k$, we have $V(x^h) = q^h$. Recall $\bar{V} = \max_{x \in X} V(x)$,

$$\mathcal{H}^{k+1}(x) = \{1 \leq h \leq k : x_i^h > x_i - \delta,\ i = 1, 2, \ldots, N\},$$

$$Q^{k+1}(x) = \min\left\{\bar{V}, \min\left\{q^h : h \in \mathcal{H}^{k+1}(x)\right\}\right\}.$$

For $\delta > 0$, and for any $x \in X_\delta = \{x : \delta \leq x_i \leq K_i,\ i = 1, 2, \ldots, n\}$, define $\bar{Q}^{k+1}(x)$ to be the optimal value of the mixed integer program

MIP(x): max φ

subject to:

$$\begin{aligned}
\varphi &\leq q^h + (\bar{V} - q^h)(1 - w_h), && h = 1, 2, \ldots, k,\\
x_i &\geq x_i^h z_i^h + \delta, && i = 1, 2, \ldots, N,\\
\textstyle\sum_{i=1}^n z_i^h &= 1 - w_h, && h = 1, 2, \ldots, k,\\
w_h &\in \{0, 1\}, && h = 1, 2, \ldots, k,\\
z_i^h &\in \{0, 1\}, && i = 1, 2, \ldots, N,\\
& && h = 1, 2, \ldots, k.
\end{aligned}$$

Proposition 1. *For every $x \in X_\delta$,*

$$\bar{Q}^{k+1}(x) = Q^{k+1}(x).$$

Proof. For a given point $x \in X_\delta$, consider w_h, z_i^h, $i = 1, 2, \ldots, N$, $h = 1, 2, \ldots, k$ that are feasible for MIP(x). If $w_h = 0$, $h = 1, 2, \ldots, k$, then $\varphi \leq \bar{V}$ is the only constraint on φ and so $\bar{Q}^k(x) = \bar{V}$. But $w_h = 0$, $h = 1, 2, \ldots, k$ means that for every such h,

$z_i^h = 1$ for some component i giving

$$x_i \geq x_i^h + \delta.$$

Thus

$$\mathcal{H}^{k+1}(x) = \{h : x_i < x_i^h + \delta \text{ for every } i\} = \emptyset.$$

Thus $Q^{k+1}(x) = \bar{V}$ which is the same value as $\bar{Q}^{k+1}(x)$.

Now assume that the optimal solution to MIP(x) has $w_h = 1$ for some h. It suffices to show that

$$\bar{Q}^{k+1}(x) = \min\{q^h : h \in \mathcal{H}^{k+1}(x)\}.$$

First if $h \in \mathcal{H}^{k+1}(x)$ then $w_h = 1$. This is because choosing $w_h = 0$ implies $z_i^h = 1$ for some i, so for at least one i

$$x_i \geq x_i^h + \delta,$$

so $h \notin \mathcal{H}^{k+1}(x)$.

Now if $h \notin \mathcal{H}^{k+1}(x)$ then any feasible solution to MIP(x) can have either $w_h = 0$ or $w_h = 1$. Observe however that if

$$q^h < \min\{q^{h'} : h' \in \mathcal{H}^{k+1}(x)\}$$

for any such h then choosing $w_h = 1$ for any of these would yield a value of φ strictly lower than the value obtained by choosing $w_h = 0$ for all of them. So $w_h = 0$ is optimal for $h \notin \mathcal{H}^{k+1}(x)$. It follows that $\mathcal{H}^{k+1}(x) = \{h : w_h = 1\}$. Thus the optimal value of MIP(x) is

$$\bar{Q}^{k+1}(x) = \min\{q^h : w_h = 1\} = \min\{q^h : h \in \mathcal{H}^{k+1}(x)\} = Q^{k+1}(x).$$

■

4.6 Summary

In this chapter, we have proposed a method called MIDAS for solving multistage stochastic dynamic programming problems with monotonic value functions. We have demonstrated the almost-sure convergence in $2T\varepsilon$-optimal first stage decision a finite number of steps.

MIDAS as a general algorithm for solving multistage stochastic integer programming (MSIPs) problems does not necessarily require a piecewise constant approximation. As long as the alternative approach provides an ε-outer approximation of the value function, we can incorporate it into a MIDAS scheme, which will converge almost surely by the same arguments above. We can also extend the current approximations of MIDAS in several ways. For instance, a more accurate approximation might be achieved using an alternative formulation, that for example uses specially ordered sets to yield a piecewise-affine approximation.

In the following chapters (Chapters 6 and 7), we study several non-convex hydro-bidding problems. These hydro-bidding problems were formulated based on various key challenges, such as modeling price uncertainty, modeling headwater effects and redefining power generation to discrete quantities involving integer state variables. We use MIDAS to solve these hydro-bidding problems and compare the policies obtained with the incumbent SDDP equivalent. Our motivation is to assess how well MIDAS performs in comparison to SDDP in solving these problems.

Chapter 5

Solving hydro-bidding problems with integer state variables

In Chapter 3, we formulated the hydro-bidding problem, and applied the stochastic dual dynamic programming (SDDP) method to solve it. However, SDDP was unable to guarantee convergence if the value function is not convex (if minimizing) or not concave (if maximizing). Our motivation to solve hydro-bidding problems with non-concave value functions has led us to develop the mixed-integer dynamic approximation scheme (MIDAS). MIDAS uses step functions to approximate the value function. In Chapter 5, we presented a general MIDAS algorithm and proved its almost sure convergence with continuous and integer state variables.

In this chapter, we apply the MIDAS algorithm in order to solve a hydro-bidding model with integer state variables. The model developed in this chapter is based on HERBS (see Chapter 4 for description). It has discrete quantities where each station can choose from a predefined set of feasible power dispatches and equivalent water discharges. Incorporating discrete production quantities makes our hydro-bidding model a Stochastic Mixed Integer Program (SMIP), which means that the state variable (i.e.

reservoir storage) is integer. We solve this model using MIDAS and study its performance with respect to the quality of the computed policy and the approximation of the value function. We compare MIDAS with SDDP and illustrate that MIDAS produces better policies than SDDP due to its more accurate representation of the value function. However, solving these models, especially for large data sets, comes at computational cost, as for large problems MIDAS takes a long time and large number of steps to converge. The decrease in computational efficiency is due to the MIP representation of the sub-problems, which introduces several binary variables whenever a new step function is added.

5.1 The SMIP hydro-bidding model formulation

Consider Model (6.1.1), a hydro-bidding model with discrete production. This model represents an M-tranche offerstack with a Markov chain of prices modeled in the same way as the Markov chain described in Section 3.1 of Chapter 3.

$$HB(t,i,x_t):$$

$$V_{t,i}(x_t) = \max \sum_{j=1}^{M} \rho_{i,j}(t)\left[\pi_j o_{t,j} + V_{t+1,j}(x_{t+1,j})\right] \tag{5.1.1}$$

subject to:

$$x_{t+1,j,r} = x_{t,r} + \sum_{s\in\mathcal{S}} A_{r,s}u_{t,j,s} + \sum_{k\in\mathcal{R}} B_{r,k}l_{t,j,k} + \omega_{t,r} \quad \text{for } j = 1,\ldots,M, \text{ and } r \in \mathcal{R}, \tag{5.1.1a}$$

(5.1.1b)

$$u_{t,j,s} = \sum_{l \in L_{t,s}} \theta_{t,s,l} z_{j,s,l} \text{ for } j = 1, \ldots, M, \text{ and } s \in \mathcal{S}, \tag{5.1.1c}$$

$$o_{t,j} = \sum_{s \in \mathcal{S}} \sum_{l \in L_{t,s}} \eta_{t,s,l} z_{j,s,l} \quad \text{for } j = 1, \ldots, M, \tag{5.1.1d}$$

$$o_{t,j} \leq o_{t,j+1} \quad \text{for } j = 1, \ldots, M-1, \tag{5.1.1e}$$

$$\sum_{l \in L_{t,s}} z_{j,s,l} \leq 1 \quad \text{for } j = 1, \ldots, M, \text{ and } s \in \mathcal{S}, \tag{5.1.1f}$$

$$x_{t+1,j,r} \in [\underline{x}_r, \overline{x}_r] \quad \text{for } j = 1, \ldots, M, \text{ and } r \in \mathcal{R}, \tag{5.1.1g}$$

$$z_{j,s,l} \in \{0,1\} \quad \text{for } j = 1, \ldots, M, \quad s \in \mathcal{S}, \text{ and } l \in L_{t,s}, \tag{5.1.1h}$$

$$x_{t+1,j,r} \in \mathbb{Z} \quad \text{for } j = 1, \ldots, M, \text{ and } r \in \mathcal{R}, \tag{5.1.1i}$$

$$V_{t+1,j}(x_{t+1,j}) \in Q_{t+1,j} \ (x_{t+1,j}) \text{ for } j = 1, \ldots, M, \tag{5.1.1j}$$

$$V_{T,j}(x_{T,j}) = \mathbf{V}_{T,j}(x_{T,j}) \quad \text{for } j = 1, \ldots, M. \tag{5.1.1k}$$

where:

T	=	the number of stages in the planning horizon (i.e. $t = 1, 2, \ldots, T$),
$\mathcal{R}$	=	the set of Reservoir node labels,
$\mathcal{S}$	=	the set of station node labels,
M	=	the number of tranches in the offer stack as well as the number of Markov states,
$L_{t,s}$	=	the set of production pair labels for period t and station s,

$\rho_{i,j}(t)$ = the transition probability of being dispatched at tranche j given the previous dispatch was at tranche i at time t,

π_j = conditionally expected price (\$/MWh) for clearing tranche j,

$[(\theta_{t,s,l}, \eta_{t,s,l})]_{l \in L_{t,s}}$ = vector of production pairs (water discharge and equivalent power production) for station s in period t,

x_t = vector of starting storage levels (cubic meter) belonging to set X at the beginning of stage t for each reservoir in the hydro scheme,

$x_{t+1,j}$ = vector of storage levels (cubic meter) at the end of period t if tranche j is dispatched, and is based on the water-balance constraints (6.1.1a),

ω_t = vector of inflows (cubic meter) for each reservoir in the hydro scheme for period t,

$l_{t,j}$ = vector of reservoir spillage (cubic meter) for each reservoir in the hydro scheme for tranche j and period t,

$u_{t,j}$ = vector of turbine water discharge (cubic meter) for each station in the hydro scheme for tranche j and period t,

$z_{j,s,l}$ = $\begin{cases} 1 & \text{if production pair } l \text{ is chosen for offer } j \text{ at station } s, \\ 0 & \text{otherwise,} \end{cases}$

$o_{t,j}$ = offer quantity for tranche j in period t,

$V_{t+1,j}(x_{t+1,j})$ = value function representing the contributions of future stages based on the storage levels $x_{t+1,j}$ if being dispatched in tranche j.

At period t and station s there are $L_{t,s}$ pairs $(\theta_{t,s,l}, \eta_{t,s,l})$ that represent possible discrete power generation values at this station. The parameter $\theta_{t,s,l}$ represents the water discharge (in cubic meters) through the turbines, and $\eta_{t,s,l}$ represents the respective power (in MWh). These pairs are coupled, through constraints (6.1.1c) and (6.1.1d), by the binary variables $z_{j,s,l}$ that dictate which water discharge and power generation pairs are selected. When $z_{j,s,l} = 1$, pair l with $(\theta_{t,s,l}, \eta_{t,s,l})$ is selected. At most one discrete production pair can be selected for each station s, therefore constraint (6.1.1f) is added to ensure that only one $z_{j,s,l}$ can be equal to 1. Like the previous hydro-bidding models, $HB(t, i, x_t)$ is a block-based dispatch model where each offer tranche $o_{t,j}$ represents the total generation of the hydro scheme (see constraint (6.1.1d)). constraints (6.1.1a), (6.1.1g) and (6.1.1e) in $HB(t, i, x_t)$ are the standard constraints present in the hydro-bidding models that were previously discussed in Chapter 3. Constraint (6.1.1h) is the terminal value function at the end of the trading horizon. It represents the value of the water beyond the trading horizon. Constraint (6.1.1j) represents an arbitrary approximation of the value function by $\mathcal{Q}_{t+1,j}(x_{t+1,j})$. This approximation can be a set of constraints, like the hyperplanes in SDDP, or step functions in MIDAS.

5.2 Structure of the value function

The value function $V_{t+1,j}(x_{t+1,j})$ in Model (6.1.1) has a unique structure. As illustrated in Figure 6.1, the structure has distinct plateaus or 'steps', where the value function remains relatively the same for a range of state values, and then suddenly jumps to a higher value. This step-like structure has been observed by [68] for stochastic programs with integer variables in the recourse functions.

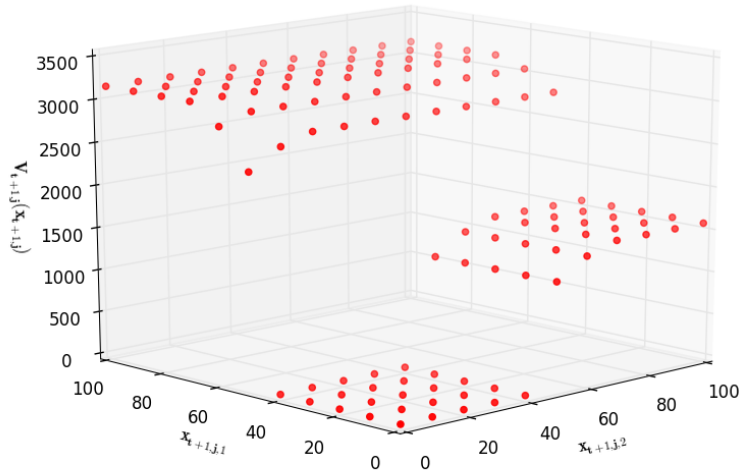


Figure 5.1: Example of the value function structure for 2 reservoir hydro scheme.

The presence of binary variables $z_{j,s,l}$ in the hydro-bidding model is the main reason why the value function exhibits the aforementioned structure. In the hydro-bidding model, each offer tranche $o_{t,j}$ defines a block dispatch that equals the total power generated across the hydro scheme. As a result of this block dispatch rule, there can be many power generation combinations across the hydro scheme which result in the same $o_{t,j}$. For example, consider the power generation combination for a hydro scheme consisting of 2 reservoirs and 2 stations in cascade, described in Table 6.2. Power generation combinations of $(55, 65)$ and $(65, 55)$ produce a block dispatch quantity of 120 MWh. It is also a similar case for other power generation combinations.

		$[\eta_{\mathbf{t},\mathbf{1},\ell}]_{\ell \in \mathbf{L_{t,1}}}$			
	$\mathbf{o_{t,j}}$	**0**	**55**	**65**	**70**
$[\eta_{\mathbf{t},\mathbf{2},\ell}]_{\ell \in \mathbf{L_{t,2}}}$	**0**	0	55	65	70
	55	55	110	120	125
	65	65	120	130	135
	70	70	125	135	140

Table 5.2: Example of the mapping between offer $o_{t,j}$ and power production η_t.

As each generation quantity $\eta_{t,s,l}$ corresponds to a respective water discharge $\theta_{t,s,l}$, it couples x_{t+1} to $o_{t,j}$ in a similar manner. Table 6.3 lists the offer $o_{t,j}$ for each x_{t+1} level for a 2 reservoir hydro-scheme. Each reservoir has a net capacity of 100 cubic meters of water, and their stations have the same vector $[\eta_{t,s,l}]_{l \in L_{t,s}}$ of power generation described by Table 6.2. For various combinations of storage levels between the upstream reservoir $x_{t+1,j,1}$ and downstream reservoir $x_{t+1,j,2}$ the offer quantity $o_{t,j}$ remains constant. For instance, when $x_{t+1,j,1} \geq 70$ the offer is 140 for all storage levels in the downstream reservoir. This is due to the maximum water discharge capacity of the upstream station being 70. Therefore, when $x_{t+1,j,1} \geq 70$ it discharges up to 70 cubic meters of water through the station in order to produce 70 MWh of power. This same volume of water is also passed through the downstream station in order to produce an equivalent amount of power, thus, resulting in $o_{t,j} = 140$. Table 6.3 also highlights similar instances of different, but constant, $o_{t,j}$ indicated by the different values. If the terminal value function $V_{T,j}(x_{T,j})$ exhibits such structure then, through recursion, this property is transferred to value functions in earlier stages, which results in the overall model inheriting this property.

		$\mathbf{x_{t+1,j,1}}$ (upstream)										
	$\mathbf{o_{t,j}}$	**0**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
$\mathbf{x_{t+1,j,2}}$ (downstream)	**0**	0	0	0	0	0	110	130	140	140	140	140
	10	0	0	0	0	0	120	135	140	140	140	140
	20	0	0	0	0	0	125	135	140	140	140	140
	30	0	0	0	0	0	125	135	140	140	140	140
	40	0	0	0	0	0	125	135	140	140	140	140
	50	55	55	55	55	55	125	135	140	140	140	140
	60	65	65	65	65	65	125	135	140	140	140	140
	70	70	70	70	70	70	125	135	140	140	140	140
	80	70	70	70	70	70	125	135	140	140	140	140
	90	70	70	70	70	70	125	135	140	140	140	140
	100	70	70	70	70	70	125	135	140	140	140	140

Table 5.3: Example of an offer $o_{t,j}$ for each storage level $x_{t+1,j}$ for a 2 reservoir hydro scheme.

5.3 Approximating the value function

As presented in Chapter 5, the MIDAS algorithm approximates the value function using step functions. In iteration H of the forward pass it generates a new state trajectory $(x_1^{H+1}, x_2^{H+1}, \ldots, x_T^{H+1}, x_{T+1}^{H+1})$ based on the current approximation of the value function $Q_{t+1,j}^H(x)$ consisting of H step functions for $j = 1, 2, \ldots, M$. Then, in the backward pass it updates $Q_{t+1,j}^H(x)$ based on the state trajectory, where $q_{t,j}^{H+1} = V_t(x_t^{H+1})$ for $t = 1, 2, \ldots, T+1$ and $j = 1, 2, \ldots, M$. It adds the new value function

estimate as a step function for $Q^H_{t+1,j}(x)$, where for a given $x_{t+1,j}$

$$V_{t+1,j}(x_{t+1,j}) \in Q^H_{t+1,j} = \min\left\{\overline{R}, q^h_{t+1,j} : \ h \in \mathcal{H}_{\mathbb{Z}}(x_{t+1,j})\right\}, \tag{5.3.1}$$

with its supporting indices

$$\mathcal{H}_{\mathbb{Z}}(x_{t+1,j}) = \{h' : \ x^{h'}_j \geq x_j \text{ for } h' = 1, 2, \ldots, H\}. \tag{5.3.2}$$

The term $\overline{R}$ is a known upper-bound of $V_{t+1,i}$. In order to illustrate how this actually works, consider a set of 4 step functions in Table 6.4 with their respective parameters x and q, that approximates the value function in Section 6.2. If $x_{t+1,j} = (40, 50)$ then $\mathcal{H}_{\mathbb{Z}}(x_{t+1,j}) = \{1, 3, 4\}$ because $x_{t+1,j,r} \leq x^h_{t+1,j,r}$ for $h = 2, 3, 4$. Based on $\mathcal{H}_{\mathbb{Z}}(x_{t+1,j})$, $Q^H_{t+1,j}(x_{t+1,j}) = \min\{\overline{R}, q^2, q^3, q^4\}$. Therefore, the estimate of the value function is 1462.5, which is the value of the second step function.

$h = 1, 2, \ldots, H$	x^h	q^h
1	$(40, 40)$	0
2	$(50, 60)$	1462.5
3	$(50, 100)$	1575
4	$(100, 100)$	3150

Table 5.4: Value function approximation using 4 step functions.

By placing the step functions at state points where there are adjacent plateaus, MIDAS can represent the step-like structure observed in the value function. Figure 6.2 compares the approximation by the step function in Table 6.4 with the exact value function. With 4 step functions, the structure of the value function can be approximated with relatively good accuracy. Adding more step functions will improve the accuracy of the approximation, but it does not always guarantee significant improve-

ments in the policy. Furthermore, it may add to the computation time to solve the MIP sub-problem.

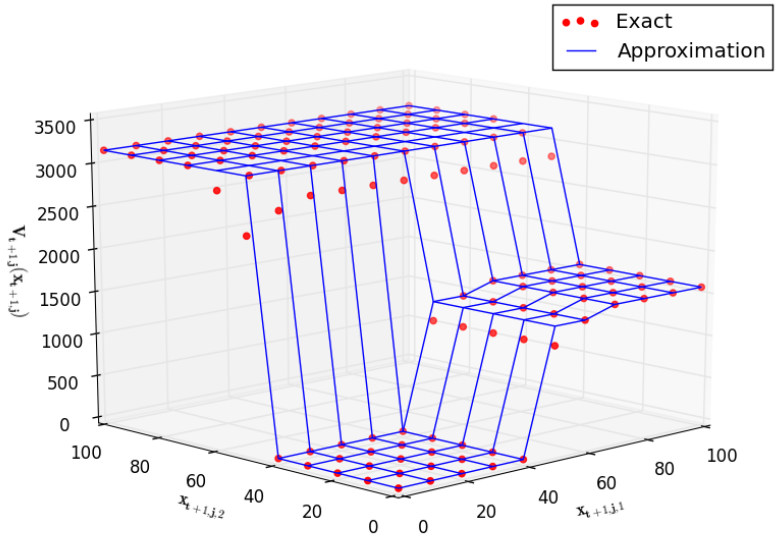


Figure 5.2: Comparison of the value function approximation with the exact value function.

In order to explicitly represent the value function approximation using the step functions the following constraints and variables need to be added to Model (6.1.1).

$$\hat{V}_{t+1,j} \leq q^h_{t+1,j} + (\overline{R} - q^h_{t+1,j})(1 - y_{h,j}) \text{ for } j = 1, \ldots, M, \text{ and } h = 1, \ldots, H, \tag{5.3.3}$$

$$\sum_{r \in R} w_{j,h,r} = 1 - y_{j,h} \quad \text{for } j = 1, \ldots, M, \text{ and } h = 1, \ldots, H, \tag{5.3.4}$$

$$x_{t+1,j,r} \geq (x^h_{t+1,j,r} + 1) w_{j,h,r} \quad \text{for } j = 1, \ldots, M,\ h = 1, \ldots, H, \text{ and } r \in \mathcal{R}, \tag{5.3.5}$$

$$y_{j,h} \in \{0, 1\} \quad \text{for } j = 1, \ldots, M, \text{ and } h = 1, \ldots, H, \tag{5.3.6}$$

$$w_{j,h,r} \in \{0, 1\} \quad \text{for } j = 1, \ldots, M,\ h = 1, \ldots, H, \text{ and } r \in \mathcal{R}. \tag{5.3.7}$$

where:

H	=	the iteration of the MIDAS algorithm, which also corresponds to the total number of step functions that have been added,
$\hat{V}_{t+1,j}$	=	continuous variable for the approximate value function for period $t+1$ and tranche j,
$y_{j,h}$	=	binary variable which chooses if step function h will be used to estimate the value function for tranche j,
$w_{j,h,r}$	=	binary variable which turns on or off the constraint $x_{t+1,j,r} \geq x^h_{t+1,j,r} w_{j,h,r}$ for tranche j, step function h, and reservoir r.

The variable $\hat{V}_{t+1,j}$ represents the $Q^H_{t+1,j}(x_{t+1,j})$ approximation. The binary variable $y_{j,h}$ selects which step function will be used to estimate the value of $\hat{V}_{t+1,j}$. Whenever $y_{j,h} = 1$ for the step function h, constraint (6.3.3) becomes binding by removing the term $\overline{R} - q^h_{t+1,j}$ making $\hat{V}_{t+1,j} \leq q^h_{t+1,j}$. Otherwise, it becomes non-binding with $\hat{V}_{t+1,j} \leq \overline{R}$. If $y_{j,h} = 1$ then constraint (6.3.4) forces $w_{j,h,r} = 0$ for all $r \in \mathcal{R}$, and in turn makes Constraint (6.3.5) non-binding. However, when $y_{j,h} = 0$ constraint (6.3.5) becomes non-binding, and constraint (6.3.4) enforces that $\sum_{r \in R} w_{j,h,r} = 1$. This entails that for at least one r, $w_{j,h,r} = 1$ and in turn enforces the storage variable in the r'th reservoir to be $x_{t+1,j,r} \geq (x^h_{t+1,j,r} + 1)w_{j,h,r}$. Based on the values of $y_{j,h}$ and $w_{j,h,r}$ for $h = 1, 2, \ldots, H$, the approximate value function $\hat{V}_{t+1,j} \leq \min\left\{\overline{R}, q^h_{t+1,j} : \; x^h_{t+1,j} \geq x_{t+1,j}, \text{ for } h = 1, 2, \ldots, H\right\}$, which is equivalent to definition of the $Q^H_{t+1,j}$ in Equation (6.3.1). The reader will notice that (6.3.3) to (6.3.7) are essentially the same as the MIP representation of $Q^H(x)$ in Appendix 5.5.

In order to visually illustrate how these constraints and variables represent the step-like structure of the value function, consider an example contour plot of $\hat{V}_{t+1,j}$ in Figure 6.3. Suppose that $x_{t+1,j} = (3, 1.5)$ indicated by the cross in the diagram. Then, $y_{j,2}$, $y_{j,3}$ and $y_{j,4}$ will all be equal to 1 making $\hat{V}_{t+1,j} \leq \min\left\{q^2_{t+1,j}, q^3_{t+1,j}, q^4_{t+1,j}, \overline{R}\right\} \leq 20$. As for the first step function $y_{j,1} = 0$, which makes the respective constraint $\hat{V}_{t+1,j} \leq q^1_{t+1,j} + \overline{R} - q^1_{t+1,j} \leq \overline{R}$ (i.e. non-binding).

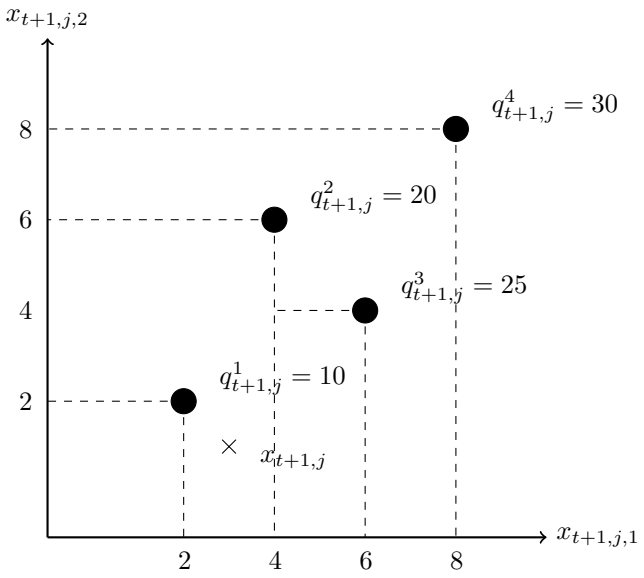


Figure 5.3: Example of upper bounded approximate of $\hat{V}_{t+1,j}$ by 4 step functions.

However, the step function 1 makes sure that $\sum_{r \in R} w_{j,1,r} = 1$ and $x_{t+1,j,r} \geq (x^1_{t+1,j,r} + 1)w_{j,1,r}$ for all $r \in \mathcal{R}$. This enforces that either $x_{t+1,j,1} \geq 3$ and $x_{t+1,j,2} \geq 1$ if $w_{j,1,1} = 1$, or $x_{t+1,j,1} \geq 1$ and $x_{t+1,j,2} \geq 3$ if $w_{j,1,2} = 1$. This restricts $x_{t+1,j}$ to the region between step function 1 and 2.

Essentially, constraints (6.3.4) and (6.3.5) create a set of hyperplanes which act as lower bounds on the value of variable $x_{t+1,j,r}$. They partition the feasible space of $x_{t+1,j}$ into regions of constant value function estimates, which mimics the observed structure of the value function. The contour plot in Figure 6.4 illustrates the shape of the example $\hat{V}_{t+1,j}$.

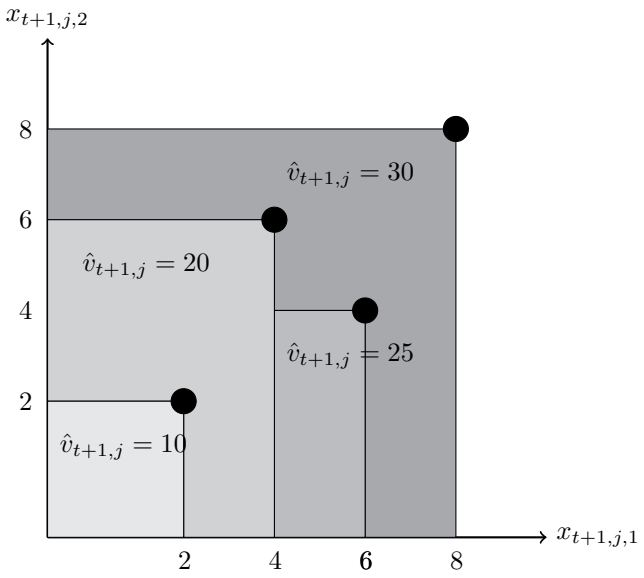


Figure 5.4: Contour of $\hat{V}_{t+1,j}$ by the 4 step functions.

By adding the aforementioned constraints and variables to Model (6.1.1) we can define its MIDAS equivalent, $\overline{HB}(t, i, x_t)$, defined by Model (6.3.8).

$\overline{HB}(t, i, x_t)$:

$$\begin{aligned} \overline{V}_{t,i}(x_t) = \quad & \max \sum_{j=1}^{M} P_{i,j}(t) \left[\pi_{t,j} o_{t,j} + \hat{V}_{t+1,j} \right] \\ & \text{subject to:} \end{aligned} \tag{5.3.8}$$

$$x_{t+1,j,r} = x_{t,r} + \sum_{s\in\mathcal{S}} A_{r,s}u_{t,j,s} + \sum_{k\in\mathcal{R}} B_{r,k}l_{t,j,k} + \omega_{t,r} \text{ for } j = 1,\ldots,M, \text{ and } r \in \mathcal{R}, \tag{5.3.8a}$$

$$u_{t,j,s} = \sum_{l\in L_{t,s}} \theta_{t,s,l}z_{j,s,l} \text{ for } j = 1,\ldots,M, \text{ and } s \in \mathcal{S}, \tag{5.3.8b}$$

$$o_{t,j} = \sum_{s\in S}\sum_{l\in L_{t,s}} \eta_{t,s,l}z_{j,s,l} \quad \text{for } j = 1,\ldots,M, \tag{5.3.8c}$$

$$o_{t,j} \leq o_{t,j+1} \quad \text{for } j = 1,\ldots,M-1, \tag{5.3.8d}$$

$$\sum_{l\in L_{t,s}} z_{j,s,l} \leq 1 \quad \text{for } j = 1,\ldots,M, \text{ and } s \in \mathcal{S}, \tag{5.3.8e}$$

$$x_{t+1,j,r} \in [\underline{x}_r, \overline{x}_r] \quad \text{for } j = 1,\ldots,M, \text{ and } r \in \mathcal{R}, \tag{5.3.8f}$$

$$z_{j,s,l} \in \{0,1\} \quad \text{for } j = 1,\ldots,M, \quad s \in \mathcal{S}, \text{ and } l \in L_{t,s}, \tag{5.3.8g}$$

$$x_{t+1,j,r} \in \mathbb{Z} \quad \text{for } j = 1,\ldots,M, \text{ and } r \in \mathcal{R}, \tag{5.3.8h}$$

$$\hat{V}_{t+1,j} \leq q^h_{t+1,j} + (\overline{R} - q^h_{t+1,j})(1 - y_{h,j}) \text{ for } j = 1, \ldots, M, \text{ and } h = 1, \ldots, H, \tag{5.3.8i}$$

$$\sum_{r \in R} w_{j,h,r} = 1 - y_{j,h} \quad \text{for } j = 1, \ldots, M, \text{ and } h = 1, \ldots, H, \tag{5.3.8j}$$

$$x_{t+1,j,r} \geq (x^h_{t+1,j,r} + 1) w_{j,h,r} \quad \text{for } j = 1, \ldots, M,\ h = 1, \ldots, H, \text{ and } r \in \mathcal{R}, \tag{5.3.8k}$$

$$y_{j,h} \in \{0, 1\} \quad \text{for } j = 1, \ldots, M, \text{ and } h = 1, \ldots, H, \tag{5.3.8l}$$

$$w_{j,h,r} \in \{0, 1\} \quad \text{for } j = 1, \ldots, M,\ h = 1, \ldots, H, \text{ and } r \in \mathcal{R}, \tag{5.3.8m}$$

$$V_{T,j}(x_{T,j}) = \mathbf{V}_{T,j}(x_{T,j}) \text{ for } j = 1, \ldots, M. \tag{5.3.8n}$$

5.4 MIDAS algorithm description for hydro-bidding

Algorithm 8 describes the MIDAS algorithm for solving Model (6.3.8). It is very similar to the SDDP algorithm, but is rewritten here for convenience and to illustrate the modelling of the Markov process of prices. As in SDDP, convergence is reached if the upper bound $\overline{V}_{1,\hat{i}_1}(\hat{x})$ is within a defined confidence level of the lower bound $\overline{V}_{\text{low}}$. In this chapter, we use a confidence level of 95% to test for convergence. MIDAS terminates by default if convergence is not reached before reaching the maximum iterations $H_{\max}$,

Algorithm 8 : MIDAS algorithm for solving the hydro-bidding model (6.3.8).

1. Initialize $\hat{x}$, $H_{\max}$, k, $\hat{i}_1$, N
2. Set $q^1_{2,i} = \overline{R}$ at $x^1_{1,i} = [\overline{x}_r]_{r\in R}$ for $i = 1, 2, \ldots, M$
3. Set $H = 2$
4. **Forward pass:**
   (a) Randomly generate a sequence of Markov states $\{\hat{i}_t\}_{t=1,2,\ldots,T}$ starting from the initial state $\hat{i}_1$.
   (b) For $t = 1, 2, \ldots, T$ do,
      i. solve $\overline{\mathrm{HB}}(t, \hat{i}_t, x^H_t)$ and set $x^{H+1}_{t+1} = x_{t+1,\hat{i}_{t+1}}$.
5. **Backward pass:**
   (a) For $t = T, \ldots, 3, 2, 1$ and for $i = 1, 2, \ldots, M$ do,
      i. solve $\overline{\mathrm{HB}}(t, i, x^{H+1}_t)$ and compute $\overline{V}_{t,i}(x^{H+1}_t)$,
      ii. update $\overline{\mathrm{HB}}(t-1, i, x^{H+1}_t)$ by adding the step function $q^{H+1}_{t,i} = \overline{V}_{t,i}(x^{H+1}_t)$ at point $x^{H+1}_{t,i} = x^{H+1}_t$.
6. **Convergence test:** If $H(\mod k) = 0$ then do the following:
   (a) Independently generate N sequences of Markov states $\{\hat{i}_t\}_N$ starting from the initial state $\hat{i}_1$,
   (b) for $l = 1, \ldots, N$ do,
      i. solve $\overline{\mathrm{HB}}(1, \hat{i}_1, \hat{x})$ and compute $\overline{V}_{1,\hat{i}_1}(\hat{x})$,
      ii. set $x_{1,l} = \hat{x}$, then for $t = 1, 2, \ldots, T+1$ do,

A. solve $\overline{\text{HB}}(t, \hat{i}_{t,l}, x_t)$, set $V_{\text{low},l} = V_{\text{low},l} + \pi_{t,\hat{i}_{t,l}} q_{t,\hat{i}_{t,l}}$ and $x_t = x_{t+1,\hat{i}_{t,l}}$.

(c) Compute $\overline{V}_{\text{low}} = \mathbb{E}_{\{1,\ldots,N\}}[V_{\text{low}}]$ and $\sigma_{\text{low}} = \text{SD}_{\{1,\ldots,N\}}(V_{\text{low}})$.

(d) If $\overline{V}_{\text{low}} - \frac{1.96}{\sqrt{N}}\sigma_{\text{low}} \leq \overline{V}_{1,\hat{i}_1}(\hat{x}) \leq \overline{V}_{\text{low}} + \frac{1.96}{\sqrt{N}}\sigma_{\text{low}}$ or $H > H_{\max}$, then stop, otherwise set $H = H + 1$ and go to Step 4.

Observe that this algorithm is different the MIDAS algorithm in Chapter 5 for solving SMIPs. The key difference between these two algorithms is the criteria of convergence. In the original algorithm, it converges when it visits an already sampled state trajectory across all the scenarios of the scenario tree. In this algorithm, it converges when the upper bound and the lower bound are within a certain level of confidence. This, less strict convergence criteria, does not guarantee the policy is the optimum policy after convergence. However, it does indicate the degree of optimality. Moreover, if the user of the algorithm is happy with the gap between the upper and lower bound, they could terminate the algorithm early to obtain a reasonably good policy.

5.5 Comparison of MIDAS and SDDP

In the previous section, we described the standard algorithm of MIDAS for solving the hydro-bidding problem with discrete productions. In this section, we compare the performance of MIDAS with SDDP with respect to the quality of the value function approximation, and the performance of the policies of each method.

We apply Algorithm 8 to the following hydro scheme, illustrated by Figure 6.5. The data for this hydro scheme are based on the same elementary reservoir used in Section 3.3 of Chapter 3 for the same reason, that this makes it easier to analyze, compare, and illustrate the performance of each of the algorithms.

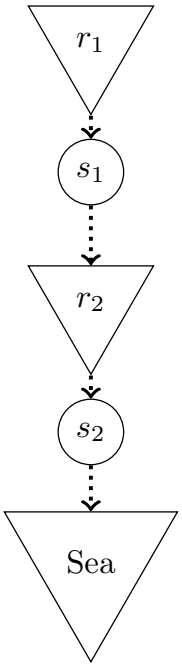


Figure 5.5: Topology of the hydro scheme.

In Figure 6.5, the two cascading reservoirs, called r_1 and r_2, each have their own respective hydropower production stations s_1 and s_2. In order to compute an analytical solution to this stochastic program, the reservoirs and stations are identical, and have the following reservoir storage bounds and discrete power production states listed in Table 6.6. We solve this problem using the same three tranche offer stack price process as in Section 3.3 of Chapter 3 (see Figure 3.9) in a 4 stage (i.e. $T = 4$) trading horizon.

Parameter	Value
$\underline{x}$	$\begin{bmatrix} 0 & 0 \end{bmatrix}$
$\overline{x}$	$\begin{bmatrix} 200 & 200 \end{bmatrix}$
(θ, η)	$\begin{pmatrix} 0 & 0 \end{pmatrix}$ $\begin{pmatrix} 50 & 55 \end{pmatrix}$ $\begin{pmatrix} 60 & 65 \end{pmatrix}$ $\begin{pmatrix} 70 & 70 \end{pmatrix}$
$\mathbf{V}_{T,j}(x_{T,j})$	0 for $j = 1, \ldots, M$

Table 5.6: Model parameters for the 2 reservoir, 2 station of the test hydro scheme.

We carry out a similar analysis to Chapter 3 for measuring the optimality of the policies computed by MIDAS and SDDP. First, an analytical solution to Model (6.1.1) is computed using a standard scenario tree based model as in Section 3.2.2 of Chapter 3. This represents a baseline measure, which can be used to compare the optimality of the policies computed by MIDAS and SDDP.

In this analysis, we executed both MIDAS and SDDP for a maximum of 500 iterations (i.e. $H_{\max} = 500$) for each combination of the initial storage levels. We began with the starting Markov state at 1 (i.e. $\hat{i}_0 = 1$) and tested for convergence after every five iterations $k = 5$ with a sample size $N = 30$. For all initial storage states, MIDAS converged under $H_{\max}$, whereas SDDP did not. We discuss why this is the case further in this section. Once these algorithms converged, or terminated due to reaching the maximum iteration, we extracted the first stage objective and then simulated each policy across all scenarios in the scenario tree. By simulating the policies across the complete scenario tree, we can remove any sampling errors. Then the differences in the expected return from the policy can be purely attributed to their respective policies.

We ran a slightly different SDDP algorithm. As the sub-problem is a mixed-integer program, we relaxed the sub-problem in the backward pass in order to obtain the dual variable values in order to construct the hyperplanes. This is a similar approach to that taken by [52, 53] who used SDDP to solve power system investment planning problems. The issue with this approach is that the convergence criteria based on the confidence interval is no longer theoretically valid. The relaxed upper bound may never fall within the confidence interval of the integer lower bound. However, this does not mean that the problem will never converge.

Figures 6.6 and 6.7 illustrate the respective upper bounds of the policy computed by MIDAS and SDDP. These are obtained from the objective of the first stage sub-problem when the algorithm converges. As seen in Figure 6.6, MIDAS, through its

step functions, approximates the plateaus of the exact optimum value (red points), except for highest storage levels. For these points MIDAS cannot produce any better upper bound values because adding additional step functions does not improve the value function approximation. This is observed in the figure by high upper bound values indicated by the sudden 'spikes'. As depicted in Figure 6.7, SDDP convexifies the overall value function, which is reflected in the upper bound value of the SDDP policy. As result of this convexification, SDDP misrepresents the plateaus observed in the exact values. There are several regions in the storage state space where the upper bound shows non-concavity. These are depicted as five clear 'humps', and arise from SDDP failing to converge within the maximum number of iterations because of a very poor approximation of the value function.

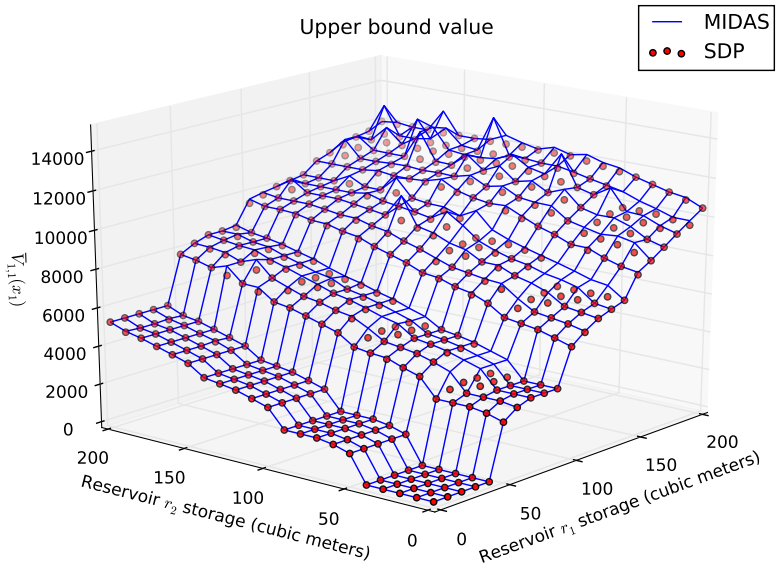


Figure 5.6: Upper bound value of MIDAS across various initial storage states.

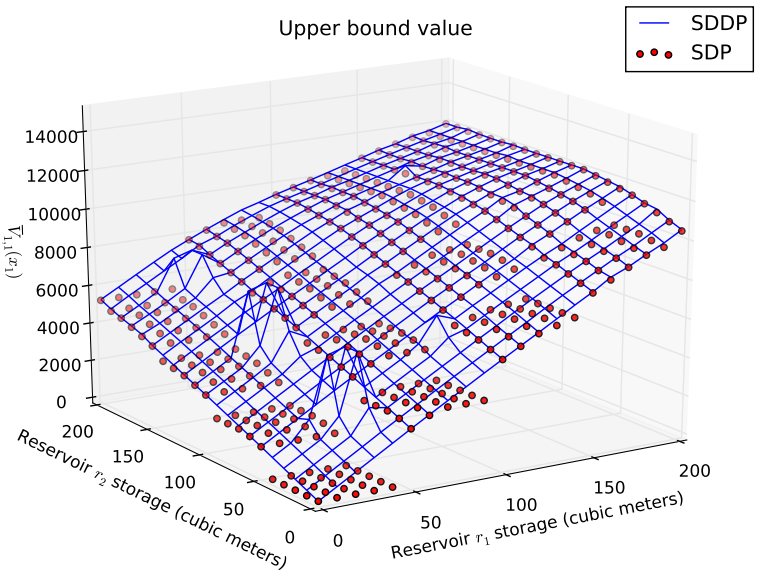


Figure 5.7: Upper bound of SDDP across various initial storage states.

The 'humps' occur in regions of the storage state space where there are large differences in the value function estimate between adjacent plateaus. Since SDDP uses hyperplanes to approximate the value function, it overestimates the value function in these regions and, when testing for convergence, the upper bound never reaches the 95% confidence interval of the lower bound. For example, when the initial storage $x_1 = (40, 40)$ the exact value function is 0 whereas SDDP estimates the value function as 6100.25. In these regions SDDP starts to cycle. Due to the bad approximation of the value function, the sub-problems produce the same state trajectory in each iteration. Therefore, after a finite number of iterations SDDP starts to generate the same cuts that fail to improve the value function approximation and the upper bound. Since MIDAS uses step functions, it is able to estimate the value function at $x_1 = (40, 40)$ as a step function with a value function estimate of 0, thus producing a better upper bound on the policy.

$T = 4$, $M = 3$	**Lower quartile**	**Mean**	**Median**	**Upper quartile**
MIDAS	93%	96%	98%	100%
SDDP	82%	89%	90%	97%

Table 5.7: Simulated profit from the offer policy of MIDAS and SDDP as a proportion of the optimal profit, averaged across the various storage states.

In order to measure how close each of the computed policies of MIDAS and SDDP is to the optimal policy, we calculated how close the expected profit, from the simulated policy, is to the profit of the optimum policy and averaged this measure among all initial storage level combinations. Table 6.7 summarizes, on average, how well the simulated policies of MIDAS and SDDP performed in relation to the true optimum policy. Based on Table 6.7, the offer policies produced by MIDAS yielded on average 96% of the optimum policy value, while the offer policies produced by SDDP yielded on average 89%. In fact, in all measures, from Median to upper and lower quartiles, MIDAS consistently produced policies that were better than SDDP for this hydro-bidding problem. A closer inspection of the simulated policies, illustrated by Figures 6.8 and 6.9, indicates that SDDP under performs in areas where there is strict non-concavity, such as when $x_1 = (100, 40)$, where the SDDP value is 57.4% of the optimum policy value as opposed to MIDAS which is 99%. However, MIDAS under performs for high storage levels, where the approximation is poor.

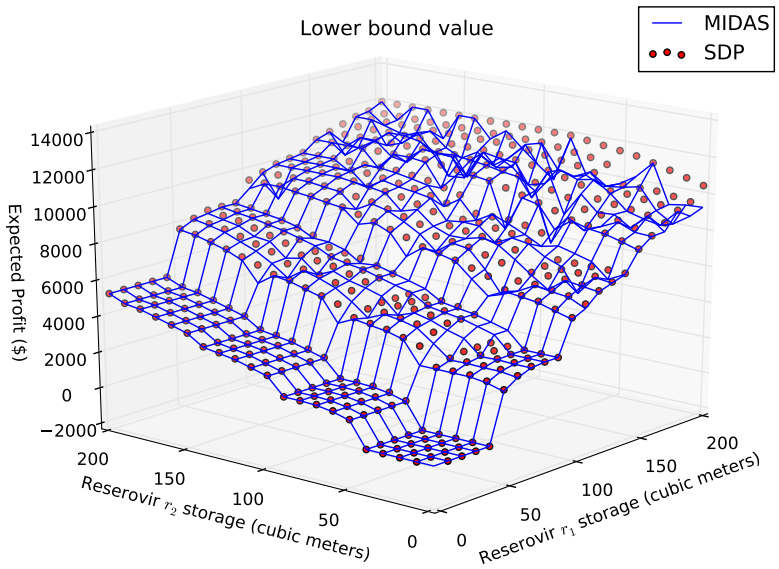


Figure 5.8: Expected Profit from a policy computed by MIDAS under simulation.

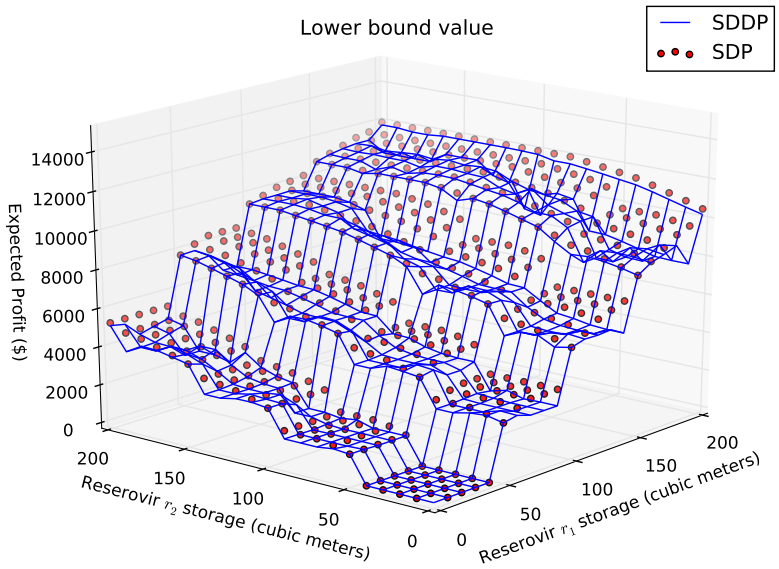


Figure 5.9: Expected Profit from a policy computed by SDDP under simulation.

Based on these findings, we can conclude that MIDAS indeed can produce better policies than SDDP for solving multistage stochastic mixed-integer programming problems. MIDAS is able to approximate the non-concave value function unlike SDDP.

This gives MIDAS a greater advantage to computing near-optimum policies, as we have observed in Table 6.7. However, MIDAS is slower than SDDP due to its continuously increasing MIP sub-problem, and for this reason SDDP is computationally more efficient. We can see that further developments of the MIDAS algorithm need to address the issue of speed. In Chapter 8, we present some heuristics to speed up the solution time of MIDAS.

5.6 Summary

In this chapter we presented a stochastic, multistage hydro-bidding model with integer state variables. We presented a multistage hydro-bidding model, Model (6.1.1), which is based on the underlying HERBS model in Chapter 4. In this model the actions, which are water discharged for power generation, were modeled as a set of discrete production quantities, similar to HERBS. Due to the integer state variables, the structure of the value function is non-concave (Section 6.2), where it exhibits 'steps' or plateaus. This makes step functions ideal for approximating the value function.

We reformulated the hydro-bidding model into a MIDAS-based hydro-bidding model and applied a variant of the MIDAS algorithm presented in Chapter 5. The key difference between the original MIDAS algorithm and the one proposed in this chapter is the convergence criterion. The original MIDAS algorithm converges when it samples a state trajectory that has already been sampled previously for all scenarios of the scenario tree. In Algorithm 8 the convergence criterion is the same as that of SDDP, where if the upper bound is within a confidence level of the lower bound then the algorithm stops.

In order to analyze the performance of MIDAS, we solved a 2 reservoir hydro-bidding model and compared its offer policies with the SDDP equivalent. MIDAS approx-

imates the plateaus of the exact value function with greater accuracy than SDDP (Figures 6.6 and 6.7). SDDP produces a convex value function approximation and therefore misrepresents these non-concave plateaus. Based on their approximations, we demonstrate that the offer policies of MIDAS were better than the policies computed by SDDP (Table 6.7). However, MIDAS was slower in computation time than SDDP. This was due to the sub-problem, which is a mixed-integer program, that continuously added binary variables to represent new step function in the model. As a consequence, the time it took to solve a sub-problem increased with each iteration. In the next chapter, Chapter 7, we study non-concave hydro-bidding models with continuous state variables.

Chapter 6

Solving hydro-bidding problems with continuous state variables

In Chapter 5, we introduced the MIDAS algorithm for solving non-convex multistage stochastic optimization problems, and proved its convergence for integer and continuous state variables. Then, in Chapter 6 we illustrated that MIDAS can produce better policies than SDDP for hydro-bidding problems when the state variable is integer. Through the use of step functions, MIDAS constructs more accurate approximations of the monotonic, non-concave value functions than SDDP, and produces near-optimal policies. However, we observed MIDAS is slower to converge than SDDP. This is due to the MIP representation of the sub-problems, which increases in size when adding step functions in each iteration of the algorithm. In this chapter, we study hydro-bidding problems with continuous state variables. We focus on two modelling challenges of the hydro-bidding problem. The first is modelling complex price processes, such as a mean reverting autoregressive price process, and the other is modelling the headwater effects for the power generation function.

These two features are very important for New Zealand hydroproducers such as Mer-

dian. Since the majority of New Zealand's electricity is generated from hydropower, effective management of water levels is important for flexibility, specifically during drought situations [49]. Moreover, New Zealand has an intra-day power market, which creates opportunities for hydroproducers to arbitrage price differences between different periods of the day. Therefore, representing autocorrelated price processes (see Figure 7.1) into the hydro-bidding models will enable these hydroproducers to construct good offer policies.

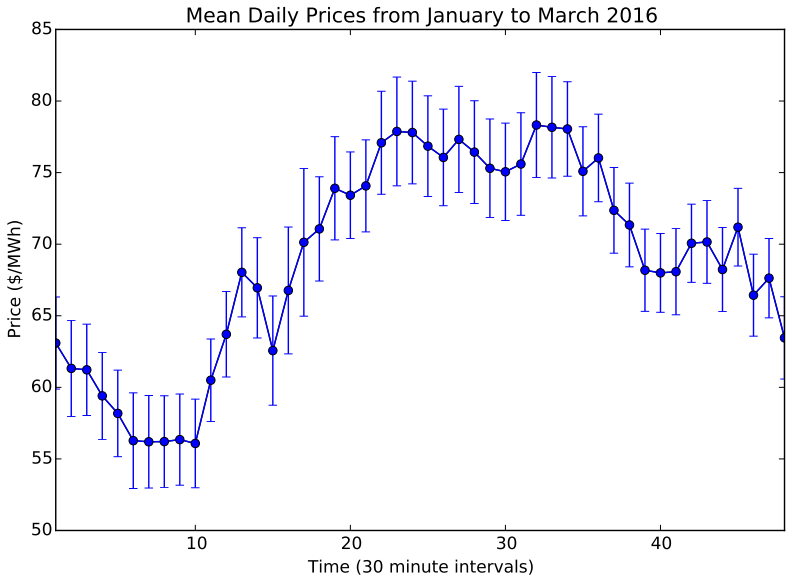


Figure 6.1: Example of mean daily prices at price node OTA2201 from January to March 2016 with 95% confidence bands.

We first present a hydro-bidding model with an autoregressive price process in Section 7.1. This model has value functions that are dependent on both price and storage, which makes them non-concave, and so difficult to incorporate into SDDP. We reformulate the model within the MIDAS framework in Section 7.1.1. We approximate the value function as a function of price and storage through the use of the step functions from MIDAS. We also extend the model to be within the SDDP framework by using a cut interpolation technique in Section 7.1.2. The cut interpolation technique super-

imposes on the scenario tree of prices a lattice of price nodes. Each of the nodes in the lattice stores the cuts generated in the backward pass. In the forward pass, the value function is interpolated between an upper and lower node from the lattice. This allows us to sample the large scenario tree that is created from discretizing the auto correlated price process.

In Section 7.2, we introduce a hydro-bidding model which incorporates headwater effects into the power generation function. We present a unique formulation which decouples the flow and head variables using differences of squares. We linearize the two quadratic equations using piecewise linear functions so that we can implement them into the linear models of MIDAS and SDDP. In the backward pass of SDDP we relax the integer variables so that we can compute the dual variables and construct its hyperplanes. This is because the formulation uses special ordered sets type II (SOS type II) which introduces integer variables [12]. We compare the MIDAS and SDDP based models on the performance of their policies and computational efficiency.

6.1 Modelling Price Uncertainty

Consider a set of reservoirs $\mathcal{R}$, a set of stations $\mathcal{S}$, and a hydro-bidding model for this system over T periods. This is formulated as

$$\begin{aligned} V_t(x_t, p_t) &= \mathbb{E}_{p_t}\left[\max_{(v_t,l_t)\in U(x_t)} \{r_t(g(v_t), p_t) + V_{t+1}(f_t(x_t, v_t, l_t, p_t))\}\right] \\ V_{T+1}(x_{T+1}, p_{T+1}) &= R(x_{T+1}, p_{T+1}). \end{aligned} \tag{6.1.1}$$

In Model (7.1.1), $(v_t, l_t) \in U(x_t) \subseteq \mathbb{R}^{|\mathcal{S}|}$ is the action which represents the volumes of water discharged for generation v_t and for spill l_t. The random variable, p_t, is the market clearing price. We assert that $f_t(x_t, v_t, l_t, p_t)$ is a continuous state transition

function. In the hydro-bidding context, (x_t, p_t) represents both a reservoir stock variable $x_t \in X \subseteq \mathbb{R}^{|\mathcal{R}|}$ and price variable p_t. We refer the reader to Section 3.1 for a description of the compact sets $U(x_t)$ and X.

In this chapter, we model p_t using an autoregressive, mean reverting price process with $f_t(x_t, v_t, l_t, p_t)$ defined as

$$\begin{bmatrix} x_{t+1} \\ p_{t+1} \end{bmatrix} = \begin{bmatrix} x_t - Av_t - Bl_t + \omega_t \\ \alpha_t p_t + (1-\alpha_t)b_t + \eta_t \end{bmatrix}, \tag{6.1.2}$$

where ω_t is a deterministic reservoir inflow (see Assumption 2 in Section 3.1), and η_t is the error term for an autoregressive price model.

The reward $r_t(g(v_t), p_t)$ in each stage is the revenue earned by the released energy $g(v_t)$ sold at price p_t, which is defined as

$$r_t(g(v_t), p_t) = p_t g(v_t). \tag{6.1.3}$$

The terminal reward $R(x_{T+1}, p_{T+1})$ represents the value of the remaining waterstock for prices beyond the time horizon. It is continuous and monotonic increasing, but not necessarily a concave function. Given an initial state $(\overline{x}, \overline{p})$, we seek an optimal policy yielding $V_1(\overline{x}, \overline{p})$. Here $V_t(x_t, p_t)$ denotes the maximum expected reward from the beginning of stage t onwards given the waterstock is x_t and price p_t. We discharge v_t for power generation and l_t for spill after observing the random price p_t. For such a model, $V_t(x_t, p_t)$ is continuous and monotonic increasing in x_t and p_t, but in general $V_t(x_t, p_t)$ is not concave (Figure 7.2), which makes it hard to approximate in SDDP using cutting planes.

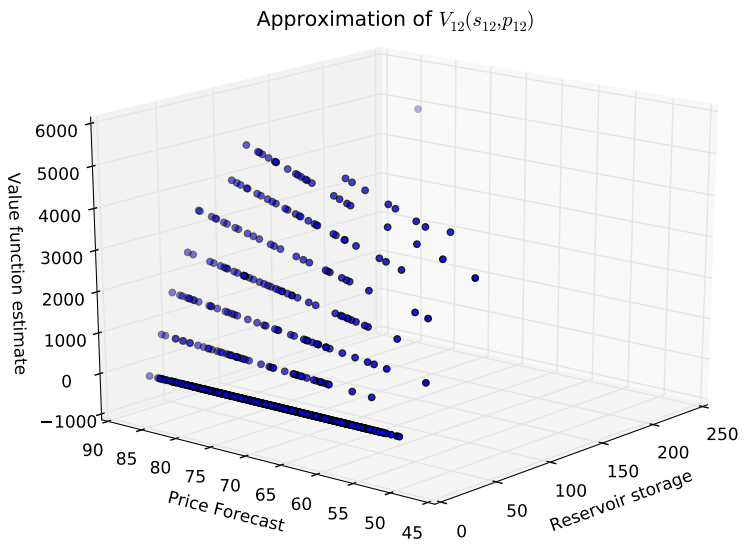


Figure 6.2: Example of a value function $V_t(x_t, p_t)$.

In order to incorporate the autoregressive, mean reverting price process within MIDAS and SDDP, the noise term η_t is approximated as a discrete random variable $\tilde{\eta}$. The discretized noise term $\tilde{\eta}$ consists of a set of L values ($\Omega = \{\hat{\eta}_1, \hat{\eta}_2, \ldots, \hat{\eta}_L\}$ for $t = 1, 2, \ldots, T$) with a corresponding set of probabilities $\{\rho_1, \rho_2, \ldots, \rho_L\}$. Then, the new price process is

$$p_{t+1} = \alpha_t p_t + (1 - \alpha_t) b_t + \tilde{\eta}, \text{ with } p_1 = \overline{p}. \tag{6.1.4}$$

The price process can now be represented as a finite scenario tree, but its size could be astronomically large. Figure 7.3 illustrates an example of a 12 stage price process with 5 discrete $\tilde{\eta}$ values.

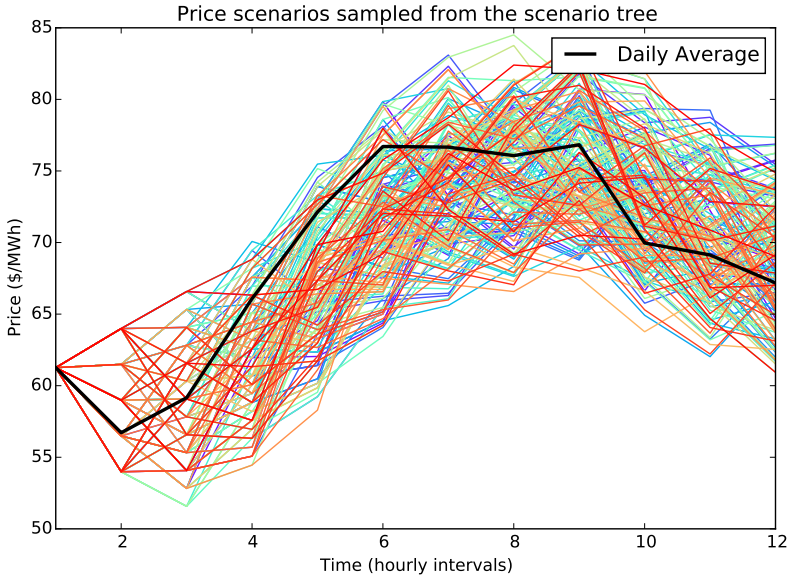


Figure 6.3: Sampled price scenarios from a 12 stage price process with 5 discrete η_t values.

The approximate price process now can be incorporated inside MIDAS and SDDP.

6.1.1 MIDAS-based approach

The value function $V_t(x_t, p_t)$ is approximated in MIDAS by adding step functions with a price dimension. The estimate q_t^h is the value function estimate at x_t^h and p_t^h. An example is illustrated in Figure 7.4 for a single reservoir, where at $(x_t, p_t) = (175, 17.5)$ the approximated value function is 4000.

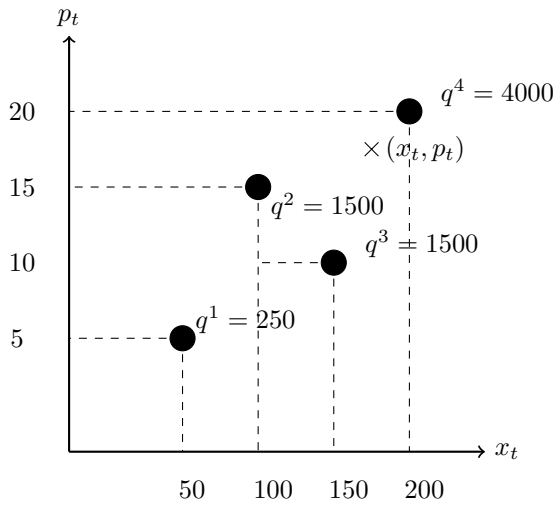


Figure 6.4: Value function approximation at (x_t, p_t), as shown by the cross is $q^4 = 4000$.

The value function approximation $Q^H_{t+1}(x_{t+1}, p_{t+1})$ is represented by the objective of Model (7.1.5) based on x_{t+1} and p_{t+1}. This model is similar to the MIP representation of $Q^H(x)$ in Chapter 5, which is discussed in greater detail in Section 5.5. There are two different δ values, one in the price dimension p_{t+1} and one in the state dimension x_{t+1} in order to reflect the differences in their scale.

$$Q^H_{t+1}(x_{t+1}, p_{t+1}) = \max \quad \varphi \tag{6.1.5}$$

$$\text{s.t.}$$

$$\varphi \leq \quad q^h_{t+1} + (\overline{R} - q^h_{t+1})(1 - w^h) \quad \text{for } h = 1, 2, \ldots, H, \tag{6.1.5a}$$

$$x_{t+1,r} \geq \quad x^h_{t+1,r} z^h_r + \delta_x \quad \text{for } h = 1, 2, \ldots, H, \tag{6.1.5b}$$

$$\text{and } r \in \mathcal{R},$$

$$p_{t+1} \geq \quad p^h_{t+1} z^h_p + \delta_p \quad \text{for } h = 1, 2, \ldots, H, \tag{6.1.5c}$$

$$\sum_{r\in\mathcal{R}} z_r^h = \quad 1 - w^h \quad \text{for } h = 1, 2, \ldots, H, \tag{6.1.5d}$$

$$z_p^h = \quad 1 - w^h \quad \text{for } h = 1, 2, \ldots, H, \tag{6.1.5e}$$

$$z_r^h,\ z_p^h \in \quad \{0, 1\} \quad \text{for } h = 1, 2, \ldots, H, \tag{6.1.5f}$$

$$\text{and } r \in \mathcal{R}.$$

In Model (7.1.5) three types of binary variables are introduced, these are z_p^h, z_r^h and w^h for $h = 1, 2, \ldots, H$. These binary variables yield

$$Q_{t+1}^H(x_{t+1}, p_{t+1}) = \min\left\{q^{h'} : h' \in \mathcal{H}_\delta(x_{t+1,} p_{t+1})\right\}, \tag{6.1.6}$$

where the set of supporting indices of Q_{t+1}^H is

$$\mathcal{H}_\delta(x, p) = \{h^{'} :\ x^{h'} > x - \delta_x,\ p^h > p - \delta_p \text{ for } h' = 1, 2, \ldots, H\}.$$

If $w^h = 1$ for step function h, then $\varphi \leq q_{t+1}^h$ (i.e. $(\overline{R} - q_{t+1}^h)(1 - w^h) = 0$), which makes constraints (7.1.5b) and (7.1.5c) nonbinding as both $z_p^h = 0$ and $z_r^h = 0$ for all $r \in \mathcal{R}$. On the other hand if $w^h = 0$, then constraint (7.1.5a) is nonbinding (i.e. $\varphi \leq \overline{R}$). Alternatively, constraints (7.1.5b) and (7.1.5c) become binding constraints, where $p_{t+1} \geq p_{t+1}^h + \delta_p$ (i.e. $z_p^h = 1 - w^h = 1$) and, for at least one $r \in \mathcal{R}$, $x_{t+1,r} \geq x_{t+1,r}^h + \delta_s$ (i.e. $\sum_{r\in\mathcal{R}} z_r^h = 1 - w^h = 1$). The MIP formulation of Q_t^H models $\mathcal{H}_\delta(s, p)$, the supporting index set of Q_{t+1}^H, by setting $w^h = 1$ when $h \in \mathcal{H}_\delta(x, p)$. Figure 7.5 depicts an example of $Q_{t+1}^H(x_{t+1}, p_{t+1})$, which is computed along the line $p_{t+1} = \hat{p}_{t+1}$ using 4 step functions.

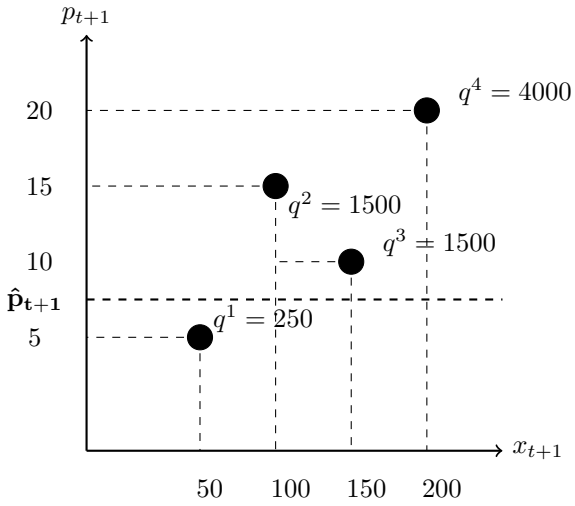


Figure 6.5: Computing $Q^H_{t+1}(x_{t+1}, \hat{p}_{t+1})$ for $\hat{p}_{t+1} = 7.5$ in the increasing step function along the horizontal line.

$$
\begin{aligned}
V_t(x_t, p_t) = \quad & \max \quad p_t g\,(v) + Q^H_{t+1}\,(x_{t+1}, p_{t+1})\,, \qquad (6.1.7)\\
& \text{s.t.}\\
& \qquad x_{t+1} = x_t + Av + Bl + \omega_t,\\
& \qquad p_{t+1} = \alpha_t p_t + (1-\alpha_t) b_t + \tilde{\eta},\\
& \qquad x_{t+1} \in X_\delta,\\
& \qquad (v, l) \in U(x_t),\\
& \qquad (x_1, p_1) = (\overline{x}, \overline{p}),\\
& \qquad Q^H_{T+1}(x_{T+1}, p_{T+1}) = R(x_{T+1}, p_{T+1}).
\end{aligned}
$$

Like SDDP, MIDAS has three main phases, which is the forward pass, the backward pass and the convergence test. In the forward pass of iteration H it first generates a price scenario $\left\{p^H_1, \ldots, p^H_2, \ldots, p^H_T\right\}$. Then, it computes the state trajectory

$\{x_1^H, x_2^H, \ldots, x_T^H\}$ by solving for $V_t(x_t, p_t^H)$ (see Model (7.1.7) above) for $t = 1, 2, \ldots, T$. In the backward pass, it solves $V_t(x_t^H, p_t^H)$ over the discrete distribution of $\tilde{\eta}$, where $p_{t+1} = \alpha_t p_t^H + (1 - \alpha_t)b_t + \hat{\eta}_i$ for $i = 1, 2, \ldots, L$. Then, it computes the expected value function estimate $q_t^H = \sum_{i=1}^{L} \rho_i V_t(\hat{x}_t, \hat{p}_t)$ for the point (x_t^H, p_t^H), where ρ_i is the probability of $\hat{\eta}_i$. Lastly, MIDAS measures the quality of the candidate policy by computing the upper and lower bound. It calculates the upper bound by taking the expectation of the objective $V_1(\overline{x}, \overline{p})$ for the first stage problem ($\overline{x}$ and $\overline{p}$ are the initial state and price) over the discrete distribution of $\tilde{\eta}$. Then, MIDAS computes the lower bound by simulating the candidate policy under N price scenarios and calculates the sample average reward. Algorithm 9 describes the MIDAS algorithm used to solve the hydro-bidding model (7.1.7).

Algorithm 9 : MIDAS algorithm for solving the hydro-bidding model (7.1.7).

1. Initialize $\overline{x}$, $\overline{p}$, $H_{\max}$, $H_{\text{convergence}}$, L, N
2. Initialize $Q_t^1(s_t, p_t)$ for $t = 1, 2, \ldots, T$
3. Set $H = 2$
4. **Forward Pass:**
   (a) Randomly sample a price scenario $\{p_t^H\}_{t=1,2,\ldots,T}$ starting from the initial state $\overline{p}$
   (b) Set $x_1^H = \overline{x}$
   (c) For $t = 1, 2, \ldots, T$:
      i. Solve $V_t(x_t^H, p_t^H)$ and set $x_{t+1}^H = x_{t+1}$
5. **Backward pass:**

(a) For $t = T, T-1, \ldots, 1$:

i. For $k = 1, 2, \ldots, L$:

A. Set $p_{t+1} = \alpha_t p_t^H + (1-\alpha_t) b_t + \hat{\eta}_k$ in $V_t(x_t^H, p_t^H)$ and solve (7.1.7)

B. Compute $\hat{V}_k = p_t^H g(v) + \rho_k Q_{t+1}^H (x_{t+1}, p_{t+1})$

ii. Update Q_t^H by adding the step function $q_t^H = \sum_{k=1}^{L} \hat{V}_k$ at point (x_t^H, p_t^H)

6. **Convergence test:** If $H(\mod H_{\text{convergence}}) = 0$ then:

(a) For $k = 1, 2, \ldots, L$:

i. Set $p_2 = \alpha_2 \overline{p} + (1-\alpha_2) b_2 + \hat{\eta}_k$ in $V_1(\overline{x}, \overline{p})$ and solve

ii. Compute $\hat{V}_k = \overline{p} g(v) + \rho_k Q_2^H (x_2, p_2)$

(b) Let $V^{\text{up}} = \sum_{k=1}^{L} \hat{V}_k$

(c) For $l = 1, \ldots, N$:

i. Randomly generate a price scenario $\{p_t\}_{t=1,2,\ldots,T}$ starting from the initial state $\overline{p}$

ii. Set $x_1 = \overline{x}$ and $V_{\text{low},l} = 0$

iii. For $t = 1, 2, \ldots, T$:

A. Solve $V_t(x_t, p_t)$, compute $V_{\text{low},l} = V_{\text{low},l} + p_t g(v)$, and set x_{t+1}

(d) Compute $\overline{V}_{\text{low}} = \mathbb{E}_{\{1,\ldots,N\}}[V_{\text{low}}]$ and $\sigma_{\text{low}} = \text{SD}_{\{1,\ldots,N\}}(V_{\text{low}})$

(e) If $\overline{V}_{\text{low}} - \frac{1.96}{\sqrt{N}}\sigma_{\text{low}} \leq V^{\text{up}} \leq \overline{V}_{\text{low}} + \frac{1.96}{\sqrt{N}}\sigma_{\text{low}}$ or $H > H_{\max}$, then stop, otherwise set $H = H + 1$ and go to Step 4

6.1.2 SDDP-based approach

Modelling the price process, described in Section 7.1, inside SDDP is not trivial. The value function $V_t(x_t, p_t)$ cannot be approximated using hyperplanes in both x_t and

p_t. This is because, as we can see from Figure 7.2 the value function is bilinear due to multiplying the offers being multiplied by the price in the objective, as well as being present in the right hand side of the constraints. Due to this bilinearity, Model (7.1.7) becomes non-linear and non-convex, which makes computing the dual variables difficult and incorrect to construct cutting planes.

In this thesis, we propose to discretize the price variable in order to construct a scenario tree of prices. Then, we introduce a novel interpolation technique that represents the value function for a particular price value as the interpolated value function between neighboring price nodes that is greater and less than the price.

This interpolation technique first discretizes the price domain into K price values,

$$(\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_K). \tag{6.1.8}$$

Based on these values for period t, p_t can be interpolated from two neighboring price node values as

$$p_t = \lambda \hat{p}_i + (1 - \lambda)\hat{p}_{i+1}, \tag{6.1.9}$$

where $\hat{p}_i \leq p_t \leq \hat{p}_{i+1}$, and $\lambda \in [0, 1]$. Similarly, $V_t(x_t)$ can be interpolated between i and $i + 1$, which are the SDDP outer-approximation of the value function at $\hat{p}_i$ and $\hat{p}_{i+1}$ respectively. Then,

$$V_t(x_t) = \lambda \theta_i + (1 - \lambda)\theta_{i+1}, \tag{6.1.10}$$

where

$$\theta_{i+1} \leq \beta^h_{t+1,i+1} + \left(\pi^h_{t+1,i+1}\right)^\top \left(x_{t+1} - \alpha^h_{t+1,i+1}\right) \text{ for } h = 1, 2, \ldots, H, \tag{6.1.11}$$

and

$$\theta_i \leq \beta^h_{t+1,i} + \left(\pi^h_{t+1,i}\right)^\top \left(x_{t+1} - \alpha^h_{t+1,i}\right) \text{ for } h = 1, 2, \ldots, H. \tag{6.1.12}$$

Figure 7.6 illustrates how the value function $V_t(x_t)$ at p_t is computed based on the interpolation technique. The cutting planes θ_i and θ_{i+1} approximate the value function $V_t(x_t)$ at the discrete price values $\hat{p}_i$ and $\hat{p}_{i+1}$. By discretizing the price domain and storing cuts at each of these price nodes we are able to represent $V_t(x_t, p_t)$ as $V_t(x_t)$ using the SDDP hyperplanes for along both the x_t and still model the autoregressive price process for p_t.

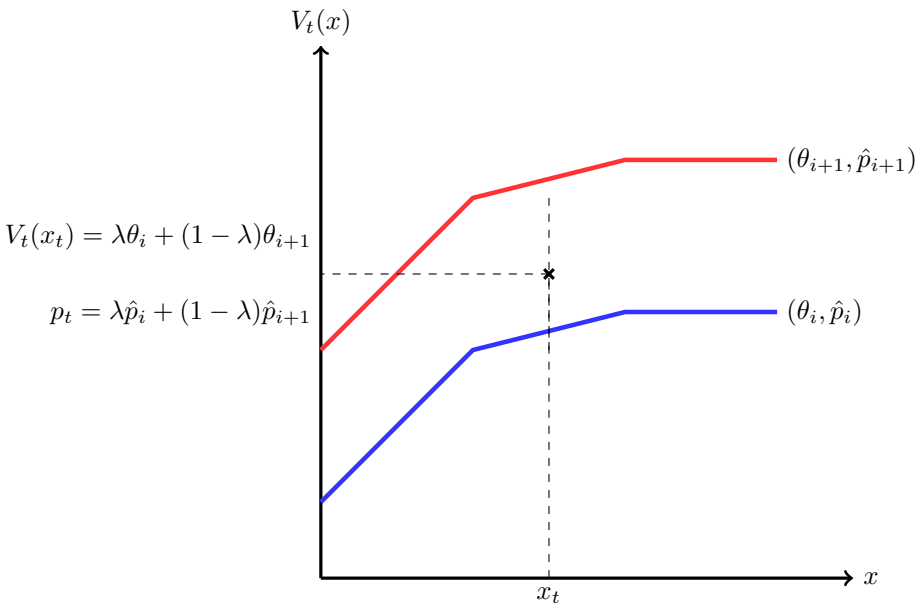


Figure 6.6: Interpolating $V_t(x_t)$ by two sets of cutting plane approximations i (blue line) and $i+1$ (red line) at $\hat{p}_i$ and $\hat{p}_{i+1}$.

In order to implement this technique, a few alterations are made to the SDDP-based hydro-bidding model and algorithm discussed in Section 3.2.3 of Chapter 3. We begin by formulating a new sub-problem (7.1.13), which is the SDDP equivalent of Model (7.1.7). In this model, the value function $V_{t+1}(x_{t+1})$ is represented by the variable φ, which is interpolated between θ_i and θ_{i+1}. The parameters $\beta^h_{t+1,i}$, $\beta^h_{t+1,i+1}$, $\alpha^h_{t+1,i}$,

$\alpha^h_{t+1,i+1}$, $\pi^h_{t+1,i}$, and $\pi^h_{t+1,i+1}$, for $h = 1, 2, \ldots, H$, define the hyperplanes for i and $i+1$ respectively.

$$V_t(x_t) = \max \quad p_t g(v) + \varphi, \tag{6.1.13}$$

$$\textbf{s.t.} \tag{6.1.14}$$

$$\tag{6.1.15}$$

$$x = x_t,$$

$$x_{t+1} = x + Av + Bl + \omega_t,$$

$$\varphi = \lambda\theta_i + (1-\lambda)\theta_{i+1},$$

$$\theta_{i+1} \le \beta^h_{t+1,i+1} + \left(\pi^h_{t+1,i+1}\right)^\top \left(x_{t+1} - \alpha^h_{t+1,i+1}\right) \quad \text{for } h = 1, 2, \ldots, H,$$

$$\theta_i \le \beta^h_{t+1,i} + \left(\pi^h_{t+1,i}\right)^\top \left(x_{t+1} - \alpha^h_{t+1,i}\right) \quad \text{for } h = 1, 2, \ldots, H,$$

$$x_{t+1} \in X,$$

$$(v, l) \in U(x_t),$$

$$(x_1, p_1) = (\overline{x}, \overline{p}),$$

$$V_{T+1}(x_{T+1}, p_{T+1}) = R(x_{T+1}, p_{T+1}).$$

In the forward pass of iteration H, SDDP randomly samples a price scenario $\{p^H_t\}_{t=1,2,\ldots,T}$. For each price value p^H_t, it identifies the neighboring price nodes $\hat{p}_{t,i}$ and $\hat{p}_{t,i+1}$. In addition to the nodes, it also identifies the corresponding cutting plane approximations defining θ_i and θ_{i+1}. It sets

$$\lambda = \frac{p_{t+1}^H - \hat{p}_{t,i+1}}{\hat{p}_{t,i} - \hat{p}_{t,i+1}},$$

and solves $V_t(x_t)$ to get the state x_{t+1}^H for the next period $t+1$.

In the backward pass, SDDP updates the approximation of the value function for $x_t = x_t^H$ at the discrete price nodes $\hat{p}_{t,i}$ and $\hat{p}_{t,i+1}$. For $\hat{p}_t = \hat{p}_{t,i}$ and $\hat{p}_t = \hat{p}_{t,i+1}$, for each discrete value of $\tilde{\eta}$, it sets

$$\hat{p}_{t+1} = \alpha_{t+1}\hat{p}_t + (1 - \alpha_{t+1})b_{t+1} + \tilde{\eta},$$

and then determines the neighboring nodes $\hat{p}_{t+1,j}$, $\hat{p}_{t+1,j+1}$ and the λ parameter. Then, it solves Model (7.1.16) and obtains the hyperplane parameters $\pi_t = \mathbb{E}_{\tilde{\eta}}[\hat{\pi}_t]$, $\beta_t = \mathbb{E}_{\tilde{\eta}}[\hat{p}_t g(v) + \varphi]$ and $\alpha_t = x_t^H$. It then adds the hyperplane to its respective price node in order to improve the value function approximation at $[x_t, \hat{p}_t)$.

$$\overline{V}_t(x_t, \hat{p}_t) = \max \quad \hat{p}_t g(v) + \varphi, \tag{6.1.16}$$

$$\mathbf{s.t.} \tag{6.1.17}$$

$$\tag{6.1.18}$$

$$
\begin{aligned}
& x = x_t \qquad [\hat{\pi}_t], \\
& x_{t+1} = x + Av + Bl + \omega_t, \\
& \varphi = \lambda\theta_j + (1-\lambda)\theta_{j+1}, \\
& \theta_{j+1} \le \beta^h_{t+1,j+1} + \left(\pi^h_{t+1,j+1}\right)^\top \left(x_{t+1} - \alpha^h_{t+1,j+1}\right) \quad \text{for } h = 1, 2, \ldots, H, \\
& \theta_j \le \beta^h_{t+1,j} + \left(\pi^h_{t+1,j}\right)^\top \left(x_{t+1} - \alpha^h_{t+1,j}\right) \quad \text{for } h = 1, 2, \ldots, H, \\
& x_{t+1} \in X, \\
& (v, l) \in U(x_t), \\
& (x_1, p_1) = (\overline{x}, \overline{p}), \\
& V_{T+1}(x_{T+1}, p_{T+1}) = R(x_{T+1}, p_{T+1}).
\end{aligned}
$$

In the convergence testing phase, SDDP calculates the upper bound by solving $\overline{V}_1(\overline{x})$ for each discrete value of $\tilde{\eta}$ in the same way as carried out in the backward pass, with $\hat{p}_2 = \alpha_2\overline{p} + (1-\alpha_2)b_2 + \tilde{\eta}$. Then, it calculates the lower bound by simulating the candidate policy under N price scenarios for $V_t(x_t)$ and calculates the sample average reward. It stops when the upper bound is within the 95% confidence interval of the lower bound. Algorithm 10 describes this version of the SDDP algorithm with the implemented cut interpolation technique.

Algorithm 10 : SDDP algorithm with cut interpolation for solving the hydro-bidding model (7.1.13).

1. Initialize $\overline{x}$, $\overline{p}$, $H_{\max}$, $H_{\text{convergence}}$, L, N
2. Set $H = 2$
3. **Forward Pass:**
   (a) Randomly sample a price scenario $\{p_t^H\}_{t=1,2,\ldots,T}$ starting from the initial state $\overline{p}$.
   (b) Set $x_1^H = \overline{x}$.
   (c) For $t = 1, 2, \ldots, T$ do,
      i. determine θ_i, θ_{i+1}, and λ for p_{t+1}^H,
      ii. solve (7.1.13) for $V_t(x_t^H)$ and set $x_{t+1}^H = x_{t+1}$,
4. **Backward pass:**
   (a) For $t = T, T-1, \ldots, 1$, and for $\hat{p}_t = \hat{p}_{t,i}$ and $\hat{p}_t = \hat{p}_{t,i+1}$, do the following:
      i. for $k = 1, 2, \ldots, L$ do,
         A. set $\hat{p}_{t+1} = \alpha_t \hat{p}_t + (1 - \alpha_t) b_t + \hat{\eta}_k$,
         B. determine θ_i, θ_{i+1}, and λ for $\hat{p}_{t+1}$,
         C. solve (7.1.16) for $\overline{V}_t(x_t^H, \hat{p}_t)$, and obtain $\hat{\pi}_{t,k}$,
         D. compute $\hat{V}_k = \hat{p}_t g(v) + \varphi$.
      ii. Update the outer approximation by adding the cutting plane with $\pi_t = \sum_{k=1}^{L} \rho_k \hat{\pi}_{t,k}$, $\beta_t = \sum_{k=1}^{L} \rho_k \hat{V}_k$, and $\alpha_t = x_t^H$.
5. **Convergence test:** If $H(\mod H_{\text{convergence}}) = 0$ then:

(a) For $k = 1, 2, \ldots, L$ do,

i. set $\hat{p}_2 = \alpha_2 \overline{p} + (1 - \alpha_2) b_2 + \hat{\eta}_k$,

ii. determine θ_i, θ_{i+1}, and λ for $\hat{p}_2$,

iii. solve $\overline{V}_1(\overline{x})$ and let $\hat{V}_k = \hat{p}_t g(v) + \varphi$.

(b) Let $V^{\text{up}} = \sum_{k=1}^{L} \rho_k \hat{V}_k$.

(c) For $l = 1, \ldots, N$ do the following:

i. randomly sample a price scenario $\{p_t\}_{t=1,2,\ldots,T}$ starting from the initial state $\overline{p}$,

ii. set $x = \overline{x}$ and $V_{\text{low},l} = 0$,

iii. for $t = 1, 2, \ldots, T$ do,

A. determine θ_i, θ_{i+1}, and λ for p_{t+1},

B. solve $V_t(x)$, compute $V_{\text{low},l} = V_{\text{low},l} + p_t g(v)$, and set $x = x_{t+1}$.

(d) Compute $\overline{V}_{\text{low}} = \mathbb{E}_{\{1,\ldots,N\}}[V_{\text{low}}]$ and $\sigma_{\text{low}} = \text{SD}_{\{1,\ldots,N\}}(V_{\text{low}})$.

(e) If $\overline{V}_{\text{low}} - \frac{1.96}{\sqrt{N}}\sigma_{\text{low}} \leq V^{\text{up}} \leq \overline{V}_{\text{low}} + \frac{1.96}{\sqrt{N}}\sigma_{\text{low}}$ or $H > H_{\max}$, then stop, otherwise set $H = H + 1$ and go to step 3.

6.1.3 Comparison of MIDAS and SDDP

Both MIDAS and SDDP were used to solve two hydrobidding problems, each with a planning horizon of 12 periods (i.e. $T = 12$). A single reservoir (i.e. $|\mathcal{R}| = 1$ and $|\mathcal{S}| = 1$), and two reservoirs in cascade (i.e. $|\mathcal{R}| = 2$ and $|\mathcal{S}| = 2$) hydro schemes were used as test cases. We used the elementary reservoir configuration from Section 3.3 of Chapter 3 so as to make it easier to analyze, compare, and illustrate the performance of each of the algorithms.

The spill limit was set to 0 in order to prevent spilling (i.e. $l = 0$), and it was assumed that there were no inflows (i.e. $\omega_t = 0$). This isto help us analyse and compare

the policies and state trajectories of MIDAS and SDDP. With no inflows the hydro generator only begins with an initial stock of water, which the must consume efficiently throughout the planning horizon. In this situation the hydro generator would look to generate only in the optimal periods as their stock will not be replenished. Therefore, we should expect to see similar offer strategies by both MIDAS and SDDP.

The price was modeled as an autoregressive lag 1 process that reverts to the mean defined by b_t. The noise term η is discretized into 5 outcomes, which are independently, and identically distributed (i.e. i.i.d). We chose $\delta_p = 1.5$, and set the reservoir storage bounds to be between the value of δ_x and 200. The generation function $g(v)$ is a convex piecewise linear function of the turbine flow v, which is between 0 and 70. It is the same piecewise linear function from Section 3.3 of Chapter 3. Table A.1, in Section A.1 of Appendix A, lists the values of the model parameters.

As illustrated by Figures 7.7 and 7.8, MIDAS converges under various δ_x values for the single reservoir hydro scheme. It converges quicker for higher δ_x values. This is because with a higher δ_x, the value function has a coarser approximation. The state space X_δ is covered by sampled points x_t^h, for $h = 1, 2, \ldots, H$, which are within δ_x distance. Hence, the smaller the δ_x, the closer the sampled state points are which improves the approximation and decreases the error ε value (see Chapter 5 Section 5.3). It is also a similar case for the 2 reservoir hydro scheme illustrated by Figures 7.9 and 7.10.

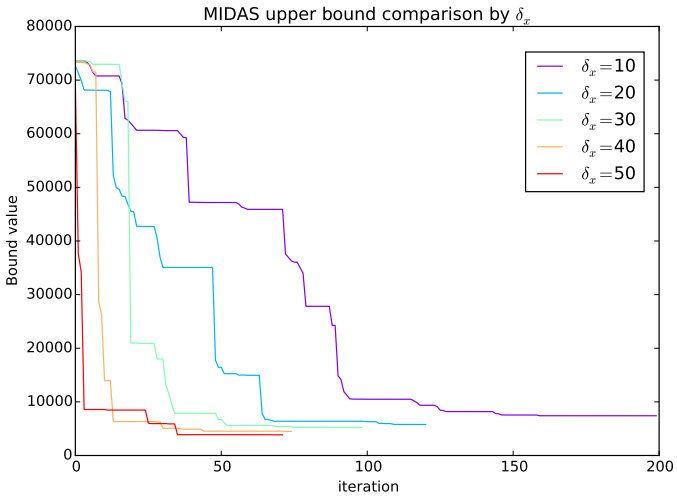


Figure 6.7: Comparison of upper bound for various δ_x values of the MIDAS algorithm for a single reservoir hydro scheme.

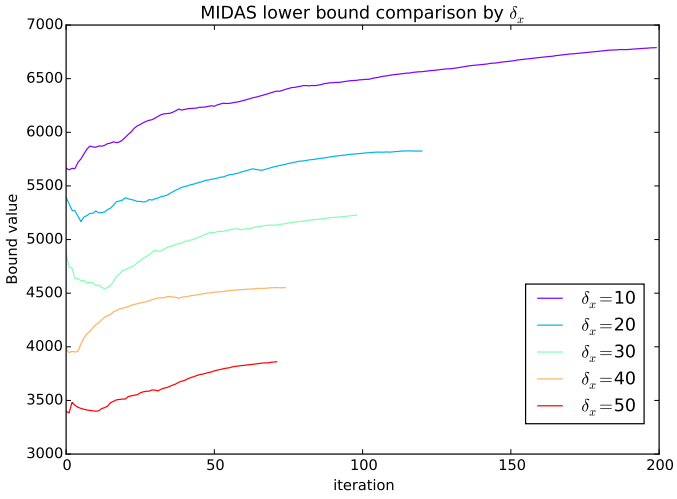


Figure 6.8: Comparison of lower bound for various δ_x values of the MIDAS algorithm for a single reservoir hydro scheme.

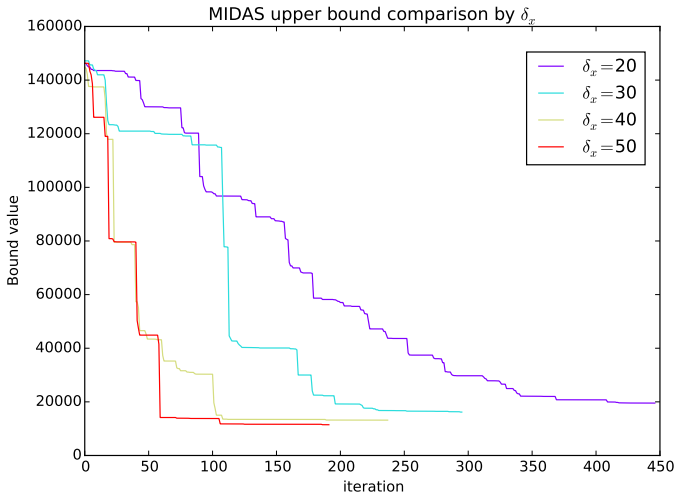


Figure 6.9: Comparison of upper bound for various δ_x values of the MIDAS algorithm for 2 reservoir hydro scheme.

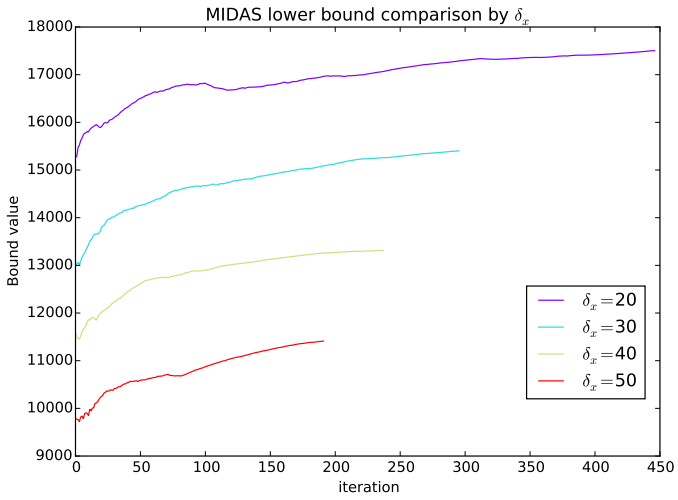


Figure 6.10: Comparison of lower bound for various δ_x values of the MIDAS algorithm for 2 reservoir hydro scheme.

The SDDP algorithm converges to similar upper and lower bounds in fewer iterations than MIDAS, as illustrated by Figures 7.12, and 7.11. The reason why SDDP converges quicker than MIDAS is because it exploits the gradient of the hyperplanes in order to generate new state trajectories in the forward pass. MIDAS on the other hand generates new state trajectories that are δ_x distance apart. This takes longer to

converge as it has to thoroughly cover the state space in order to accurately approximate the value function. Figures 7.13 and 7.14 show the state trajectories generated in the forward pass by MIDAS and SDDP respectively. MIDAS covers the feasible state space in points that are δ_x distance apart, thus creating a grid of visited states. SDDP on the other hand explores different states only in the peak price periods when it is discharging water for generation.

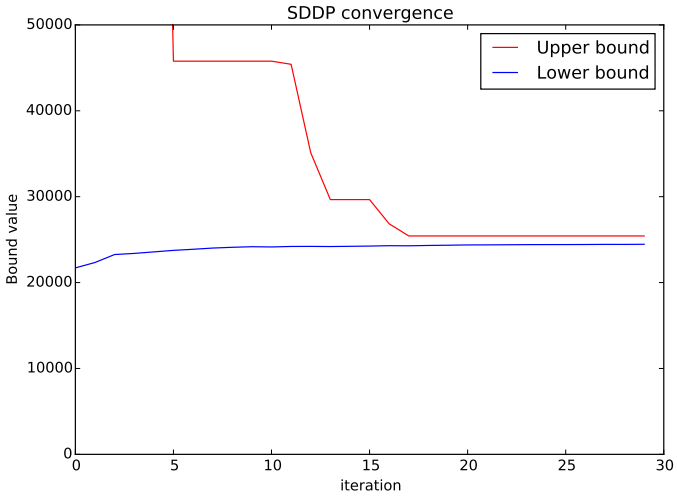


Figure 6.11: Comparison of upper and lower bound SDDP algorithm for 2 reservoir hydro scheme.

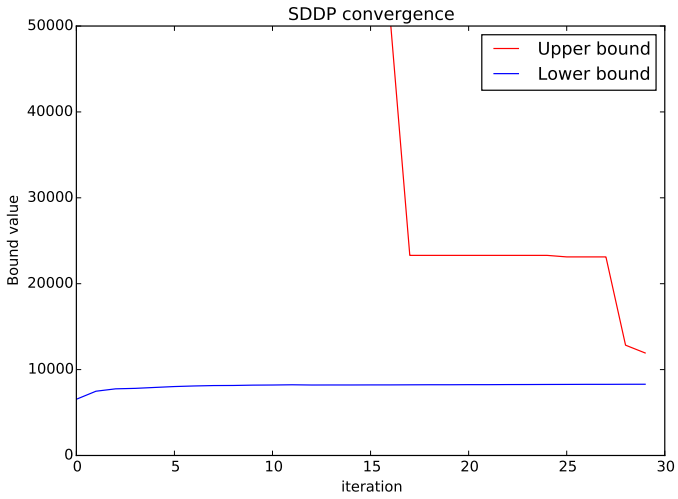


Figure 6.12: Comparison of upper and lower bound of the SDDP algorithm for single reservoir hydro scheme.

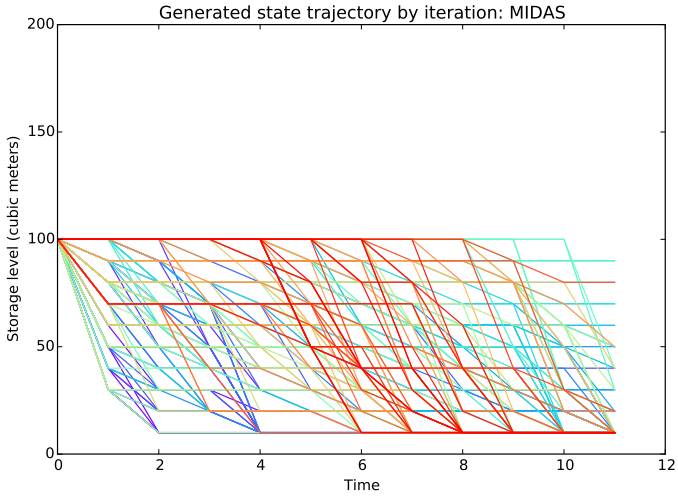


Figure 6.13: Generated state trajectory by iteration of the MIDAS algorithm ($\delta_x = 10$) for a single reservoir hydro scheme.

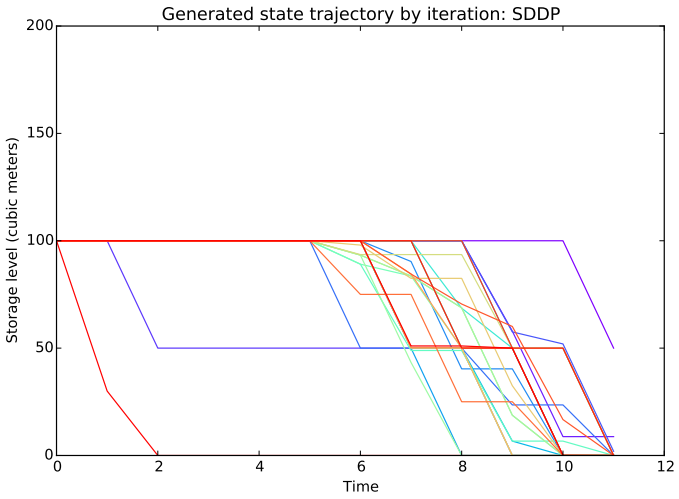


Figure 6.14: Generated state trajectory by iteration of the SDDP algorithm for a single reservoir hydro scheme.

When the policies of MIDAS and SDDP are simulated they illustrate similar strategies, which offer the greatest quantity of power during the peak price periods. The policy of MIDAS produces offer quantities in increments of δ_x, whereas SDDP has greater variations in its quantity. Moreover, MIDAS (unlike SDDP) offers energy in early time periods when the price is just about to increase, thus it anticipates the higher price peaks.

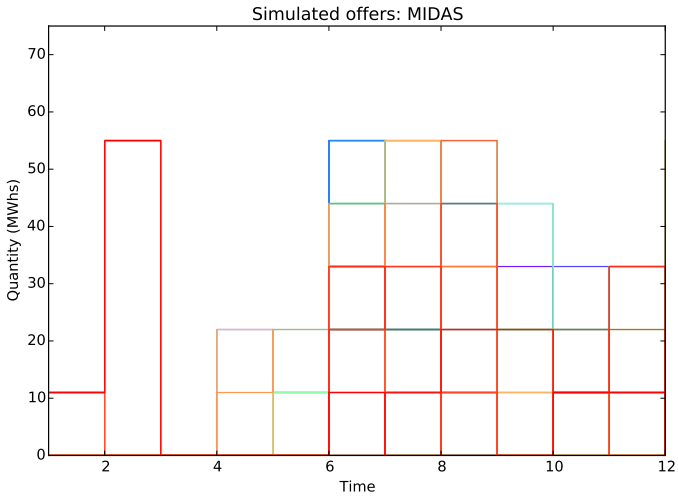


Figure 6.15: Offers from the simulated policy of the MIDAS algorithm ($\delta_x = 10$) for a single reservoir hydro scheme.

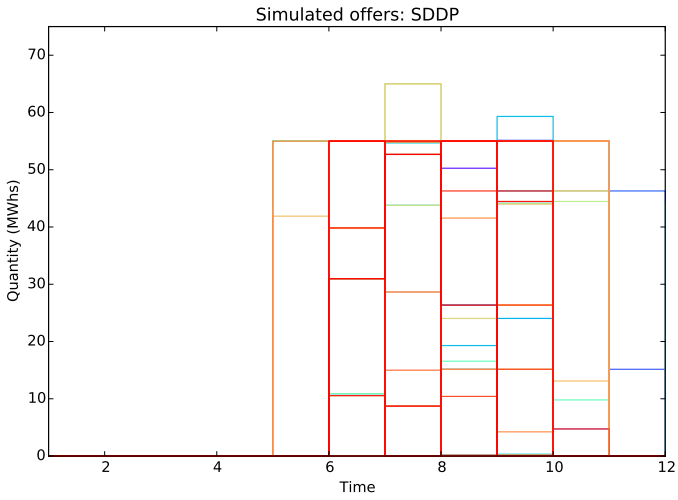


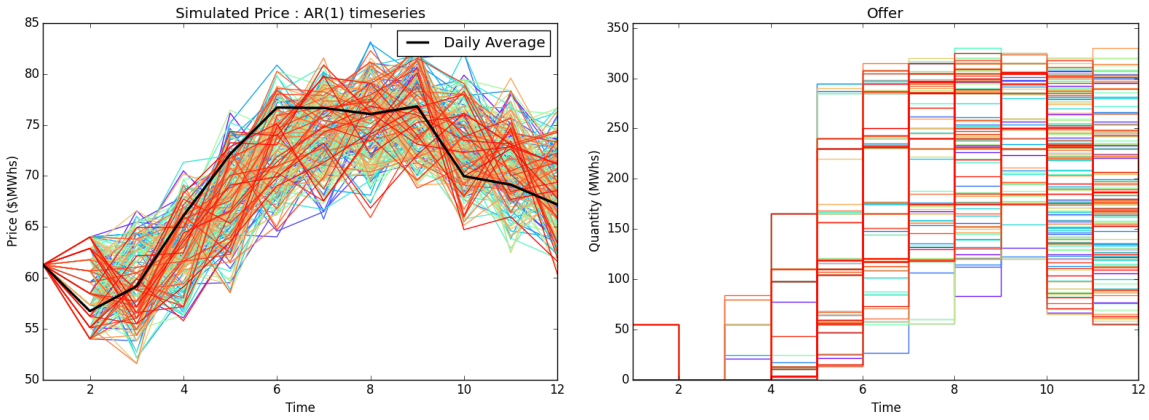
Figure 6.16: Offers from the simulated policy of the SDDP algorithm for a single reservoir hydro scheme.

Through the use of the cut interpolation technique, SDDP is computationally more efficient than MIDAS at constructing similar offer policies. However, this technique can only model univariate price processes. It is not easy to implement the cut interpolation technique in multiple price dimensions as it requires carrying out multivariate interpolation among the neighboring price nodes. For MIDAS, adding more dimen-

sions is trivial to model, as it only requires adding new dimensions inside the value function. There is no need for additional calculations, because the formulation of the step function approximation of the value function is still the same.

6.1.4 Solving a large scale hydro scheme using SDDP

Using the cut interpolation technique SDDP can represent an autoregressive, mean reverting price processes. Since it is computationally more efficient than MIDAS, it was extended to solve a hydro-bidding problem for a 5 reservoir hydro scheme, and with a higher resolution of $\tilde{\eta}$ disturbances (Figure 7.17a). Each reservoir has an initial storage volume of 100 cubic meters, and there is no inflow and spill in this model. The offer policy computed by SDDP, as illustrated in Figure 7.17b, submits bids with higher quantity of power during periods of peak prices. Between period 1 to 5 there is only one single bid quantity of 55 MWh. Then, the quantity of offers starts to increase in time, where at period 8 the maximum quantity is submitted during the peak when the price is at its highest.



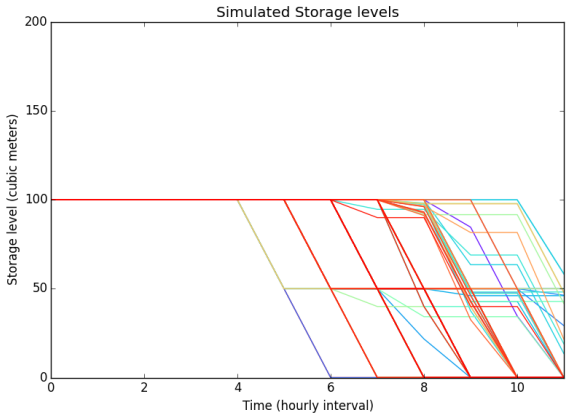
(a) Simulated prices.

(b) Simulated offers.

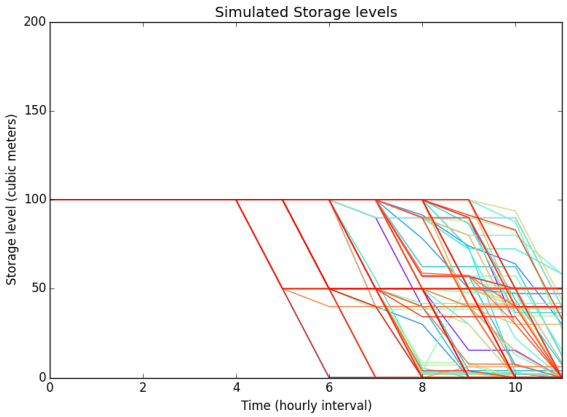
Figure 6.17: Simulated price and offers of 5 reservoir hydro scheme from a policy constructed through SDDP.

Every offer quantity submitted must be feasible for generation across the hydro scheme.

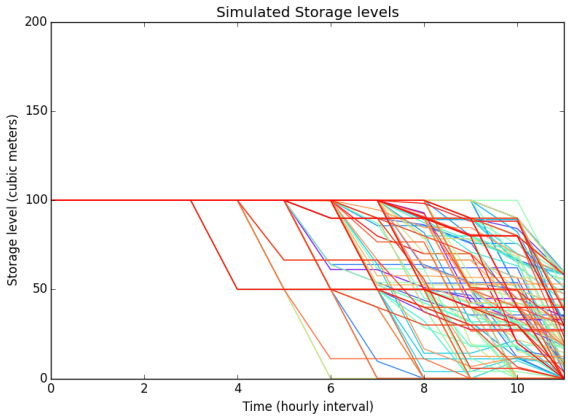
Based on the simulated storage volumes, illustrated between Figures 7.18a to 7.18e, one can see that water in upstream reservoirs is conserved for consumption for the peak prices. The upstream reservoirs 1 and 2 (Figures 7.18a and 7.18b) release their water last, compared to the downstream reservoirs in order to extract the maximum value. Since the stored water flows from upstream to downstream, it is utilized at multiple stations to produce power. This makes the volume of water stored upstream more valuable than if it were stored downstream, therefore it is released just before the price peaks in order to construct offers with maximum quantities. The downstream reservoirs 3, 4, and 5 (Figures 7.18c, 7.18d, and 7.18e) release water earlier in order to have enough storage capacity to receive the upstream releases during the price peaks. In fact, reservoir 5, in some of the scenarios, reaches maximum storage volume (200 cubic meters) in early trading periods due to early upstream releases, which it depletes over time.



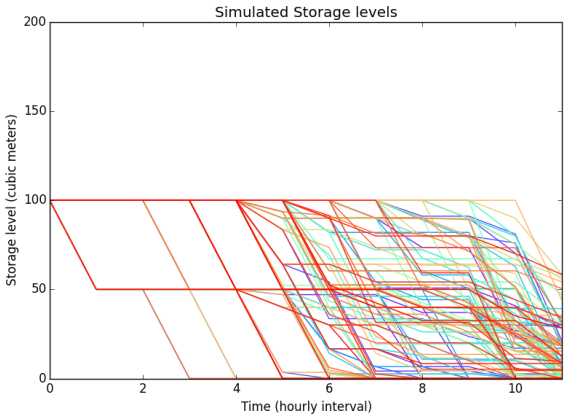
(a) Reservoir 1.



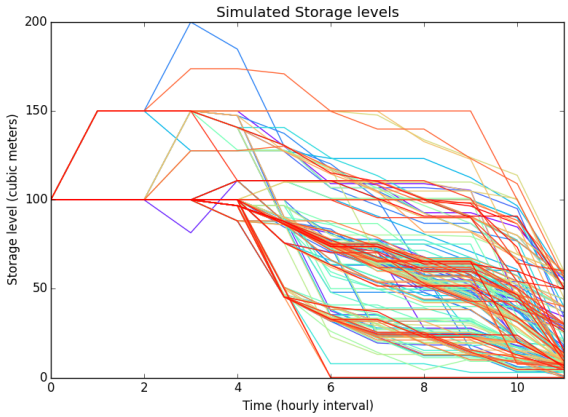
(b) Reservoir 2.



(c) Reservoir 3.



(d) Reservoir 4.



(e) Reservoir 5.

Figure 6.18: Simulated state trajectories of each reservoir in the 5 reservoir hydro scheme.

Based on the the water releases observed in this SDDP model, the hydro scheme is transferring a set volume of water across its reservoirs over time. This is what is referred to as cycling, where reservoirs are periodically emptied and then filled back up to consume a constant volume of water for power generation. This is common practice for hydropower management, which efficiently consumes the water to produce the maximum power.

6.2 Modelling headwater effects

Accurately modelling the power generation function in hydro-bidding models is important because it estimates the capacity of power available for generation based on the flow and the reservoir water level. This is particularly significant for hydro schemes of reservoirs with low storage capacity, or where the water level of their reservoirs vary significantly. As shown in Figure 7.20, the power generation varies with the water level of the reservoir for a particular flow profile. If we assume fixed headwater and model the power as function of only flow, then the increases and decreases of the power based on the headwater will be absent. This can produce sub-optimal policies in low storage conditions, when efficient generation is crucial. The significance of incorporating headwater effects has been observed by [60], where their models retained water (i.e. keep headwater levels high) in reservoirs with low storage capacity in order to produce maximum power. In this section, we take the perspective that headwater is significant.

The power generation function, as defined in Equation (7.2.1) and discussed in Chapter 3, depends on the flow through the turbines u, the efficiency factor η, and the net water head h (the constant 9.81×10^{-3} is the acceleration due to gravity constant multiplied by the density of water converted from meters to MJ). The efficiency factor η represents

the plant capability curves, also known as hill curves. Combined with the hill curves, the power generation function can be represented as a function of h and station flow (i.e. total flow across all turbines) based on empirical measurements. This creates a surface which shows the actual power that can be generated for a particular station flow and head level (Figure 7.19).

$$g_s(h, u) = 9.81 \times 10^{-3} \eta u h \tag{6.2.1}$$

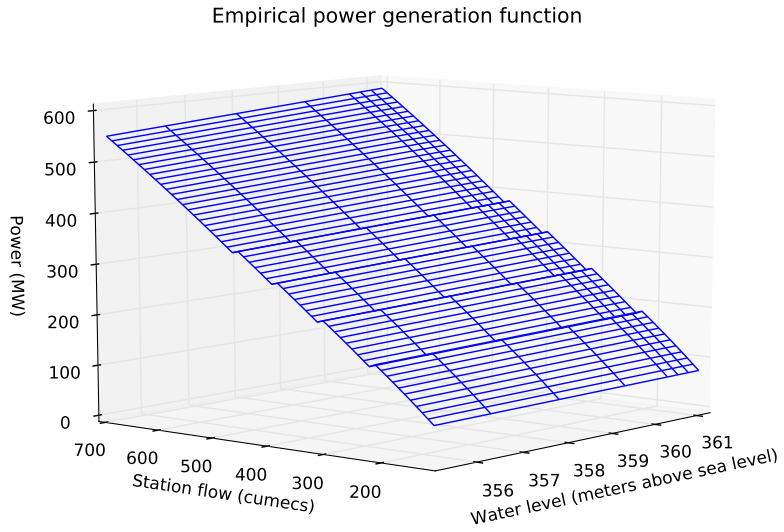


Figure 6.19: Example power function based on empirical measurements for a station with 6 turbines.

The empirical power function in Figure 7.19 is non-concave due to switching at specific station flows. These switches are due to the starting of additional turbines. As the flow across an operating turbine increases, beyond its efficient generation point, the efficiency starts to decrease. Then, switching on an additional unit increases the efficiency and the power, which is illustrated by a strict increase in generation. This creates a hill surface with a similar shape similar to the hill curves depicted in Figure 3.6 of Chapter 3.

The power function is also bilinear due to the multiplication of u and h. As illustrated in Figure 7.20, the power is increasing with the head level. Whenever the head level is below the minimum (i.e. $h = 0$) the power is zero. But, at higher station flows, the power depends on the head level. Due to the cloud-like shape, and the bilinear property, the power function is non-concave, which is difficult to model inside a SDDP-based hydro-bidding model.

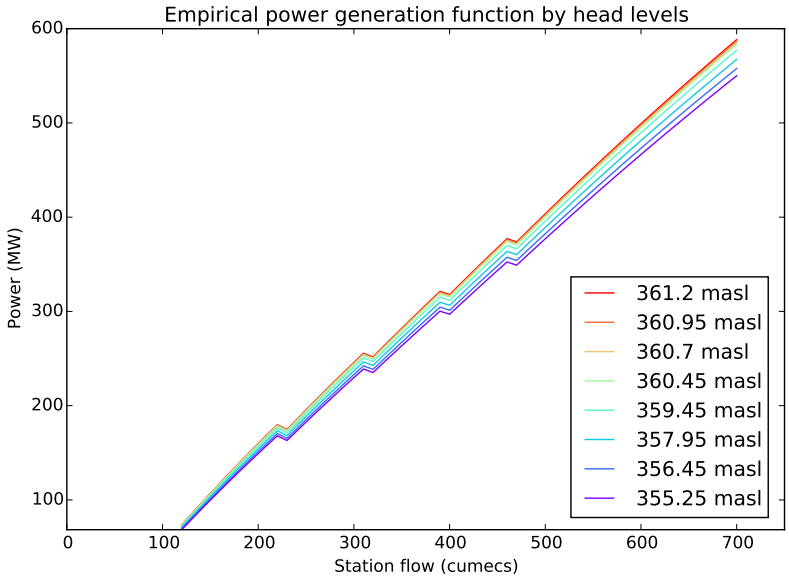


Figure 6.20: Example power function based on empirical measurements for each measured head level.

In this section, we present a new approximation of the non-concave power functions and implement this formulation inside MIDAS and SDDP. This formulation is based on the work of [60], who used it to analyze whether headwater effects on the power generation capacity were significant for the Waikato hydro scheme in New Zealand. The benefit of using such an approximation is that it enables us to include unit commitment, where the states of the individual turbines can be controlled.

Consider the following approximated power function formulation in Equation (7.2.2). In this formulation $g_{h_0}(u)$ is the piecewise linear power function, based on station

discharge u, at the nominal head level h_0. This function is linearly scaled by the differences in the current head level h and h_0 by gradient $\frac{\partial g}{\partial h}$. For a given u, Equation (7.2.2) is a linear function in the h dimension.

$$g(h,u) = g_{h_0}(u) + g_{h_0}(u)\frac{\partial g}{\partial h}(h - h_0) \tag{6.2.2}$$

The overall function $g(h,u)$ is bilinear due to the term $g_{h_0}(u)\frac{\partial g}{\partial h}(h-h_0)$, which $g_{h_0}(u)$ and h is being multiplied. The bilinear terms in Equation (7.2.2) can be separated using differences of squares. First, we define the following variables

$$a = g_{h_0}(u) + \left(\frac{\partial g}{\partial h}(h - h_0) + 1\right), \tag{6.2.3}$$

and

$$b = g_{h_0}(u) - \left(\frac{\partial g}{\partial h}(h - h_0) + 1\right). \tag{6.2.4}$$

Then, $g(h,u)$ can be represented as

$$g(h,u) = \frac{a^2 - b^2}{4}. \tag{6.2.5}$$

The a^2 and b^2 terms are quadratic and convex, since a and b are now linear with separated $g_{h_0}(u)$ and h terms. Both a^2 and b^2 can be approximated as a piecewise linear function. Then, $g(h,u)$ can be implemented inside the linear sub-problems of MIDAS and SDDP. Furthermore, the piecewise linear approximation of a, b and $g_{h_0}(u)$

can be represented as special ordered set of type two (SOS type II) constraints inside the solvers for efficient computation [12].

The nominal power function $g_{h_0}(u)$ is approximated as a single power function for a single turbine, and then it is multiplicatively scaled based on the number of turbines that are in use. For a single turbine in use, the nominal power is defined in Equation (7.2.6) for L_s^{power} pieces, where $\frac{dg_1}{du}$ is the gradient for the line between $(\hat{u}_1, \hat{g}_2)$ and $(\hat{g}_1, \hat{g}_2)$.

$$g_{h_0}(u) = \begin{cases} \hat{g}_1 + \frac{dg_1}{du}(u - \hat{u}_1) & \text{if } u \in [\hat{u}_1, \hat{u}_2), \\ \hat{g}_2 + \frac{dg_2}{du}(u - \hat{u}_2) & \text{if } u \in [\hat{u}_2, \hat{u}_3), \\ \dots & \dots, \\ \hat{g}_{L_s^{\text{power}}-1} + \frac{dg_{L_s-1}}{du}(u - \hat{u}_{L_s^{\text{power}}-1}) & \text{if } u \in [\hat{u}_{L_s^{\text{power}}-1}, \hat{u}_{L_s^{\text{power}}}). \end{cases} \tag{6.2.6}$$

The nominal power function, $g_{h_0}(u)$, can be multiplicatively scaled, where $\hat{g}$ and $\hat{u}$ values are multiplied by the number of units that are turned on. Figure 7.21 illustrates how the nominal power function for a single turbine is scaled based on the number of units being used. It is an approximation of the power generation at a head level of 359.45 (meters above sea level).

Unit switching can be incorporated inside the power generation function based on this type of formulation. As is shown in Figure 7.21, this type of formulation represents the incremental shape of the power function. The efficient generation frontier, which is formed by edges of the piecewise linear functions for each turbine in use, closely follows the empirically based power generation function shown in black.

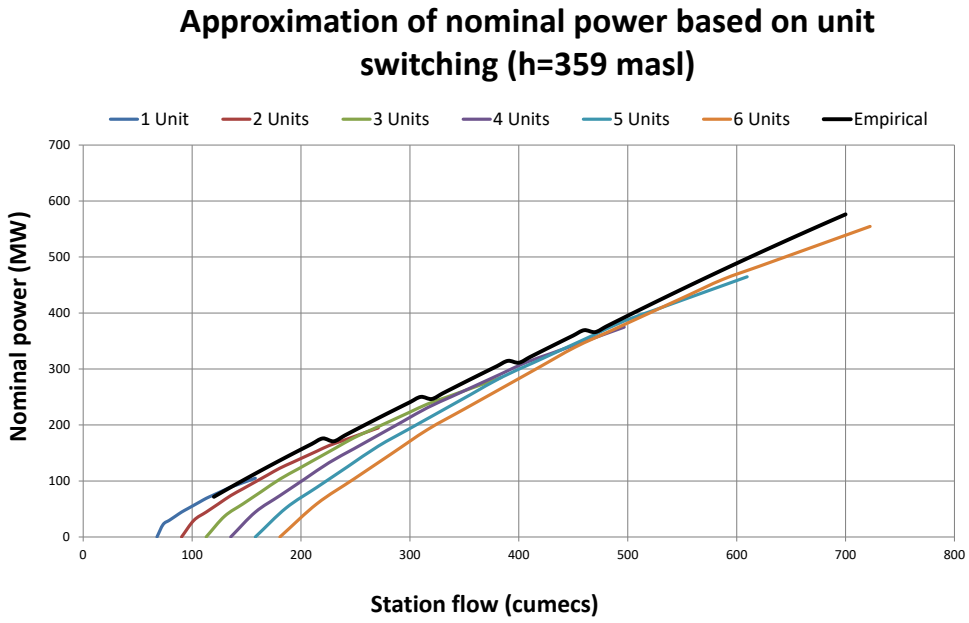


Figure 6.21: Piecewise linear approximation of $g_{h_0}(u)$ scaledby the number of units in use ($h_0 = 359.45$).

There is an underlying assumption associated with this formulation, which is that all the turbines are identical. In most occasions this is not a bad assumption because it is common for stations in a hydro scheme to be constructed with identical turbines. But all the turbines are not equally used, which can result in variations in their performance over time. The reader, if implementing this approximation, is advised to first test this assumption.

In the next section, Section 7.2.1, we introduce the linear programming formulation for representating the approximate generation function $g(h, u)$ in Equation (7.2.5).

6.2.1 Linear formulation of $g(h, u)$

For a station s and its upstream reservoir r, let $g_s(h_r, u_s)$ be the approximate generation function in (7.2.5) that can be modelled by the following set of constraints:

$$g_{s,h_0} = \sum_{i \in \mathcal{T}_s} g^{\text{unit}}_{s,h_0,i} \ , \tag{6.2.7a}$$

$$u_s = \sum_{i \in \mathcal{T}_s} u^{\text{unit}}_{s,i} \ , \tag{6.2.7b}$$

$$g^{\text{unit}}_{s,h_0,i} = i \sum_{l=1}^{L_s} \hat{g}_{s,l} \lambda^{\text{power}}_{s,i,l} \qquad \text{for } i \in \mathcal{T}_s, \tag{6.2.7c}$$

$$u^{\text{unit}}_{s,i} = i \sum_{l=1}^{L_s} \hat{u}_{s,l} \lambda^{\text{power}}_{s,i,l} \qquad \text{for } i \in \mathcal{T}_s, \tag{6.2.7d}$$

$$1 = \sum_{l=1}^{L_s} \lambda^{\text{power}}_{s,i,l} \qquad \text{for } i \in \mathcal{T}_s, \tag{6.2.7e}$$

$$u^{\text{unit}}_{s,i} \leq i \hat{u}_{s,L^{\text{power}}_s} z_{s,i} \qquad \text{for } i \in \mathcal{T}_s, \tag{6.2.7f}$$

$$u^{\text{unit}}_{s,i} \geq i \hat{u}_{s,1} z_{s,i} \qquad \text{for } i \in \mathcal{T}_s, \tag{6.2.7g}$$

$$1 \geq \sum_{i \in \mathcal{T}_s} z_{s,i}, \tag{6.2.7h}$$

$$h_r = \frac{x_{t,r} + x_{t+1,r} - 2\underline{x}_r}{2K}, \tag{6.2.7i}$$

$$a_r = g_{s,h_0} + \frac{\partial g}{\partial h}(h_r - h_{r,0}) + 1, \tag{6.2.7j}$$

$$b_r = g_{s,h_0} - \frac{\partial g}{\partial h}(h_r - h_{r,0}) - 1, \tag{6.2.7k}$$

$$a_r = \sum_{l=1}^{L_r^b} \hat{a}_{r,l} \lambda_{r,l}^a, \tag{6.2.7l}$$

$$b_r = \sum_{l=1}^{L_r^b} \hat{b}_{r,l} \lambda_{r,l}^b, \tag{6.2.7m}$$

$$a_r^{\text{squared}} = \sum_{l=1}^{L_r^a} \hat{a}_{r,l}^{\text{squared}} \lambda_{r,l}^a, \tag{6.2.7n}$$

$$b_r^{\text{squared}} = \sum_{l=1}^{L_r^b} \hat{b}_{r,l}^{\text{squared}} \lambda_{r,l}^b, \tag{6.2.7o}$$

$$g_s = \frac{a_r^{\text{squared}} - b_r^{\text{squared}}}{4}, \tag{6.2.7p}$$

$$1 = \sum_{l=1}^{L_s^a} \lambda_{r,l}^a, \tag{6.2.7q}$$

$$1 = \sum_{l=1}^{L_s^b} \lambda_{r,l}^b, \tag{6.2.7r}$$

$$\lambda_{s,i,l}^{\text{power}},\ \lambda_{r,l}^b,\ \lambda_{r,l}^a \in [0,1], \tag{6.2.7s}$$

$$z_{s,i} \in \{0,1\}. \tag{6.2.7t}$$

where:

$\mathcal{T}_s$ = set of turbines for station s,

h_r = average net headwater level adjusted from the minimum storage $\underline{x}_r$ for reservoir r,

$u_s,\ u_{s,i}^{\text{unit}}$ = station flow (cumec) and turbine flow (cumec) for the i'th turbine,

$z_{s,i}$ = $\begin{cases} 1 & \text{if } i\text{'th number of turbines are on,} \\ 0 & \text{otherwise,} \end{cases}$

$L_s^{\text{power}},\ L_r^a,\ L_r^b$ = the number of linear function intervals for the piecewise linear approximations of $g_{s,h_0,i}^{\text{unit}}$, a_r, and b_r respectively,

$x_{t,r},\ x_{t+1,r}$ = volume of water (cubic meter) of reservoir r starting period t and $t+1$ respectively,

$[\lambda_{s,i,l}^{\text{power}}]_{l\in L_s^{\text{power}}}$, $[\lambda_{r,l}^a]_{l\in L_r^a}$, $[\lambda_{r,l}^b]_{l\in L_r^b}$ = interpolation variables for the piecewise linear approximations of nominal power, and the quadratic functions,

$[\hat{u}_{s,l}]_{l\in L_s^{\text{power}}}$, $[\hat{g}_{s,l}]_{l\in L_s^{\text{power}}}$ = turbine flow and power points for the piecewise linear approximation of a single turbine,

$[\hat{a}_{r,l}]_{l\in L_r^a}$, $[\hat{b}_{r,l}]_{l\in L_r^b}$ = a and b points for the piecewise linear approximation of the quadratic variable,

$[\hat{a}_{r,l}^{\text{squared}}]_{l\in L_r^a}$, $[\hat{b}_{s,l}^{\text{squared}}]_{l\in L_r^b}$ = points for a^2 and b^2 of the piecewise linear approximation of the variable a and b.

Constraints (7.2.7a) to (7.2.7h) and (7.2.7t) model the unit switching as binary variables and the nominal power function g_{s,h_0} as a piecewise linear approximation. It uses the binary variable $z_{s,i}$ to select the number, indicated by the value of i, turbines that are turned on. Whenever $z_{s,i} = 1$, it indicates that i turbines are being used to generate power. In this case the turbine flow $u_{s,i}^{\text{unit}}$ of a single unit is be-

tween the minimum $i\hat{u}_{s,1}$ and maximum $i\hat{u}_{s,L_s^{\text{power}}}$ flow, set by constraints (7.2.7g) and (7.2.7f), with all other turbine flow variables $u_{s,\cdot}^{\text{unit}}$ set to 0. Then, the nominal power $g_{s,h_0} = g_{s,h_0,i}^{\text{unit}}$, through constraint (7.2.7a), and the station flow is equivalent to $u_{s,i}^{\text{unit}}$ (constraint (7.2.7b)). The tubine flow $g_{s,h_0,i}^{\text{unit}}$ is equivalent to the single turbine piecewise linear function, based on $u_{s,i}^{\text{unit}}$, and is scaled by i through constraints (7.2.7c) to (7.2.7e). The variables $\lambda_{s,i,l}^{\text{power}}$ interpolate between adjacent nominal power $\hat{g}_{s,l}$ and flow $\hat{u}_{s,l}$ points through constraint (7.2.7e). Additional SOS type II constraints are added to $\lambda_{s,i,l}^{\text{power}}$ to ensure that the interpolated nominal power is not greater than the piecewise approximation. These SOS constraints enforce an ordering of $\lambda_{s,i,\cdot}^{\text{power}}$, where only two adjacent variables can be positive [12].

Based on the nominal power g_{s,h_0} and the head level h_r, the actual power g_s across the station s is represented as the piecewise linear approximation of the difference of the squares, which is represented by the terms a^{squared} and b^{squared} in the model. The head level is the average of the head level of the net volume of water at period t and $t+1$ through constraint (7.2.7i), where K is the surface area of the reservoir. This assumes that the reservoir is cylindrical in shape, and therefore has a constant rate of change in head level with respect to change in volume (i.e. constant K). Constraints (7.2.7j) and (7.2.7k) represent the definition of variables a and b, and are the same as equations (7.2.3) and (7.2.4) respectively. Constraints (7.2.7l) to (7.2.7o), and (7.2.7q) to (7.2.7r) are the piecewise linear approximations of the quadratic functions for a and b. SOS type II constraints are also added for $\lambda_{r,\cdot}^{a}$ and $\lambda_{r,\cdot}^{b}$ weight variables in order to ensure that the interpolated value does not exceed the piecewise approximation. The overall power g_s, generated from the station, is equivalent to the difference of a^{squared} and b^{squared} set by the constraint (7.2.7p). This is equivalent to Equation (7.2.5).

Even though this approximation includes the switching of individual turbines, it does not take into account cost of their switching. It is expensive to switch turbines on

and off, referred to as unit switching, as it damages the turbines and reduces their lifespan. Therefore, production schedules that have less frequent unit switching are desirable. Potential ways to incorporate unit switching inside hydro-bidding models are discussed in Section A.2.1 of Appendix A. Including switching costs inside the objective introduces an additional binary state variables, which represent the state of the turbine for periods t and $t+1$. This adds another layer of complexity on the sub-problems solved in both MIDAS and SDDP. Therefore, it is not explicitly included inside the objective of the hydro-bidding models presented in this chapter. Since, the objective maximizes hydro generator surplus, it should naturally reduce the number of unit switching in order to generate efficiently. Incorporating turbine state variables inside the value function of MIDAS and SDDP provides for a good research direction in the future.

In the next section, Section 7.2.2, we describe how the aforementioned approximation was numerically carried out, and analyse its accuracy for a single hydrostation.

6.2.2 Analysis of the approximation of $g(h, u)$

We applied this approximation to a test hydro station with the empirical power function illustrated in Figure 7.19. The nominal power function was based on the power flow profile for the average head level. The nominal power flow profile was then partitioned into 6 individual profiles based on the observed power increments, which indicated the starting of additional turbines. These partitions were then adjusted to represent a single unit. A 5 point (4 linear function pieces) piecewise linear function was then fitted to these data points. This piecewise linear function approximates nominal power function g^{unit}_{s,h_0} for a single turbine. The rate of change in the power with the headwater was calculated as the rate of change between the measured power at the measured head level with respect the nominal power at the nominal head level.

This was then averaged across all the individual measurements to obtain an average rate. Based on these measures the terms a and b along with their respective quadratic terms were calculated. Similar to the power approximation, a 5 point (4 linear function pieces) piecewise linear function was fitted to the quadratic functions to the term a^{squared} and b^{squared}. The piecewse approximation parameters are listed on Table A.3 in Appendix A.

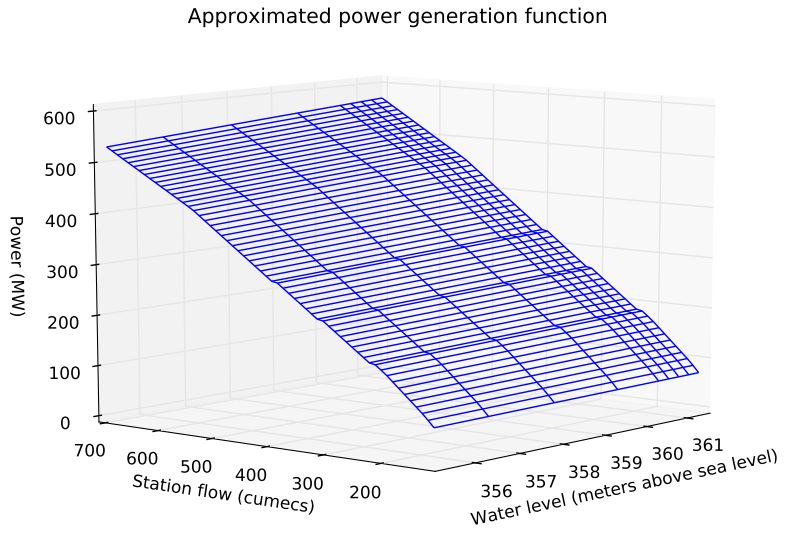


Figure 6.22: Approximated power function from empirical measurements for a station with 6 turbines.

Figure 7.22 illustrates the approximate power generation function based on the measured flow and head levels. As observed, the shape of the approximation closely resembles the original function. This approximation captures the power increments, and the cloud-like shape from the original function, albeit with less accuracy. This is illustrated in Figure 7.23 where there are distinct peaks, which indicate the starting of additional turbines. In addition to representing the price peaks, this approximation also captures the bilinear nature of the original power function, also illustrated in Figure 7.23.

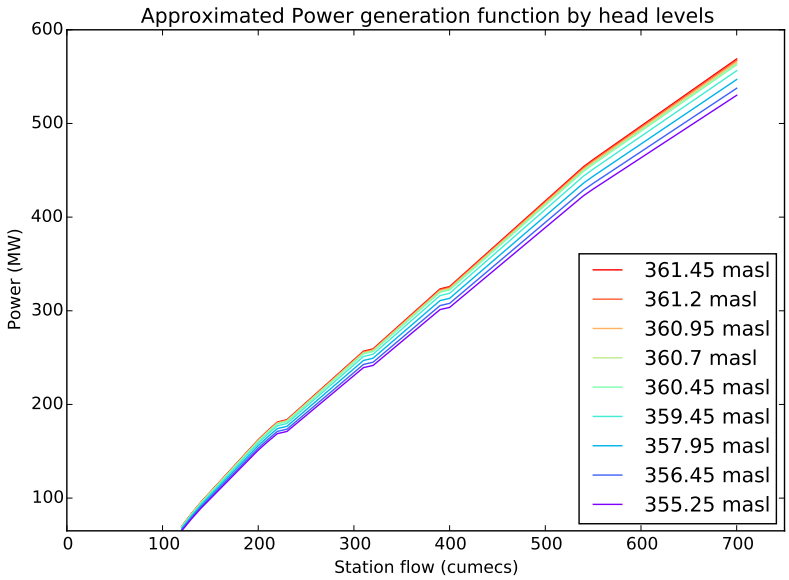


Figure 6.23: Approximated power function for each measured head levels.

Measuring the accuracy of the approximation indicates a maximum error of 8% at the lowest head level, and at the minimum flow. As illustrated in Figure 7.24, the errors are the highest at low head levels and during flows when an additional turbine is turned on. This is due to the formulation of the nominal power function. Recall, that the nominal power function is modeled for a single turbine, which is then scaled multiplicatively by the number of tubines in use. This introduces additional errors at flow ranges when the power function is scaled by the addition of a turbine. The error is greatest just after the power increments, when a new turbine is switched on, and are at its lowest just before the additional turbine is turned on.

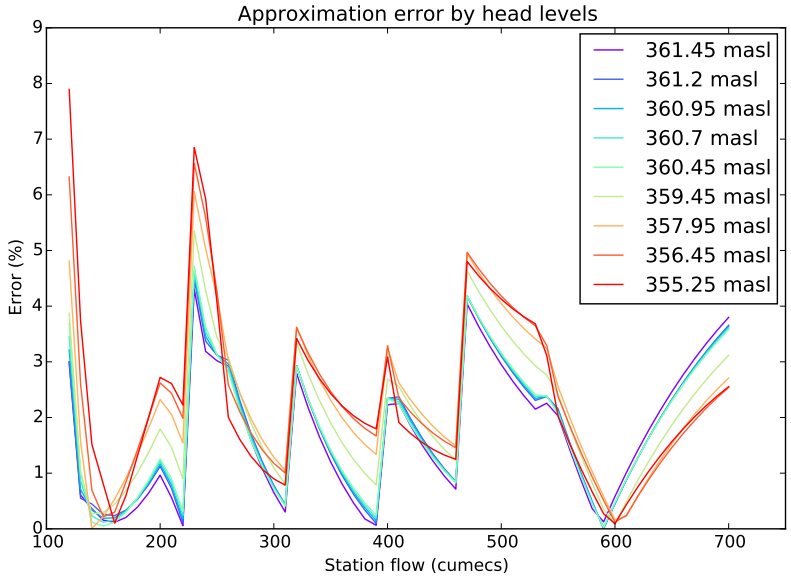
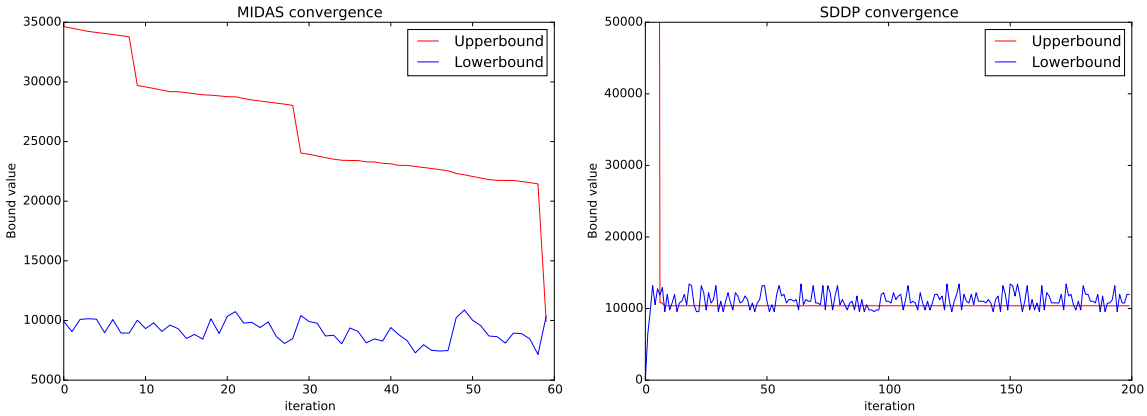


Figure 6.24: Measured error of the approximate power function for each measured head levels.

6.2.3 Comparison of MIDAS and SDDP

We implemented the head level approximation inside the 3 tranche offerstack hydro-bidding model, Model (6.3.8) in Chapter 6, with zero inflow (i.e. $\omega_t = 0$) and zero spill assumptions along with the same Markov price process. The original objective function was replaced with that of (A.2.2) in the model as well as setting the x_{t+1} to be a continuous variable. Algorithm 8 was used to solve this model for a single reservoir hydro scheme with $\delta_x = 10,000$. The initial storage level was set to 1% of the maximum level in order to simulate drought conditions. This is because, during these conditions, when there is a limited volume of water available, efficieny in generation is important. We also added the new power function formulation into an SDDP equivalent of Model (6.3.8). In the SDDP model, the binary turbine state variables and the SOS type II constraints were relaxed in the backward pass in order to extract the dual values and construct the hyperplanes.



(a) MIDAS

(b) SDDP.

Figure 6.25: Convergence comparison between MIDAS and SDDP for a single reservoir hydro scheme with headwater effects.

Figure 7.25a illustrates the convergence of MIDAS within 60 iterations. The upper bound steadily decreases as the value function approximation improves, and the lower bound increases towards the optimal policy. Similar to the price uncertainty model, MIDAS takes longer to converge than SDDP (Figure 7.25b). For the same reason discussed in Section 7.1.3 it has to thoroughly explore the state space in order to construct the optimal policy. As illustrated in Figure 7.26, it begins with a greedy policy, where it offers all of the station capacity in early time periods. Then, it improves on this policy by exploring state values that are δ_x distance apart, which takes longer than SDDP to converge.

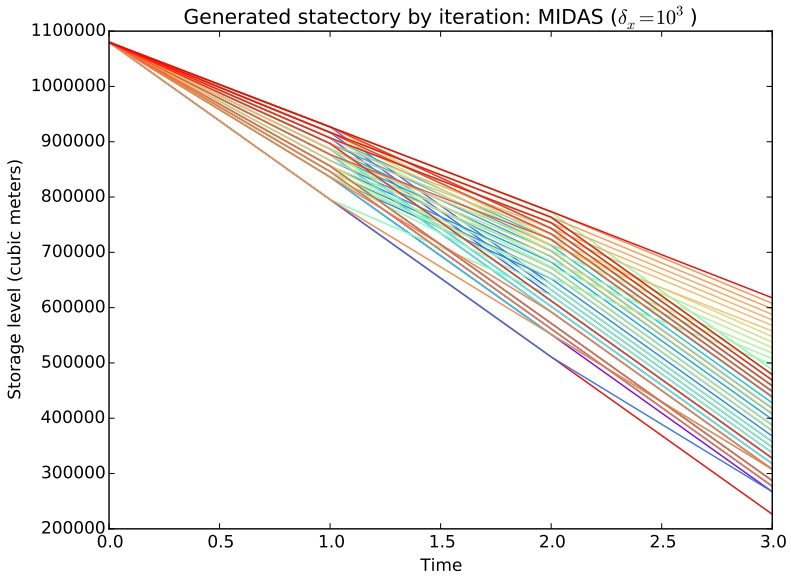


Figure 6.26: Generated state trajectory by iteration of the MIDAS algorithm of single reservoir hydro scheme with headwater effects.

6.3 Summary

In this chapter, we presented different approaches to modelling a mean reverting autoregressive price process, and headwater effects into the hydro-bidding problem. In both of these cases the state variable was continuous, but the overall hydro-bidding problem is non-convex. We first illustrated that MIDAS can incorporate these price processes by adding a price dimension into the state variable. We also introduced a cut interpolation technique in SDDP, which enabled us to incorporate these price processes. The interpolation technique discretized the price domain by K price values. Based on these price values, the price outcomes from the mean reverting autoregressive price process could be interpolated by two neighboring values, as stated by Equation (7.1.9). Using the same interpolation weights, the value function could then be interpolated between two approximate value functions of adjacent price nodes (see Figure 7.6). By storing the cuts at these price nodes, we can compute the value function for

a wider range of price outcomes.

We compared MIDAS and the interpolated SDDP for a 1, 2, and 5 reservoir hydro scheme. We observed that both methods produce similar optimum policies, but SDDP was more computationally efficient than MIDAS. SDDP, with its cutting plane approximation, exploited the gradient of the cuts to explore new states, where as MIDAS used the δ_x. This required MIDAS to more thoroughly explore the state space, as illustrated by Figure 7.13, in order to construct an optimum policy.

Although MIDAS is slower compared to SDDP, it can model multivariate price processes. For SDDP, incorporating these price processes requires carrying out multivariate interpolation among the neighboring price nodes between each price dimensions. MIDAS on the other hand simply adds another price dimension to the state variable. There is no need for additional calculations because the formulation of the step function approximation of the value function would still be the same.

We then incorporated headwater effects into the power generation function. We presented an alternative approximate power function formulation. The formulation first modeled the power function for a single turbine as a piecewise linear function of turbine flow at a nominal head level. Then, we linearly scaled this piecewise power function based on the changes in the head level. This approach made the overall power function bilinear, therefore we re-formulated the function as a difference of squares to separate the bilinear terms. In order to implement it inside the linear sub-problems of MIDAS and SDDP, we approximated each of the squares using piecewise linear functions. We used SOS type II constraints to represent the piecewise linear function approximation inside the solvers.

We solved a 3 tranche offerstack hydro-bidding model similar to Model (6.3.8) in Chapter 6 using MIDAS and SDDP. We observed a similar convergence characteristics as the price uncertainty case, where SDDP was computationally more efficient than MIDAS.

MIDAS iteratively improved a starting greedy policy by exploring state values that were δ_x distance apart. This took longer than SDDP. However, MIDAS had a tighter upper bound of 9931.022, than SDDP, which had upper bound of 10389. This was because MIDAS could explicitly represent the approximate power function, whereas SDDP, through its relaxation of the binary SOS type II constraints, convexified the approximation.

In Chapters 6 and 7 we observed that MIDAS, although it produces better upper bounds and optimum policies, is slower than the SDDP equivalent. It does not exploit any gradient information of the value function, unlike SDDP. This makes it slower because it enumerates across the state space in order to construct an optimum policy. Furthermore, it solves SMIP sub-problems, which grow in size with additional step functions. Hence, industrial solvers take longer to solve the sub-problems due to the increasing number of binary variables and big-M constraints. In the next chapter, Chapter 8, we analyze MIDAS and propose some heuristics to improve its computational inefficiency.

Chapter 7

Improving the computation of MIDAS

In Chapter 5, we introduced the Mixed-Integer Dynamic Approximation Scheme (MIDAS), and proved its finite convergence for continuous and integer state variables. Then in Chapters 6 and 7 we used MIDAS to model various versions of the hydro-bidding problem for continuous and integer state variables. The policies that were constructed using MIDAS were compared to the policies constructed using the Stochastic Dual Dynamic Programming (SDDP) equivalent. For models with integer state variables (i.e. Chapter 6), the offer policies constructed by MIDAS were superior to the policies of SDDP. However, MIDAS took much longer to converge than SDDP, and in many cases for large hydro-schemes the sub-problem became too large to solve using industrial solvers such as CPLEX [23]. As for the continuous state variable case, it was noticeably slower than SDDP. Its computation time depended on the value of δ. A larger δ would ensure less computation time for convergence but greater optimality error ε. Furthermore, the policies of SDDP and MIDAS were similar to each other.

At each iteration of MIDAS, and with each step function, the MIP sub-problems adds

$M(2 + |R|)$ constraints and $M(1 + |R|)$ binary variables. As a result, the MIP sub-problems increase in the number of variables and big-M constraints with each iteration of the backward pass, which in turn increases the computation time for solving the sub-problem, as summarized in Table 8.1. The execution time for the SDDP equivalent of Model (6.3.8) in Chapter 6, is significantly less. MIDAS on the other hand takes several hundred seconds to converge, where as SDDP takes on average a maximum of 30 seconds to solve the problem with the largest data set (2 reservoir hydro-scheme, 5 Markov states, and 9 stages).

		Mean execution Time (sec)		**Mean Upper Bound (lower is better)**	
M	T	**SDDP**	**MIDAS**	**SDDP**	**MIDAS**
3	5	4.990	107.484	8588.410	7876.953
3	7	9.954	228.398	8960.435	8446.267
3	9	20.093	354.986	9444.515	8454.877
5	5	8.197	440.858	9310.741	8054.981
5	7	19.547	563.756	10106.507	8444.402
5	9	30.630	617.277	10636.140	9117.780

Table 7.1: Comparing the computation time of SDDP and MIDAS with increasing T and M for 2 reservoir hydro-scheme in Chapter 6.

Unlike SDDP, MIDAS does not utilize information associated with the gradient of the value function. As a result, it has to explore the state space rigorously in order to yield a good approximation. Furthermore, the sub-problems are MIPs which increase with the number of step functions present. This increases the size of the branch and bound tree inside the solvers like CPLEX [23], which causes many of these solvers to run out of memory. It is due to this memory error that MIDAS has been limited to

solving hydro-schemes with up to 3 reservoirs for integer state variables.

Both the size of the sub-problem and the computation time can be reduced if the step function implementation is re-formulated as a linear program. However, this is not achievable with the structure of the step functions. Fortunately, we can improve the computation of MIDAS by applying similar techniques to those used in SDDP, such as cut selection. In this chapter, we present two heuristics to improve the computational efficiency of MIDAS and study their impact. The first is similar to SDDPs cut selection described in [26], where step functions are filtered out based on a specific criteria proposed in Section 8.1. The benefit of such a method is that it removes redundant step functions and reduces the size of the MIPs without hindering the accuracy of the value function approximation.

The second is a heuristic that solves the sub-problem iteratively, discussed in Section 8.2. It is based on the L-shaped method proposed by [45]. The benefit of using this heuristic is that it no longer requires using the $Q^H(x)$ MIP formulation, in Section 5.5 of Chapter 5, in the sub-problem. Instead it uses the cuts proposed by [45] to directly integrate the binary variables associated with the discrete production with each of the step functions. This significantly reduces the branch and bound tree inside the solvers, and increases their capability to solve large sub-problems in MIDAS.

7.1 Step function selection scheme

Unlike SDDP, MIDAS does not utilize information associated with the gradient of the value function. As a result, MIDAS has to explore the state space rigorously in order to yield a good approximation. Figure 8.1 illustrates the step functions added by MIDAS in period 8 of a hydro-bidding problem with a continuous state variable. They are color coded by iteration, with the color bar in the figure indicating in which

iteration the step functions were added to the sub-problem. As illustrated in the figure, the step function, computed in earlier iterations, are redundant in later stages of the MIDAS algorithm. They do not improve the approximation of the value function. This provides an opportunity for us to filter out the redundant step functions in order to reduce the size of the sub-problems and time it takes to solve them in solvers such as CPLEX [23].

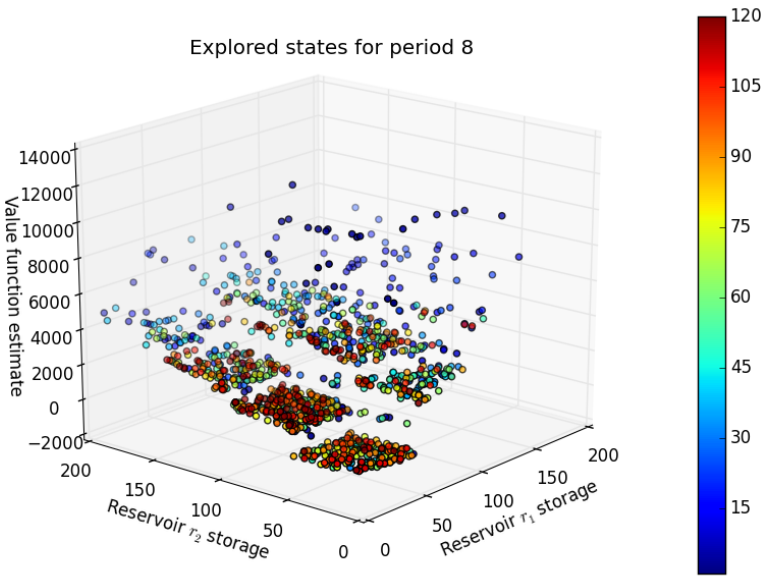


Figure 7.1: Step functions used to approximate the value function of a 2 reservoir hydro scheme with continuous state variables.

Similar to the cut selection approach in SDDP, we can apply a step function selection scheme, which removes redundant step functions in order to reduce the size of the sub-problem. The step function selection scheme works by removing a step function h_1, which is dominated by another step function h_2 (i.e. $x_{t,r}^{h_1} \leq x_{t,r}^{h_2}$ for $r \in \mathcal{R}$), that breaks the monotonicity property of the value function (i.e. $q_t^{h_1} \geq q_t^{h_2}$). In these conditions h_1 does not improve the value function approximation as $Q_t^H(x) \leq q_t^{h_2}$ for $x \leq x_t^{h_1}$. Figure 8.2 illustrates an example, where step function 2 breaks the monotonicity property (i.e. $q_t^3 < q_t^2$) but is dominated by step function 3 where

$x_t^2 \leq x_t^3$. As a result the $Q_t^H = \min\{q_t^3, q_t^2, M\} = q_t^3$ for points which are dominated by x_t^2. Therefore, having step function 2 in the sub-problem does nothing to improve the approximation.

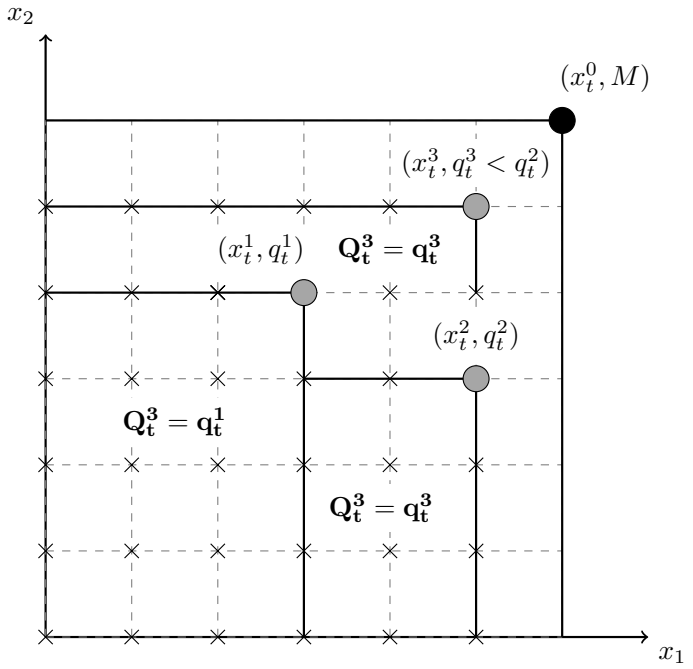


Figure 7.2: Example of a redundant step function function (x_t^2, q_t^2).

The algorithm for step function selection is described by Algorithm 11. It is a routine that can be added to MIDAS between the forward and the backward pass, or even after the convergence testing phase.

Algorithm 11 : step function selection scheme.

1. Set $\mathcal{C} = \emptyset$.

2. For $h_1 \in \{1, \ldots, H\}$ and $h_2 \in \{1, \ldots, H\} \setminus \{h_1\}$ do,

   (a) if $x_t^{h_1} \leq x_t^{h_2}$ and $q_t^{h_1} \geq q_t^{h_2}$ then $\mathcal{C} = \mathcal{C} \cup \{h_1\}$.

3. Redefine $Q_t^H(x)$ with (x_t^h, q_t^h) for $h \in \{1, \ldots, H\} \setminus \mathcal{C}$ points.

7.1.1 Analyzing computational performance of step function selection

Applying this step function selection scheme to MIDAS yields the following improvements summarized by Table 8.2. The results in the table are obtained by executing the model under the same initial conditions 30 times in order to obtain an averaged measurement of the execution time and the upper bound. Based on the results, the step function selection scheme reduces the solution time of MIDAS by a minimum of 40%. The average time, ranges from 65 to 312 seconds with step function selection, where it originally took between 107 to 617 seconds.

		Mean execution Time (sec)			**Mean upper bound (lower is better)**	
M	T	**Original**	**step function selection**	**Reduction (%)**	**Original**	**step function selection**
3	5	107.484	65.626	39%	7876.953	7876.953
3	7	228.398	135.730	41%	8446.267	8271.274
3	9	354.986	198.538	44%	8454.877	8337.396
5	5	440.858	118.092	51%	8054.981	8102.071
5	7	563.756	222.160	64%	8444.402	8702.847
5	9	617.277	312.613	45%	9117.780	9146.019

Table 7.2: Comparison of the execution time of MIDAS without and with step function selection, across increasing T and M for a 2 reservoir hydro scheme.

For instance, applying the step function selection scheme to the 5 Markov state 9 stage hydro-bidding problem reduces the number of step functions from 41 to 17. As illustrated in Figure 8.3, there are essentially two layers of step function functions on

the original $Q_2(x)$. One layer has a higher value function estimate, and is dominated by the second layer. The step function selection scheme removes this layer in order to tighten the value function approximation, while also reducing the number of binary variables and big-M constraints in the sub-problem.

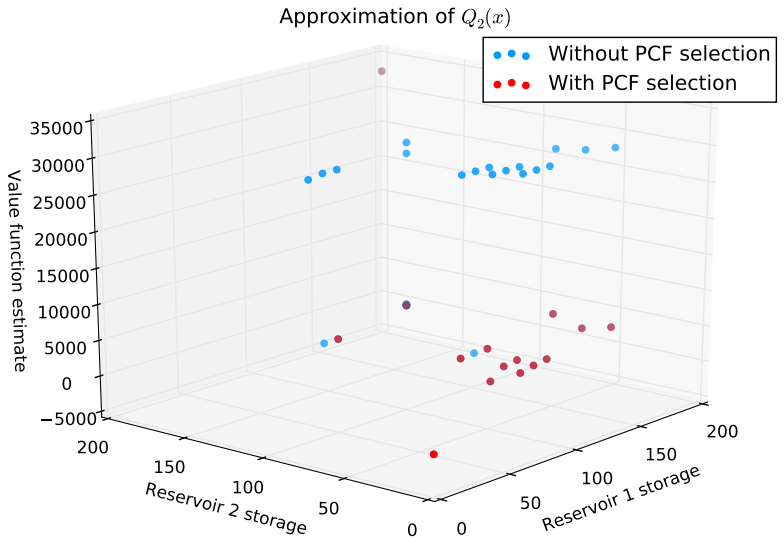
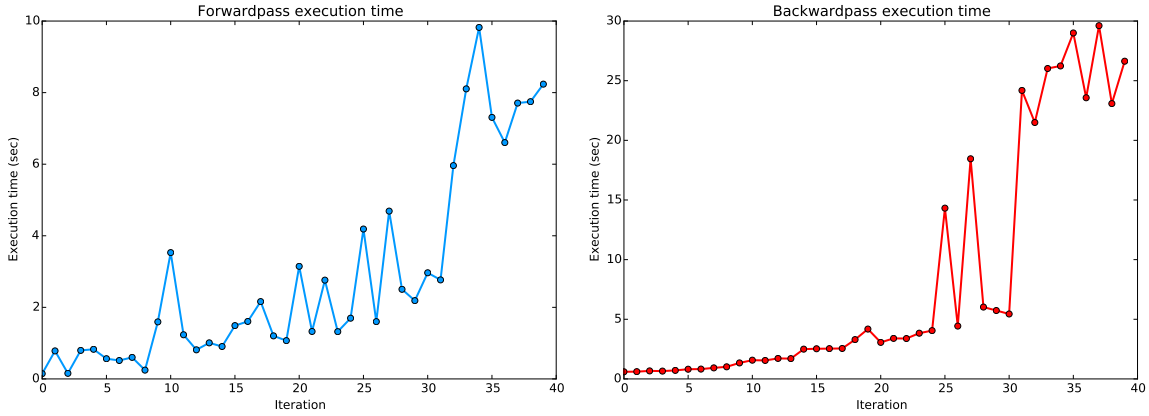


Figure 7.3: Comparison of the step function selection scheme for the 5 Markov state and 9 stage problem for Markov state 0.

However, the solution time with step function selection is still higher than for SDDP. This is because, even with step function selection, the sub-problems are still MIPs, which are always computationally more expensive than solving a linear program equivalent. Even with step function selection, MIDAS could take a while to solve for large hydro-schemes, as eventually it could accumulate a large set of step functions. The step function selection scheme also has an order of n^2 time complexity because it cross references step functions, which may add to the overall execution time of the algorithm. Although the results show a great improvement, the level of this improvement may be marginal for large data sets. Moreover, the improvement depends on both the model and initial conditions.

7.2 MIP sub-problem solver

Due to the ever increasing size of the MIP sub-problem in terms of the number of binary variables, industrial solvers such as CPLEX [23] are limited by their capability to store the branch and bound tree in memory. Moreover, the branch and bound tree will become astronomically large, which increases the time to solve the sub-problems. This is illustrated by figures 8.4a and 8.4b, where the execution time of the forward and backward passes exponentially increase with the addition of step functions in each iteration.



(a) Forwardpass (b) Backwardpass

Figure 7.4: Forward and backward pass execution time by iteration for the 2 reservoir hydro-scheme, 5 Markov state, and 10 staged model.

One of the reasons for such a rapid increase in the execution time of the forward and backward pass is due to the slow reduction of the upper bound (UB) of the branch and bound tree in order to reduce the large MIP gap. In comparison to the lower bound (LB), which reaches the optimal integer solution early in a particular node of branch and bound tree, the upper bound, which is the relaxed linear program equivalent at this node, takes longer to converge. This is due to the fathoming of the other nodes.

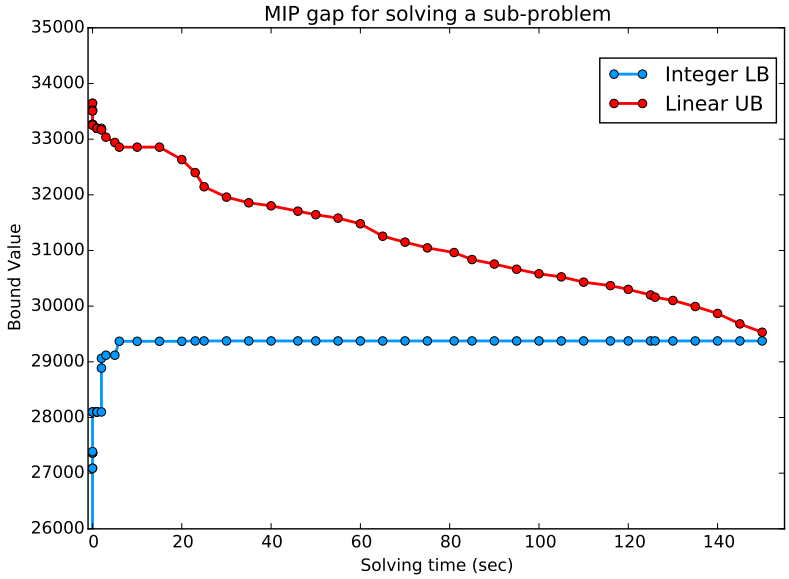


Figure 7.5: Lower and upper bound of the branch and bound tree for solving a MIP sub-problem in MIDAS using CPLEX.

To address the limitations of the increasing MIP sub-problem, and the lingering high upper bound for closing the MIP gap, we propose a heuristic for solving the MIP sub-problems iteratively, without adding any additional binary variables. It is based on the optimal cuts, which we refer to as binary optimal cuts in the L-shaped method defined by [45]. The reader is directed to [70], which provide a detailed summary of the method with numerical examples. Furthermore, this method has been further extended to solve multistage stochastic problems, in similar fashion to SDDP and MIDAS, for binary state variables by [83].

This heuristic operates like two stage Benders decomposition. It first decomposes the MIP hydro-bidding sub-problem into a master problem $\overline{MP}(t, i, x_t)$ (Model (8.2.1)), which is the same as Model (6.3.8) but without the step function constraints for representing $\hat{V}_{t+1,j}$ (i.e. constraints (6.3.8i) to (6.3.8k) in Chapter 6).

$$\overline{MP}(t,i,x_t):$$

$$\begin{aligned} \overline{V}_{t,i}(x_t) = \quad & \max \sum_{j=1}^{M} P_{i,j}(t)\left[\pi_{t,j} o_{t,j} + \hat{V}_{t+1,j}\right] \\ & \text{subject to:} \\ & (6.3.8a), \\ & \dots \\ & (6.3.8h). \end{aligned} \tag{7.2.1}$$

The heuristic iteratively improves the value function approximation by adding binary optimality cuts into the master problem until convergence. For instance, at iteration K it solves $\overline{MP}(t,i,x_t)$ and obtains the set of first stage binary variable values $z^K_{j,s,l}$ and state values $x^K_{t+1,j}$ for each Markov state j, station s, and production state l. Then, it computes the set I^K of binary variables that are equal to one, defined in Equation (8.2.2).

$$I^K = \left\{(j,s,l) \mid z^K_{j,s,l} = 1, \text{ for } j = 1,2,\dots,M,\ s \in S,\ l \in L_{t,s}\right\}. \tag{7.2.2}$$

Once it has constructed the sets I^K, it computes the new estimate for $\hat{V}_{t+1,j}$ as

$$q^K_{t+1,j} = \min\left\{\mathbf{M}, q^h_{t+1,j} :\ h \in \mathcal{H}^H_{\mathbb{Z}}(x^K_{t+1,j})\right\} \tag{7.2.3}$$

based on the step functions for the state $x^K_{t+1,j}$ value for each j Markov state. After computing $q^K_{t+1,j}$, it adds a binary optimal cut defined by Equation 8.2.4 to $\overline{MP}(t,i,x_t)$.

$$\hat{V}_{t+1,j} \le q^K_{t+1,j} + \left(\mathbf{M} - q^K_{t+1,j}\right)\left(\sum_{(j,s,l)\notin I^K} z_{j,s,l} - \sum_{(j,s,l)\in I^K} z_{j,s,l} + |I^K|\right) \text{ for } j = 1,2,\dots,M. \tag{7.2.4}$$

The binary optimal cut approximates $\hat{V}_{t+1,j}$ at the binary solution $z^K_{j,s,l}$ and the big-M (i.e. $\mathbf{M}$) for all other solutions. The term

$$\sum_{(j,s,l)\notin I^K} z_{j,s,l} - \sum_{(j,s,l)\in I^K} z_{j,s,l} + |I^K|$$

equals to zero whenever the binary solution $z_{j,s,l} = z^K_{j,s,l}$ for all $j = 1, 2, \ldots, M$, $s \in S$, and $l \in L_{t,s}$. When this happens $\hat{V}_{t+1,j} \leq q^K_{t+1}$. However if $z_{j,s,l} \neq z^K_{j,s,l}$ for some $(j, s, l) \in I^K$, then there is at least one binary variable $z_{j,s,l}$ not in the set I^K which switches from 0 to 1. This makes the term

$$\sum_{(j,s,l)\notin I^K} z_{j,s,l} - \sum_{(j,s,l)\in I^K} z_{j,s,l} + |I^K|,$$

at least one, which makes the binary optimal cut unbounded and $\hat{V}_{t+1,j}$ being at least bounded by the big-M term, where

$$\hat{V}_{t+1,j} \leq q^K_{t+1} + \mathbf{M} - q^K_{t+1} = \mathbf{M}.$$

An illustration of how the binary optimal cut approximates the value function at the point $x^K_{t+1,j}$ for the binary solution $z^K_{j,s,l}$ is provided in Figure 8.6. At the point $x^K_{t+1,j}$ with the solution $z^K_{j,s,l}$, $\hat{V}_{t+1,j}$ is projected onto $q^K_{t+1,j}$ from the big-M upper bound $\mathbf{M}$ by the difference $\mathbf{M} - q^K_{t+1,j}$, where $q^K_{t+1,j}$ is the value function estimate based on the step functions.

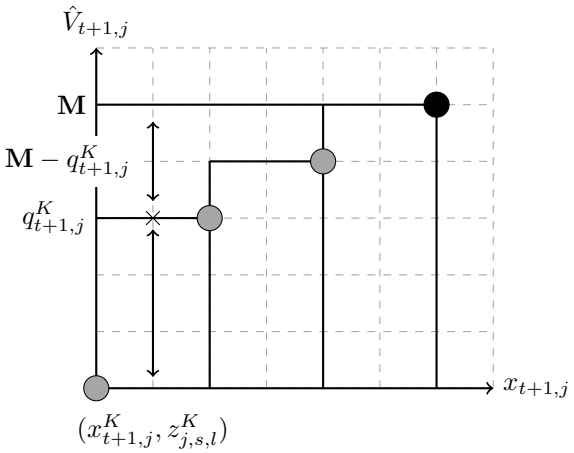


Figure 7.6: Example of a binary optimal cut approximating $\hat{V}_{t+1,j}$ at point $x^K_{t+1,j}$ for a 1 reservoir hydro-scheme.

The pseudocode below describes the proposed heuristic for solving the MIP sub-problems in MIDAS. Algorithm 12 is similar to the Bender's decomposition. At each iteration it computes a new state value $x^K_{t+1,j}$ by solving $\overline{MP}(t, i, x_t)$. Then, it updates $\hat{V}_{t+1,j}$ by adding a binary optimal cut based on value function approximation $Q^H_{t+1}(x^K_{t+1,j})$. Since each binary optimal cut applies to a particular binary solution $z^K_{j,s,l}$ and state $x^K_{t+1,j}$, it naturally explores a new binary solution and a new state in consecutive iterations. It keeps iterating between the different binary solutions and states until the upper bound (V^{up}) and the lower bound (V^{low}) is within $\epsilon_{\text{convergence}}$. Because there are a finite number of binary solutions then Algorithm 12 will always converge within a finite number of iterations.

Algorithm 12 : Solving the MIP hydro-bidding subproblem in MIDAS using the binary optimal cuts and the L-shaped method.

1. **Input:** t, i , x_t, K_{Max}, $\epsilon_{\text{convergence}}$.

2. Set $K = 1$, $V_{\text{up}} = \mathbf{M}$, $V_{\text{low}} = 0$.

3. While $V_{\text{up}} - V_{\text{low}} \leq \epsilon_{\text{convergence}}$ and $K \leq K_{\text{Max}}$ do,

(a) solve $\overline{MP}(t, i, x_t)$ set $V^{\text{up}} = \min\{V_{\text{up}}, \mathbf{M}\}$ and $V^{\text{low}} = \sum_{j=1}^{M} P_{i,j}(t)\pi_{t,j}o_{t,j}$,

(b) compute $z^K_{j,s,l}$ for $s \in \mathcal{S}$ and $l \in L_{t,s}$, its set I^K, and $x^K_{t+1,j}$ for $j = 1, 2, \ldots, M$,

(c) compute $q^K_{t+1,j}$ based on (8.2.3) for $j = 1, 2, \ldots, M$,

(d) add the binary optimal cuts based on (8.2.4) to $\overline{MP}(t, i, x_t)$ for $j = 1, 2, \ldots, M$;

(e) set $V_{\text{low}} = V_{\text{low}} + \sum_{j=1}^{M} P_{i,j}(t)q^K_{t+1,j}$.

4. **Output**:V_{up} and $x_{t+1,j}$ for $j = 1, 2, \ldots, M$.

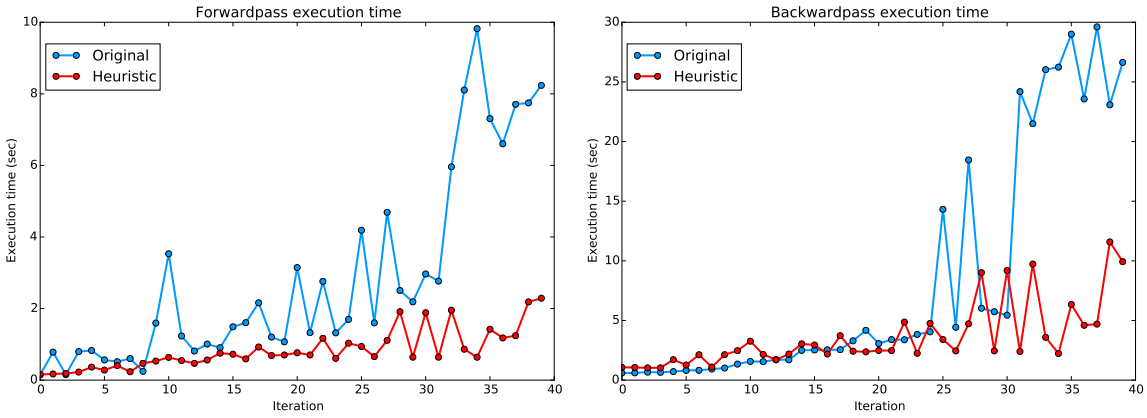
7.2.1 Analyzing computational performance of the heuristic

Applying this heuristic to the algorithm yields the following improvements, summarized by Table 8.3. The results in the table are obtained by executing the model under the same initial conditions 30 times in order to get an averaged measurement of the solution time and the upper bound. Based on the results, the heuristic reduces the solution time of MIDAS by a minimum of 17%. The average time, ranges from 90 to 291 seconds with the heuristic, where it originally took between 107 to 617 seconds.

		Mean execution Time (sec)			**Mean upper bound (lower is better)**	
M	T	**Original**	**Heuristic**	**Reduction (%)**	**Original**	**Heuristic**
3	5	107.484	89.603	17%	7876.953	8243.320
3	7	228.398	146.461	36%	8446.267	8842.056
3	9	354.986	168.622	52%	8454.877	9086.276
5	5	440.858	139.986	42%	8054.981	8701.660
5	7	563.756	247.230	60%	8444.402	9359.539
5	9	617.277	291.242	48%	9117.780	9836.071

Table 7.3: Comparing the computation time of MIDAS with and without using the heuristic, across increasing T and M for 2 reservoir hydro-scheme.

In both the forward pass (Figure 8.7a) and the backward pass (8.7b) the heuristic is significantly faster than the original. Initial iterations of MIDAS show the heuristic has a slightly higher execution time than the original MIP. However, with larger MIP sub-problems the heuristic is computationally more efficient than solving the original MIP, with lower execution times.

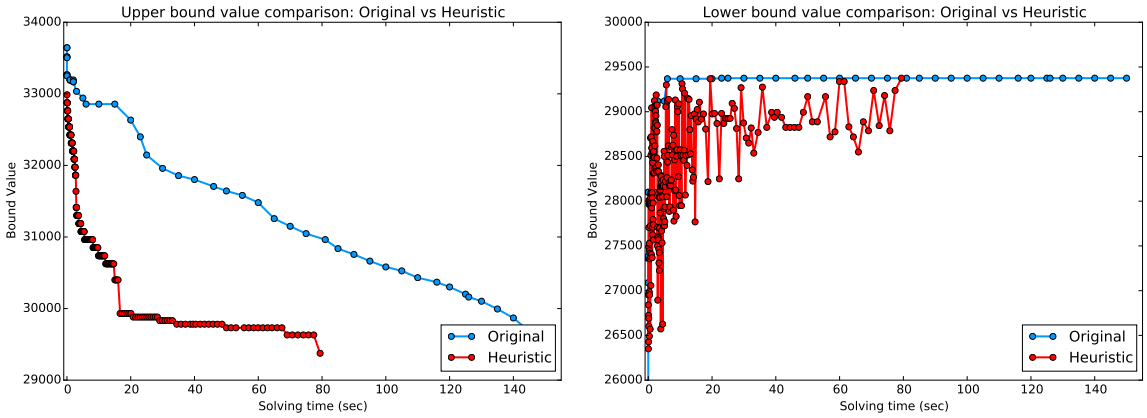


(a) Forward pass

(b) Backward pass

Figure 7.7: Comparison of the forward and backward pass execution time by iteration between the original algorithm and the implemented heuristic.

The benefit of using the heuristic is that it removes the $M(2+|\mathcal{R}|)$ constraints and $M(1+|\mathcal{R}|)$ binary variables that are added to the original sub-problems in MIDAS for each step function. This drastically reduces the branch and bound tree of the MIP, and enables the heuristic to solve $\overline{MP}(t,i,x_t)$ rapidly in each iteration. Moreover, with each addition of the binary optimal cuts, the heuristic is not adding any additional integer or binary variables, only M constraints. Therefore, the heuristic reduces the upper bound faster than the linear program relaxed upper bound of the original MIP. This is illustrated in Figure 8.8a, where the upper bound from the heuristic decreases at a faster rate than the upper bound of the original MIP sub-problem. However, the lower bound (Figure 8.8b) takes longer to converge using the heuristic compared to the original MIP because of the point estimate of the binary optimal cuts. Since a binary optimal cut is only valid at its respective point $x^K_{t+1,j}$ and $z^K_{j,s,l}$ (for $j = 1, 2, \ldots, M$, $s \in S$, and $l \in L_{t,s}$), then $\overline{MP}(t,i,x_t)$ only has to change one of the $z_{j,s,l}$ variables that were originally 0 to 1 in order to make $\hat{V}_{t+1,j} \leq \mathbf{M}$. Therefore, the heuristic has to enumerate across many $x^K_{t+1,j}$ points in order to improve the lower bound.



(a) upper bound

(b) lower bound

Figure 7.8: Lower and upper bound comparison between the branch and bound tree for solving a MIP sub-problem and the heuristic in MIDAS using CPLEX.

7.3 Solving a 4 reservoir hydro-scheme with MIDAS

As observed, both the step function selection scheme, and the sub-problem solver heuristic improves the computational efficiency of MIDAS. However, implementing them both inside MIDAS does not add their individual improvements. Nonetheless, using both of these heuristics have enabled MIDAS to solve a hydro-bidding problem consisting of 4 reservoirs, 3 Markov states, and 4 stages. The model parameters are the same as those described in Table 6.6 and Figure 3.9. The model took 850 iterations (see Figure 8.9) to converge with a total time of $367,000$ seconds. Although the model takes approximately 4 days to solve, its execution time for the forward and backward passes does not exponentially increase with the number of step functions. In fact, its execution time for each of the phases increase linearly by iteration, and plateaus towards an average of 500 seconds for the backward pass, and 100 seconds for the forward pass. This is due to the synergy of both heuristics. For instance, the step

function selection heuristic reduces the number of iterations that the sub-problem solver (Algorithm 12) takes to converge by reducing the number of step functions used to approximate the value function. And the sub-problem solver heuristic only solves MIPs with a small, and constant, number of discrete variables, which reduces the total time it takes to solve the sub-problem. Therefore, the computation times for each of the sub-problems are no longer exponentially increasing with additional step functions.

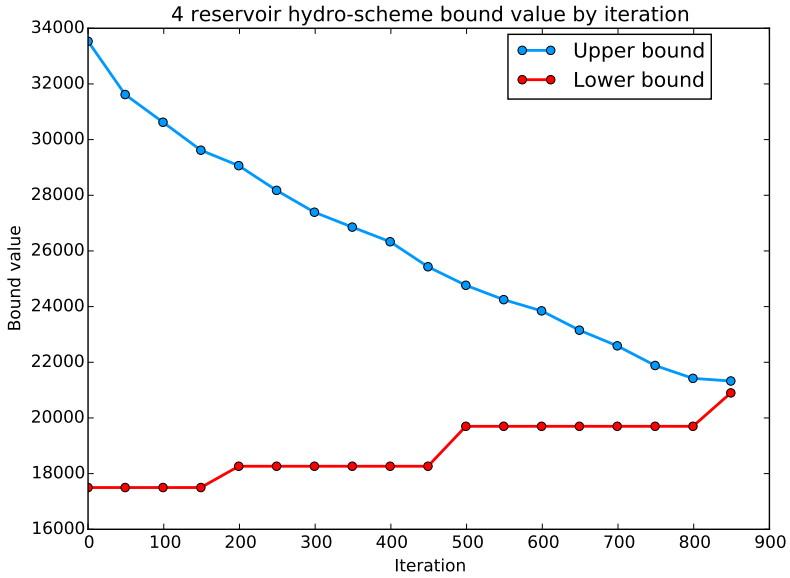


Figure 7.9: Lower and upper bound of the branch and bound tree for solving a MIP sub-problem in MIDAS using CPLEX.

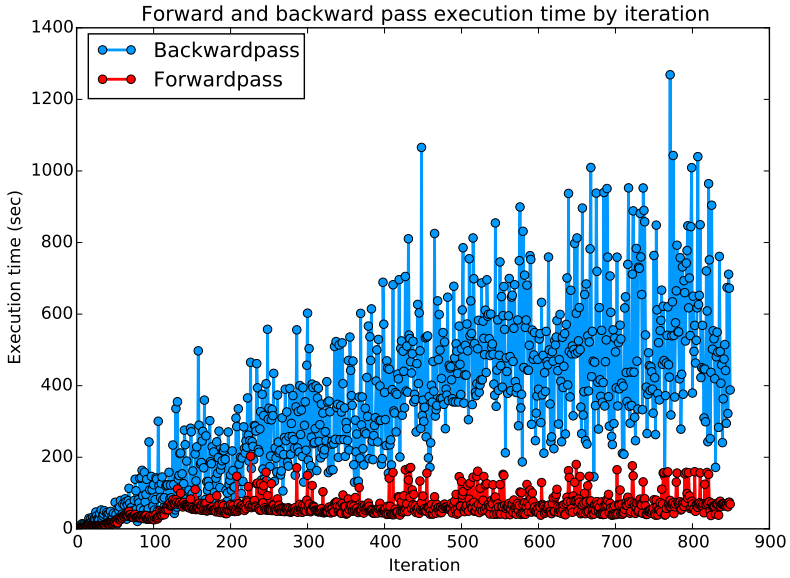
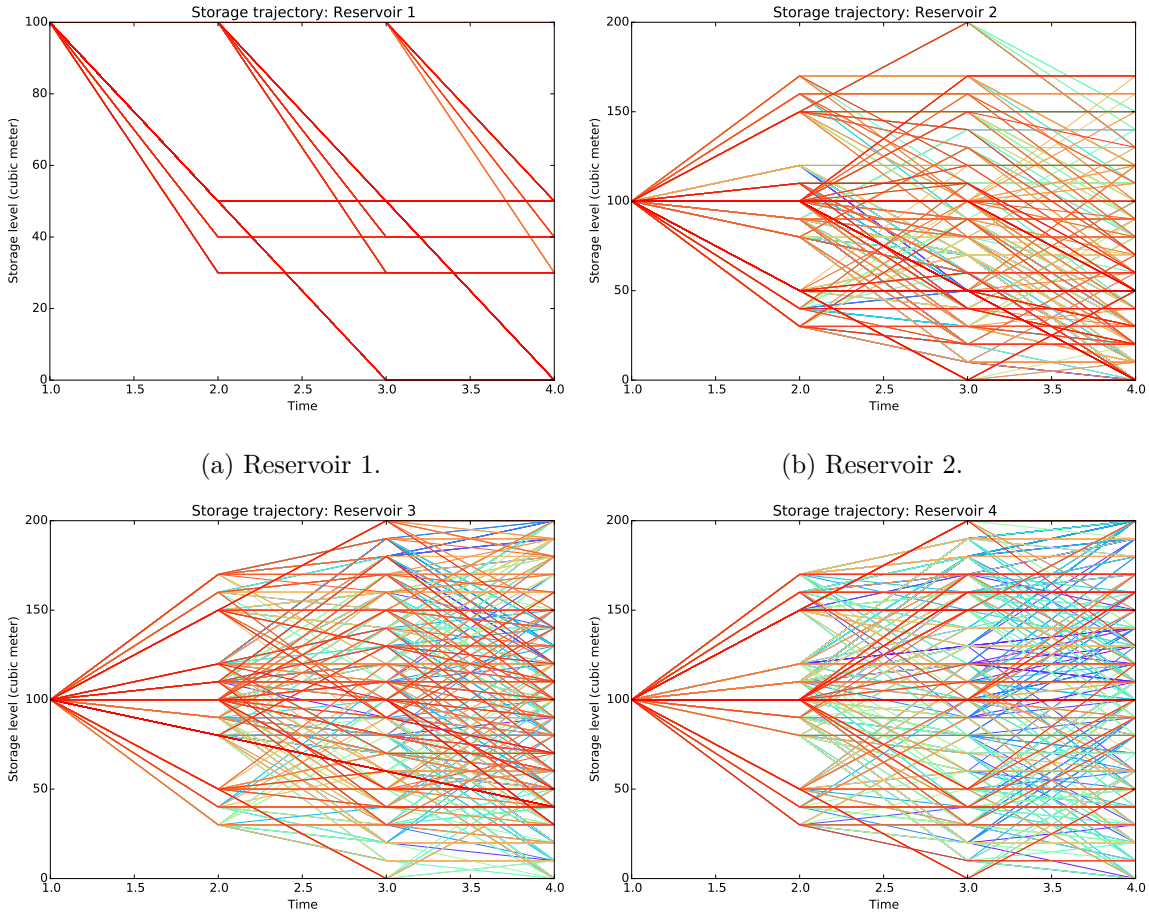


Figure 7.10: Execution time for the forward and backward pass by iteration for a 4 reservoir hydro-bidding problem in MIDAS.

However, is execution time in reality is inefficient for the size of the problem. This is due to MIDAS enumerating across many state points. Figures 8.11a to 8.11d show the state trajectories visited during the execution of the MIDAS algorithm. They visit significantly more state points for the downstream reservoirs 3 and 4 than the upstream reservoirs 1 and 2, and as a result MIDAS spends many iterations enumerating these points. Such a phenomenon has also been observed for the continuous state variable case in Chapter 7. This is detrimental to solving large models. The more state points MIDAS explores the larger the MIP sub-problem, and the longer it takes to converge.



(a) Reservoir 1.

(b) Reservoir 2.

(c) Reservoir 3.

(d) Reservoir 4.

Figure 7.11: Simulated state trajectories of each reservoir in the 4 reservoir hydro-scheme.

7.4 Summary

This chapter analyzes the computational efficiency of MIDAS. It discusses its shortcomings, which prevent it from solving large hydro-bidding models. These shortcomings are mainly due to the MIP formulation of the sub-problems, which adds $M(2+|R|)$ constraints and $M(1+|R|)$ binary variables. The MIP sub-problems increase in the number of variables and big-M constraints with each iteration of the backward pass, which in turn increases the execution time of the algorithm, as summarized in Table

8.1. This causes two issues for MIDAS: first the solution time for the sub-problem increases exponentially over the execution of the algorithm; second the MIP becomes to big for solvers such CPLEX [23] to handle.

In order to address these two issues we introduce two heuristics. The first is the step function selection scheme, much like the cut selection scheme in SDDP, which removes redundant step functions that break the monotonicity property of the value function (see Algorithm 11). Applying this step function selection scheme to MIDAS reduces its execution time by a minimum of 40%, as summarized in Table 8.2, with average time ranging from 65 to 312 seconds as opposed to 107 to 617 seconds originally. The advantage of using step function selection is that it minimizes the number of step functions needed to retain the current approximation of the value function. This in turn reduces the number of constraints and binary variables added to the sub-problems, thus reducing their solution times.

The second heuristic is a sub-problem solving heuristic based on the L-shaped method by [45]. We observed that the reason the sub-problems took a slow time to solve was due to the long convergence of the upper bound of the branch and bound tree to the optimal integer solution. In comparison to the lower bound, which reaches the optimal integer solution early, the upper bound took longer to converge, as it had to fathom all other potential nodes (Figure 8.5). By using the L-shaped method and the binary optimal cuts, we were able to remove all the binary variables needed to add the step functions, and iteratively converge to an optimum solution. This way we were able to solve the sub-problem by further decomposing it into smaller MIPs, and reducing the solutions times within the forward and backward passes (Algorithm 12). Applying the heuristic yielded a reduction in the execution time from 16% up to 60% with average times ranging from 90 to 291, where it originally took between 107 to 617 seconds. The improvements for a range of different sized problems are presented in Table 8.3.

Using both of these heuristics we were able to solve a hydro-bidding problem consisting of 4 reservoirs, 3 Markov states, and 4 stages with MIDAS. It took 850 iterations (see Figure 8.9) to converge with a total time of $367,000$ seconds. Even though the execution time of MIDAS is large, the execution times for the forward and backward passes do not exponentially increase with the number of step functions, as was observed originally. The execution time for each of the phases increase linearly by iteration, but plateaus towards an average of 500 seconds for the backward pass, and 100 seconds for the forward pass. This is due the step function selection scheme reducing the number of iterations that the sub-problem solver (Algorithm 12) takes to converge by removing redundant step functions. Furthermore, the sub-problem solver heuristic only solves MIPs with a small, constant number of discrete variables which reduces the total time to model solution time of solving the MIP.

Based on the improvements provided by these heuristics, MIDAS has great potential to be a fully scalable algorithm for solving large scale, non-convex, multistage, stochastic, optimization problems with both continuous and integer state variables. However, it does require further developments, mainly in developing tight formulations for the sub-problems. These, along with other aspects of MIDAS, and its underlying contribution to solving a variety of multistage, stochastic, optimization problems are discussed in the final chapter of this thesis, Chapter 9.

Chapter 8

Concluding remarks

A hydro producer, participating in an electricity market, has to make the trade-off between selling water in current periods and selling it in future periods when the wholesale price of electricity might be higher [75]. Their offers of energy to the market when dispatched must be feasible across the hydro-scheme, as well as take into account uncertainty in information such as inflows and the market-clearing price, which is realized over time [47]. The hydro-bidding problem seeks optimal offer policies in order to maximize the expected profit of a hydroelectric producer participating in these electricity markets.

The hydro-bidding problem suffers from:

1. uncertain prices, inflow and demand,
2. non-convex or non-concave value functions due to the presence of integer variables, price, or non-concave power generation functions,
3. being computationally expensive to solve for large hydro-schemes

In this thesis, we study these limitations of hydro-bidding problem, and propose a new stochastic optimization algorithm called the Mixed-Integer Dynamic Approximation

Scheme (MIDAS). MIDAS solves nonconvex, discrete-time, finite horizon, stochastic optimal control problems. It works in similar fashion to the Stochastic Dual Dynamic Programming (SDDP), but instead of using cutting planes, it uses step functions to create an outer approximation of the value function. Using these step functions as approximations it can solve stochastic optimal control problems with monotonic value functions.

We used MIDAS to study three variants of the hydro-bidding problem. In all of these variants the hydro-bidding model is non-convex, which limits SDDP from guaranteeing convergence to an optimum policy. The first variant had discrete production states. This makes the state variables integer, which in turn makes the hydro-bidding problem a multistage stochastic mixed-integer program (SMIP). The second variant incorporates continuous price distributions, allowing us to model more complex price processes such as autoregressive models. And the last variant is modelling headwater effects, which approximates the power generation function based on both the turbine flow and the water level of the upstream reservoir. In the last two variants the state variable is continuous, but the hydro-bidding models are non-convex. We compared the policies of MIDAS with its SDDP equivalent for all three variants. Our overall observation was that MIDAS indeed produced optimum policies that were better or at least equivalent to the policies produced by the SDDP equivalent.

8.1 Main results

The key findings from this thesis are described in the following passages.

As observed in Chapter 3, the SDDP algorithm produced offer policies which were near optimal (Table 3.4) compared to the exact optimum policy. SDDP solved these hydro-bidding models with varying reservoirs, time, and Markov states in order of

magnitudes less than the deterministic equivalent. Therefore, the Stochastic Dual Dynamic Programming (SDDP) algorithm is a good method to use for solving convex, linear, hydro-bidding problems.

Unless the forecasted load in the day-ahead market is significantly different from the actual load with high balancing prices, then there is little incentive for any hydro generators to incur the penalty and deviate away from their reference schedule. It was observed that the rolling horizon, two-stage, hydro-bidding model called HERBS produces offer policies which have very little deviation from the reference dispatch.

Through MIDAS, we approximated the monotonic increasing value functions using step functions and guarantee convergence within finite iterations. We were able to show that MIDAS can obtain policy that is $2T\varepsilon$-optimal in its first stage decisions, when solving a multistage stochastic programming problem by sampling the scenario.

We used MIDAS to solve various hydro-bidding models with discrete state variables. We observed that through the step functions MIDAS was able to approximate the unique structure of the value function. It produced near-optimal policies that were on average within 4% of the optimum policy (Table 6.7). SDDP on the other hand convexified the value function, and in some instances did not converge. MIDAS produced near-optimal policies, which are better than those produced by SDDP.

For hydro-bidding problems with continuous state variables, MIDAS can model complex price processes and guarantee convergence within finite iterations. Due to the step functions, MIDAS can represent single variate and multivariate, autoregressive price processes by incorporating price as part of the state variable (Section 7.1.1). It does not need to alter its approach to approximating the value function through its step functions. Through this approach MIDAS produces optimum offer policies that arbitrage between price differences in different trading periods.

SDDP can also model complex price processes using a cut interpolation technique.

It can represent the scenario tree by a coarser scenario tree of predefined price outcomes. Then, it can represent the value function by interpolating between two of the coarser price nodes based on the autoregressive price outcome illustrated by Figure 7.6. The coarser price nodes store the actual SDDP cuts generated in the backward pass. Through this cut interpolation method, SDDP can compute offer policies which are similar to that produced by MIDAS.

Since the sub-problems in MIDAS are mixed-integer programs, we can incorporate any formulation of the power generation function as long as the value function is still monotonic increasing. We were able to represent the power generation function using the approach of [60], and solved a 3 tranche offerstack hydro-bidding model using MIDAS. Therefore, MIDAS can incorporate headwater effects into the power generation function and guarantee convergence within finite iterations.

The sub-problems that MIDAS solves are mixed-integer programs, which naturally take longer to solve than the linear programs of SDDP. In MIDAS, additional big-M constraints and binary variables are added to the sub-problem whenever a new step function is introduced. This lengthens the time to solve the sub-problems as the algorithm progresses (Table 8.1). Furthermore, MIDAS does not exploit any gradient information of the value function, unlike SDDP. This makes it slower, because it needs to explore the state space thoroughly in order to produce an accurate approximation of the value function and converge to an optimal policy.

In order to speed up MIDAS and make it a scalable method, we introduced a step function selection scheme. The step function selection scheme functions like the cut selection in SDDP, where it removes redundant step functions that break the monotonicity property of the value function (see Algorithm 11). The advantage of using step function selection is that it reduces the size of the MIP sub-problem in the number of binary variables and constraints. We observed that applying the step function

selection scheme to MIDAS reduced its execution time by a minimum of 40%, as summarized in Table 8.2, with average time ranging from 65 to 312 seconds compared with the original time of 107 to 617 seconds.

We observed that the industrial solvers, such as CPLEX, took a long time to reduce the MIP gap when solving the sub-problems in MIDAS. This was due to the long time the solvers took fathoming nodes in the branch-and-bound tree. In order to reduce the slow solver time of the solvers, we introduced a heuristic to iteratively solve the MIP sub-problems. Since binary variables are used to model the discrete generation states of the hydro-bidding model in Chapter 6, we can use the binary optimality cuts of [45] to further decompose MIP sub-problems into two-stage stochastic programs. We can then use the L-shaped method (Algorithm 12) to iteratively solve the MIP sub-problem. Because we are using the binary optimality cuts, we no longer need the big-M constraints and additional binary variables to model the step functions, which keeps the size of the branch and bound tree in the MIP small and constant. Using this heuristic, we were able to reduce the execution time of MIDAS by up to 60%, with average time ranging from 90 to 291 seconds compared with 107 to 617 seconds.

Using the step function selection scheme and the sub-problem solver heuristic, MIDAS was able to solve a 4 reservoir hydro-bidding problem. The model took 850 iterations (see Figure 8.9) to converge with a total time of $367,000$ seconds. With the range of experiments we observed a linear behavior of the execution times for the forward and backward passes up to the 600th iteration. Beyond this iteration, we observed that the execution time remained relatively constant with an average of 500 seconds for the backward pass, and 100 seconds for the forward pass.

8.2 Contributions

This thesis has contributed to the understanding of non-convex, short-term, hydro-bidding problems under uncertainty, and solving discrete time, stochastic optimal control problems with non-concave or non-convex value functions. The specific contributions of this thesis are described in the following passages.

We have developed a hydro-bidding model for hydroproducers participating in an intra-day balancing market. The model called HERBS (Hydro-Electric Reservoir Bidding System) is a two-stage stochastic mixed-integer program, which constructs balancing bids that maximize the expected profit. HERBS is executed in a rolling horizon fashion, where at each period it computes a balancing bid.

HERBS is a non-convex SMIP which cannot be solved easily using existing methods such stochastic dynamic programming (SDP), or stochastic dual dynamic programming (SDDP). In pursuit of solving models like HERBS, we have developed an algorithm called the Mixed-Integer Dynamic Approximation Scheme (MIDAS). MIDAS is a sampling algorithm similar to SDDP that solves multi-stage stochastic programming models with monotonic, non-concave value functions. It works similarly to SDDP. We prove that MIDAS converges within finite iterations to produce policies that produce $2T\varepsilon$-optimal first stage decisions through Theorem 5.

Using MIDAS, we solve a variety of non-convex hydro-bidding problems and show that MIDAS converges in finitely many iterations. The first hydro-bidding problem we solve is based on HERBS, where the power generation function is represented by a discrete set of power productions and equivalent turbine-water discharges. This makes the state variables integer and the overall problem non-convex. We demonstrated that MIDAS produced policies that were better than policies computed using SDDP.

The second hydro-bidding problem we solve involves modelling complex price pro-

cesses, such as an autoregressive, mean-reverting price process. The step functions in MIDAS can incorporate these price processes by adding a price dimension into the state variable. In order to incorporate the price process inside SDDP, we introduce a cut interpolation technique, which interpolates the value function based on two sets of cuts. We demonstrated that both of the policies, one computed by MIDAS and the other by SDDP, indicated strategic offers that arbitraged between price differences in different trading periods.

The last hydro-bidding problem we solved using MIDAS incorporated headwater effects into the power generation function. By using the formulation of [60], we were able to model both the unit states and the water levels of the reservoirs. Solving this hydro-bidding model yielded similar policies from both MIDAS and SDDP.

Due to the MIP sub-problems in MIDAS, it is computationally inefficient, which limited MIDAS to solving small hydro schemes of up to 3 reservoirs. In order to address this limitation, we have developed two heuristics, the first tightens the value function approximation by removing redundant step functions; and the second uses a sub-problem solver heuristic based on the optimal cuts and the L-shaped method of [45]. Using these heuristics, we demonstrated improvements in the execution time of MIDAS. When we applied the step function selection scheme, we observed a reduction of 40% in the execution time with average time ranging from 65 to 312 seconds as opposed to 107 to 617 seconds originally. We also demonstrated a reduction in the execution time up to 60% with average times ranging from 90 to 291, when we applied the sub-problem solving heuristic. Using both of these heuristics, we were successfully able to scale MIDAS to solve a 4 reservoir hydro-scheme with integer state variables.

Based on these contributions we have demonstrated that MIDAS can solve non-convex stochastic optimal control problems. Moreover, it can solve multistage, stochastic mixed-integer programs in similar fashion to SDDP. This shows that monotonicity of

the value function is a key property required to solve multistage stochastic programs using algorithms like SDDP. This opens up possibilities for studying other models using MIDAS that approximate the value function.

8.3 Future research directions

Although MIDAS is in its early days, it already shows some promise in terms of solving non-convex stochastic optimal control problems. Here we suggest a few ideas that might take MIDAS from a proof of concept to an industrially scalable algorithm capable of solving hydro-bidding problems with many reservoirs, many stations, and complex topologies.

Since MIDAS has the same algorithmic framework as SDDP, it can incorporate parallelization. We could implement the parallelization techniques discussed in [26] inside MIDAS and analyze its performance. In addition to parallelization, we could extend the models already developed with real hydro schemes. The hydro schemes used in this thesis are contrived, where each reservoir and station is the same. This makes the hydro-bidding problems more symmetric than usual, where there are multiple state trajectories which produce the same offer strategies. We could analyze MIDAS performance with more realistic data, and see if it generates state trajectories similar to those observed in the SDDP equivalent.

One limitation of the sub-problem solving heuristic is that the binary optimal cuts are point estimates, where the cut is only valid at a particular state point. This formulation is not a tight formulation. We could use the improved L-shaped method proposed in [7], which introduces a new class of binary optimal cuts. We may also be able to strengthen the sub-problem by re-formulating it as a forbidden vertices problem as proposed by [8]. The forbidden vertices problem, in essence, produces a

set of optimal points on an existing polytope, that does not belong to a predefined set of vertices of the same polytope. We could formulate the master problem $\overline{MP}(t, i, x_t)$ in the heuristic as a forbidden vertices problem, where the polytope is the overall state space, and the set of forbidden vertices are the already visited state points. This may speed up the heuristic by producing a set of binary optimal cuts per iteration instead of just one.

The value function, when the state variable is continuous, is piecewise convex. Using this property, we can represent the state space by a finite number of polytopes in which the value function is convex. Then, instead of using step functions, we can use the SDDP hyper-planes within these polytopes to approximate the value function. The formulations of the piecewise linear functions proposed by [80, 81] may be potential avenues for formulating value function approximation. As long these alternative approximations still produce an ε-outer approximation of the value function, MIDAS will still converge within finitely many iterations based on Theorem 5. We can further complement these formulations and solve the sub-problems through disjunctive programming [11], where we can represent the set of hyperplanes within each polytope as disjunctive constraints and use branch-and-bound methods to branch on specific polytopes in order to solve the sub-problem [70].

In summary, we have produced an innovative algorithm called MIDAS, which can solve stochastic optimal control problems with non-concave value functions. We have proven its almost sure convergence for continuous and integer state variables, and have applied it to various hydro-bidding models to verify that it works and is comparable with SDDP methods. Our hope is that this approach might lead to other suitable methods that can be applied to a variety of stochastic optimization problems at an industrial scale.

Appendix A

Model parameters Chapter 7, Section 7.2

A.1 Modelling price uncertainty

Table A.1: Model parameters for each of the three hydro-scheme configurations.

Parameters	**Value**
T	12
α_t for $t = 1, 2, \ldots, T$	0.5
η_t for $t = 1, 2, \ldots, T$	$\sim$ Norm$(0, 1)$
b_t	[61.261, 56.716, 59.159, 66.080, 72.131, 76.708, 76.665, 76.071, 76.832, 69.970, 69.132, 67.176]
Ω	10
δ_p	5
X_δ	$\{x \in \mathbb{R} \ : \ x \in [\delta_x, 200]\}$

l_t for $t = 1, 2, \ldots, T$	0
ω_t for $t = 1, 2, \ldots, T$	0
$U(x_t)$	$\{u \in \mathbb{R} \ : \ u \in [0, \min\{x_t, 70\}]\}$
$g(v)$	$\begin{cases} 1.1v & \text{if } v \in [0, 50], \\ v + 5 & \text{if } v \in (50, 60], \\ 0.5(v - 60) + 65 & \text{if } v \in (60, 70], \\ 0 & \text{otherwise.} \end{cases}$
$\overline{x}_r$ for $r \in \mathcal{R}$	100
$\overline{p}$	61.261
N	30

A.2 Modelling headwater effects

Parameters	Value
$\hat{g}$	$\begin{bmatrix} 0, & 22.58, & 45.16, & 67.74, & 90.32 \end{bmatrix}$
$\hat{u}$	$\begin{bmatrix} 0, & 29.456, & 53.988, & 74.153, & 89.95 \end{bmatrix}$
$\hat{a}$	$\begin{bmatrix} 69.174, & 198.233, & 319.176, & 444.203, & 556.511 \end{bmatrix}$
$\hat{b}$	$\begin{bmatrix} 67.174, & 196.233, & 317.176, & 442.203, & 554.510 \end{bmatrix}$
$\hat{a}^{\text{squared}}$	$\begin{bmatrix} 4785.050, & 39296.614, & 101873.815, & 197316.623, & 309704.411 \end{bmatrix}$
$\hat{b}^{\text{squared}}$	$\begin{bmatrix} 4512.354, & 38507.679, & 100601.108, & 195543.809, & 307482.367 \end{bmatrix}$

Table A.3: Piecewise linear function parameters of the nominal power, and the quadratic functions.

A.2.1 Modeling unit committment

In order to model unit switching, (A.2.1) can be added as a constraint inside the hydro-bidding models. This constraint represents the change in the number of turbines being either switched on or off between period $t-1$ and t.

$$\sum_{i \in \mathcal{T}_s} i z_{t,s,i} - \sum_{i \in \mathcal{T}_s} i z_{t-1,s,i} = w_{t,s}^{\text{startup}} - w_{t,s}^{\text{shutdown}} \tag{A.2.1}$$

As already defined, $z_{t,s,i}$, which is now also indexed by period t, represents the state of the turbine and indicates when it is on ($z_{s,i} = 1$) or off ($z_{s,i} = 0$). The variables $w_{t,s}^{\text{startup}}$ and $w_{t,s}^{\text{shutdown}}$, which can be either continuous or integer, represents the number of units that have been started up or shut down in period t. A unit is considered to have been started up (i.e. $w_{t,s}^{\text{startup}} \geq 1$) when it transitions from being off (i.e. $z_{t-1,s,i} = 0$) in the previous period to being on ($z_{t,s,i} = 1$) in the current period. Similarly, a unit is considered to have been shut down (i.e. $w_{t,s}^{\text{shutdown}} \geq 1$) when it transitions from being on (i.e. $z_{t-1,s,i} = 1$) to being off ($z_{t,s,i} = 0$).

Constraint (A.2.1) is based on the formulation presented in [50], and ensures that if the unit status variable $z_{t-1,s,i}$ is binary then the unit switching variables $w_{t,s}^{\text{startup}}$ and $w_{t,s}^{\text{shutdown}}$ will naturally be integer. As reported in [50], such a formulation does reduce the computation time, because when more variables are relaxed from discrete to continuous the fewer variables are branched on resulting in a smaller branch and bound tree. However [54] has reported that this is not always the case. A reason may be because it is easier to generate strong cutting planes for the convex hull of integer points when more binary variables are present. If the solver does not know that certain quantities are inherently binary, it is not able to look for opportunities to

exploit this fact [9]. For instance $z_{t-1,s,i}$ variables can be prioritized to branch first and then variables $w_{t,s}^{\text{startup}}$ and $w_{t,s}^{\text{shutdown}}$, assuming they are integer. This would maintain the advantages of the presence of the $w_{t,s}^{\text{startup}}(\xi)$ and $w_{t,s}^{\text{shutdown}}$ variables for generating the cutting planes.

The inclusion of unit commitment, through the use of Constraint (A.2.1), makes variable $z_{t,s,i}$ a state variable along with the reservoir storage variable x_t. Because this variable is binary it makes the sub-problems in MIDAS and SDDP stochastic, mixed-integer, linear programs. As discussed throughout this thesis stochastic, mixed-integer, linear programs are difficult to solve due their non-convex and discontinuous structure. Including $z_{t,s,i}$ in the state variable can be avoided by adding a fixed running cost to each turbine. This changes the objective function in Model (6.1.1) of Chapter 6, to the new of the objective function defined by Equation (A.2.2). The new function adds the term $C \sum_{s \in S} \sum_{i \in \mathcal{T}_s} z_{t,s,i}$, which prices all the turbine state variables with a fixed running cost C. This new objective does not require the addition of Constraint (A.2.1), and therefore we no longer require the $z_{t,s,i}$ to be a state variable. However, it does not minimize unit switching explicitly, but rather incentivizes efficient generation by limiting the number of turbines that are used in each period.

$$V_{t,i}(x) = \max \sum_{j=1}^{M} \rho_{i,j}(t) \left[\pi_{t,j} o_{t,j} - C \sum_{s \in S} \sum_{i \in \mathcal{T}_s} z_{t,s,i} + \hat{V}_{t+1,j} \right] \tag{A.2.2}$$

However, if one were to consider unit switching, then implementing the variable $z_{t-1,s,i}$ as a state variable would be trivial in MIDAS. A similar approach to Section 7.1 can be taken, where an additional dimension is added as an input to the value function, which represents the number of turbines that are in use. This additional state variable can be an integer, and enable the implementation of the Constraint (A.2.1). The step functions in MIDAS would simply have an additional dimension to represent this variable.

References

[1] E.K. Aasgård, G.S Andersen, S.E. Fleten, and D. Haugstvedt. Evaluating a stochastic-programming-based bidding model for a multireservoir system. *Power Systems, IEEE Transactions on*, 29(4):1748–1757, 2014.

[2] H. Abgottspon, K. Njalsson, M.A. Bucher, and G. Andersson. Risk-averse medium-term hydro optimization considering provision of spinning reserves. In *2014 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, pages 1–6. IEEE, 2014.

[3] International Energy Agency. Energy policies of IEA countries: France 2009 review. Technical report, IEA, 2009.

[4] International Energy Agency. Technology roadmap hydropower. Technical report, International Energy Agency, 2012.

[5] International Energy Agency. Energy policies of IEA countries: European Union 2014 review, 2015.

[6] W.T. Alley. Hydroelectric plant capability curves. *IEEE Transactions on Power Apparatus and Systems*, 96(3):999–1003, 1977.

[7] G. Angulo, S. Ahmed, and S.S. Dey. Improving the integer l-shaped method. *INFORMS Journal on Computing*, 28(3):483–499, 2016.

[8] G. Angulo, S. Ahmed, S.S. Dey, and V. Kaibel. Forbidden vertices. *Mathematics of Operations Research*, 40(2):350–360, 2014.

[9] M.F. Anjos. Recent progress in modeling unit commitment problems. In *Modeling and Optimization: Theory and Applications*, pages 1–29. Springer, 2013.

[10] L. Bacaud, C. Lemaréchal, A. Renaud, and C. Sagastizábal. Bundle methods in stochastic optimal power management: A disaggregated approach using preconditioners. *Computational Optimization and Applications*, 20(3):227–244, 2001.

[11] E. Balas. Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51, 1979.

[12] E.M Beale and J.J. Forrest. Global optimization using special ordered sets. *Mathematical Programming*, 10(1):52–69, 1976.

[13] A. Belloni, A.D. Lima, P. Maceira, and C. Sagastizábal. Bundle relaxation and primal recovery in unit commitment problems. the brazilian case. *Annals of Operations Research*, 120(1-4):21–44, 2003.

[14] D.P. Bertsekas and S.E. Shreve. *Stochastic optimal control: The discrete time case*, volume 23. Academic Press New York, 1978.

[15] A. Borghetti, C. D'Ambrosio, A. Lodi, and S. Martello. An milp approach for short-term hydro scheduling and unit commitment with head-dependent reservoir. *IEEE Transactions on Power Systems*, 23(3):1115–1124, 2008.

[16] S. Brignol and A. Renaud. A new model for stochastic optimization of weekly generation schedules. In *Advances in Power System Control, Operation and Management, 1997. APSCOM-97. Fourth International Conference on (Conf. Publ. No. 450)*, volume 2, pages 656–661. IET, 1997.

[17] J.P.S. Catalão, S.J.P.S. Mariano, V.M.F. Mendes, and L.A.F.M. Ferreira. Scheduling of head-sensitive cascaded hydro systems: a nonlinear approach. *IEEE Transactions on Power Systems*, 24(1):337–346, 2009.

[18] S. Cerisola, J.M. Latorre, and A. Ramos. Stochastic dual dynamic programming applied to nonconvex hydrothermal models. *European Journal of Operational Research*, 218(3):687–697, 2012.

[19] G.W. Chang, M. Aganagic, J.G. Waight, J. Medina, T. Burton, S. Reeves, and M. Christoforidis. Experiences with mixed integer linear programming based approaches on short-term hydro scheduling. *IEEE Transactions on power systems*, 16(4):743–749, 2001.

[20] H.C. Chang and P.H. Chen. Hydrothermal generation scheduling package: a genetic based approach. *IEE Proceedings-Generation, Transmission and Distribution*, 145(4):451–457, 1998.

[21] Z. Chen and W.B. Powell. Convergent cutting-plane and partial-sampling algorithm for multistage stochastic linear programs with recourse. *Journal of Optimization Theory and Applications*, 102(3):497–524, 1999.

[22] A.I. Cohen and V.R. Sherkat. Optimization-based methods for operations scheduling. *Proceedings of the IEEE*, 75(12):1574–1591, 1987.

[23] CPLEX, IBM ILOG. 12.6 user manual. *ILOG. See ftp://ftp.software.ibm.com/software/websphere/ilog/docs/optimization/cplex/ps_usrmancplex.p* 2010.

[24] A.K. David and F. Wen. Strategic bidding in competitive electricity markets: a literature survey. In *Power Engineering Society Summer Meeting, 2000. IEEE*, volume 4, pages 2168–2173. IEEE, 2000.

[25] D. De Ladurantaye, M. Gendreau, and J. Potvin. Strategic bidding for price-taker hydroelectricity producers. *Power Systems, IEEE Transactions on*, 22(4):2187–2203, 2007.

[26] V.L. De Matos, A.B. Philpott, and E.C. Finardi. Improving the performance of stochastic dual dynamic programming. *Journal of Computational and Applied Mathematics*, 290:196–208, 2015.

[27] A.L. Diniz and M.E.P. Maceira. A four-dimensional model of hydro generation for the short-term hydrothermal dispatch problem considering head and spillage effects. *IEEE transactions on power systems*, 23(3):1298–1308, 2008.

[28] C.J. Donohue and J.R. Birge. The abridged nested decomposition method for multistage stochastic linear programs with relatively complete recourse. *Algorithmic Operations Research*, 1(1), 2006.

[29] L. Dubost, R. Gonzalez, and C. Lemaréchal. *A Primal-Proximal Heuristic Applied to the Unit-Commitment Problem*. PhD thesis, INRIA, 2003.

[30] IEA Renewable Energy Essentials. Hydropower, 2010.

[31] E.C. Finardi and E.L. Da Silva. Unit commitment of single hydroelectric plant. *Electric Power Systems Research*, 75(2):116–123, 2005.

[32] S.E. Fleten, D. Haugstvedt, M.M. Belsnes, J.A. Steinsbø, and F. Fleischmann. Bidding hydropower generation: Integrating short-and long-term scheduling. In *17th Power Systems Computations Conference PSCC*, pages 352–358, 2011.

[33] S.E. Fleten and T.K. Kristoffersen. Stochastic programming for optimizing bidding strategies of a nordic hydropower producer. *European Journal of Operational Research*, 181(2):916–928, 2007.

[34] J. Garcia-Gonzalez and G.A. Castro. Short-term hydro scheduling with cascaded and head-dependent reservoirs based on mixed-integer linear programming. In *Power Tech Proceedings, 2001 IEEE Porto*, volume 3. IEEE, 2001.

[35] L. Gaudard and F. Romerio. The future of hydropower in europe: Interconnecting climate, markets and policies. *Environmental Science & Policy*, 37:172–181, 2014.

[36] P. Girardeau, V. Leclere, and A.B. Philpott. On the convergence of decomposition methods for multistage stochastic convex programs. *Mathematics of Operations Research*, 40(1):130–145, 2014.

[37] A. Gjelsvik, M.M. Belsnes, and A. Haugstad. An algorithm for stochastic medium-term hydrothermal scheduling under spot price uncertainty. In *Proceedings of 13th Power Systems Computation Conference*, 1999.

[38] A. Gjelsvik, B. Mo, and A. Haugstad. Long-and medium-term operations planning and stochastic modelling in hydro-dominated power systems based on stochastic dual dynamic programming. In *Handbook of Power Systems I*, pages 33–55. Springer, 2010.

[39] M. Hindsberger and A.B. Philpott. Resa: A method for solving multistage stochastic linear programs. *Journal of Applied Operational Research*, 6(1):2–15, 2014.

[40] N.A. Iliadis, A. Tilmant, R. Chabar, S.O Granville, and M. Pereira. Optimal operation of hydro-dominated multireservoir systems in deregulated electricity markets. In *Proceedings of the International Conference for Reservoir Operation and River Management*. Guanghzou, China, 2005.

[41] K. Imran and I. Kockar. A technical comparison of wholesale electricity markets in north america and europe. *Electric Power Systems Research*, 108:59–67, 2014.

[42] D.R. Jiang and W.B. Powell. An approximate dynamic programming algorithm for monotone value functions. *arXiv preprint arXiv:1401.1590*, 2014.

[43] D.R. Jiang and W.B. Powell. Optimal hour ahead bidding in the real time electricity market with battery storage using approximate dynamic programming. *arXiv preprint arXiv:1402.3575*, 2014.

[44] T.K. Kristoffersen and S.E. Fleten. Stochastic programming models for short-term power generation scheduling and bidding. In *Energy, Natural Resources and Environmental Economics*, pages 187–200. Springer, 2010.

[45] G. Laporte and F.V. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations research letters*, 13(3):133–142, 1993.

[46] P. Lino, L.A. Barroso, M.V.F Pereira, R. Kelman, and M.H.C Fampa. Bid-based dispatch of hydrothermal systems in competitive markets. *Annals of Operations Research*, 120(1-4):81–97, 2003.

[47] N. Löhndorf, D. Wozabal, and S. Minner. Optimizing trading decisions for hydro storage systems using approximate dual dynamic programming. *Operations Research*, 61(4):810–823, 2013.

[48] L. Meeus, R. Belmans, and J. Glachant. Regional electricity market integration france-belgium-netherlands. *Revue E-Societe Royale Belge des Electriciens*, 122(3):17, 2006.

[49] Innovation Ministry of Business and Employment. Energy in New Zealand 2015. Technical report, Ministry of Business, Innovation and Employment, 2015.

[50] G. Morales-España, J.M. Latorre, and A. Ramos. Tight and compact milp formulation of start-up and shut-down ramping in unit commitment. *IEEE Transactions on Power Systems*, 28(2):1288–1296, 2013.

[51] P.J. Neame, A.B. Philpott, and G. Pritchard. Offer stack optimization in electricity pool markets. *Operations Research*, 51(3):397–408, 2003.

[52] N. Newham. *Power system investment planning using stochastic dual dynamic programming.* phdthesis, 2008.

[53] N. Newham and A. Wood. Transmission investment planning using sddp. In *Power Engineering Conference, 2007. AUPEC 2007. Australasian Universities*, pages 1–5. IEEE, 2007.

[54] J. Ostrowski, M.F. Anjos, and A. Vannelli. Tight mixed integer linear programming formulations for the unit commitment problem. *IEEE Transactions on Power Systems*, 1(27):39–46, 2012.

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

[56] A.B. Philpott, M. Craddock, and H. Waterer. Hydro-electric unit commitment subject to uncertain demand. *European Journal of Operational Research*, 125(2):410–424, 2000.

[57] A.B. Philpott, A. Dallagi, and E. Gallet. On cutting plane algorithms and dynamic programming for hydroelectricity generation. In *Handbook of Risk Management in Energy Production and Trading*, pages 105–127. Springer, 2013.

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

[59] A.B. Philpott, Z. Guan, J. Khazaei, and G. Zakeri. Production inefficiency of electricity markets with hydro generation. *Utilities Policy*, 18(4):174–185, 2010.

[60] N. Porter, A.B. Philpott, A. Downward, and G. Zakeri. River Valley Optimization for Hydro-Electricity Generation. Research report, Department of Engineering Science, University of Auckland, 2012.

[61] W.B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[62] W.B. Powell. Clearing the jungle of stochastic optimization. *Informs TutORials*, 2014.

[63] Mighty River Power. Hydro stations, 2009.

[64] G. Pritchard. Stochastic inflow modeling for hydropower scheduling problems. *European Journal of Operational Research*, 246(2):496–504, 2015.

[65] G. Pritchard and G. Zakeri. Market offering strategies for hydroelectric generators. *Operations Research*, 51(4):602–612, 2003.

[66] RTE. Generation adequecy report 2014. Technical report, RTE, 2014.

[67] S. Ruzic and R. Rajakovic. Optimal distance method for Lagrangian multipliers updating in short-term hydro-thermal coordination. *IEEE transactions on power systems*, 13(4):1439–1444, 1998.

[68] N.V. Sahinidis. Optimization under uncertainty: state-of-the-art and opportunities. *Computers & Chemical Engineering*, 28(6):971–983, 2004.

[69] M.S. Salam, K.M. Nor, and A.R. Hamdam. Hydrothermal scheduling based lagrangian relaxation approach to hydrothermal coordination. *IEEE Transactions on Power Systems*, 13(1):226–235, 1998.

[70] S. Sen. Algorithms for stochastic mixed-integer programming models. *Handbooks in operations research and management science*, 12:515–558, 2005.

[71] A. Shapiro. On complexity of multistage stochastic programs. *Operations Research Letters*, 34(1), 2006.

[72] A. Shapiro. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209(1):63–72, 2011.

[73] A. Shapiro and D. Dentcheva. *Lectures on stochastic programming: modeling and theory*, volume 16. SIAM, 2014.

[74] Z.K. Shawwash, T.K. Siu, and S.D. Russell. The BC Hydro short term hydro scheduling optimization model. *IEEE Transactions on Power Systems*, 15(3):1125–1131, 2000.

[75] G. Steeger, L.A. Barroso, and S. Rebennack. Optimal bidding strategies for hydroelectric producers: A literature survey. *Power Systems, IEEE Transactions on*, 29(4):1758–1766, 2014.

[76] G. Steeger and S. Rebennack. Strategic bidding for multiple price-maker hydroelectric producers. *IIE Transactions*, 47(9):1013–1031, 2015.

[77] G. Steeger and S. Rebennack. Dynamic convexification within nested Benders decomposition using Lagrangian relaxation: An application to the strategic bidding problem. *European Journal of Operational Research*, 257(2):669–686, 2017.

[78] F. Thome, M.V.F. Pereira, S. Granville, and M. Fampa. Non-convexities representation on hydrothermal operation planning using SDDP. Technical report, working paper, available: http://www. psr-inc. com. br/portal/psr/publicacoes, 2013.

[79] K. Verhaegen, L. Meeus, and R. Belmans. Development of balancing in the internal electricity market in europe. In *Proceedings 2006 European Wind Energy Conference, Athens, Greece*, 2006.

[80] J.P. Vielma, S. Ahmed, and G. Nemhauser. Mixed-integer models for nonseparable piecewise-linear optimization: unifying framework and extensions. *Operations research*, 58(2):303–315, 2010.

[81] J.P. Vielma and G.L. Nemhauser. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming*, 128(1-2):49–72, 2011.

[82] S.W. Wallace and S.E. Fleten. Stochastic programming models in energy. *Handbooks in operations research and management science*, 10:637–677, 2003.

[83] J. Zou, S. Ahmed, and X.A Sun. Nested decomposition of multistage stochastic integer programs with binary state variables. *Available on Optimization Online*, 2016.

université PARIS-SACLAY | École doctorale de mathématiques Hadamard (EDMH)

Titre : Optimisation de la rivière : enchères à court terme de l'hydroélectricité sous incertitude

Mots clés : Optimisation stochastique, Hydro-ordonnancement, Optimisation non convexe

Résumé :

Le problème de l'hydro-offre consiste à calculer des d'offre optimales afin de maximiser le bénéfice attendu d'un producteur hydroélectrique participant à un marché de l'électricité. Ces problèmes peuvent être difficiles à résoudre lorsque la fonction de valeur n'est pas concave. Dans cette thèse, nous étudions quelques-unes des limites du problème hydro-bidding et proposons une nouvelle méthode d'optimisation stochastique appelée le Mixed-Integer Dynamic Approximation Scheme (MIDAS). MIDAS résout des programmes stochastiques non convexe avec des fonctions de valeurs monotones. Il fonctionne de manière similaire à la Stochastic Dual Dynamic Programming (SDDP), mais au lieu utiliser des hyperplans, il utilise des fonctions d'étape pour créer une approximation externe de la fonction de valeur. MIDAS converge « presque sûrement » à $2T\varepsilon$ solution optimale des première étage décisions.

Nous utilisons MIDAS pour résoudre trois types de problèmes hydro-bidding non convexes. Le premier modèle d'hydro-bidding que nous résolvons a des variables d'état entier car les productions sont discrètes. Dans ce modèle, nous démontrons que MIDAS est meilleur que SDDP. Le modèle suivant d'hydro-bidding utilise des processus de prix autorégressifs au lieu d'une chaîne de Markov. Le dernier modèle d'hydro-bidding intègre les effets de hauteur d'eau, où la fonction de production d'énergie dépend du niveau de stockage du réservoir et du débit d'eau de la turbine. Dans tous ces modèles, nous démontrons que la convergence de MIDAS en un nombre fini d'itération.

Title: River Optimization: short-term hydro-bidding under uncertainty

Keywords: Stochastic optimization, Hydro-scheduling, Nonconvex optimization

Abstract:

The hydro-bidding problem is about computing optimal offer policies in order to maximize the expected profit of a hydroelectric producer participating in an electricity market. These problems can be difficult to solve when the value function is not concave. In this thesis, we study some of the limitations of the hydro-bidding problem, and propose a new stochastic optimization method called the Mixed-Integer Dynamic Approximation Scheme (MIDAS). MIDAS solves nonconvex, stochastic programs with monotonic value functions. It works in similar fashion to the Stochastic Dual Dynamic Programming (SDDP), but instead of using cutting planes, it uses step functions to create an outer approximation of the value function. We show that MIDAS will converge almost surely to $2T\varepsilon$ optimal first stage decisions.

We use MIDAS to solve three types of nonconvex hydro-bidding problems. The first hydro-bidding model we solve has integer state variables due to discrete production states. In this model we demonstrate that MIDAS constructs offer policies which are better than SDDP. The next hydro-bidding model has a mean reverting autoregressive price processs instead of a Markov chain. The last hydro-bidding incorporates headwater effects, where the power generation function is dependent on both the reservoir storage level and the turbine waterflow. In all of these models, we demonstrate convergence of MIDAS in finite iterations.