Université catholique de Louvain
École Polytechnique de Louvain

Center for Operations Research and Econometrics

Optimization Under Uncertainty in Power Systems Supported by Parallel Computing

Daniel Felipe Ávila Girardot

Thesis submitted in partial fulfillment of the requirements for the degree of
Docteur en sciences de l'ingénieur et technologie

Supervisor:
Anthony Papavasiliou
(NTUA, Greece)
Philippe Chevalier
(UCLouvain, Belgium)

Jury:
Mathieu Van Vyve (UCLouvain, Belgium)
Daniele Catanzaro (UCLouvain, Belgium)
Mario Pereira (PSR, Brazil)
Lazaros Exizidis (ENTSO-E, Belgium)
Mauricio Junca (Universidad de los Andes, Colombia)
Nils Löhndorf (University of Luxembourg, Luxembourg)

Chair:
Mathieu Van Vyve (UCLouvain, Belgium)

PhD Organization

Daniel Ávila
UCLouvain
École Polytechnique de Louvain
Center for Operations Research and Econometrics

Thesis Supervisor

Anthony Papavasiliou
Assistant Professor, National Technical University of Athens
Department of Electrical and Computer Engineering

Philippe Chevalier
Professor, UCLouvain
École Polytechnique de Louvain
Center for Operations Research and Econometrics

Supervisory Committee

Mauricio Junca
Associate Professor, Universidad de los Andes
Mathematics Department

Nils Löhndorf
Associate Professor, University of Luxembourg
Luxembourg Centre for Logistics and Supply Chain Management

En memoria de mi abuelo, Luis Girardot.

Abstract

Power systems present a variety of characteristics which makes their operation challenging. One complication of power system operation is the necessity of the overall structure to react to uncertain events, which are becoming increasingly important as the energy transition results in the expansion of renewable technologies, and decision makers are faced with the task of reacting to unforeseen events. From a methodological point of view, this novel energy landscape implies that the scale of the scheduling and planning problems at hand is becoming increasingly large, which poses challenges to state-of-the-art optimization techniques. This dissertation aims at proposing novel algorithmic schemes, supported by high performance computing, that help with addressing the increasingly relevant paradigm of optimization under uncertainty in power systems.

The first part of this thesis considers a class of problems referred to as multistage stochastic optimization problems. We specifically focus on the long-term hydrothermal scheduling problem. This class of problems is known to be difficult to tackle. We build upon the SDDP algorithm, and extend the algorithm through high performance computing. We specifically exploit high performance computing in order to study a variety of strategies for speeding up the overall algorithmic performance. We benchmark our proposed algorithmic scheme against PSR SDDP, an industrial scale implementation, and report favorable performance comparisons. Furthermore, we discuss the connections between our techniques and the reinforcement learning framework.

The second part of this dissertation considers the ongoing European Resource Adequacy Assessment. This study aims at measuring the capacity of the power system network to react to future uncertain conditions. Institutionally, the adequacy concerns identified through such a study support Member States in determining the need for national capacity mechanisms. A critical methodological step of the overall study is to determine the generation capacity expansion opportunities as well as decommissioning decisions of existing capacity that will occur in the upcoming years, thus naturally leading us to a framework of optimization under uncertainty. As a first step to address the problem, this thesis considers the single-year setting used for the 2021 version of the study. We leverage the two-stage stochastic programming framework, and propose parallel computing algorithmic schemes for tackling the problem.

Acknowledgements

It has been a long journey; a journey which began long before starting my Ph.D., but one which feels fulfilled than ever. As I remember have written in my statement of purpose, more than 5 years ago, "curiosity is the trail to knowledge", a path that has lead me to this, almost surreal, point in life where I'm overjoyed (and struggling to find the words) to write this section; one where my thoughts are on the people that helped me, and guided me, throughout this last five years.

I first met Prof. Anthony Papavasiliou back in 2017, while at INFORMS conference in Houston. We had a brief chat where he introduced me to quite an interesting topic: stochastic programming within power systems. Not only did he pointed me the interesting methodological aspects behind it, but also how these kind of techniques could lead to a positive impact in our society. Fast forward a couple of months and I was in Louvain-la-Neuve, Belgium, starting my Ph.D. being advised by Prof. Anthony. Some introspection has lead me to realize how lucky I'm to have find such a great researcher, professor and, most importantly, human being as my advisor. Despite a first challenging year, where I was faced by the harsh criticisms, Prof. Anthony taught me an important lesson: it is through the harshest truth that you embrace the obstacles, and become a better version of yourself. It is at this point where I would like to express my deepest gratitude, for guiding me throughout these years; for our weekly meetings; for providing me the opportunity to grow; for correcting my grammar; and finally for the opportunity to collaborate with so many people during these last 5 years.

These last years offered me the amazing opportunity to collaborate with several people, I would like to dedicate a few lines to them.

I would like to thank Prof. Nils Löhndorf, I really enjoyed our weekly meetings, you taught me a whole range of stochastic optimization ideas (I must admit your reasoning is so quick I had troubles following-up). Your involvement in a big portion of this thesis is greatly appreciated and wouldn't have been possible without your help.

Being my former master's studies advisor, Prof. Mauricio Junca has been a constant guide. I would like to thank you, for guiding me with your remarkable ideas and out-of-box thinking; for your help explaining me, and clarifying, a variety of concepts; and finally for your constant availability, which helped me

to find the way at several points where the path was obscure.

I would also like to thank Prof. Philippe Chevalier, I really enjoyed our reinforcement learning seminar, which served as inspiration for the developments carried out in chapter 3. Furthermore, our collaboration in the demand response household topic served as a nice opportunity to test the concepts I learnt on the stochastic optimization front.

Dr. Lazaros Exizidis, I would like to thank you for providing me the opportunity to carry out an internship at ENTSO-E. An opportunity where I had the experience to closely appreciate industry operations, and to understand which are the industry needs. It has proven to be a fruitful experience. I hope we can further collaborate in future endeavours.

To the Jury members, Prof. Mathieu Van Vyve, Prof. Daniele Catanzaro and Prof. Mario Pereira, I would like to express my deepest gratitude for seeking the time in your agendas to participate in this defense. I have found your input extremely helpful, as well as enlightening. In particular, I would like to thank you for your detailed comments in the manuscript (special thanks to Prof. Daniele for providing me such detailed comments in the first draft of this thesis).

Besides research activities, there is a wide range of people I would like to thank, whose involvement during these last years was crucial to overcome difficult times, as well as to celebrate joyful moments.

I would like to thank the Energy Group (Céline, Yuting, Gilles, Ilyes, Quentin, Jacques, Jehum and Nicolas). It has been a nice experience to share several informal discussions at lunches, breaks and bars (and also thanks for the car rides when needed).

To my childhood friends, Fuentes, Umaña; to my college friends (Los Patrones) Andrés, Hernán, Jerson, "el tío", Sebastián, Simon, Rodolfo, Gustavo, Nicolás, Weimar. To my (now ex) neighbour, Sixto; To my Peruvian friend while in Belgium, Martin; your support during these years, sharing good moments (as well as memes), sharing thoughts and disappointments, have allowed me to easen the path during these years.

I would like to thank my parents, Luz and Antonio; my siblings, Paola and Camilo; my grandmother, Lal; my in-laws, Silvio and Estela; for their guidance and words of support throughout these years (and for taking care of Sashita while I was away). I would like to thank my grandfather, Luis Girardot; whose presence is missed, but whose remembrance has inspired me throughout his intellect, wisdom and vast knowledge.

Finally, I would like to thank my lovely wife, Estefany; for your support and patience despite being thousands of kilometers apart, with 7 hours of time zone difference; for our constant calls (while I was cooking dinner and you taking the bus to the university); and foremost for bringing colour and happiness to my life, happiness which is now embraced by our son. These last words, are for our son Arthur (sitting right now in my lap), without his help I would have finished this thesis earlier, but why finish earlier if I have the joy that his eyes disseminate throughout my soul.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Context and motivation

In an uncertain world, how can we make good decisions? Decision-makers are often faced with the responsibility of taking actions within complex systems, actions which have to be taken so as to react properly to future uncertain events. One such complex system, often described as the largest device ever built by humankind, is the electrical power system. The increased supply in energy derived from power systems has boosted our technological development, leading to a remarkable progress of society. However, consequences have emerged as well, unveiling the crucial importance of the decision-making processes that take place in power systems.

On the one hand, the greenhouse gas emissions resulting from the electrical infrastructure have impacted our environment. In order to mitigate such an adverse effect, during recent years, a proliferation of renewable resources has been observed and governments have put legislation in place in order to reduce greenhouse gas emissions, an example being the European Commission's target to reduce the European Union carbon footprint by 55% by 2030 [Com19a]. These renewable resources are inevitably controlled by nature's unpredictable availability of natural resources, and decision-makers are now faced with the task of making optimal decisions in such an increasingly uncertain environment. Moreover, unfortunate recent events such as the Russian invasion in Ukraine unveil the lack of preparedness of the electrical power system for reacting to unforeseen events. In fact, this conflict has created a disruption in gas supply, leading to a massive increase in electricity prices all across Europe [Com22], and leading to major efforts in order to secure enough gas reserves for the winter of 2022. As such, the electrical power system is expected to react to a variety of uncertain conditions, and tools must be developed that allow decision-makers to reach optimal decisions in such a complex and uncertain setting.

An approach for modeling the decision-making process is through mathematical programming. Specifically, stochastic programming [BL11] is a branch of math programming that focuses on optimization under uncertainty. The

past few decades have witnessed a proliferation of stochastic programming applications within power system operations. Systems where a decision must be reached before the realization of uncertainty have often been modeled as two-stage stochastic problems. Examples range from early applications in generation expansion planning [Blo82] or stochastic unit commitment [TBL96], to recent developments that target real-world capacity generation expansion problems [GAC14], large-scale stochastic unit commitment [POR14] or resource adequacy assessments [EE21]. Systems where multiple decisions have to be reached over time, each one without knowledge of the future, have been modeled as multi-stage stochastic programs. From early developments aiming at long-term hydrothermal scheduling [PP91], to recent developments targeting trading decisions within hydro systems [LWM13], real-time storage dispatch [PMCS17], gas storage valuation [LW20] or demand side response [GÁM$^+$22], both industry and academia have seen a widespread usage of these techniques. Nonetheless, limitations have become a matter of concern as well.

In the context of multi-stage stochastic programming, with specific applications in hydrothermal scheduling, questions have emerged regarding the computational tractability of finding an optimal solution [STdCS13]. Concretely, it has been observed that after days of computation, state of the art algorithms can be far from reaching an optimal solution. Such an observation is not an isolated one, and in fact our research has detected a similar behaviour using the state-of-the-art commercial software.

Insofar as applications in resource adequacy are concerned, regulation (EU) 943\2019 [Com19c] of the European Commission foresees the European Resource Adequacy Assessment (ERAA) [EE21], which aims at measuring the ability of the electric power system to react to adverse uncertain conditions. In particular, the adequacy concerns identified through this study can become the basis for member states to apply for capacity mechanisms. This consequently results in an institutional urge to develop an accurate resource adequacy assessment methodology. The scale of the problem at hand is of significant magnitude. Empirical evidence suggests that high performance computing servers equipped with the state-of-the-art commercial software are not able to tackle the problem.

The aforementioned empirical evidence indicates that, despite the existence of tools which aim at enabling the decision making process, as the scale of power system models increases and as the amount of uncertainty increases, state of the art commercial tools are becoming insufficient to address these novel concerns. The present thesis aims at bridging this methodological gap, by proposing novel algorithmic methods based on ideas from stochastic programming and high performance computing in order to provide feasible computational solutions for a variety of problems. We specifically focus on two applications: the multi-stage stochastic programming formulation of the long-term hydrothermal planning problem, and the two-stage stochastic capacity expansion problem, which is part of the European Resource Adequacy Assessment.

The present chapter aims at providing the relevant background for the

methods that are subsequently described in this dissertation. The organization of the chapter is as follows. Section 1.2 introduces two-stage stochastic programs, where two stages of decision-making are allowed, as well as standard techniques used to tackle them. Section 1.3 considers an extension of previous setting, multi-stage stochastic programs, where extra stages of decision-making are allowed. Algorithmic strategies to deal with these problems are introduced as well. Section 1.4 introduces Markov decisions processes. The introductory material is finalized with section 1.5 which introduces a key concept within the thesis: parallel computing. The chapter is concluded by presenting the contributions and organization of the dissertation in section 1.6.

1.2 Two-stage stochastic programming

Two-stage stochastic programming problems model the decision-making process by allowing for two stages of decisions. The first-stage decisions are reached before the realization of uncertainty, and once the uncertainty realizes a set of second-stage decisions is reached. The goal is to select first-stage decisions which minimize the current stage cost, while minimizing the expected costs of the second stage. Mathematically, we can write the problem as follows.

$$\min_{x_1,y_1} u_1^T x_1 + v_1^T y_1 + \mathbb{E}\left[\min_{x_2(\omega)} u_2^T(\omega)x_2(\omega)\right] \tag{1.1}$$

$$\text{s.t. } A_1x_1 + D_1y_1 = b_1 \tag{1.2}$$

$$x_1, y_1 \geq 0 \tag{1.3}$$

$$B_2(\omega)x_1 + A_2(\omega)x_2(\omega) = b_2(\omega), \text{ for all } \omega \in \Omega \tag{1.4}$$

$$x_2(\omega) \geq 0, \text{ for all } \omega \in \Omega \tag{1.5}$$

Here u_1, v_1, b_1 are vectors and B_1, A_1, D_1 are matrices which define the first-stage objective function and first-stage constraints. The set of uncertainty realizations is given by Ω, which we assume to be a finite set, and for $\omega \in \Omega$ we have the stochastic data $u_2(\omega), b_2(\omega), B_2(\omega), A_2(\omega)$. Note that the set of variables is distinguished between first-stage decisions x_1, y_1 and second-stage decisions x_2, y_2. Note that first-stage decisions do not depend on the uncertainty realization, while second-stage decisions do depend on $\omega \in \Omega$. Furthermore, second-stage decisions depend upon the first-stage decision x_1, as described by constraint 1.4.

In theory problem (1.1)-(1.5) can be formulated as a large linear program (and thus solved by standard linear programming solvers). However, it is often the case that the uncertainty set is prohibitively large, to the point loading the problem into memory is infeasible due to hardware constraints. Therefore, several decomposition strategies have been proposed in the literature. We proceed by describing some relevant strategies. Subsection 1.2.1 describes the

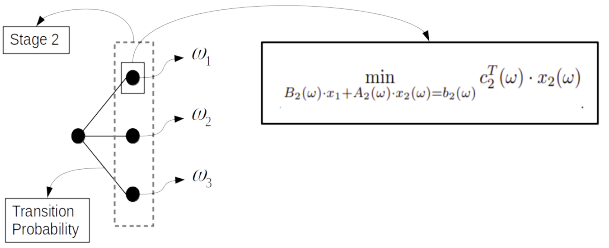


Figure 1.1: Decomposition of the two-stage stochastic problem. Each node represents an uncertainty realization and has associated to it an optimization problem that aims at minimizing the costs of the stage.

l-shaped method, which is the cornerstone of the cutting plane methods that we develop later in this dissertation. Subsection 1.2.2 describes the subgradient method while subsection 1.2.3 considers the progressive hedging scheme.

1.2.1 The L-Shaped algorithm

The L-shaped method breaks down the two-stage problem into smaller subproblems. The idea is to first consider separating the first and second stage, and then to consider a separate subproblem for each uncertainty realization of the second stage, as presented in Figure 1.1. Given an uncertainty realization $\widehat{\omega} \in \Omega$ and a first-stage decision $\widehat{x}_1$, the second-stage subproblem is as follows:

$$\begin{aligned} \mathcal{V}(\widehat{x_1}, \widehat{\omega}) = \min_{x_2(\widehat{\omega})} & u_2^T(\widehat{\omega}) x_2(\widehat{\omega}) \\ \text{s.t. } & B_2(\widehat{\omega})\widehat{x}_1 + A_2(\widehat{\omega}) x_2(\widehat{\omega}) = b_2(\widehat{\omega}) \qquad (\lambda_{\widehat{\omega}}) \\ & x_2(\omega) \geq 0 \end{aligned}$$

Here, $\lambda_{\widehat{\omega}}$ are the dual multipliers, which happen to be subgradients for $\mathcal{V}(x, \hat{\omega})$ around $\hat{x}$ [BL11], and are the basis for the iterative schemes that will be developed. These subproblems allow us to re-write problem (1.1)-(1.5) as:

$$\begin{aligned} \min_{x_1, y_1} & u_1^T x_1 + v_1^T y_1 + \mathbb{E}\Big[\mathcal{V}(x_1, \omega)\Big] \\ \text{s.t. } & A_1 x_1 + D_1 y_1 = b_1 \\ & x_1, y_1 \geq 0 \end{aligned} \tag{R}$$

The function $\mathbb{E}\Big[\mathcal{V}(x_1, \omega)\Big]$ is piece-wise linear convex in x_1 [BL11], and can therefore be under-approximated by supporting hyperplanes, commonly referred as to cuts. These cuts are computed by calculating the subgradients of

the second-stage functions $\mathcal{V}(x_1, \omega)$ [BL11]. Consequently, given a collection of N cuts $\{c_i(x_1)\}_{i=1}^N$, the previous problem can be approximated as follows:

$$\begin{aligned} \min_{x_1, y_1, \theta} \; & u_1^T x_1 + v_1^T y_1 + \theta \\ \text{s.t.} \; & A_1 x_1 + D_1 y_1 = b_1 \\ & \theta \geq c_i(x_1) \text{ for all } i \in 1, \ldots, N \end{aligned} \tag{M}$$

The L-Shaped algorithm advances by finding, at each iteration, a new supporting hyperplane for problem M. Each iteration begins by solving problem M, which leads to a trial $\widehat{x}_1^i$. The second-stage subproblems $\mathcal{V}(x_1, \omega)$ are then solved given $\widehat{x}_1^i$. The dual multipliers λ_ω are subgradients of the value functions $\mathcal{V}(x_1, \omega)$ at $\widehat{x}_1^i$, and can therefore be used for computing a supporting hyperplane [BL11], which is added to problem M. As we move through iterations, the method starts building an accurate representation of $\mathbb{E}\Big[\mathcal{V}(x_1, \omega)\Big]$ around the optimal region, eventually finding the optimal value. In fact, the method converges after finitely many iterations [PG08]. The algorithm is illustrated in pseudo-code as follows:

Algorithm 1: L-Shaped

INPUT: Provide a lower bound for θ. A maximum number of iterations N.
OUTPUT: Set of cuts $\{c_i(x_1)\}_{i=1}^N$ and first-stage solution $\widehat{x}_1^N$.

for $i = 1, \cdots, N$

1. **Forward Pass**: Solve problem M and get the optimal action $\widehat{x}_1^i$.
2. **Backward Pass for** $\omega \in \Omega$:

   (2.1) Solve the second stage subproblem $\mathcal{V}(\widehat{x}_1^i, \omega)$ at trial action $\widehat{x}_1^i$.

   (2.2) Using the dual multipliers λ_ω compute a supporting hyperplane around $\widehat{x}_1^i$.

   (2.3) Add the supporting hyperplane to problem M.

1.2.2 Projected stochastic subgradient algorithm

The idea of this algorithm is as follows: given an initial trial first-stage solution $\widehat{x}_1^1$, calculate a subgradient of the objective function of problem (1.1)-(1.5) around $\widehat{x}_1$ and update the trial first-stage solution along the direction of such a subgradient. This method has the following advantages: (i) it does not require a

hyperplane description of $\mathbb{E}\Big[\mathcal{V}(x_1,\omega)\Big]$ in order to advance to the next candidate $\widehat{x}_1^2$; (ii) it can be initialized around a trial $\widehat{x}_1^1$ which is known in advance to be somewhat close to the optimal value, thus ensuring that if the starting value is close to the optimal solution then the iterates remain near the optimal point and lead to few iterations of the algorithm.

We start by decomposing problem (1.1)-(1.5) by rewriting it as problem R. Note that a subgradient of the objective function along the x_1 coordinate is:

$$\rho = u_1 + \mathbb{E}\Big[\lambda_\omega\Big]$$

Due to the reformulation R, the calculation of these slopes can be decomposed into calculating several subproblems, concretely by solving $\mathcal{V}(x_1,\omega)$ for each $\omega \in \Omega$. This decomposition allows us to apply the following scheme described as pseudo-code in algorithm 2:

Algorithm 2: Projected stochastic subgradient algorithm

INPUT: Provide an initial trial first-stage solution $\widehat{x}_1^1$, and a maximum number of iterations N.
OUTPUT: First-stage solution $\widehat{x}_N^1$.

for $i = 1, \cdots, N$

1. **for** $\omega \in \Omega$: solve the subproblem $\mathcal{V}(\widehat{x}^i,\omega)$ at trial $\widehat{x}^i$.
2. Using the dual multipliers, calculate the objective function slope ρ^i.
3. Apply a projected subgradient step as:

$$\widehat{x}_1^{i+1} = \max\{0, \widehat{x}_1^i + \alpha^i \cdot \rho^i\}$$

We commence by providing an initial candidate action $\widehat{x}_1^1$. During each iteration, the subproblems $\mathcal{V}(\widehat{x}_1^i,\omega)$ are solved for all $\omega \in \Omega$. Using the dual multipliers of these subproblems, the subgradients ρ are calculated. Finally, the trial action is updated through a projected subgradient step: $\widehat{x}_1^{i+1} = \max\{0, \widehat{x}_1^i + \alpha^i \cdot \rho^i\}$. The term α^i is a stepsize which is crucial to the performance of the algorithm [Boy]. For example one can select the Polyak stepsize [Boy], which ensures convergence. The Polyak stepsize can be described as follows: $\alpha^i = \frac{W^* - W^i}{||p^i||^2}$. Here, W^* is the optimal value of the R problem, while W^i is the objective value of the current iterate. As the optimal value W^* is not known in advance, an approximate value can be used. No lower bound is obtained, thus upper bound stabilization is used as stopping criterion.

1.2.3 Progressive hedging

Progressive hedging [RW91] has emerged as an alternative approach for tackling large-scale stochastic programming problems. For this approach we introduce the so-called non-anticipativity constraints, for which extra first-stage variables are introduced by defining a variable $x_1(\omega)$ for all $\omega \in \Omega$. Problem (1.1)-(1.5) is reformulated as:

$$\min_{x_1, y_1} u_1^T x_1 + v_1^T y_1 + \mathbb{E}\left[\min_{x_1(\omega), x_2(\omega)} u_2^T(\omega) x_2(\omega)\right] \tag{1.6}$$

$$\text{s.t. } A_1 x_1 + D_1 y_1 = b_1 \tag{1.7}$$

$$x_1, y_1 \geq 0 \tag{1.8}$$

$$x_1 = x_1(\omega) \text{ for all } \omega \in \Omega \tag{1.9}$$

$$B_2(\omega) x_1(\omega) + A_2(\omega) x_2(\omega) = b_2(\omega), \text{ for all } \omega \in \Omega \tag{1.10}$$

$$x_2(\omega) \geq 0, \text{ for all } \omega \in \Omega \tag{1.11}$$

Equation 1.9 are referred as the non-anticipativity constraints. The progressive hedging algorithm relaxes these constraints, thereby allowing us to consider an independent subproblem for each scenario. The algorithm approximates the dual multipliers of the relaxed non-anticipativity constraints and additionally introduces a quadratic regularization term ρ. The steps of the algorithm can be interpreted as a process of building consensus between scenarios until reaching convergence.

The progressive hedging scheme proceeds by proposing the following set of subproblems for each $\omega \in \Omega$:

$$\begin{aligned} f(\bar{x}_1, \omega, w_\omega^T) = \min_{x_1, y_1, x_2} & \left[u_1^T x_1 + v_1^T y_1 + u_2^T(\omega) x_2\right] + w_\omega^T (x_1 - \bar{x}_1) + \frac{1}{2}\rho ||x_1 - \bar{x}_1||^2 \\ \text{s.t. } & A_1 x_1 + D_1 y_1 = b_1 \\ & x_1, y_1 \geq 0 \\ & B_2(\omega) x_1 + A_2(\omega) x_2 = b_2(\omega) \\ & x_2 \geq 0 \end{aligned}$$

The first term corresponds to the objective cost of problem (1.1)-(1.5) for a single $\omega \in \Omega$, the second term corresponds to the relaxation of the non-anticipativity constraint associated to x_1 (consequently w_ω is the dual multiplier associated to this constraint), and the latter part is a regularization term associated to x_1. During each iteration of the progressive hedging algorithm, an approximation of the dual multipliers w_ω is found, and a first stage decision approximation $\bar{x}_1$ is found by averaging the first-stage solutions found by each subproblem. A description of the algorithm is presented in Algorithm 3.

Algorithm 3: Progressive hedging

INPUT: Provide an initial trial action $\bar{x}_1^0$ and set w_ω^0 equal to 0 for all $\omega \in \Omega$, and maximum number of iterations N.
OUTPUT: First-stage solution $\bar{x}_1^N$

for $i = 1, \cdots, N$

1. **for** $\omega \in \Omega$: solve the subproblem $f(\bar{x}_1^0, \omega, w_\omega^0)$, thus obtaining a first-stage solution $x_{1,\omega}$.
2. Aggregate the first-stage decision as: $\bar{x}_1^i = \mathbb{E}[x_{1,\omega}]$.
3. Update dual multipliers as:

$$w_\omega^i = w_\omega^{i-1} + \rho(x_{1,\omega} - \bar{x}_1^{i-1})$$

Tuning the parameter ρ is critical for the performance of the algorithm. In particular, an adequate tuning can reduce the number of iterations required for convergence. As discussed in [BSdG$^+$22] tuning such a parameter is problem dependant, and tuning strategies are followed for each particular problem.

1.3 Multi-stage stochastic programming

Multi-stage stochastic programming involves several stages of decision making. Let us consider a multistage stochastic linear program over T stages, given by:

$$\min_{\substack{A_1x_1+D_1y_1=b_1\\ x_1,y_1\geq 0}} u_1^Tx_1 + v_1^Ty_1 + \mathbb{E}\left[\min_{\substack{B_2x_1+A_2x_2+D_2y_2=b_2\\ x_2\geq 0}} u_2^Tx_2 + v_2^Ty_2 \right.$$
$$\left. +\mathbb{E}\left[\cdots + \mathbb{E}\left[\min_{\substack{B_Tx_{T-1}+A_Tx_T+D_Ty_T=b_T\\ x_T\geq 0}} u_T^Tx_T + v_T^Ty_T\right]\right]\right] \quad \text{(MSP-P)}$$

For each stage, we have vectors u_t, v_t, b_t as well as matrices B_t, A_t, D_t that constitute the stochastic data process $\xi_t = (u_t, v_t, b_t, B_t, A_t, D_t)$. We assume that $u_1, v_1, b_1, B_1, A_1, C_1$ are deterministic, and that we are given an initial condition x_0. To avoid notational clutter, we drop the dependence of B_t on ω_t that indicates stochasticity of the data process. Let us assume that, at each stage, there are finitely many outcomes Ω_t, and that the data process follows a Markov chain. We therefore consider that we can define transition matrices $P(\xi_{t+1}|\xi_t)$.

Similarly as for two-stage stochastic linear problems, multistage stochastic linear problems can be formulated as large linear problems. Unfortunately, it

is typically the case that the exponential growth in the number of scenarios, as the number of stages increase, makes such an approach infeasible due to hardware restrictions. Leveraging on the success of cutting-plane methods on two-stage settings the literature has proposed algorithmic techniques to deal with the multi-stage setting. Subsection 1.3.1 introduces the so-called nested benders decomposition algorithm that builds off of the l-shaped method, while subsection 1.3.2 introduces the Stochastic Dual Dynamic Programming (SDDP) algorithm, a central concept in the thesis, which proposes a sampling methodology as to provide a tractable algorithmic scheme.

1.3.1 Nested L-Shaped

The Nested L-Shaped method, described in [BL11], is a generalization of the L-Shaped method for the multi-stage setting. Similarly as for the two-stage setting, the algorithm works by finding supporting hyperplanes that provide tight outer approximations of the value functions of the dynamic programming formulation in regions of the state space that can be reached by the optimal policy.

The dynamic programming equations of problem MSP-P can be written as

$$\begin{aligned} V_t(x_{t-1}, \xi_t) = &\min_{x_t, y_t} u_t^T x_t + v_t^T y_t + \mathcal{V}_{t+1}(x_t, \xi_t) \\ &\text{s.t. } B_t x_{t-1} + A_t x_t + D_t y_t = b_t \\ &\quad x_t, y_t \geq 0 \\ \mathcal{V}_{t+1}(x_t, \xi_t) = &\mathbb{E}\Big[V_{t+1}(x_t, \xi_{t+1}) \Big| \xi_t\Big] \end{aligned}$$

for $t = 2, \cdots, T-1$. There is no associated function $\mathcal{V}_T$ in the last stage. Note that, due to the Markov property, the cost-to-go functions $V_t(x_{t-1}, \xi_t)$ and the expected value cost-to-go functions $\mathcal{V}_{t+1}(x_t, \xi_t)$ depend only on ξ_t, and not on the entire history of the data process. Moreover $\mathcal{V}_{t+1}(x_t, \xi_t)$ is a convex function of x_t [BL11] and can therefore be approximated by a piecewise linear function. The idea of the Nested L-Shaped method is to generate approximations of the expected value cost-to-go functions through supporting hyperplanes, commonly referred to as cuts. Given a collection of N cuts $\{c^i_{\xi_t}(x_t)\}_{i=1}^N$ the cost-to-go functions can be approximated as:

$$\begin{aligned} \widehat{V}_t(x_{t-1}, \xi_t) = &\min_{x_t, y_t} u_t^T x_t + v_t^T y_t + \theta_{\xi_t} \\ &\text{s.t. } B_t x_{t-1} + A_t x_t + D_t y_t = b_t \\ &\quad \theta_{\xi_t} \geq c^i_{\xi_t}(x_t) \text{ for all } i \in 1, \ldots, N \\ &\quad x_t, y_t \geq 0 \end{aligned}$$

To describe the algorithm, we introduce the set $\Phi = \{(\xi_1, \ldots, \xi_T) : \xi_t \in \Omega_t \text{ and } P(\xi_t|\xi_{t-1})) \neq 0\}$, which is the set of all possible (non-zero probability) uncertainty paths of the multistage stochastic problem. The cuts are generated

through an iterative process which consists of forward and backward passes. During forward passes, the algorithm moves forward in the number of stages, visiting the different paths of uncertainty in set Φ, and computes trial solutions for each stage. During backward passes, the algorithm proceeds backwards in the number of stages, building cuts around the trial solutions obtained in the forward pass. A pseudo-code description of the algorithm can be found in Algorithm 4.

Algorithm 4: Nested L-Shaped

INPUT: A lower bound for θ_{ξ_t} for $t = 1, \ldots, T-1$, $\xi_t \in \Omega_t$. A maximum number of iterations N.
OUTPUT: A cutting-plane approximation $\{c^i_{\xi_t}(x_t)\}_{i=1}^N$ for $t = 1, \ldots, T$, $\xi_t \in \Omega_t$.

for $i = 1, \cdots, N$.

1. **Forward Pass**.

   (1.1) **for** $(\xi_1^k, \ldots, \xi_T^k) \in \Phi$:

   for $t = 1, \cdots, T$:

   Solve $\widehat{V}_t(\widehat{x}_{t-1}^k, \xi_t^k)$ and store the trial action $\widehat{x}_t^k$.

2. **Backward Pass**.

   (2.1) **for** $t = T, \ldots, 2$:

   for $(\xi_1^k, \ldots, \xi_T^k) \in \Phi$:

   (2.1.1) **for** $\xi_t \in \Omega_t$: Solve $\widehat{V}_t(\widehat{x}_{t-1}^k, \xi_t)$ and store the dual multipliers.

   (2.2.2) Using the dual multipliers compute a supporting hyperplane around $\widehat{x}_{t-1}^k$.

   (2.2.3) Add the supporting hyperplane to problem $\widehat{V}_{t-1}$.

One way to understand the algorithm is by relying on a lattice representation (see definition 1.1), which is a graphical way of representing the underlying structure of the problem.

Definition 1.1. *A lattice is an undirected graph. Each column of nodes represents a stage of the problem and each node represents an uncertainty realization ξ_t. Each node is associated to an optimization problem which aims at minimizing the current period costs plus the future costs, given x_{t-1}, for the uncertainty realization ξ_t. This problem corresponds to the calculation of the cost-to-go function $V_t(x_{t-1}, \xi_t)$. Note also that each node has an expected cost-to-go function associated to it.*

The nested L-Shaped algorithm is described graphically over a lattice in Figure 1.2. The problem has three stages, and two uncertainty realizations for stages 2 and 3. The algorithm starts in the forward pass. During this step, all uncertainty paths are visited, 4 in this example. Next, we proceed with the backward pass. Starting from the last stage (the third stage), the second-stage decision obtained in the red path is evaluated in all nodes of the third stage. Using the dual solution information of the subproblems, a cut is built. The process is repeated with the second-stage decisions obtained by the other uncertainty paths. Once all third-stage nodes have been solved, the procedure continues with the second stage.

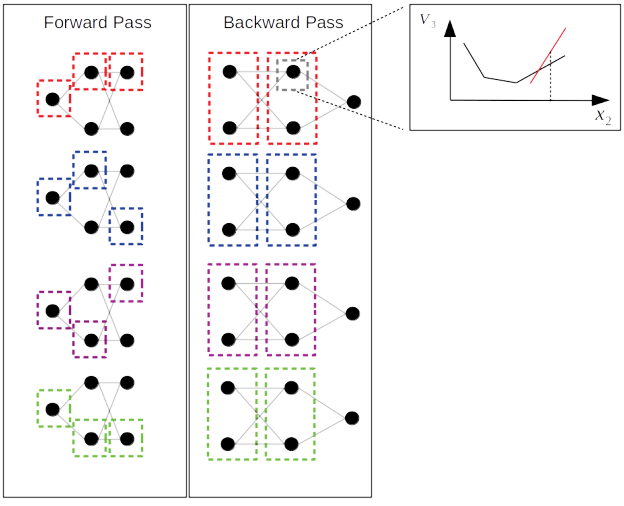


Figure 1.2: The nested L-Shaped algorithm described graphically over a lattice. In the forward pass, trial points are obtained for every uncertainty path. The backward pass proceeds backward in the number of stages. Starting form the last stage, the algorithm solves the subproblems at the trial points obtained during the forward pass, producing cuts for the expected value functions corresponding to the nodes of stage 2. The backward pass continues in this manner throughout the remaining stages.

1.3.2 Stochastic Dual Dynamic Programming (SDDP)

The nested l-shaped algorithm provides a decomposition approach to tackle multistage problems. However, the forward search across all possible uncertainty paths translates in a large number of subproblems that need to be solved during each iteration of the algorithm. In particular, as the number of stages increase the amount of uncertainty paths increases exponentially, thus leading to an intractable approach. The Stochastic dual dynamic programming (SDDP) algorithm proposes a sampling strategy at the level of the forward pass to cope with this exponential growth.

The SDDP algorithm has emerged as a scalable algorithm for solving multistage stochastic programming problems. Since the seminal work of [PP91], SDDP has captivated the interest of the stochastic programming community and achieved widespread adoption in industrial applications [FBP10, DMPFG10, PBM13, LWM13, LW20, DPMD19]. PSR, a consulting firm based in Rio de Janeiro, has invented and pioneered the SDDP algorithm, which is at the core of its hydro-thermal planning software that is marketed under the same name [PSR]. Similarly as for the nested L-Shaped method, the SDDP algorithm proceeds by building supporting hyperplane approximations of the value functions of the dynamic programming equations. Furthermore, each iteration also consists of forward and backward passes, but as opposed to the nested L-Shaped method, a Monte Carlo sampling is introduced at the level of the forward pass. The SDDP algorithm is described in pseudo-code in algorithm 5.

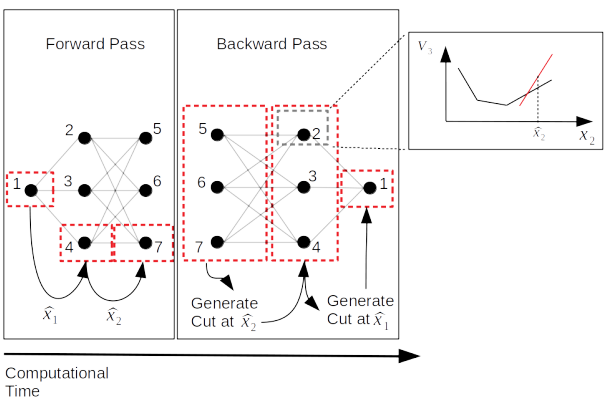


Figure 1.3: The SDDP algorithm described graphically over a lattice. The computational time evolves along the x-axis. The length of the red dashed boxes represents the elapsed computational time. In the forward pass, a scenario is drawn and trial points are obtained. The backward pass starts by solving the last stage, which produces a cut for the expected value functions corresponding to the nodes of stage 2. The backward pass continues in this manner throughout the remaining stages.

Algorithm 5: SDDP

INPUT: Provide a lower bound for θ_{ξ_t} for $t = 1, \ldots, T-1$, $\xi_t \in \Omega_t$. A maximum number of iterations N.
OUTPUT: A cutting-plane approximation $\{c^i_{\xi_t}(x_t)\}_{i=1}^N$ for $t = 1, \ldots, T$, $\xi_t \in \Omega_t$.

for $i = 1, \cdots, N$.

1. **Forward Pass**.

   (1.1) Draw K Monte Carlo scenarios of the realization of uncertainty throughout the entire time horizon of the problem. This yields sequences $\xi_1^k, \cdots, \xi_T^k$, where $\xi_t^k \in \Omega_t$ for $k = 1, \cdots, K$.

   (1.2) **for** $k = 1, \cdots, K$:

   for $t = 1, \ldots, T$:

   Solve $\widehat{V}_t(\widehat{x}_{t-1}^k, \xi_t^k)$ and store the trial action $\widehat{x}_t^k$.

2. **Backward Pass**.

   (2.1) **for** $t = T, \ldots, 2$:

   for $k = 1, \ldots, K$:

   (2.1.1) **for** $\xi_t \in \Omega_t$: Solve $\widehat{V}_t(\widehat{x}_{t-1}^k, \xi_t)$ and store the dual multipliers.

   (2.2.2) Using the dual multipliers compute a supporting hyperplane around $\widehat{x}_{t-1}^k$.

   (2.2.3) Add the supporting hyperplane to problem $\widehat{V}_{t-1}$.

Figure 1.3 presents the SDDP algorithm as it traverses the lattice. The computational time evolves along the x-axis. The forward pass starts by sampling a scenario. The transition probabilities are used for this purpose, selecting as a result the nodes that are indicated with a red dashed box, namely nodes 1, 4, 7. The cost-to-go function V_t corresponding to these nodes is calculated, starting from the first, and ending with the last stage, that is to say beginning with node 1, then 4, and finally 7. As a result, we obtain trial points $\widehat{x}_1, \widehat{x}_2, \widehat{x}_3$. The backward pass is then executed. Starting from the last stage, the value functions for all nodes, evaluated at the trial point $\widehat{x}_2$, are calculated. In Figure 1.3, this corresponds to solving nodes 5, 6 and 7. The solution information of the nodes is used to build a cut of the expected value function at the trial point

$\widehat{x}_2$. The backward pass then proceeds in the same way for stage 2.

1.4 Markov decision processes

Markov decision processes (MDP) are a framework for analyzing decision making over time and under uncertainty. An MDP is defined by the tuple $(\mathcal{S}_t, \mathcal{A}_t, C_t, P)$, where $\mathcal{S}_t$ is the set of states, $\mathcal{A}_t$ is the set of actions, $C_t : \mathcal{S}_t \times \mathcal{A}_t \to \mathbb{R}$ is the reward function of an agent, and $P(s_{t+1}|a_t, s_t)$ is the probability of transitioning to state $s_{t+1} \in \mathcal{S}_t$ if we are in state $s_t \in \mathcal{S}_t$ and select action $a_t \in \mathcal{A}(s_t)$. The set $\mathcal{A}(s_t)$ indicates the set of feasible actions when in state s_t. A policy $\pi = (\pi_1, \pi_2, \ldots)$ is a vector of functions, where each function maps states to actions[1], namely $\pi_t : \mathcal{S}_t \to \mathcal{A}_t$. We will focus on finite horizon models with a time horizon T. Further details on the definition of MDPs can be found in [Pow07, SB18, Put14]. The objective in MDPs is to obtain a policy $\pi \in \Pi$ that minimizes the expected cost over the decision-making horizon:

$$\min_{\pi \in \Pi} \mathbb{E}\Big[\sum_{t=1}^{T} C_t(s_t^\pi, a_t^\pi)\Big]$$

The optimal expected reward is often calculated with the help of the value functions of the problem. Concretely, the value function $V_t^\pi(s_t)$ is the expected reward when starting in state s_t and following policy π. Mathematically, value functions are defined as

$$V_t^\pi(s_t) = \mathbb{E}\Big[\sum_{n=t}^{T} C_n(s_n^\pi, a_n^\pi)\Big| s_t\Big]$$

We will use the notation

$$\mathcal{V}_{t+1}^\pi(s_t, a_t) = \mathbb{E}\Big[V_{t+1}^\pi(s_{t+1})|s_t, a_t\Big]$$

to refer to the value functions after taking expectation. The value functions that correspond to an optimal policy satisfy the following optimality conditions [SB18, Put14]:

$$\begin{aligned} V_t^*(s_t) &= \min_{a_t \in \mathcal{A}(s_t)} C_t(s_t, a_t) + \mathbb{E}\Big[V_{t+1}^*(s_{t+1})|s_t, a_t\Big] \\ &= \min_{a_t \in \mathcal{A}_t(s_t)} C_t(s_t, a_t) + \mathcal{V}_{t+1}^*(s_t, a_t) \end{aligned}$$

Another noteworthy concept within the MDP framework are Q-factors. A Q-factor $Q^\pi(s_t, a_t)$ represents the expected reward that is obtained after selecting action a_t while being in state s_t, and then following policy π. Formally,

[1]The literature presents more general policies, where a state is mapped to a probability measure over the set of actions. However we will focus on deterministic policies.

we can define Q-factors as

$$Q_t^\pi(s_t, a_t) = \mathbb{E}\Big[\sum_{n=t}^{T} C_n(s_n^\pi, a_n^\pi)\Big| s_t, a_t\Big].$$

Note that we can express our Q-factors in terms of the value functions as

$$\begin{aligned} Q_t^\pi(s_t, a_t) &= C_t(s_t, a_t) + \mathbb{E}\Big[V_{t+1}^\pi(s_{t+1})|s_t, a_t\Big] \\ &= C_t(s_t, a_t) + \mathcal{V}_{t+1}^\pi(s_t, a_t) \end{aligned}$$

Note also that the optimal value functions can be expressed in terms of the optimal Q-factors as

$$V_t^*(s_t) = \min_{a_t \in \mathcal{A}(s_t)} Q_t^*(s_t, a_t)$$

1.5 Parallel computing

High performance computing has become critical in tackling large-scale power system optimization problems. Early work on the topic [MPG87, TPPM90] describes parallel computing schemes for addressing security-constrained optimal power flow and hydro system scheduling, respectively, using Bender's decomposition. Parallel computing schemes for Lagrangian decomposition have been developed for solving optimal power flow problems [KB97, BB03]. In recent years, parallel computing has enabled tackling large-scale instances of stochastic unit commitment [POR14]. In particular, the use of asynchronous parallel computing has resulted in a reduction of computation time from weeks to a few hours for certain stochastic unit commitment instances [AP20]. Within the hydrothermal scheduling framework, and by means of the stochastic dual dynamic programming algorithm, a widespread usage of parallel computing has emerged as well [dSF03, PBM13, HB15, DK21]. This has allowed the research community to reduce the computational time for a variety of configurations and to open further areas of research.

This section aims at developing the high performance computing terminology that will be used throughout the dissertation. Subsection 1.5.1 will describe distributed and shared memory configurations, while subsection 1.5.2 will describe synchronous and asynchronous computing.

1.5.1 Distributed and shared memory computing

Hardware configurations are crucial in order to establish parallel computing strategies. One such configuration is the distribution of computing cores across the HPC infrastructure.

Physical cores are commonly arranged in the so-called computing nodes. Each node is a computational device with its own motherboard, processor (a processor typically contains a few tens of physical cores) and RAM. Thus, each

computing node can be considered as a single computer. A high performance computing infrastructure is built by connecting several such nodes (or computers), leading to a device with a large array of cores. A peculiarity of such a configuration is noteworthy to emphasize: each core will have access just to the RAM memory and data available within the node. This leads to two paradigms within parallel computing: distributed memory and shared memory configurations. The difference between these two paradigms is of practical relevance.

Distributed configurations are those where algorithms are run using several computing nodes. Naturally, these configurations have access to a larger array of cores. However, an important drawback arises: there is a non-negligible start-up and communication time between two different cores. This implies that the algorithm has to be designed so as to minimize data transfers and ensure that enough work is allocated within each core so as to limit the start-up costs.

On the other hand, shared memory configurations are those where a single computing node is used. These schemes have the advantage that no start-up and communication time is present, as well as no data transfers, between two different cores. These advantages come at the cost of the limitation on the number of cores that are available (typically a few tens of cores per node).

1.5.2 Synchronous and asynchronous computing

The previous subsection discussed hardware configurations and how these affect the communication across CPUs. The present subsection aims at describing two common software paradigms within CPU communication: synchronous and asynchronous computing [BT89].

Let us consider a parallel computing algorithm. A synchronous computing scheme is one in which a CPU commences a task, finishes, but before proceeding to the next task has to wait for all other CPUs to finish their respective task. One may encounter this situation when the subsequent task requires the information gathered by all CPUs before being able to commence. An advantage of these schemes is that all CPUs have access to the information collected by all CPUs. A disadvantage is the computational time lost due to the need to wait for the slowest CPU, thus leading to bottlenecks which may harm the performance of the algorithm. On the other hand, asynchronous algorithms are those were a CPU does not need to wait for the full set of CPUs in order to proceed to the next task, but can proceed using the CPU information of the CPUs that have already finished their job. The main advantage of these algorithms is that no computational time is lost, which allows such algorithms to use the computational resources more effectively. A disadvantage is that the CPUs may not have access to the most updated information of other CPUs, and implementation in code is more challenging.

As an example, let us consider an iterative algorithm, where the work required to complete each iteration is distributed among the available CPUs.

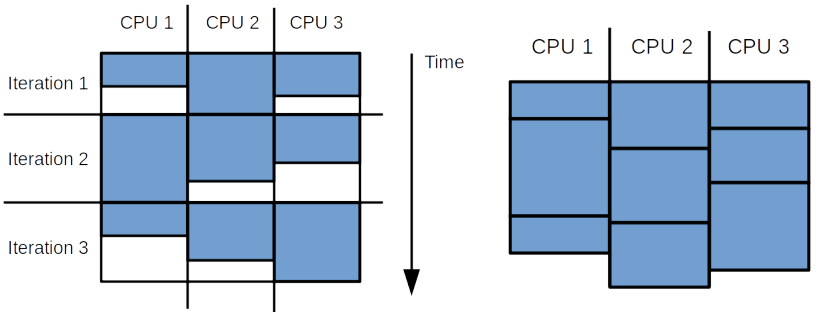


Figure 1.4: Parallel computing iterative algorithm. The synchronous scheme is presented in the left panel, while the asynchronous scheme is shown in the right panel.

Figure 1.4 presents the evolution of the algorithm in a synchronous and asynchronous fashion. The left figure presents the former setting while the right figure presents the latter configuration. In the synchronous setting, during each iteration, all 3 CPUs have to wait for the slowest CPU before starting a new iteration, thus leading to idle times for some CPUs. On the other hand, the asynchronous setting does not have idle times, as each CPU continues to the next iteration without waiting for others. Note that, in the synchronous setting, the CPUs have access to the information of the preceding iteration of other CPUs before starting a new iteration, while in the asynchronous setting each CPU only has the information of the CPUs that have already finished their iteration.

1.6 Structure of the dissertation and contributions

This dissertation is organized into four chapters. We make the distinction between the developments used to tackle two-stage and multi-stage stochastic problems. Consequently, the thesis is organized into two parts. Following the research evolution throughout the years, the first part focuses on multi-stage stochastic problems, the latter on two-stage stochastic problems.

Part I - Parallel Computing and Multi-stage Stochastic Programming: Hydrothermal Scheduling Applications.

- *Chapter 2. Parallel and Distributed Computing for Stochastic Dual Dynamic Programming.*

  The stochastic dual dynamic programming (SDDP) algorithm, developed by [PP91], has emerged as a scalable approximation method for tackling multistage stochastic programming problems. While the algorithm has

shown success in a wide range of areas, computational challenges have also been raised in the literature [STdCS13]. This chapter aims at proposing and investigating various parallel computing strategies for easing the computational burden. The results of this chapter have been published in the following work:

  - ◇ Ávila, D., Papavasiliou, A., Löhndorf, N., Parallel and Distributed Computing for Stochastic Dual Dynamic Programming, *Comput Manag Sci*, 19, 199–226 (2022).

- *Chapter 3. Batch Learning SDDP for Long-Term Hydrothermal Planning*: The parallelization strategies that are studied in Chapter 2 point towards a common point: parallelism in its own may not be able to overcome convergence issues. Inspired by this observation, this chapter introduces a variant of SDDP, named *Batch Learning SDDP (Bl-SDDP)*, which proposes a diligent approach for computing value functions with the objective of reducing the back-propagation of errors, and in turn providing faster convergence. These notions are connected to ideas in the reinforcement learning community, which are described in the chapter. This chapter is based on the following publication:

  - ◇ Ávila, D., Papavasiliou, A., Löhndorf, N., Batch Learning SDDP for Long-Term Hydrothermal Planning, to appear in *IEEE Transactions on Power Systems* (2023).

Part II - Parallel Computing and Two-stage Stochastic Programming: European Resource Adequacy Assessment.

- *Chapter 4. Applying high performance computing to the ERAA study*: This work considers the European Resource Adequacy Assessment (ERAA), which is a pan-European resource adequacy assessment process that is being developed by the European Networks of Transmission System Operators for Electricity (ENTSO-E). A critical part of this process is the so-called Economic Viability Assessment (EVA) model, which aims at determining future capacity expansion and retirement opportunities for the entire European network. As such, the problem is stochastic. Nevertheless, due to computational constraints, simplified approaches have been followed by ENTSO-E. Our work formulates the problem as a two-stage stochastic program and proposes two decomposition algorithms for solving the problem. These algorithms are implemented in high-performance computing infrastructure. The first is a subgradient-based algorithm, and the second uses a relaxation of the second stage (the economic dispatch) in order to speed up subgradient calculation, thus achieving a considerable reduction in solution time. We compare our schemes against the commonly used L-Shaped algorithm and against the progressive hedging algorithm. We compare the obtained stochastic solution against the

deterministic solution proposed by ENTSO-E for their 2021 study and analyze the impact of the stochastic solution on various adequacy indicators. The results of this chapter have been submitted for publication in the following manuscript:

  - ◇ Ávila, D., Papavasiliou, A., Junca, M., Exizidis, L., Applying High Performance Computing to the European Resource Adequacy Assessment, to appear in *IEEE Transactions on Power Systems* (2023).

- *Chapter 5. Applying scenario reduction to the ERAA study*: The high computational complexity of the Economic Viability Assessment (EVA), within the ERAA, has led ENTSO-E to seek scenario reduction strategies in order to reduce the overall computational complexity of the problem. This chapter proposes a scenario reduction methodology for the EVA study, which is still under development by the candidate. Our preliminary research show advantages as compared to the approach followed by ENTSO-E for ERAA 2021, in particular related to the accuracy of quantifying load shedding, which is critical for adequacy metrics. We highlight that this is a preliminary work, where we explore preliminary aspects. Our proposed scenario reduction scheme has been implemented by ENTSO-E for their 2022 edition and has been published as part of the ERAA 2022 methodology as follows:

  - ◇ Methodology for the European resource adequacy assessment, *ENTSO-E*, 35-36 (2022).

Part I

Parallel Computing and Multi-stage Stochastic Programming: Hydrothermal Scheduling Applications

2 Parallel and Distributed Computing for SDDP

2.1 Introduction

The stochastic dual dynamic programming (SDDP) algorithm has emerged as an attractive approach to deal with multistage stochastic problems. Although, other alternatives have been proposed (such as the nested L-Shaped method), in practice the sampling strategy introduced by the SDDP algorithm has proven to be better suited from a computational perspective. Nonetheless, multistage stochastic programming problems are generally computationally intractable and therefore pose serious computational challenges, even for SDDP. Such challenges are documented in [STdCS13], where SDDP is unable to close the optimality gap for the problem under investigation, even after several hours of run time. In order to speed up convergence, several approaches have been proposed in the existing literature. Such methods include cut selection techniques for removing redundant hyperplanes [DMPF15, Gui17, GB19, LWM13], regularization techniques for selecting better trial points during the forward pass [AP18], forward sampling schemes that exploit problem structure [DB06, HP14], as well as parallel computing [dSF03, PBM13, HB15, MDBB21].

2.1.1 Parallelism in Large-Scale Optimization

Parallelism can be crucial for tackling large-scale optimization problems, but is often undermined by synchronization bottlenecks. In power system applications, for instance, parallelism has allowed tackling large-scale day-ahead stochastic unit commitment problems [POR14]. While synchronous parallel computing algorithms require run times in the order of weeks for certain instances of stochastic unit commitment, asynchronous implementations of Lagrange relaxation have been shown to reduce these run times to a few hours [AP20]. This allows us to hope for an eventual deployment of stochastic operational planning models in actual operations, where run time constraints are critical.

This objective motivates our research on the parallelism attributes of SDDP.

Nevertheless, the existing literature on SDDP shows limited results in this front. The literature provides a narrow set of parallel schemes, which rely on increasing the number of Monte Carlo samples that are used in the forward pass of the algorithm [dSF03, PBM13, HB15, DK21]. The aforementioned literature provides evidence that such schemes are superior relative to a serial approach. The literature, however, does not focus on how different schemes may compare relative to each other.

In [PBM13] and [dSF03] the authors propose a synchronous parallel scheme, according to which subproblems are solved in parallel at every stage. Parallelization is also applied in the backward pass of the algorithm. This introduces a natural synchronization bottleneck at each stage. In [HB15] the authors propose a relaxation in the synchronization points of this synchronous parallel scheme. According to the proposed scheme, a worker waits for a subset of subproblems at each stage of the backward pass. The authors provide empirical evidence that demonstrate that their approach achieves performance gains relative to the synchronous setting. However, the analysis is not sufficiently robust, since the authors declare convergence once the lower bound is within the 95% confidence interval of the upper bound. This convergence criterion has been criticized in [Sha11].

In our work, we propose a richer family of parallelizable algorithms for SDDP. Our analysis considers synchronous as well as asynchronous computation. We develop a taxonomy of (i) parallelization by scenario of Monte-Carlo samples in the forward pass of the algorithm, and (ii) parallelization by node of the underlying stochastic process. Our taxonomy encompasses the traditional parallel schemes that are encountered in the literature, and gives rise to new parallel formulations. We present an analysis for the resulting class of algorithms, and compare the relative strengths and weaknesses of the proposed algorithms.

2.1.2 Limitations of Parallelism

Parallelism is often viewed as a one-way procedure, where more processors necessarily imply better performance. The SDDP literature tends to consider an increase in the number of Monte Carlo samples in the forward pass, in order to be able to rely on more processors [dSF03, PBM13, HB15, DK21]. Nevertheless, there is a lack of evidence for assessing the effect of such an increase on the performance of the SDDP algorithm. In our work, we present empirical evidence which indicates that increasing the number of Monte Carlo samples in the forward pass of the SDDP algorithm may in fact undermine performance. This indicates a serious drawback with traditional parallel schemes that have been proposed in the literature, since these traditional parallel schemes may not scale well.

2.1.3 Contributions

Our contribution to the literature on SDDP parallelization is two-fold. First, we enrich the set of available parallel schemes that have been considered in the literature, by considering both synchronous as well as asynchronous computation, and we present a taxonomy that categorises existing and new schemes. Second, we conduct an extensive numerical experiment in order to compare the relative performance of these schemes when the objective is to achieve tight optimality gaps with high confidence. Moreover, our analysis provides empirical evidence that indicates that increasing the number of parallel processors may harm the performance of traditional parallel schemes that have been proposed for SDDP.

The chapter is organized as follows. In section 2.2 we describe our proposed parallelization schemes for SDDP. In section 2.3 we present numerical case studies which constitute the basis for our empirical observations. Finally, in section 2.4 we summarize our conclusions and outline future directions of research that are inspired by this work.

2.2 Parallel Schemes For SDDP

We begin this section by presenting parallel strategies for SDDP and then proceed to explain how these strategies can be implemented in a synchronous and asynchronous setting. These schemes span the different strategies that have been proposed in the literature [dSF03, PBM13, HB15, DK21, MDBB21] and some new schemes that, to the best of our knowledge, have not yet been considered.

2.2.1 Parallel computing attributes for SDDP

The SDDP algorithm presents various opportunities for parallelization. The main bottleneck within SDDP appears when building the cuts to approximate the value functions of each stage. Fortunately, such a calculation can be highly parallelized. On the one hand, the value function is composed of several cuts, thus one can envision strategies that distribute the cut calculation among cores. Furthermore, the work required to calculate each cut can be parallelized as well. On the other hand, each stage carries an associated value function approximation. The value function approximation related to different stages can be parallelized as well.

These different parallel computing strategies for SDDP can be combined within common high performance computing paradigms, such as synchronous and asynchronous computing, or distributed and shared memory configurations. These different flavors of parallelism open the path for a rich set of parallel computing configurations which we intend to explore in the present chapter.

2.2.2 Parallelizing by Scenario and by Node

Parallelizing by Scenario (PS)

We start by defining the notion of scenario, which is a trajectory of the stochastic process from the beginning to the end of the time horizon. As we demonstrate graphically in panel (a) of Figure 2.1, in this approach, each processor generates a cut that supports the expected value cost-to-go functions at different sample points of the state space. The forward and backward steps are executed as follows:

- Forward pass: The forward pass consists of N Monte Carlo scenarios. Each processor computes a different scenario, thus producing trial points $x_1^n, \cdots, x_T^n$ in the state space, for $n = 1, \cdots, N$.
- Backward pass: At stage t, the n-th processor generates a cut for the expected cost-to-go functions of stage $t-1$ at point x_{t-1}^n.

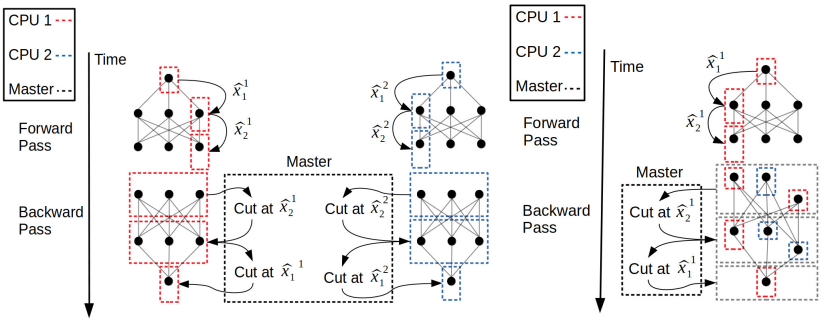
The Parallel Scenario approach appears to be the most common parallelization strategy for SDDP in the literature. Different variants have been proposed, ranging from synchronous schemes [dSF03, PBM13] to relaxations in the synchronization points [HB15], to asynchronous schemes [DK21].

Parallelizing by Node (PN)

In panel (b) of Figure 2.1 we can observe that, as opposed to the PS strategy, the idea in PN strategies is to use the available processors in order to generate a single cut at a single trial point. The forward and backward steps are executed as follows:

- Forward Pass: The main processor computes trial points along a single scenario. This produces a sequence $x_1, \cdots x_T$. Note that there is no parallelization at this step.
- Backward Pass: Moving backwards in time through the lattice, each processor selects a node of the lattice that has not yet been updated and solves the corresponding subproblem.

A competitive implementation for this scheme is unfortunately limited to a shared memory setting. This is due to the fact that, when a CPU commences a task in a distributed memory setting, there is a non-negligible communication startup time involved with receiving the required data for commencing the task. This implies that the task executed by each processor must require significantly more time than this start up time if parallelism is to deliver benefits, otherwise the latency of the network becomes an important factor in slowing down the algorithm. In the PN scheme, at stage t, each processor withdraws a subproblem from the list of $|\Omega_t|$ problems and proceeds to solve it. In a



(a) Parallelizing by Scenario (b) Parallelizing by Node

Figure 2.1: Representation of SDDP parallel schemes. The height of the red and blue dashed boxes represents the elapsed time. Panel (a) presents the parallel scenario scheme. At each iteration a cut is built at 2 different points, and each cut is computed by a different CPU. Panel (b) presents the parallel node scheme. The grey dashed boxes represent outcomes that belong to the same time stage. At each iteration, a cut is computed at a single point of the state space. The work that is required for computing such a cut is distributed among the available CPUs.

distributed memory setting, this solve time is comparable to the startup time of the processor. Thus, the latency of the network becomes problematic.

In [MDBB21] a similar approach is followed, where a single scenario is considered and the work required to compute the scenario is distributed among the workers. However, the authors consider a different scheme to distribute the nodes among the processors. They allow processors to be attached to a stage. The processors are then constantly generating cuts for the given stage.

2.2.3 Synchronous and Asynchronous Computing

As is commonly the case in parallel computing algorithms [BT89], the interaction between processors in our proposed schemes can unfold synchronously or asynchronously. In what follows, we propose synchronous and asynchronous schemes for both the parallel scenario (PS) and parallel node (PN) versions of SDDP. This leads to a variety of algorithms, which are summarized in table 2.1.

Table 2.1: SDDP schemes compared in this chapter.

	Synchronous	Asynchronous
PS	Each processor produces a cut, processors have access to all cuts [dSF03, PBM13] [HB15]	Each processor produces a cut, processors may not have access to all cuts [DK21]
PN	Each processor computes a node, processors have access to the solution of all nodes	Each processor computes a node, processors may not have access to the solution of all nodes [MDBB21]

Note that no reference is attached to the synchronous PN scheme. To the best of our knowledge, there such a parallel scheme has not been proposed previously in the literature.

Synchronous Parallel Scenario (Sync PS)

As we discuss in subsection 2.2.2, in the PS scheme each processor builds a cut. The difference between the synchronous and asynchronous version of the algorithm is how these cuts are exchanged between processors. Given N processors, the forward and backward procedures for the synchronous PS scheme are described in pseudo code in Algorithm 6.

In panel (a) of Figure 2.2 we present the evolution of the algorithm over a lattice. In the forward pass, the processors compute a scenario and synchronize at the end of the forward pass. In the backward pass, at stage 3, both processors compute a cut which is shared in order to approximate the expected value cost to go functions. Because of the synchronization, both processors must wait until receiving the cut of the other processor. If one processor is faster when computing a cut, then it must stay idle until all other processors have computed their cut. Note that, apart from the synchronization at the end of each stage during the backward pass, synchronization also occurs at the end of stage 1. Consequently, in the next iteration, all processors commence with the same set

of cuts.

Algorithm 6: SDDP Sync PS

INPUT: Provide a lower bound for θ_{ξ_t} for $t = 1, \ldots, T-1$, $\xi_t \in \Omega_t$, and maximum number of iterations K.
OUTPUT: A cutting-plane approximation $\{c^i_{\xi_t}(x_t)\}_{i=1}^K$ for $t = 1, \ldots, T$, $\xi_t \in \Omega_t$.

for $i = 1, \cdots, K$.

1. **Forward Pass**.

   (1.1) The n-th processor computes a Monte Carlo scenario, thus obtaining a sequence $\xi_1^n, \cdots, \xi_T^n$, where $\xi_t^n \in \Omega_t$.

   (1.2) **for** $t = 1, \cdots, T$:

   Solve the linear problem associated to $\widehat{V}_t(\widehat{x}_{t-1}^n, \xi_t^n)$ and store the trial action $\widehat{x}_t^n$.

   At the end of this step, the processors synchronize[a]

2. **Backward Pass**.

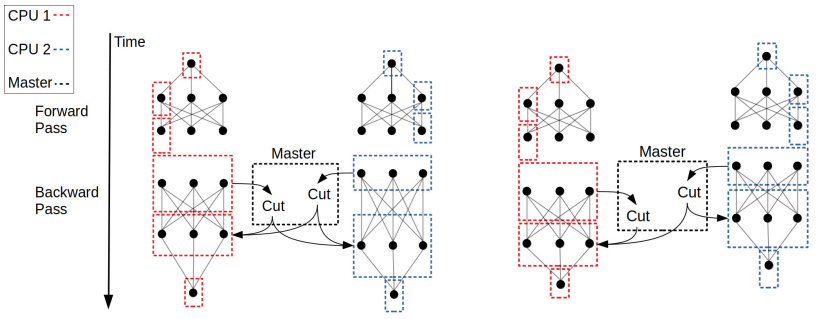
   The n-th processor solves:

   (2.1) **for** $t = T, \ldots, 2$:

   (2.1.1) **for** $\xi_t \in \Omega_t$: Solve the linear problem associated to $\widehat{V}_t(\widehat{x}_{t-1}^n, \xi_t)$ and store the dual multipliers.

   (2.2.2) The multipliers are used for computing a cut. The processors synchronize, and the expected value cost-to-go function is updated with the gathered cuts.

[a] The processors could in fact start as soon as possible. Nevertheless, since the forward pass represents a small part of the computational effort, the algorithm is implemented as described here.



(a) Sync PS (b) Async PS

Figure 2.2: Synchronous and Asynchronous Parallel Scenario Schemes. The height of the red and blue dashed boxes represents the elapsed time.

The synchronous version appears in a number of publications [dSF03, PBM13]. In [HB15] the authors propose a relaxation in the synchronization points of each stage of the backward pass, whereby a processor waits for a subset of processors. This work discusses empirical evidence that indicate benefits relative to a fully synchronous version.

Asynchronous Parallel Scenario (Async PS)

The difference between Async PS and the synchronous version is that processors do not wait for cuts that have not been computed yet. The algorithm is described in Algorithm 7.

As we can observe in panel (b) of Figure 2.2, in the forward pass each processor computes a sample and proceeds immediately to the backward pass. In the backward pass, once a processor computes a cut, this cut is shared with the master process. The processor then asks for available cuts and proceeds without waiting for cuts that have not been computed yet. For instance, at stage 3, the blue processor computes a cut faster than the red processor, sends the cut and asks if the cut provided by the red processor is already available. Since the red processor has not finished its job, the blue processor proceeds to stage 2 without waiting for the cut provided by the red processor. On the other hand, once the red processor finishes stage 3, it will receive the cut provided by the blue processor. A disadvantage of this scheme is that, since every processor operates with a different set of cuts, it is not clear how to estimate an upper bound. In section 2.3 we discuss how the convergence evolution is measured.

In [DK21] the authors follow the aforementioned asynchronous strategy, nevertheless no evidence of its benefits is developed in detail.

Algorithm 7: SDDP Async PS

INPUT: Provide a lower bound for θ_{ξ_t} for $t = 1, \ldots, T-1$, $\xi_t \in \Omega_t$, and maximum number of iterations K.
OUTPUT: A cutting-plane approximation $\{c^i_{\xi_t}(x_t)\}_{i=1}^K$ for $t = 1, \ldots, T$, $\xi_t \in \Omega_t$.

for $i = 1, \cdots, K$.

1. **Forward Pass**. The n-th processor performs the same steps as in the synchronous setting, the difference is that there is no synchronization.
2. **Backward Pass**.

   The n-th processor solves:

   (2.1) **for** $t = T, \ldots, 2$:

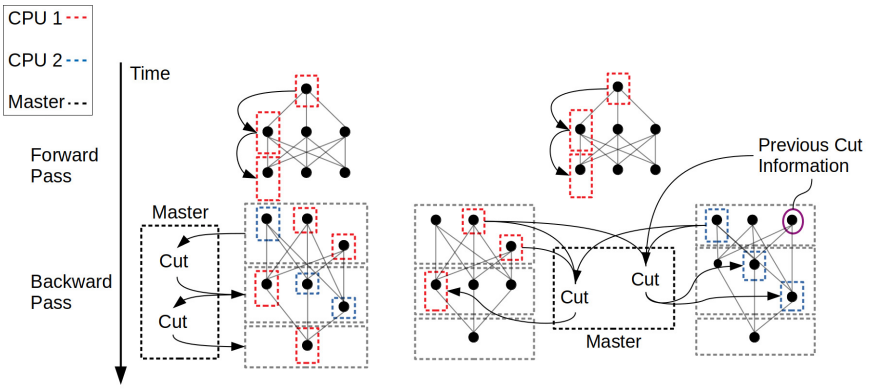
   (2.1.1) **for** $\xi_t \in \Omega_t$: Solve the linear problem associated to $\widehat{V}_t(\widehat{x}^n_{t-1}, \xi_t)$ and store the dual multipliers.

   (2.2.2) The processor asks for available cuts, and updates the expected value cost-to-go function with available cuts.

Synchronous Parallel Node (Sync PN)

In section 2.2.2 we present the PN scheme, according to which different processors are allocated to different nodes of the lattice for a given stage. The synchronous and asynchronous schemes then differ on whether a processor waits for the nodes computed by other processors. The forward and backward passes are presented in Algorithm 8.

In Figure 2.3, panel (a), we present this process graphically over a lattice. Note that, in stage 3, the blue processor solves the subproblem associated with the first node, while the red processor solves the subproblem associated with the second node. The red processor finishes first and proceeds with the third node. Note that, once the blue processor finishes, it must stay idle as there are no more nodes available for that stage. The solution information of all the nodes is then used in order to compute a cut, which is then transmitted to stage 2. Note that, before passing to stage 2, all the subproblems of the third stage must be solved.



(a) Sync PN (b) Async PN

Figure 2.3: Synchronous and asynchronous parallel node schemes. The height of the red and blue dashed boxes represents the elapsed time. The dashed grey box represents outcomes that belong to the same stage.

Algorithm 8: SDDP Sync PN

INPUT: Provide a lower bound for θ_{ξ_t} for $t = 1, \ldots, T-1$, $\xi_t \in \Omega_t$, and maximum number of iterations K.
OUTPUT: A cutting-plane approximation $\{c^i_{\xi_t}(x_t)\}_{i=1}^K$ for $t = 1, \ldots, T$, $\xi_t \in \Omega_t$.

for $i = 1, \cdots, K$.

1. **Forward Pass**.

   (1.1) The main process computes a single Monte Carlo scenario, thus obtaining a sequence $\xi_1, \cdots, \xi_T$, where $\xi_t \in \Omega_t$.

   (1.2) **for** $t = 1, \cdots, T$:

   Solve the linear problem associated to $\widehat{V}_t(\widehat{x}_{t-1}, \xi_t)$ and store $\widehat{x}_t$.

   Note that there is no parallelization in this step.

2. **Backward Pass**.

   (2.1) **for** $t = T, \ldots, 2$:

   (2.2.1) There is a list of $|\Omega_t|$ problems. The n-th processor selects a $\xi_t \in \Omega_t$ that has not been selected yet, and solves the linear problem associated to $\widehat{V}_t(\widehat{x}_{t-1}, \xi_t)$. The dual multiplier is stored.

   (2.2.2) The processors synchronize, and the multipliers are collected and used for building the cut.

   (2.2.3) The expected value cost-to-go function is updated with the generated cut.

Asynchronous Parallel Node (Async PN)

In contrast to the synchronous version, in the asynchronous version the processors do not wait for nodes that have not been solved. The procedures in the backward and forward passes can be described in Algorithm 9.

This process is presented graphically in panel (b) of Figure 2.3. At stage 3, the blue processor solves the subproblem associated to the first node. When it completes its computation, there are no more subproblems available for that stage, since the red processor has already solved the second node and is now working on the third node. Then the blue processor starts processing the nodes of the second stage. However, in order to compute a cut for stage 2, without having access to the solution information from the subproblem of node 3, the processor uses the subproblem information of node 3 of the cut obtained in the previous iteration. The blue processor is thus able to build a cut, and can start processing the nodes of stage 2.

The following lemma shows that the proposed cuts are valid. The proof can be found in the appendix.

Lemma 2.1. *The cuts built in the Async PN scheme are valid cuts.*

Following a similar argument as the one presented in [PG08], we can show that, after a finite number of iterations, no new cuts will be added. The proof can be found in the appendix.

Lemma 2.2. *Let $\mathcal{G}_k^{t,\omega}$ be the set of cuts at stage t, node ω and iteration k. There exists $m_{t,\omega}$ such that $|\mathcal{G}_k^{t,\omega}| \leq m_{t,\omega}$ for all k, $1 \leq t \leq T-1$.*

It is worth mentioning that although convergence after finitely many iterations is ensured, it may not be to the optimal value. However, as the value function is a lower approximation, we can always ensure that the convergence will be an under-estimation. The reason is two fold. As described in [PG08] to guarantee convergence to the optimal value we require three properties (i) the cut generation property, (ii) the backward pass sampling property and (iii) the forward pass sampling property. The proposed approach does not satisfy properties (i) and (ii). Property (i) states that missing information is completed by using the previous' iterations dual multipliers that maximize the cut at the current trial point. In our case missing information is completed by previous iteration dual multiplier. Property (ii) states that each node, during the backward pass, is visited infinitely many times. Our approach doesn't include a scheduling scheme that allows each processor to visit infinitely many times each node during the backward pass.

Following the cut generation property, as in [PG08], we have also tested the approach that uses the dual multiplier that maximizes the cut at the current trial point, however no considerable difference is observed. In practice, the considered test cases have shown that Async PN presents a convergence behaviour comparable to the one obtained by the other schemes that are implemented in the chapter.

As stated previously, the authors in [MDBB21] consider a variant in the distribution of the nodes among the processors. Instead of distributing the nodes at each stage, the processors are attached to the nodes of a fixed stage. The authors propose an asynchronous version. The results of the authors vary. In certain instances, such an approach exhibits superior performance relative to a synchronous PS implementation. In other instances, the performance of the proposed method is comparable to a synchronous PS implementation.

Algorithm 9: SDDP Async PN

INPUT: Provide a lower bound for θ_{ξ_t} for $t = 1, \ldots, T-1$, $\xi_t \in \Omega_t$, and maximum number of iterations K.
OUTPUT: A cutting-plane approximation $\{c^i_{\xi_t}(x_t)\}_{i=1}^K$ for $t = 1, \ldots, T$, $\xi_t \in \Omega_t$.

for $i = 1, \cdots, K$.

1. **Forward Pass**. The same process as in the synchronous PN setting is executed. There is no parallelization in this step.

2. **Backward Pass**.

   (2.1) **for** $t = T, \ldots, 2$:

   (2.2.1) There is a list of $|\Omega_t|$ problems. The n-th processor selects a $\xi_t \in \Omega_t$ that has not been selected yet, and solves the linear problem associated to $\widehat{V}_t(\widehat{x}_{t-1}, \xi_t)$. The dual multiplier is stored.

   (2.2.2) If there are no more subproblems available, the available multipliers are collected. The multipliers of the previous iteration are used for a subproblem that has not been computed yet.

   (2.2.3) The cut is computed and the expected value cost-to-go functions are updated.

2.3 Case Studies

In this section we present results for an instance of the Brazilian hydrothermal scheduling problem. The results are further complemented with an inventory control problem. Our experimental results can be summarized as follows.

i Asynchronous computing is not helpful for achieving tight optimality gaps faster. Nevertheless, in certain cases, there is a temporary advantage in the asynchronous PS scheme in early stages of the execution of the algorithm.

ii The PN scheme performs better than the PS scheme during early stages of the execution of the algorithm.

iii The PS scheme scales poorly when increasing the number of Monte Carlo samples.

iv The PN scheme exhibits desirable parallel efficiency properties, nevertheless a competitive implementation is limited to a shared memory setting.

We proceed by briefly introducing the test cases that we analyze in this work. The models are presented in further detail in the appendix.

Hydrothermal Scheduling Problem: The Brazilian interconnected power system is a multistage stochastic programming problem that has been analyzed extensively in the literature due to its practical relevance [PP91, STdCS13, PBM13, DMPF15, LS19]. The Brazilian power systems comprises, as of 2010, more than 200 power plants. Among these, 141 are hydro units.

The objective of the problem is to determine optimal operation policies for power plants, while minimizing operation costs and satisfying demand. Representing the 141 hydro plants as well as their associated inflows results in a high-dimensional dynamic problem. In order to tackle this problem, the literature typically separates it into long-term, medium-term and short-term operational planning. The value functions obtained in the long-term operational planning problem are used as input for medium-term planning. The value functions from medium-term planning are then used, in turn, as input for the short-term operational planning problem. The SDDP algorithm is applied in the long-term operational planning problem. The problem is simplified by aggregating reservoirs into equivalent energy reservoirs [AR70]. The literature typically considers four energy equivalent reservoirs for this problem instance: North, Northeast, Southeast, South and a Transshipment node. The Transshipment node has no loads or production. The system is presented in Figure 2.4.

The problem aims at satisfying the demand at each node by using the hydro and thermal power of that node, as well as power that is imported from other nodes. However, there is a limit in the power that can flow trough the

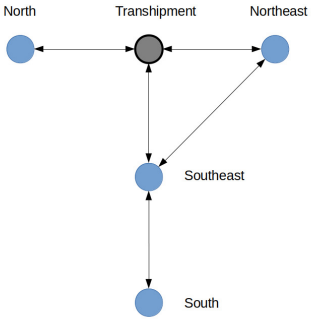


Figure 2.4: Brazilian hydrothermal test case - equivalent reservoirs

transmission lines of the electricity network. In the literature, the problem is typically solved for a 60-month planning period. However, in order to represent the continuation of operations at the end of the planning horizon, 60 additional months are considered. Following [LS19], the uncertainty model is represented by considering a Markov chain, the so-called MC-SDDP, thus leading to an uncertainty lattice. This leads to a multi-stage stochastic program in 4 dimensions and 120 stages. We consider a setting with 100 nodes per stage.

Inventory Control Problem: We model a stochastic inventory problem with Markovian demand. The objective of the problem is to maximize expected profits by placing optimal order quantities x_{tn} for products $n \in \mathcal{N}$ over periods $t \in H$. Demand is satisfied from on-hand inventory $v_{t-1,n}$ by selling a quantity s_{tn} of each product. Any excess demand is considered as being lost. We consider a case with 10 products, which is the dimension of the random vector. The problem horizon is equal to 10 stages, with 100 nodes per stage.

2.3.1 Experimental Results

The computational work is performed on the Lemaitre3 cluster, which is hosted at the Consortium des Equipements de Calcul Intensif (CECI). It comprises 80 compute nodes with two 12-core Intel SkyLake 5118 processors at 2.3 GHz and 95 GB of RAM (3970MB/core), interconnected with an OmniPath network (OPA-56Gbps). The algorithms are implemented in Julia v0.6 [BEKS17] and JuMP v0.18 [DHL17]. The chosen linear programming solver is Gurobi 8.

Synchronous and Asynchronous Computation

Figure 2.5 presents the evolution of the optimality gap for all algorithms against run time. The algorithms are run with 20 CPUs. Obtaining a reliable upper bound at each point in time can be very time consuming. Thus, providing a

reliable gap evolution can be time-consuming as well. Moreover, estimating an upper bound for the Async PS scheme is difficult, since there is a policy that is evolving differently on each CPU. Therefore, in order to provide a fair comparison between the different schemes, we have pre-calculated a best available lower bound and compared the lower bound evolution of all the algorithms against this best known solution. Concretely, the gap is measured as the relative difference between the lower bound evolution of the algorithm and the "best available lower bound" L that we are able to compute for the problem. What we refer to as the "best available lower bound" L is a lower bound that corresponds to a high-quality policy. The way in which we verify a high-quality policy is by verifying that the relative difference between the upper bound estimate of said policy and L is below 1%. The upper bound estimate for this policy is calculated with a sufficient number of samples so as to ensure a 1% difference between the performance of this policy and L with a confidence of 95%. Once this best lower bound L is calculated, the reported gaps are calculated as follows: $(L - L_t) \cdot 100/L$, where L_t is the lower bound calculated as each algorithm progresses. More specifically, for the PN schemes and the Sync PS scheme, L_t is the lower bound at the end of iteration t. For the Async PS scheme, each CPU is performing its own SDDP run and sharing cuts whenever they are available. Namely, on each CPU the policy is evolving differently. Therefore, L_t corresponds to the lower bound at the end of iteration t of the fastest CPU.

1. Parallelizing by Scenario: For the inventory test case, the asynchronous schemes tend to perform better during early iterations, as indicated in panels (a) and (b) of Figure 2.5. Instead, for the hydrothermal test case there is no considerable difference, see panels (c) and (d) of Figure 2.5. The difference in the behaviour between both test cases can be explained as follows. As pointed out in [DMPF15], the expected value cost-to-go function approximations tend to be myopic at early iterations, when the gap is high. This implies that the trial points and the cuts obtained when there is a high gap tend to produce low-quality information. Therefore, the following possibilities can occur:

    - When the algorithm struggles to decrease the gap during early iterations, the synchronous version tends to perform poorly. This is due to the fact that the processors wait for the generation of loose cuts. Instead, an asynchronous version benefits from the fact that the fastest processor is not waiting for these low-quality cuts.
    - On the other hand, when the algorithm manages to reduce the optimality gap during early iterations, the disadvantage of the synchronous version diminishes. This is due to the fact that, since the gap reduces quickly, the value functions are of good quality. Consequently, the synchronous version will wait for useful information.

    The inventory test case corresponds to the former case, whereas the

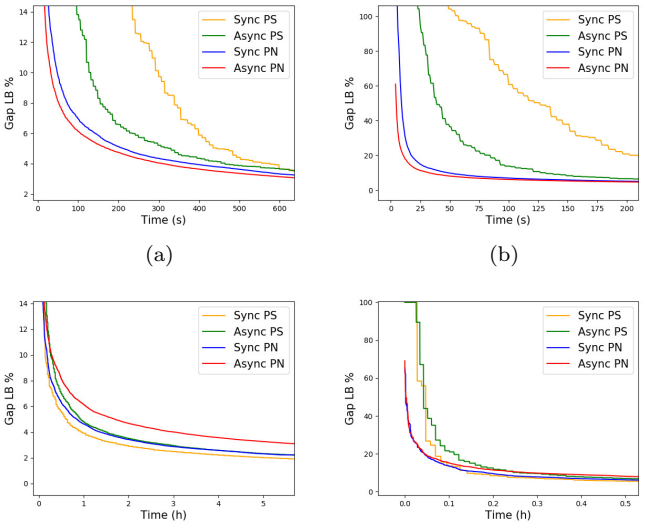


Figure 2.5: Comparison of algorithms. Panels (a) and (b) present the evolution of the optimality gap against time for the inventory test case. Panels (c) and (d) present the evolution of the optimality gap for the hydrothermal problem. Panels (a) and (c) show the gap evolution throughout the entire execution time, with emphasis on presenting the differences when the gap is low. Panels (b) and (d) present a close up at the beginning of the run time, emphasizing the differences between the PS and PN schemes at the early steps of execution.

Brazilian hydrothermal test case corresponds to the latter. Panel (a) of Figure 2.6 demonstrates that, after 500 scenarios, the gap for the PS methods is above 100% for the inventory test case. Instead, the same number of scenarios analyzed results in a gap below 20% for the hydrothermal test case, as we can observe in panel (c) of Figure 2.6. Moreover, as we can observe in panel (b) of Figure 2.6, the PS scheme is able to process more scenarios compared to the PN scheme for the inventory test case. Nevertheless, the PS gap is worse, thus supporting the observation that the value functions computed during early steps of the algorithm are poorly approximated.

Despite these observations, we note that there is no significant difference between the synchronous or asynchronous schemes when aiming for tight optimality gaps. This can be observed in Figure 2.5. When we target tight gaps, many additional scenarios are required, as we can observe in panels (a) and (c) of Figure 2.6. Unfortunately, Async PS is not able to visit many more scenarios than the synchronous counterpart, see panels (b) and (d) of Figure 2.6. As a consequence, although Async PS may

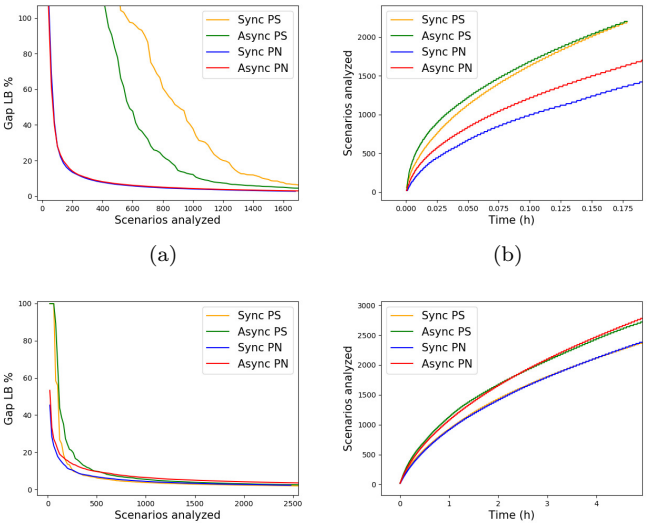
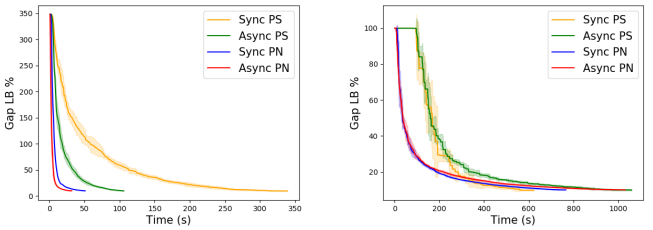


Figure 2.6: Scenarios Analyzed by each method. The scenarios analyzed refers to the number of scenarios visited during the training process for each method. The first row corresponds to the inventory test case, the second row corresponds to the hydrothermal test case. Panels (a) and (c) present the evolution of the optimality gap against the number of scenarios analyzed. Panels (b) and (d) present the number of scenarios analyzed against time.

reduce the optimality gap faster during early iterations, both synchronous and asynchronous schemes achieve similar performance after a significant amount of computation time has elapsed.

2. Parallelizing by Node: As seen in panels (b) and (d) of Figure 2.6 the asynchronous version is able to process more scenarios as compared to the synchronous counterpart. Nevertheless, the main observation of Figure 2.5 is that there is no significant benefit in an asynchronous implementation for parallelizing by node. Both the synchronous and asynchronous parallel node algorithms are attaining comparable performance in terms of gap throughout the entire course of the execution of the algorithms.

In order to evaluate the reproducibility of the results under different runs, 5 repetitions are performed for each method. We use Student's t-distribution to construct 95% confidence intervals. The results are presented in Figure 2.7. As we can observe, the convergence trend of each algorithm remains.



(a) Inventory Test Case (b) Hydrothermal Test Case

Figure 2.7: Reproducibility of the results

Parallel Node versus Parallel Scenario

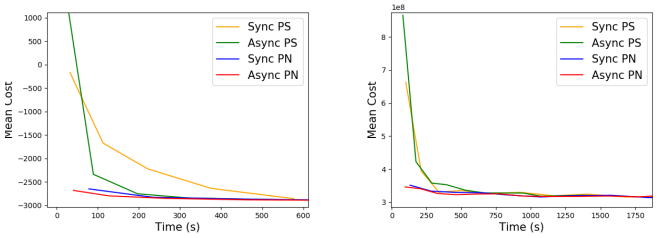
Interestingly, as we can observe in panels (b) and (d) of Figure 2.5, the PN strategy is behaving much better than the PS strategy during early iterations. The reason is that, as we discuss previously, the value function approximations tend to be poor in early iterations [DMPF15]. Thus, at each iteration, the PS version generates several cuts that are loose approximations of the value functions, whereas the PN setting generates a single cut. Nevertheless, if the goal is to obtain tight gaps, the difference is not significant. The reason is that, once value functions of better quality have been obtained by either algorithm, there is a benefit of visiting more than one scenario per iteration. This works in favor of the PS setting.

In addition to the experiments shown, an out-of-sample estimation was performed. Concretely, for every x visited scenarios (i.e. for every x scenarios that are used for training) we perform an out-of-sample simulation by considering 2000 scenarios that are not used during the training process. For the inventory test case, the out-of-sample simulation is performed every 400 scenarios, while for the hydrothermal test case the out-of-sample is performed every 60 scenarios. We chose to select a finer granularity for the hydrothermal test case because the PS policies improve faster in terms of visited scenarios for this test case. The results are presented in Figure 2.8.

From the performed experiments, one can observe that the PN methods tend to perform better during early iterations. However, there is a point in which both PN and PS attain comparable performance, as our previous experiments also demonstrate. We also observe that, for the inventory test case, the Async PS scheme behaves considerably better than the Sync PS counterpart, which our previous experiments have also demonstrated.

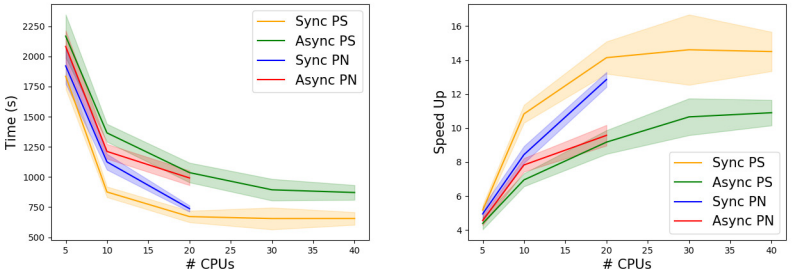
Scalability of Parallel Scenario

Panel (a) of Figures 2.9, 2.10 presents the evolution of the PS algorithms when increasing the number of CPUs. Figure 2.9 corresponds to the hydrothermal test case, and Figure 2.10 corresponds to the inventory test case. The target



(a) Inventory Test Case (b) Hydrothermal Test Case

Figure 2.8: Out Of Sample Simulation. After every x scenarios used for training, an out-of-sample upper bound estimate is performed. For the Inventory test case, x is chosen to be 400 scenarios. For the hydrothermal test case, x is chosen to be 60 scenarios.



(a) (b)

Figure 2.9: Hydrothermal test case - CPU scalability.

gap for terminating the algorithms is set to 10%. As in the previous experiments, the gap refers to the relative difference between the lower bound and the best available solution. The algorithms present an inherent uncertainty due to Monte Carlo sampling in the forward pass. Consequently, the run time itself is random. Therefore, for each CPU count we conduct the experiment 5 times, in order to construct 95% confidence intervals, which are based on Student's t-distribution. Panels (a) and (b) of Figures 2.10 and 2.9 demonstrate that an initial increase in the number of CPUs results in a notable performance improvement. Nevertheless, there is a point at which this trend is reversed. This is especially true for the inventory test case. We arrive to the same observation when reporting the speedup[1] of the algorithms. Note, in panel (b) of the Figures, that there is a point beyond which the speedup decreases. This is due to the fact that, as more CPUs are introduced, more samples are introduced per iteration. As we have argued, the expected value cost-to-go functions are my-

[1]The speedup is defined as the ratio between a serial run (i.e. a run using a single CPU) and a parallel run using X CPUs.

opic at early steps. Consequently, more samples tend to introduce information that is not entirely useful for the algorithm, more so in early iterations. This results in a slowdown of the PS algorithm.

In order to tackle the poor quality of cuts that is produced as a result of the myopic value function that is found at early steps, the cut selection methodology proposed in [LWM13] is implemented for the Sync PS algorithm. Our choice to focus on the Sync PS algorithm is motivated by the fact that it suffers the most due to the aforementioned effect. This cut selection technique rejects a cut, calculated at point x_t, if the value function is not improved by some $\epsilon > 0$ when adding the cut at point x_t. The strategy has been applied to the inventory test case, which is the case study for which the performance deteriorates the most. As one can observe in panel (c) of Figure 2.10, we observe that introducing such a cut selection method has a damping effect: the issue is diminished but is not solved. The cut selection technique helps by not adding some non-useful cuts to the linear programs, nevertheless computational resources are wasted as several scenarios end up building non-useful cuts.

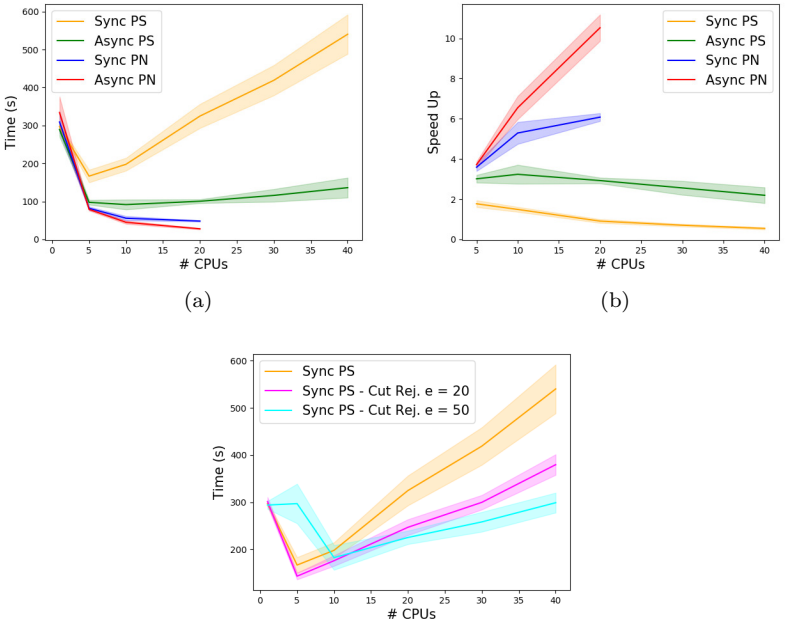


Figure 2.10: CPU scalability for the inventory test case.

Parallel Node Scalability

Panel (a) of Figures 2.9, 2.10 presents the performance when increasing the number of CPUs for the parallel node setting. As in the previous case, the target optimality gap is set to 10% and 95% confidence intervals are presented. Note that the performance of the algorithm improves with additional CPUs. Since the algorithm is building one cut per iteration, as more CPUs are introduced, that single cut is computed faster. This is an important difference as compared to the PS scheme, where the speedup can decrease.

Although this scheme exhibits favorable scalability behavior, it is limited to a shared memory setting. As stated in subsection 3.1.2, the reason behind this limitation is that the solve time of a subproblem is comparable to the startup cost of a worker in a distributed memory setting. Concretely, after some hundreds of iterations the solve time per subproblem for both test cases ranges in the order of a few milliseconds, a time which is comparable to the startup cost which is about 3 milliseconds. Therefore, the latency of the network becomes an issue.

2.4 Conclusions

In this chapter we propose a family of parallel schemes for SDDP. These schemes are differentiated along two dimensions: (i) using parallel processors in order to distribute computation per Monte Carlo sample of the forward pass (per scenario) or per node of the lattice at every stage of the problem (per node); (ii) implementing the exchange of information among processors in a synchronous or asynchronous fashion. We compare the performance of these algorithms in two case studies: (i) an inventory management problem, and (ii) an instance of the Brazilian hydrothermal scheduling problem. The case studies deliver consistent messages, which we summarize below in the form of four conclusions.

(i) Asynchronous computing is not helpful in the studied experiments, when the goal is to achieve a tight optimality gap in a shorter time. Asynchronous schemes, on the other hand, may be beneficial at early stages of the Parallel Scenario strategy. (ii) We have proposed a Parallel Node strategy for SDDP, which performs better at early iterations than the traditional parallel scheme for SDDP. (iii) Parallel schemes that are based on increasing the number of scenarios which are processed during the forward pass may not scale well with extra CPUs. This is the case for the PS scheme. (iv) The Parallel Node strategy presents desirable CPU scalability properties, but only in a shared memory setting.

In our work we have indicated some of the weaknesses in the parallelization of SDDP. On the one hand, we have empirically demonstrated scalability issues with the commonly proposed parallel scenario scheme. On the other hand, We have proposed a new set of parallel schemes, the parallel node configurations, with better scalability properties. These proposed algorithms are restricted to a shared memory setting, thus limiting the amount of CPUs available for use.

However, recent advances of message passing in distributed computing infrastructures open the path for future developments of parallel node configurations in such infrastructures. These scalability issues motivate future research into scalable parallel SDDP schemes based on backward dynamic programming. These schemes, which present desirable scalability properties, are closer to the per-node strategy in the sense that a pool of fixed amount of work is distributed among processors, but are also implementable in distributed memory settings, such as high performance computing clusters.

Interestingly, we have observed that the different configurations tend to behave similarly after a certain amount of time has elapsed, in the sense that the different optimality gaps are close and reach a steady-state behaviour. This observation motivates the search for algorithmic configurations which focus on the optimality gaps rather than on speeding up the SDDP iterations.

2.A Hydrothermal Scheduling Problem

We use a transportation model to approximate the operation of the transmission network. The cost to go function $V_t(v_{t-1}, \xi_t)$ can then be computed by solving the following problem:

$$
\begin{aligned}
\min \; & \sum_{i \in \mathcal{G}} M_i \cdot g_{t,i} + \text{VOLL} \sum_{n \in \mathcal{N}} ls_{t,n} + \mathcal{V}_{t+1}(v_t, \xi_t) \\
s.t. \; & v_{t,n} = v_{t-1,n} + A_{t,n}(\xi_t) - q_{t,n} - s_{t,n} && n \in \mathcal{N} \\
& q_{t,n} + \sum_{i \in \mathcal{G}} g_{t,i} + \sum_{i \in \mathcal{F}_n} f_{t,i} = L_{t,n} && n \in \mathcal{N} \\
& g_t \leq \bar{G} \\
& v_t \leq \bar{V} \\
& q_t \leq \bar{Q} \\
& f_t \leq \bar{F} \\
& g_t, ls_t, v_t, q_t, s_t, f_t \geq 0
\end{aligned}
$$

The variables can be described as follows:

v_t: The state variable vector, which represents the stored energy of the equivalent reservoir.

q_t, s_t : Decision variables which represent the generated hydro energy and the spillage, respectively.

ls_{tn} : Decision variable which represents load shedding.

g_t: The vector of generated power from thermal plant $i \in \mathcal{G}$.

The parameters can be described as follows:

M_i: The generation cost of thermal plan $i \in \mathcal{G}$.

$VOLL$: The value of lost load.

P_i: The hydro generation coefficient of hydro plant $i \in \mathcal{H}$.

A_t: The inflow vector.

L_t: Load at stage t.

$\bar{G}, \bar{V}, \bar{Q}, \bar{F}$: Physical upper limits on the variables.

Further details for the Brazilian hydrothermal model, and the description of the data, can be found in [STdCS13].

2.B Inventory Control Problem

The inventory control problem can be modeled as a multistage stochastic problem. The uncertainty in the problem is due to stochastic demand, which is assumed to follow a Markov Chain. The cost-to-go function $V_t(v_{t-1}, \xi_t)$ is computed by solving the following problem:

$$
\begin{aligned}
\max \ & \sum_{n \in \mathcal{N}} P \cdot s_{t,n} - HC \cdot v_{t,n} - PC \cdot x_{t,n} + \mathcal{V}_{t+1}(v_t, \xi_t) & \\
s.t. \ & v_{t,n} = v_{t-1,n} + x_{t,n} - s_{t,n} & n \in \mathcal{N} \\
& s_{t,n} \leq v_{t-1,n} & n \in \mathcal{N} \\
& s_{t,n} \leq D_{t,n}(\xi_t) & n \in \mathcal{N} \\
& v_{t,n} \leq C & n \in \mathcal{N} \\
& v_{t,n}, s_{t,n}, x_{t,n} \geq 0 & n \in \mathcal{N}
\end{aligned}
$$

The variables are given as follows:

$v_{t,n}$: The state variable, which represents the on-hand inventory for product $n \in \mathcal{N}$.

$s_{t,n}$: Variable representing the amount of sold items for product $n \in \mathcal{N}$.

$x_{t,n}$: Variable representing the ordered quantities for product $n \in \mathcal{N}$.

The parameters can be described as follows:

P: The sales price.

HC: The inventory holding cost.

PC: the purchase cost.

$D_{t,n}$: The demand for product $n \in \mathcal{N}$.

C: The inventory capacity.

The problem is set up for 10 products. This is the dimension of the random vector. The time horizon is equal to 10 stages. We consider 100 nodes per stage.

2.C Convergence of the Async PN Scheme

The proposed strategy to build cuts for the Async PN scheme is a valid strategy, as the following lemma shows.

Lemma 2.1. *The cuts built in the Async PN scheme are valid cuts.*

Proof. Let $\widehat{x}_{t-1}^k$ be the obtained trial point for stage $t-1$ at iteration k. Consider the cut for the cost-to-go function $V_t(x_{t-1}, \xi_t)$, which is given by

$$V_t(x_{t-1}, \xi_t) \geq \alpha_{\xi_t,t}^k + \beta_{\xi_t,t}^k \cdot x_{t-1} \tag{2.1}$$

The expected cost-to-go function satisfies

$$\mathcal{V}_t(x_{t-1}, \xi_{t-1}) = \sum_{\xi_t \in \Omega_t} V_t(x_{t-1}, \xi_t) \cdot P(\xi_t | \xi_{t-1}) \tag{2.2}$$

Then, a cut for $\mathcal{V}_t(x_{t-1}, \xi_{t-1})$ is given by

$$\mathcal{V}_t(x_{t-1}, \xi_{t-1}) \geq \sum_{\xi_t \in \Omega_t} \alpha_{\xi_t,t}^k \cdot P(\xi_t | \xi_{t-1}) + \sum_{\xi_t \in \Omega_t} \beta_{\xi_t,t}^k \cdot P(\xi_t | \xi_{t-1}) \cdot x_{t-1} \tag{2.3}$$

Let us now build a cut for the expected cost-to-go function at iteration $k+1$ using incomplete information. Assume that, at iteration $k+1$, we do not have access to the solution information of outcome $\widehat{\xi}_t$. Note that, as equation 2.1 holds for any x_{t-1}, and given equation 2.2, we can write

$$\begin{aligned} \mathcal{V}_t(x_{t-1}, \xi_{t-1}) \geq & \sum_{\xi_t \in \Omega_t - \{\widehat{\xi}_t\}} \alpha_{\xi_t,t}^{k+1} \cdot P(\xi_t | \xi_{t-1}) + \alpha_{\widehat{\xi}_t,t}^k \cdot P(\widehat{\xi}_t | \xi_{t-1}) \\ & + \Big[\sum_{\xi_t \in \Omega_t - \{\widehat{\xi}_t\}} \beta_{\xi_t,t}^{k+1} \cdot P(\xi_t | \xi_{t-1}) + \beta_{\widehat{\xi}_t,t}^k \cdot P(\widehat{\xi}_t | \xi_{t-1}) \Big] \cdot x_{t-1} \end{aligned} \tag{2.4}$$

Note that, in equation 4, the missing information, at iteration $k+1$, of outcome $\widehat{\xi}_t$ is completed by using the information of the previous iteration, namely by using the cut coefficients $\alpha_{\widehat{\xi}_t,t}^k, \beta_{\widehat{\xi}_t,t}^k$ of the previous iteration. □

In [PG08] the authors show that any sequence of cuts will necessarily be finite, in the sense that after a finite number of iterations no new cuts will be computed. Concretely, the following lemma, which is just an adaptation of the proof shown in [PG08], proves that after a finite number of iterations the algorithm will not produce new cuts.

Lemma 2.2. *Let $\mathcal{G}_k^{t,\omega}$ be the set of cuts at stage t, node ω and iteration k. There exists $m_{t,\omega}$ such that $|\mathcal{G}_k^{t,\omega}| \leq m_{t,\omega}$ for all k, $1 \leq t \leq T-1$.*

Proof. We proceed by induction on t. For $t = T-1$. Note that, as there are no cuts in the last stage, the set of multipliers in the last stage is given by,

$$\{\pi | \pi^T \cdot A_T(\omega) \leq c_T^T(\omega)\}$$

Thus, there are at most say $M_{T,\omega}$ possibilities for multipliers. Assuming there are N_t nodes for stage t, then this implies that at stage T there are at most $\prod_{i=1}^{N_T} M_{T,\omega_i}$ combinations of multipliers. As a consequence, for any node ω at stage $T-1$, there are at most $m_{T-1,\omega} := \prod_{i=1}^{N_T} M_{T,\omega_i}$ possibilities to build a cut. Let us now assume that for t we have $m_{t,\omega}$ such that $|\mathcal{G}_k^{t,\omega}| \leq m_{t,\omega}$ for all k. We have to show that the property holds for $t-1$.

Due to the assumption at the end of the previous paragraph, we know that for node ω in stage t, $|\mathcal{G}_k^{t,\omega}| \leq m_{t,\omega}$, in particular this implies that there exists a $\hat{k}$ such that $\mathcal{G}_{\hat{k}}^{t,\omega} = \mathcal{G}_k^{t,\omega}$ for all $k > \hat{k}$, and so any cut after iteration $\hat{k}$ is already in the set of cuts. Then, after iteration $\hat{k}$, the linear program of node ω at stage t will not have new cuts. As a consequence, the set of multipliers for node ω at stage t is finite, say $M_{t,\omega}$. This implies that, at stage t, there are $\prod_{i=1}^{N_t} M_{t,\omega_i}$ possible combinations of multipliers. Therefore, for any node ω at stage $t-1$ there will be at most $m_{t-1,\omega} := \prod_{i=1}^{N_t} M_{t,\omega_i}$ possibilities to build a cut, which proves the result. □

3 Batch Learning SDDP for Long-Term Hydrothermal Planning

3.1 Introduction

While SDDP has exhibited superior performance in solving large-scale instances of difficult optimization problems, there are instances where the algorithm fails to converge. For instance, gaps of nearly 22% are reported for a widely studied instance of the long-term planning problem of the Brazilian power system [STdCS13]. This is despite theoretical guarantees of convergence [PG08]. Failure to converge is a problem of practical relevance in short-term planning, where solutions need to be available within a certain period of time. Strategies to speed up the algorithm includes cut selection techniques, which aim at selecting and removing redundant hyperplanes [DMPF15, Gui17, GB19, LWM13]. While these techniques have shown promising results for increasing the computational speed during each iteration, failure of convergence is not addressed. Parallel computing has been proposed as well to speed-up the algorithm [dSF03, PBM13, HB15, MDBB21], as it is has shown significant promise for solving unit commitment problems [AP20]. However, as we have observed in the previous chapter, parallelism in its own may not be able to circumvent the issue of prompt convergence. Concretely, parallelization may not scale well, and different parallelization strategies may yield somewhat similar results when targeting tight optimality gaps [ÁPL21].

In this chapter, We propose to use experience replay - a batch learning technique that is popular within the reinforcement learning framework - to improve SDDP convergence as well as its parallel efficiency. We show how batch learning makes better use of parallel computations than conventional SDDP, and we compare it with a commercial implementation of SDDP - the PSR SDDP software [PSR][1]. We argue that our findings open the door for a number of relevant applications in short-term planning, which are to be explored further in future work.

[1] PSR is a consulting firm based in Rio de Janeiro that has pioneered the commercialization of the SDDP algorithm for hydro-thermal planning

3.1.1 Batch learning

Reinforcement learning (RL) is an area of machine learning that aims at training computational agents in order to enable them to reach decisions in a dynamic and uncertain environment, so that these agents can maximize their rewards.

There is a distinction in reinforcement learning between model-free and model-based methods. In model-free RL, the objective is to train an agent by observing its interaction with an environment for which no model exists. In model-based RL, the objective is to train an agent that interacts with a model of the environment. Since stochastic programming provides us with a model of the environment, we focus on model-based methods.

Conventional RL algorithms apply a sequential strategy that updates a decision policy as soon as new information arrives. It has been found that it can be better to delay and batch these updates so as to avoid costly matrix multiplications when updating gradients or simply to reduce noise [KS07]. This so-called batch learning approach has emerged as an attractive alternative that often outperforms other reinforcement learning algorithms [Lin92, KS07, LGR12].

Experience replay [Lin92], a batch learning technique, resamples states and actions that have been visited in previous iterations, thereby replacing the old belief regarding expected cost associated with those state-action pairs with new information. In this way, the algorithm decreases the delay of revisiting previously explored states, which may have a significant impact on the decision policy [Lin92, LGR12]. Google's DeepMind algorithm uses experience replay in combination with Q-learning as a strategy for obtaining human-level performance in a series of Atari games [MKS15].

We propose to use batch learning and experience replay in order to accelerate learning and thereby the convergence of SDDP. The proposed algorithm applies experience replay during the backward pass, where experiences correspond to the trial points of previous iterations. The updates of these trial points can be batched and parallelized during the backward pass.

3.1.2 Parallel computing

As argued in the previous chapter, parallel schemes are a natural choice for countering the computational complexity of multistage stochastic programming [PBM13]. The extant literature mostly discusses parallel Monte Carlo sampling during the forward and backward passes of the SDDP algorithm [dSF03, PBM13, HB15, DK17]. This view is somewhat limited, as we have argued in last chapter. Experimental evidence suggests [DMPF15, DMPFG10, ÁPL21] that a downside of increasing the number of parallel forward passes is that it often leads to an accumulation of trial points that are similar, or that are far from the optimal region, which in turn leads to an accumulation of redundant cuts that hardly improve the approximation but severely slow down the convergence of the algorithm.

Furthermore, past research has shown that different techniques tailored for exploiting synchronous and asynchronous parallel computing present a behaviour that is dependent on the problem and may not hold for long runs of the algorithms [ÁPL21, MDBB21]. These results have allowed us to identify that forward exploration is not sufficient for creating a highly parallelizable algorithm, but that the focus should rather be on the backward pass. This motivates the idea of being accurate during backward passes, a notion which has connections with certain ideas originating from the reinforcement learning framework.

3.1.3 Organization & Contributions

In this chapter, (i) we introduce a novel variant of SDDP, which we refer to as *Batch Learning SDDP (BL-SDDP)* and show its connection to reinforcement learning (RL); (ii) we compare the algorithm to the widely used commercial SDDP implementation of PSR; (iii) we demonstrate its suitability for parallel computing; (iv) we test its performance in both risk neutral and risk-averse settings.

In Section 3.2 we formulate multistage stochastic programming problems as Markov decision processes (MDPs), and we cast SDDP as a reinforcement learning algorithm, similar to Q-learning. In Section 3.3, we introduce BL-SDDP. BL-SDDP uses experience replay, a batch learning technique from reinforcement learning, as a novel approach for accelerating the convergence of the algorithm. We describe a novel parallel scheme for BL-SDDP in Section 3.4. In Section 3.5, we benchmark the new algorithm, not only against our own implementation of SDDP, but also against the commercial PSR SDDP software. The benchmark is performed against a high-dimensional, real-world instance of a hydro-thermal planning problem.

3.2 Problem Formulation

In this section, we discuss the connection between SDDP and reinforcement learning, so as to motivate our algorithmic developments. The section is divided into two subsections. The first subsection provides a link between multistage stochastic programs and MDP. In particular, in lemma 2.1, we establish that multistage stochastic programs which can be tackled through SDDP can be cast as MDPs, which in turn can be tackled through reinforcement learning. The last subsection presents Lemma 2.2, which demonstrates that SDDP is a reinforcement learning algorithm. This development allows us to access a wealth of algorithms from reinforcement learning that have delivered impressive performance in applications outside of power systems optimization.

3.2.1 Multistage stochastic programming and MDP

We motivate the connections between multistage stochastic linear programs and MDPs using the familiar example of a two-stage hydrothermal scheduling problem. The problem can be cast as follows:

$$\begin{aligned}
&\min\ C \cdot g_1 + \text{VOLL} \cdot ls_1 + \mathbb{E}[C \cdot g_2(\xi_2) + \text{VOLL} \cdot ls_2(\xi_2)] \\
&s.t.\ q_1 + g_1 + ls_1 = L_1 \\
&\quad x_1 = X_0 + A_1 - q_1 \\
&\quad q_2(\xi_2) + g_2(\xi_2) + ls_2(\xi_2) = L_2 \\
&\quad x_2(\xi_2) = x_1 + A_2(\xi_2) - q_2(\xi_2) \\
&\quad g_t(\xi_t) \le \bar{G} \\
&\quad x_t(\xi_t) \le \bar{X} \\
&\quad g_t(\xi_t), x_t(\xi_t), q_t(\xi_t) \ge 0, \text{ for all } \xi_t \in \Omega_t
\end{aligned}$$

Here, g_t is the power generated by thermal units at a marginal cost C. The thermal generators can produce up to $\bar{G}$ units of power per period. The system can shed load at a high cost $VOLL$, and ls_t is the amount of power that is curtailed from consumers. The variable q_t is the power generated from hydro units, which is generated at zero cost. The variable x_t represents the amount of available energy in the hydro reservoir at the end of period t. The hydro reservoir can store a maximum amount $\bar{X}$ of energy, and has as initial condition X_0. The system is subject to uncertain natural inflows represented by A_t. We assume that there are finitely many outcomes Ω_t. Note that there is a single realization $\Omega_1 = \{\xi_1\}$ in the first stage. Using standard MDP notation, we can write the problem as follows.

- The set of states are defined as below:

$$\begin{aligned}
\mathcal{S}_1 &= \{(X_0, \xi_1)\} \\
\mathcal{S}_2 &= \Big\{(x_1, \xi_2) :\ \text{exists } q_1, g_1 \text{ s.t. } x_1 = X_0 + A_1 - q_1 \\
&\qquad\qquad q_1 + g_1 + ls_1 = L_1 \\
&\qquad\qquad g_1 \le \bar{G}, x_1 \le \bar{X}, g_1, x_1, q_1 \ge 0, \xi_2 \in \Omega_2\Big\}
\end{aligned}$$

- The actions for a state $s_t = (x_{t-1}, \xi_t)$ are defined as

$$\begin{aligned}
\mathcal{A}_t(s_t) = \{(x_t, q_t, g_t, l_t) : &\ q_t + g_t + ls_t = L_t \\
&\ x_t = x_{t-1} + A_t(\xi_t) - q_t \\
&\ g_t \le \bar{G}, x_t \le \bar{X}, g_t, x_t, q_t \ge 0\}
\end{aligned}$$

We use a_t to refer to an action.

- The reward function is given by

$$C_t(s_t, a_t) = C \cdot g_t + VOLL \cdot ls_t$$

• The probability of transitioning to state s_2 while being in state $s_1 = (X_0, \xi_1)$ and selecting action $a_t = (x_t, q_t, g_t, l_t)$ is defined as:

$$P(s_{t+1}|s_t, a_t) = \begin{cases} P(\xi_2|\xi_1) & s_2 = (x_t, \xi_{t+1}) \\ 0 & \text{otherwise} \end{cases}$$

These definitions allow us to translate the multistage stochastic program into the MDP framework.

Now let us recall the general case of a multistage stochastic linear program over T stages (see 1.3), given by:

$$\begin{aligned} &\min_{\substack{B_1x_0+A_1x_1+D_1y_1=b_1 \\ x_1,y_1\geq 0}} u_1^T x_1 + v_1^T y_1 + \mathbb{E}\Bigg[\\ &\min_{\substack{B_2x_1+A_2x_2+D_2y_2=b_2 \\ x_2\geq 0}} u_2^T x_2 + v_2^T y_2 + \mathbb{E}\Bigg[\cdots + \\ &\mathbb{E}\Bigg[\min_{\substack{B_Tx_{T-1}+A_Tx_T+D_Ty_T=b_T \\ x_T\geq 0}} u_T^T x_T + v_T^T y_T \Bigg]\Bigg]\Bigg] \end{aligned} \tag{MSP-P}$$

Following the standard MDP description [Put14], we cast the MSP as an MDP by defining a tuple $(\mathcal{S}_t, \mathcal{A}_t, C_t, P)$ in the following way. To simplify the description we define the following feasible set:

$$\begin{aligned} \text{Feas}_t(x_{t-1}, \xi_t) = \{(x_t, y_t) \in \mathbb{R}^n : \exists\ x_t, y_t \geq 0, \\ B_t(\xi_t)x_{t-1} + A_t(\xi_t)x_t + D_t(\xi_t)y_t = b_t(\xi_t)\} \end{aligned}$$

• **States:** The set of states is defined recursively as:

$$\begin{aligned} &\mathcal{S}_1 = \{(x_0, \xi_1)\} \\ &\mathcal{S}_t = \Big\{(x_{t-1}, \xi_t) : \xi_t \in \Omega_t \text{ and } (x_{t-1}, y_{t-1}) \in \\ &\quad \text{Feas}_{t-1}(x_{t-2}, \xi_{t-1}) \text{ for some } (x_{t-2}, \xi_{t-1}) \in \mathcal{S}_{t-1}\Big\} \end{aligned}$$

• **Actions:** For each state $s_t = (x_{t-1}, \xi_t)$, the feasible actions are defined as $\mathcal{A}_t(s_t) = \text{Feas}_t(x_{t-1}, \xi_t)$. To improve readability, we define $a_t = (x_t, y_t)$ to refer to an action.

• **Reward:** The reward function is given by $C_t(s_t, a_t) = u_t(\xi_t)^T x_t + v_t(\xi_t)^T y_t$.

• **Dynamics:** The probability of transitioning to state s_{t+1} while being in state $s_t = (x_{t-1}, \xi_t)$ and selecting action $a_t = (x_t, y_t)$ is as follows:

$$P(s_{t+1}|s_t, a_t) = \begin{cases} P(\xi_{t+1}|\xi_t) & s_{t+1} = (x_t, \xi_{t+1}) \\ 0 & \text{otherwise} \end{cases}$$

Furthermore, the state transition equation can be expressed as $(x_t, \xi_{t+1}) = f_t(s_t, a_t, \xi_{t+1})$.

Remark 3.1. *Note that problem MSP-P is a linear program. Consequently, the optimal action is at the vertex of the feasible set. Therefore, we can always restrict the actions to such a set, and have finitely many actions and states.*

With these definitions, with the presented choice of $(\mathcal{S}_t, \mathcal{A}_t, C_t, P)$, we can pose the MDP problem as one of finding a policy that minimizes the expected reward:

$$\min_{\pi \in \Pi} \mathbb{E}\Big[\sum_{t=1}^{T} C_t(s_t^\pi, a_t^\pi)\Big] \tag{MDP-P}$$

To link reinforcement learning with stochastic dual dynamic programming, we establish the following relationship.

Lemma 3.1. *The problems MDP-P and MSP-P are equivalent.*

Proof. Let us consider a feasible solution of problem MSP-P. Furthermore, we can ask such a solution to be a vertex of the polyhedra defining the linear programs. Then, for every t and $\xi_t \in \Omega_t$, we have feasible values $x_t(\xi_t), y_t(\xi_t)$ and an objective cost

$$u_1^T x_1 + v_1 y_1 + \mathbb{E}\Big[u_2^T x_2 + v_2^T y_2 + \mathbb{E}\Big[\cdots + \mathbb{E}\Big[u_T^T x_T + v_T^T y_T\Big]\Big]\Big]$$

Note that we can define a policy π_t such that for $s_t = (x_{t-1}(\xi_{t-1}), \xi_t)$ we have $\pi_t(s_t) = (x_t(\xi_t), y_t(\xi_t))$. Moreover, under such a policy,

$$\mathbb{E}\Big[\sum_{t=1}^{T} C_t(s_t^\pi, a_t^\pi)\Big] = u_1^T x_1 + v_1 y_1 + \mathbb{E}\Big[u_2^T x_2 + v_2^T y_2 + \mathbb{E}\Big[\cdots + \mathbb{E}\Big[u_T^T x_T + v_T^T y_T\Big]\Big]\Big]$$

Therefore, every vertex solution of MSP-P gives a policy π which, when evaluated in MDP-P, yields the same cost. Thus, MDP-P $\leq$ MSP-P. Similarly, given any policy π, we can define a feasible solution for MSP-P which, when evaluated, yields the same result as when evaluating the policy, and so MDP-P $\geq$ MSP-P. □

3.2.2 Stochastic dual dynamic programming as a reinforcement learning algorithm

Following a similar structure as double-pass algorithms [Pow07], we can describe SDDP as a reinforcement learning algorithm that can be used to tackle problem MDP-P. Through an iterative procedure, this algorithm approximates the Q-factors. Supporting hyperplanes are used in order to approximate the value functions $\mathcal{V}_{t+1}^*(s_t, a_t)$. The Bellman optimality equation is then used in order to approximate the value functions of preceding stages.

Lemma 3.2. *SDDP is a reinforcement learning algorithm used to solve MDP-P.*

Proof. As presented in subsection 1.4, the Q-factors satisfy

$$Q_t^*(s_t, a_t) = C_t(s_t, a_t) + \mathcal{V}_{t+1}^*(s_t, a_t)$$

Moreover, $\mathcal{V}_{t+1}^*$ can be approximated by building a supporting hyperplane around x_t^n [STdCS13], where $(x_t^n, y_t^n) = a_t^n$ is a trial action. Thus, we can consider an update rule that adds a supporting hyperplane around x_t^n to the current approximation of the value function, $\mathcal{V}_{t+1}$, and updates the Q-factor as $Q_t(s_t, a_t) = C_t(s_t, a_t) + \mathcal{V}_{t+1}(s_t, a_t)$. Building such a supporting hyperplane requires a model-based scheme. A detailed exposition on how the supporting hyperplane is built can be found in [STdCS13]. Let us express this update rule as $Q_t = U(a_t, Q_{t+1})$. We can now formulate the SDDP algorithm following a similar structure as double-pass algorithms:

Algorithm 10: SDDP as an RL algorithm

INPUT: Provide an initialization of the Q-factors $Q_t^0(s_t, a_t)$ for $s_t \in \mathcal{S}_t$, $a_t \in \mathcal{A}_t(s_t)$ and $t = 1, \cdots, T$, and a maximum number of iterations N.
OUTPUT: A cutting-plane approximation $\{c_{\xi_t}^i(x_t)\}_{i=1}^N$ for $t = 1, \ldots, T$, $\xi_t \in \Omega_t$.

for $n = 1, \cdots, N$

1. **Forward Pass**: Initialize at state s_1. **for** $t = 1, \cdots, T$

   (1.1) Find the decision using the current Q-factors.

   $$a_t^n \in \underset{a_t \in \mathcal{X}(s_t^n)}{\arg\min}\ Q_t^{n-1}(s_t^n, a_t)$$

   (1.2) Take action a_t^n and transition to state s_{t+1}^n.

2. **Backward Pass**: **for** $t = T, \cdots, 1$

   (2.1) Update Q_t^{n-1} using the update rule.

   $$Q_t^n = U_t(a_t^n, Q_{t+1}^n)$$

return Q_t^N estimates for $t = 1, \cdots T$

□

Remark 3.2. *Double-pass algorithms, with the proper update rule, include commonly known reinforcement learning algorithms such as TD(1) [Pow07]. Furthermore, single-pass algorithms, where just a forward pass is applied and*

the updates evolve forward in time, include algorithms such as Q-learning or more generally TD(0) algorithms [Pow07].

3.3 Batch Learning SDDP (BL-SDDP)

The motivation of Batch Learning SDDP is to exploit parallel computing for proposing a novel and effective approach to perform backward passes, aiming at back-propagating the information accurately across stages.

The relevance of the backward pass can be understood as follows. Poor value function approximations in the last stage, which SDDP will build at early steps, will produce even looser cuts for the previous stage, with approximation errors increasing as we move backwards in time stages. The BL-SDDP algorithm addresses this drawback by allowing previously visited trial actions or experiences to have access to the value function updates carried out in later stages. The idea of re-visiting previous experiences, a technique known as experience replay, has gained popularity in the reinforcement learning literature due to its success in accelerating learning speed [Lin92, PW16, MKS15].

This section presents a novel application of the experience replay framework in the context of SDDP as a corollary of the results presented in section 3.2. The first subsection presents the experience replay scheme, and the second subsection introduces our novel batch learning SDDP algorithm.

3.3.1 Experience replay

The experience replay framework, introduced first by [Lin92], aims at updating Q_t using previously computed states and actions, commonly referred to as experiences. The motivation behind this is that updating a state-action pair (s_t, a_t) at stage t may affect some preceding states s_{t-1}. Nevertheless, this information will not back-propagate until state s_{t-1} is re-visited. Furthermore, states preceding s_{t-1} will need to be re-visited after the update of s_{t-1} before being able to see the update carried out in the upper layers. Therefore, the back-propagation of information is not possible unless states are re-visited. Given an arbitrary update rule $Q_t = U_t(s_t, a_t, s_{t+1}, Q_{t+1})$, the experience replay algorithm, which has been adapted for the finite horizon setting, can be described as follows [Lin92, PW16, MKS15].

Algorithm 11: Experience Replay

INPUT: Initialization of the Q-factors $Q_t^0(s_t, a_t)$ for $s_t \in \mathcal{S}_t$, $a_t \in \mathcal{A}_t(s_t)$, $t = 1, \cdots, T$. For each stage, provide a set of experiences $M_t = \{(s_t^n, a_t^n, s_{t+1}^n) : n = 1, \ldots, N\}$ and let K be the batch size $K \leq |M_t|$.

for $t = T, \cdots, 1$

1. Retrieve a subset $\{(s_t^k, a_t^k, s_{t+1}^k) : k = 1, \ldots, K\} \subset M_t$.
2. **for** $k = 1, \cdots, K$ update Q_t^0 using the update rule.

$$Q_t^1 = U_t(s_t^k, a_t^k, s_{t+1}^k, Q_{t+1}^1)$$

return Q_t^1 estimates for $t = 1, \cdots T$

Note that this process can be repeated iteratively. That is to say, a set of experiences is collected, the experience replay algorithm is applied, afterwards more experiences are collected, and the experience replay algorithm is applied again. In the literature, when K equals the total set of experiences, the method is usually referred to as a full-batch update, whereas when K is less than the total size it is referred to as a mini-batch update.

3.3.2 BL-SDDP description

As presented in section 3.2.2, the SDDP algorithm can be described as a type of double-pass algorithm, of the sort that can be encountered in the reinforcement learning literature (lemma 3.2). As a consequence, we can apply known techniques for MDP algorithms, such as the experience replay scheme. This leads to the Batch Learning SDDP algorithm described as follows.

Algorithm 12: BL-SDDP

INPUT: Initialization of the Q-factors $Q_t^0(s_t, a_t)$ for $s_t \in \mathcal{S}_t$, $a_t \in \mathcal{A}_t(s_t)$, $t = 1, \cdots, T$. Let K be the batch size, Z be the number of collected experiences before applying experience replay. Let $M = \emptyset$ be the set of experiences. A maximum number of iterations N.
OUTPUT: A cutting-plane approximation $\{c^i_{\xi_t}(x_t)\}_{i=1}^N$ for $t = 1, \ldots, T$, $\xi_t \in \Omega_t$.

for $n = 1, \cdots, N$

1. Apply SDDP, collecting up to Z experiences, and add them to the replay memory M.
2. Consider a batch of K experiences of M.

   Apply experience replay on the K experiences.

return Q_t^1 estimates for $t = 1, \cdots T$

Note that, as we are using the update rule based on supporting hyperplanes, the replay memory simply requires $M = \{a_t^n : n = 1, \ldots, N\}$. The proposed scheme can also be seen as an update of cuts, which is carried out for a batch of K cuts every Z SDDP iterations. Note that the procedure can be combined with other schemes. For instance, during step 1 the forward sampling procedure could be changed to the one presented in [DB06, HP14] in order to adjust the sampling to the problem structure.

The BL-SDDP algorithm can be described using the flow chart shown in Fig. 3.1. The algorithm commences by performing usual SDDP iterations, collecting the trial actions obtained during the forward passes. These trial actions are added to the replay memory. The procedure continues until Z new trial actions are added to the replay memory. Next we proceed to the experience replay scheme. This step receives as an input the replay memory, which is a collection of trial actions, and builds a cut for a batch of these trial actions. As an output of this step, we obtain cuts around a batch of trial actions, which can then be used as an input for SDDP.

The difference between SDDP and BL-SDDP can be illustrated graphically in Figure 3.2. The red line corresponds to the cuts calculated by SDDP while the blue line corresponds to BL-SDDP. The super index j in $\widehat{x}_i^j$ refers to the SDDP iteration while the lower index i refers to the stage. Following this notation, the SDDP cut around $\widehat{x}_1^1$ is calculated using the first cut of the second-stage value function. Similarly, the SDDP cut around $\widehat{x}_2^1$ is built using the first cut of the third-stage value function. During each iteration, the third-stage value function improves. Nevertheless, this improvement is not seen by $\widehat{x}_2^1$. Similarly, the second-stage value function improves during each iteration but this is not seen by $\widehat{x}_1^1$. Concretely, improvements made in upper stages are not seen by previously visited trial actions, namely, there is lim-

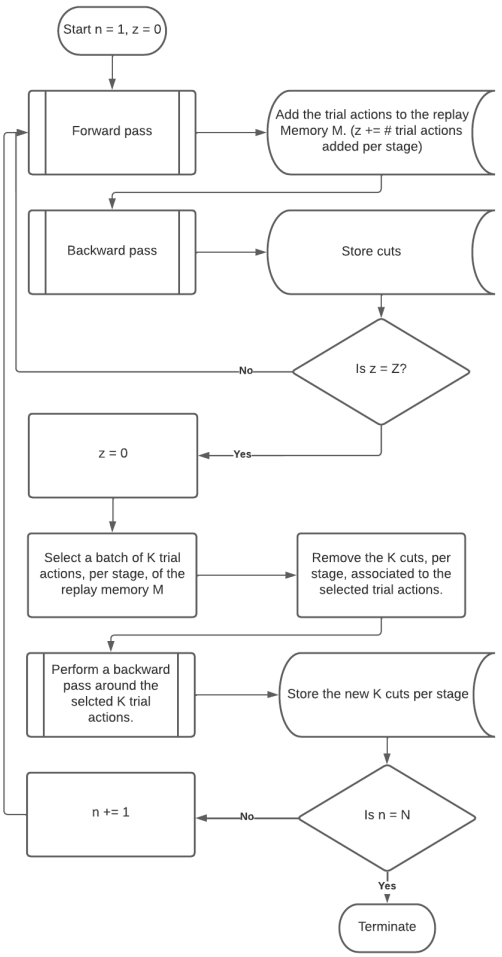


Figure 3.1: Flow chart describing the BL-SDDP scheme. The algorithm commences by performing usual SDDP iterations, until Z trial actions or experiences are collected. A batch of K trial actions is selected and the cuts around these trial actions are updated.

ited back-propagation of information. On the other hand, BL-SDDP allows a back-propagation of information. The trial actions are collected in the replay memory and re-visited, which means that the cut around $\widehat{x}_2^1$ is calculated using the most recent value function in the third stage, and thus the cut is expected to be of better quality. A similar effect takes place in the preceding stages.

Relative to standard SDDP, the improved performance of BL-SDDP stems from the fact that it approximates the value functions more diligently in the

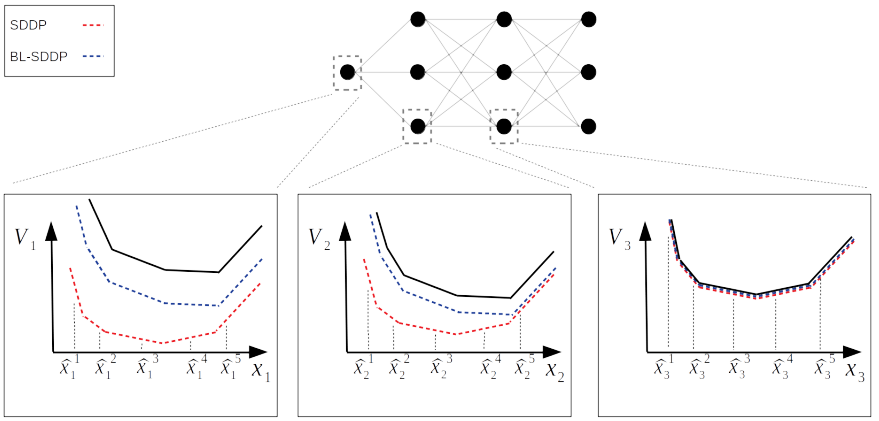


Figure 3.2: Value function differences between SDDP and BL-SDDP. The red line corresponds to the cuts calculated by SDDP while the blue corresponds to BL-SDDP. The black line corresponds to the true value function.

backward pass. It thus prevents the back-propagation of approximation errors of value functions at later stages of the problem from "contaminating" the approximations of value functions at earlier stages of the algorithm. This comes at the cost of increased computational effort at each backward pass relative to standard SDDP. We propose resorting to parallel computing in order to cope with this increased computational burden, and thus to combine the most appealing attributes of experience replay and SDDP into a single and highly parallelizable algorithmic procedure.

3.4 Parallelization Strategies

As discussed in the previous chapter, the parallel SDDP literature presents parallelization schemes that are mostly focused on increasing the number of Monte Carlo forward samples and distributing these samples among the available processors [dSF03, PBM13, HB15, DK17]. As the Sync PS parallel scheme described in Fig. 2.2 is the most commonly adopted strategy for parallelizing SDDP, we use it as a benchmark for the parallel strategy that we develop in the next subsection.

3.4.1 Parallelization of BL-SDDP

We present a novel parallelization scheme for SDDP based on the BL-SDDP algorithm that we propose in section 3.3. The developed BL-SDDP algorithm is divided into two parts. The first part is a common SDDP run. The second part employs the experience replay framework. We propose a synchronous parallel

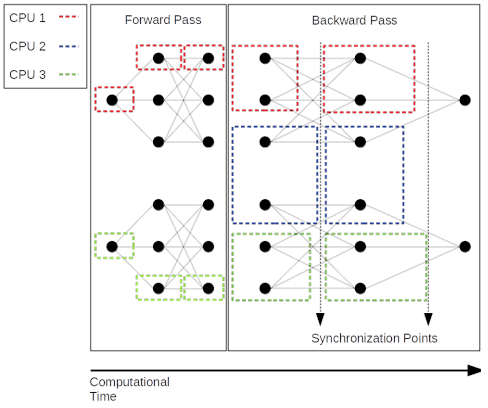


Figure 3.3: Parallelization of the BL-SDDP algorithm. The SDDP iterations are parallelized by considering a small number of samples in the forward pass, which are then distributed among the available processors. In the backward pass, the problems at each stage are distributed among the processors.

computing strategy for each one of these parts.

• **SDDP parallelization.** As we have discussed, the standard parallelization strategy for SDDP can result in inferior performance. In order to avoid a possible deterioration in performance, our proposed parallelization proceeds by fixing a small number of samples in the forward pass. We illustrate the parallelization strategy through the example depicted in Fig. 3.3. The forward pass consists of 2 samples. Each processor computes one sample. Note that the blue CPU remains idle at this point. There is a synchronization point at the end of the forward pass. During the backward pass, at every stage, there are 6 problems, since there are 2 samples and 3 nodes per stage in the lattice. These 6 problems are then distributed among the available processors. Once the problems of the stage have been computed, the processors synchronize, the cuts are built, and the obtained cuts are shared among the available processors. Note that, as presented in the picture, the scheme has several synchronization points.

• **BL parallelization**. Let us assume that the replay memory is given by $M_t = \{a_t^n : n = 1, \ldots, N\}$ where $a_t^n = (x_t^n, y_t^n)$ is an action. Let us introduce the parallelization through the example shown in Fig. 3.4. For the present example, let us assume we have a batch of size 5. Therefore, the algorithm starts by selecting a batch of 5 experiences (trial actions), for each stage, among the collected experiences in the replay memory. These selected experiences are distributed among the available processors. Then, proceeding backwards in time, each processor updates the Q-factors around the corresponding actions. That is to say, each processor builds supporting hyperplanes for the expected value functions around the experiences that it receives. Note that, at the end

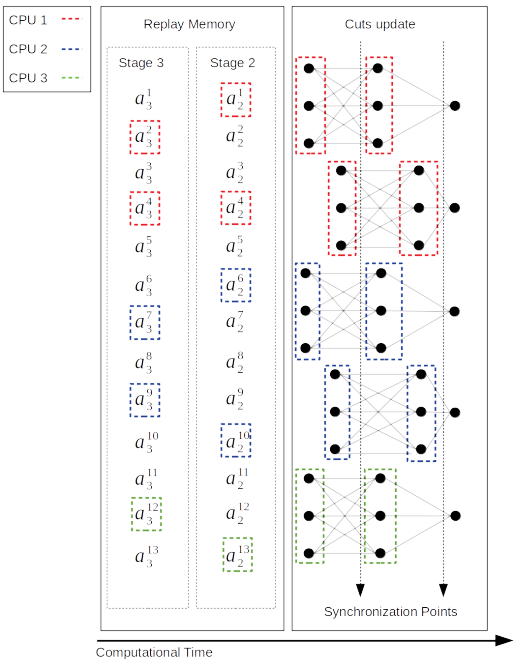


Figure 3.4: Parallelization of the BL-SDDP algorithm. The BL steps are parallelized by distributing the trial actions of the replay memory among the available processors. The processors update the cuts around those trial actions. .

of each stage, the processors synchronize in order to receive the cuts obtained by other processors. This procedure naturally scales up with the introduction of additional CPUs, since additional processors imply that each processor will have a smaller set of experiences.

3.5 Case Studies

We carry out computational experiments over a realistic instance of a long-term planning problem for a network of hydropower plants in Colombia. We further analyze the impact of the methodology on an instance of long-term hydrothermal planning from Brazil that is well-know for its difficulty. As in the case of the previous chapter, the methodology is tested under a different class of problems: an inventory control problem with lead times, the results of which can be found in the Appendix. Our experimental results can be summarized as follows: (i) The BL-SDDP algorithm is able to produce tighter gaps in less time, compared to the PSR SDDP commercial implementation. (ii) The BL-SDDP parallel scheme is better suited for parallel computing, and responds more favorably to an increase in parallel computing capacity than standard

SDDP. (iii) The superior performance of BL-SDDP can be observed in both risk-neutral and risk-averse formulations of multistage stochastic programming.

We proceed by briefly introducing the hydrothermal test cases analyzed in this chapter. The SDDP literature has focused extensively on hydrothermal scheduling problems due to their practical relevance [PP91, STdCS13, PBM13, DMPF15, LS19]. The objective of the problem is to determine optimal storage levels for the hydro reservoirs of a power system under inflow uncertainty, while respecting operational constraints and satisfying demand, in such a way that the total expected operational costs of the system are minimized. The mathematical description of the problem can be found in the Appendix, in particular it differs from the previous chapter in that the Colombian test case considers a river network. Moreover, we consider the Markov Chain approach in addition to a time series for representing uncertainty [LS19].

The first test case that we consider is an instance of the Colombian power system that has been provided by PSR. The case study comprises 32 thermal plants and 42 hydro plants, 25 of which have storage capacity. The case study considers a river network that consists of a 54-dimensional inflow vector. Since transmission network data is not available to us, it is not considered. The instance considers long-term planning and exhibits a time horizon of 10 years and monthly steps, i.e., 120 stages in total. Inflow uncertainty is modeled using PSR's SDDP software that fits a periodic autoregressive (PAR) model to historical inflow data. We expand the state space, as discussed in [STdCS13], who propose to add equality constraints into the problem. Historical inflow data from 1937 to 2019 is used to calibrate a PAR(1) model. This inflow model is then used in PSR's SDDP model as well as our own implementations, so as to arrive to a meaningful comparison. Uncertainty is represented by drawing a sample of 100 realizations of the error terms of the PAR model. The result is a high-dimensional multi-stage stochastic program. The test case is analyzed in subsections 3.5.1 to 3.5.5.

The introduction of a transmission network increases the complexity of the problem. Therefore, we consider an additional test case, described in [STdCS13], which includes a transmission network. The case study considers an instance of the Brazilian power system, where the reservoirs have been aggregated into equivalent energy reservoirs [AR70]. The instance has 4 energy equivalent reservoirs. The transmission network is formulated as a transportation network. The model spans a time horizon of 10 years. We consider monthly time increments, and 100 uncertainty realizations per stage. In [STdCS13], the users use this instance to study risk-neutral and risk-averse formulations and find that SDDP struggles to close the optimality gap in the presence of time-dependent inflows. Similar observations have been made in [LS19]. In addition to analyzing the impact of a transmission network, this test case allows us to test the methodology under two uncertainty assumptions: time series modelling and Markov Chain modelling. The last subsection analyzes this case study.

Our algorithms are implemented in Julia v0.6 [BEKS17] and JuMP v0.18 [DHL17]. The chosen linear programming solver is Gurobi 8. In order to avoid

confusion in the subsequent sections regarding which codes are being compared, the following nomenclature is adopted. **PSR SDDP:** This scheme refers to the PSR software. The language used to code PSR SDDP is FORTRAN and uses the XPRESS solver. **SDDP:** This refers to our base Julia SDDP implementation. **BL-SDDP:** This refers to our proposed algorithm, which we describe in section 3.3.

3.5.1 Comparison of BL-SDDP to PSR SDDP

The present subsection aims at comparing the performance of the BL-SDDP algorithm against the PSR SDDP commercial software. The PSR SDDP commercial software is developed for solving hydrothermal scheduling problems. We conduct our comparison on the basis of an instance of the Colombian power system that has been provided to us by PSR. The experimental results for this subsection were obtained on an Intel Core i5-6198DU CPU with 2.30 GHz and 8 GB of RAM, using a single CPU.

Both codes are run with the exact same parameters. Specifically, each SDDP iteration consists of 20 samples for the forward pass. Each stage consists of 100 uncertainty realizations, namely 100 nodes per stage. Regarding the BL-SDDP algorithm, we perform batch updates at every 5 SDDP iterations. The batch size corresponds to a full-batch update, namely all the experiences are used for the update. The time horizon is defined to be equal to 120 stages.

Fig. 3.5 presents the convergence behavior for each of the two algorithms. Panel (a) presents the evolution of the lower and upper bound over iterations for the PSR software, while panel (b) presents the evolution for our BL-SDDP code. The upper bound estimate presented in panel (a) is the one reported by the PSR software. In our approach, in order to provide reliable estimates, we estimate the upper bound every 5 iterations by simulating our current policy over a large collection of samples, concretely 4000 inflow samples. As the samples used to estimate the upper bound for both algorithms can be different, we have also performed an out-of-sample evaluation, whose results are later discussed and presented in table 3.1. Note that the PSR SDDP software reaches a steady state behaviour after approximately 150 iterations, at which point the difference between the lower and upper bound remains relatively constant. On the other hand, the BL-SDDP approach is able to reduce the difference between upper and lower bound significantly after each batch update.

Table 3.1 presents the total run time and the reported gap after 34 hours of run time. Note that the BL-SDDP algorithm is able to produce a tighter gap. The table also presents the mean cost of an out-of-sample evaluation of both policies. Concretely, 2000 out-of-sample inflow samples are generated using PSR software. The policies are then tested against these inflows. As we can observe, the improved gap of the BL-SDDP algorithm also results in a superior out-of-sample performance relative to the policy generated by the PSR SDDP commercial software.

Interestingly, such a superior performance in computing tighter optimality

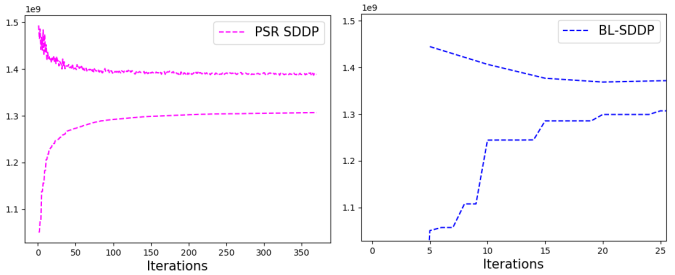


Figure 3.5: Lower and upper bound evolution over iterations for the Colombian hydrothermal test case. The left panel presents the PSR SDDP software while the right panel presents the BL-SDDP algorithm.

Table 3.1: Comparison of the policies after 34 hours of run time.

	PSR SDDP	BL-SDDP
Reported Gap (%)	6.2	4.2
Time (h)	34 (1 CPU)	34 (1 CPU)
Out-Of-Sample Inflows ($)	1.437e9	1.39e9

gaps holds even though our base SDDP implementation is considerably slower as compared to the PSR SDDP implementation. Concretely, PSR SDDP requires approximately 30 minutes in order to compute 20 iterations while our base SDDP implementation requires approximately 3 hours in order to compute the same number of iterations.

The superior performance of the BL-SDDP algorithm in computing tighter optimality gaps, despite the less optimized performance of the subproblem solvers, can be understood in terms of the amount of "work" that each approach is performing, and in particular the number of linear programs that both approaches are solving. For ease of exposition, let us ignore the forward pass as it comprises a very small part of the overall computational effort. The problems solved when performing the backward pass can be described as follows:

(i) PSR SDDP: Evaluating a single trial point involves solving, at each stage, 100 LPs. Since the time horizon is equal to 120 stages, this amounts to a total of $119 \cdot 100$ problems. Every iteration considers 20 forward samples. Thus, 20 trial points are generated per iteration. Consequently, a total of $100 \cdot 119 \cdot 20$ problems are solved per iteration. Over 370 iterations, this amounts to a total of $100 \cdot 119 \cdot 20 \cdot 370 = 88,060,000$ problems.

(ii) BL-SDDP: The algorithm performs a total of 25 usual SDDP iterations. As explained in the previous paragraph, this results in $100 \cdot 119 \cdot 20 \cdot 25 = 5950000$ problems after the execution of the 25 usual SDDP iterations. In this case,

we have to add the problems that are solved when performing the full-batch updates. After N iterations, a total of $N \cdot 20$ trial points need to be solved. As we mention previously, for each trial point a total of $119 \cdot 100$ LPs need to be solved. Thus, performing a full-batch update after N iterations would require solving $119 \cdot 100 \cdot 20 \cdot N$ LPs. As the full-batch update is performed every 5 iterations, this means that N evolves according to the following sequence: 5, 10, 15, 20, 25. Thus, the full-batch update step throughout the entire execution of the algorithm requires solving $100 \cdot 119 \cdot 20 \cdot (5 + 10 + 15 + 25) = 17,850,000$ problems. Adding the full-batch update LPs and the LPs of the usual SDDP iterations results in a total of $23,800,000$ problems, which represents 27% of the problems that the PSR software is solving. In short, the difference is due to the fact that the BL-SDDP algorithm avoids over-exploring and thus avoids redundant computation in additional iterations.

As a consequence of the aforementioned observation, we additionally note that the expected value function approximations for the BL-SDDP algorithm are considerably lighter. Concretely, PSR SDDP performs 370 iterations before terminating, thus the expected value function approximation consists of approximately $370 \cdot 20 = 7,400$ cuts per stage. Instead, the BL-SDDP code requires approximately $25 \cdot 20 = 500$ cuts per stage.

3.5.2 Comparison of parallel BL-SDDP to parallel SDDP

The present subsection aims at analyzing how the BL-SDDP algorithm responds to parallelizaiton, compared to the standard SDDP scheme. For this purpose, we resort to the same base Julia implementation. The computational work is performed on the Lemaitre3 cluster of UCLouvain, which is hosted at the Consortium des Equipements de Calcul Intensif (CECI). The cluster, where the algorithms are run, consists of 80 compute nodes, each consisting of two 12-core Intel SkyLake 5118 processors at 2.3 GHz and 95 GB of RAM (3970MB/core), interconnected with an OmniPath network (OPA-56Gbps).

Figure 3.6 presents the convergence evolution when using 20 CPUs. The algorithms are set up to compute, at each iteration, 20 samples in the forward pass. The BL-SDDP algorithm performs full-batch updates every 5 iterations. The figures demonstrate that the BL-SDDP algorithm produces considerably tighter gaps throughout the execution of the algorithm[2]. Note that for this particular test case the upper bounds are very close, so both codes produce very similar policies.

Increasing the number of CPUs produces a similar behaviour. Fig. 3.7 presents the scaling of the algorithm with respect to an increasing number of CPUs. Panel (a) of the figure presents the elapsed time until achieving a certain target optimality gap in the y-axis. Panel (b) presents the obtained gap after a fixed run time in the y-axis. As we can observe, the BL-SDDP scheme is able to attain a considerable improvement relative to the standard SDDP scheme.

[2]Note that, as the upper bound is a statistical one, there is randomness in the estimate of the true upper bound which occur during the execution of the algorithm.

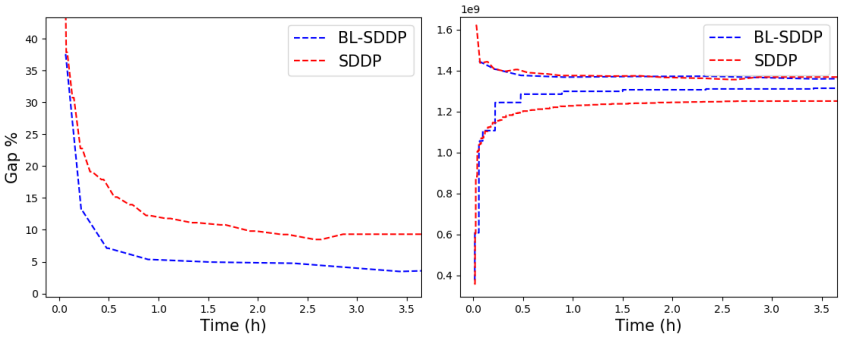


Figure 3.6: Convergence evolution for the Colombian hydrothermal test case using 20 CPUs

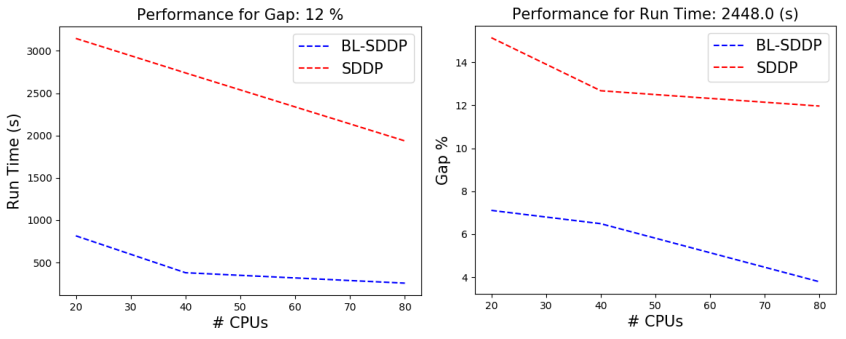


Figure 3.7: Performance of algorithms with respect to increasing CPUs for the Colombian hydrothermal test case.

A common measure for estimating the scalability of a parallel algorithm is the so-called Parallel Efficiency (PE), which measures how the reported parallel computing time would compare against a perfect parallelization of the reported serial time. Mathematically, it is defined as $PE = S_T/(N \cdot P_T)$. Here, P_T is the parallel time using N cores and S_T is the best serial time. Note that the best possible outcome would be a parallel efficiency of 1. Fig. 3.8 presents the parallel efficiency results for both test cases. We can observe that both SDDP and BL-SDDP achieve a parallel efficiency approximately equal to 0.8 for the hydrothermal test case. We observe that BL-SDDP demonstrates favorable parallel efficiency, with the added value that is able to attain tighter optimality gaps.

The literature discusses changes to the forward pass in order to improve exploration of the state space [DB06, HP14]. As described in Section 3.3, the BL-SDDP scheme can be easily combined with these exploration schemes. However, initial tests performed by the authors that compare and combine BL-SDDP with the schemes described in [DB06] and [HP14] showed only small

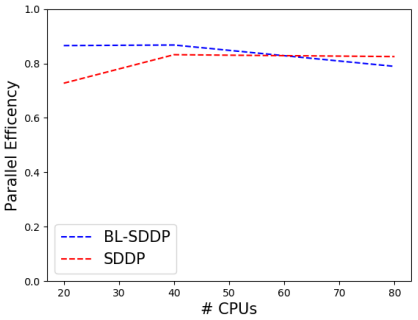


Figure 3.8: Parallel efficiency for the Colombian hydrothermal problem.

improvements in performance, so that we decide not to further explore this topic. It should be noted that the scheme presented in [DB06] appears to be better suited for problems with many realizations per stage but only few stages, which is not the case for the studied problem instances.

3.5.3 Comparison of the value functions

In order to illustrate the differences between the value functions calculated by SDDP and BL-SDDP, we compare the expected value functions obtained through SDDP and BL-SDDP after running both codes for the same amount of time. The comparison is performed as follows. Let f_t, g_t denote the expected value function approximation obtained by BL-SDDP and SDDP respectively at stage t. At stage t, we consider the trial points obtained by SDDP and evaluate them at both f_t and g_t. The relative difference, for each trial point, is computed as

$$\frac{f_t(x) - g_t(x)}{\max\{|f_t(x)|, |g_t(x)|\}} \cdot 100$$

The relative difference at each stage is then averaged among the trial points. Note that this procedure should in principle lead to an advantage for SDDP, because we are using the trial points where the SDDP cuts are calculated.

Fig. 3.9 presents the results of this procedure. The x axis corresponds to the stage number and the y axis corresponds to the relative difference. The first observation is that the relative difference is a positive number for almost all stages. This indicates that the BL-SDDP algorithm is computing tighter cuts. Moreover, negative values are only encountered for a few stages at the end of the time horizon. In certain cases the differences are significant. For example, in the hydrothermal test case, the relative difference in the last stage is approximately equal to -20%. Note that negative values are expected: the value functions are evaluated at the trial points visited by SDDP. Recall that, in the last stages, the cuts produced by SDDP are tight at the trial points of the SDDP algorithm. Therefore, it is expected that SDDP would perform

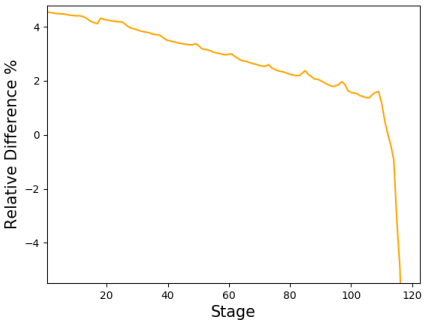


Figure 3.9: Comparison of expected value functions for the Colombian hydrothermal test case. The x axis represents the current stage. The y axis represents the relative difference between the expected value functions obtained through SDDP and BL-SDDP.

better at these points (even if these points do not correspond to where an optimal policy would have "landed" at the given stage). The second observation is that the relative difference becomes greater as we move backwards in the number of stages. SDDP builds a cut for the previous stage using the current stage approximation of the expected value function. A poor approximation at the present stage will result in an even poorer approximation for the previous stages, thereby resulting in a back-propagation of errors. BL-SDDP is less susceptible to such an effect as the batch update results in expected value function approximations that are calculated at a large collection of trial points and therefore high-quality approximations of the expected value functions.

3.5.4 Batch Choices

The present subsection aims at providing a brief study on the effect of different batch choices. We consider the following rules for selecting a batch.

F Corresponds to a full batch update.

R Corresponds to a random batch.

B Let C_i correspond to a cut calculated around x_i. Let δ_i be defined as the distance between the cut C_i and the value function at point x_i, namely:

$$\delta_i = V(x_i) - C_i(x_i)$$

The batch corresponds to the smallest δ_i: the best cuts.

W Using the same notation as in previous item, the batch corresponds to the highest δ_i. These are the worst cuts.

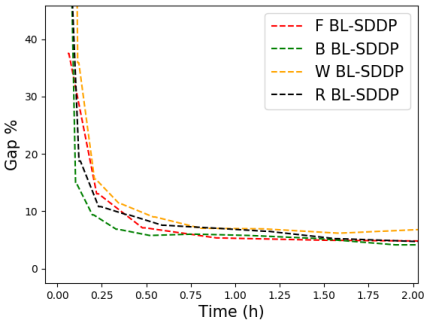


Figure 3.10: Batch choices for BL-SDDP applied to the Colombian hydrothermal test case. The batch size is set to 50%, except for the F BL-SDDP that performs a full-batch update.

The results are presented in Fig. 3.10. As in the previous experiments, 20 samples are considered in the forward pass. The batch update is performed every 5 iterations for the hydrothermal test case. The chosen batch size for batch choices R, B and W is 50%. Fig. 3.10 shows that the W and R strategies tend to behave the worst. On the other hand, there is a significant improvement in the B strategy at the early steps, as compared to the F strategy. Nevertheless, in the long run, both strategies behave similarly. We highlight that developing selection rules is of high practical relevance, as they can improve substantially the running times. Several alternative strategies for batch selection can be considered, in particular investigating strategies developed by the reinforcement learning community may prove beneficial. Such a detailed investigation is left for future research.

3.5.5 Risk-Averse SDDP

The objective of a risk-averse model is to avoid risky decisions that would lead to high costs for certain unfavorable scenarios. From a practical point of view, these measures have gained interest due their capabilities of protecting the user against catastrophic outcomes. As a practical examples we encounter instances of robust unit commitment [LSLZ16] or instances of the Brazilian hydrothermal power system [STdCS13], the latter being used by practitioners in actual operations. We consider the risk measure:

$$\rho_t[Z] = (1-\lambda)\mathbb{E}_t[Z] + \lambda \mathrm{AV@R}_\alpha[Z]$$

where $\lambda \in [0,1]$ is a weighting parameter and $\mathrm{AV@R}_\alpha[Z]$ is the Average Value-at-Risk (also referred to as Conditional Value-at-Risk). It is defined as

$$AV@R_\alpha[Z] = V@R_\alpha[Z] + \alpha^{-1}\mathbb{E}_t[Z - V@R_\alpha[Z]]_+$$

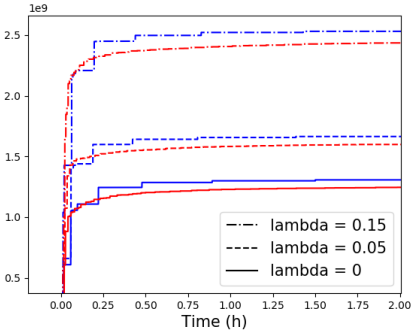


Figure 3.11: Risk averse BL-SDDP applied to the Colombian hydrothermal test case. The blue color corresponds to the BL-SDDP code, while the red color corresponds to SDDP.

Intuitively, $\text{AV@R}_\alpha[Z]$ is the expected value given that Z is greater than the $(1-\alpha)$-quantile. The minimization then aims at minimizing the costs found at the tail of the distribution. As stated in [STdCS13], the SDDP algorithm can be adapted to handle such a measure. Nevertheless, the algorithm only provides a lower bound approximation. Thus, stabilization criteria are used for deciding on convergence [STdCS13]. Being nested conditional expectations, the calculation of an upper bound would require to perform a conditional sampling process, for every stage of decision. That is to say, for every conditional event, perform a sampling process, however due to the nested structure such a calculation would have to be done from the most outer layer to the most inner layer, leading to an intractable approach [PDM12]. Some recent work proposes a tractable risk averse SDDP algorithm which allows the computation of upper bounds [GLCP23]. We are then interested in examining whether the BL-SDDP algorithm provides better lower bound estimates in a shorter amount of time.

Fig. 3.11 presents the results when employing the risk measure introduced in section 3.2.2 for different choices of weighting parameters $\lambda \in [0, 1]$. As we can observe, the BL-SDDP method, which corresponds to the blue color, is consistently able to achieve a better lower bound relative to the standard SDDP method, which corresponds to the red color.

3.5.6 BL-SDDP on models with a transmission network

The present subsection aims at testing the BL-SDDP algorithm on a known hard instance of the Brazilian interconnected power system, which is described in [STdCS13]. In particular, this allows us to test the effectiveness of BL-SDDP in a system with a transmission network. Inflow uncertainty is modeled as a geometric periodic autoregressive (GPAR) process [STdCS13]. Two scenarios are considered: (i) Following [STdCS13], inflows are modeled as decision vari-

ables. This requires additional equality equations in the optimization problem. We refer to this setting as the time series approach. (ii) Following [LS19], the GPAR process is discretized to a Markov chain using vector quantization. We refer to this setting as the Markov Chain approach.

Figures 3.12, 3.13 present the convergence evolution when using 20 CPUs. The former is the Markov Chain case, while the latter is the time series case. The algorithms use 20 forward samples and, in the case of BL-SDDP, a full batch update is performed every 5 iterations. In both cases, BL-SDDP is able to provide a notable improvement as compared to standard SDDP. As in [LWM13], we observe that convergence of the time series approach is slower than with the Markov chain approach.

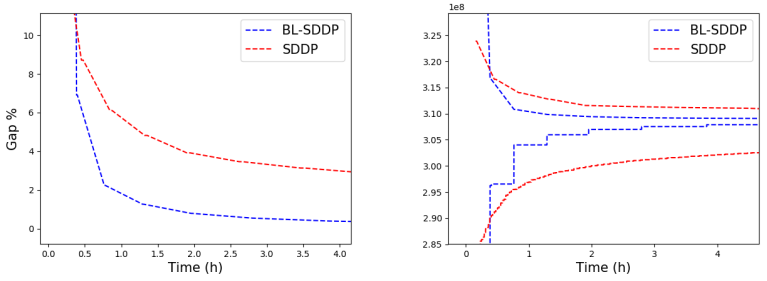


Figure 3.12: Convergence evolution for the Brazilian hydrothermal test case using 20 CPUs. The inflows are modeled as a Markov chain.

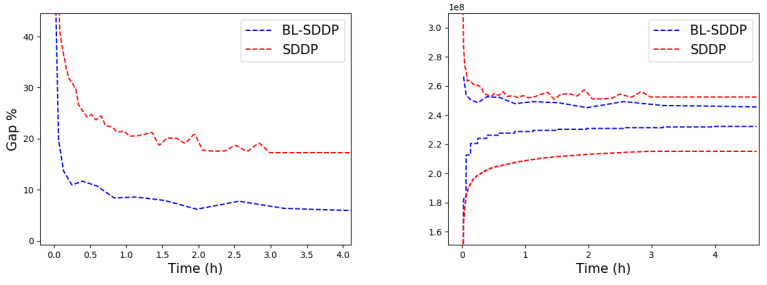


Figure 3.13: Convergence evolution for the Brazilian hydrothermal test case using 20 CPUs. The inflows are modeled using the time series approach.

The discussed Brazilian time series model is known to be a difficult problem. To finalize this section, we have let our BL-SDDP algorithm run for a longer period of time. The convergence evolution is presented in Figure 3.14. The total run time was 37 hours using 80 CPUs. The deviations in the gap are due to the fact that, for this particular problem, obtaining a reliable upper bound requires a considerable amount of samples. Therefore, for computational reasons, we

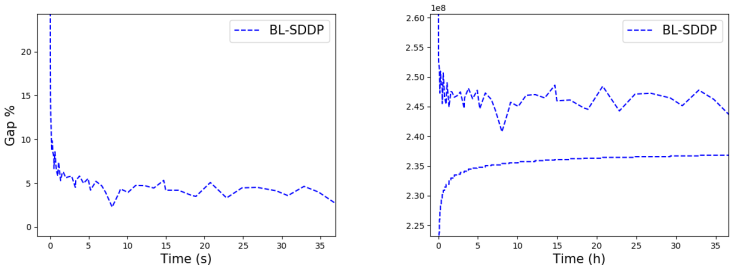


Figure 3.14: Convergence evolution for the Brazilian hydrothermal test case using 80 CPUs for 37 hours of run time. The upper panel shows the case with serial independence. The lower panel shows the Markov Chain case.

have restricted the upper bound measurement to 5000 inflow scenarios. This measurement is made every few iterations. At the end of the run time, we perform a reliable upper bound measurement by testing the policy on 200 000 inflow scenarios. Table 3.2 presents the results of this measurement, where we observe that we were able to solve the problem up to a 3.6% optimality gap. To the best of our knowledge, this is the best gap achieved so far for this particular instance.

Table 3.2: Obtained solution: Brazil time series model

	cost	lower 95 % C.I.	upper 95 % C.I.
lower bound	2.369e8	-	-
upper bound	2.454e8	2.449e8	2.459e8
gap (%)	3.58		

3.6 Conclusions

In this chapter we propose a novel variant of the SDDP algorithm by applying ideas from the reinforcement learning framework in the multistage stochastic programming framework. The SDDP variant that we propose, which we refer to as *Batch Learning SDDP*, is shown to improve the convergence behavior of the SDDP algorithm. The algorithm is evaluated on a realistic instance of hydrothermal scheduling in Colombia and Brazil. The main observations of this chapter are summarized below, in the form of the following conclusions.

(i) We propose a description of SDDP as a reinforcement learning technique and (ii) propose an algorithm that applies ideas of reinforcement learning in the SDDP framework. (iii) The BL-SDDP algorithm converges faster than the

commercial PSR SDDP software. (iv) The BL-SDDP parallel scheme exhibits favorable performance in terms of parallel efficiency. (v) The proposed BL-SDDP algorithm can handle different risk measures such as risk neutral and risk averse formulations. (vi) The BL-SDDP algorithm narrows the optimality gap of the tested instances by relying on fewer iterations than SDDP and thus provides lighter descriptions of the value functions.

In our work, we demonstrate how the combination of reinforcement learning ideas and the standard SDDP algorithm can lead to a novel scheme with superior convergence for a variety of case studies. This opens a research path for investigating whether other reinforcement learning techniques can be integrated into the SDDP framework. In particular, strategies for selecting a proper batch could prove to be of great help for the performance of the algorithm. Furthermore, we have proposed a parallel variant with positive results. Nevertheless, as the algorithm is synchronous, it remains susceptible to well-known synchronization issues [BT89]. This motivates the question of how to improve the scheme so as to avoid synchronization bottlenecks and thus improve the usage of computational resources. These questions will be investigated in future research.

3.A Results for an inventory test case with lead times

In this section we present the results obtained for an inventory control problem with lead times and lost sales. Given T stages, and assuming uncertain demand, which is assumed to follow a Markov Chain, the problem can be modeled as a multistage stochastic program. We denote the expected value function as $\mathcal{V}_{t+1}(z_t, \xi_t)$. We then compute the value function $V_t(z_{t-1}, \xi_t)$ by solving the following problem at each stage:

$$
\begin{aligned}
V_t(z_{t-1}, \xi_t) = \max \sum_{n \in \mathcal{N}} & P \cdot s_{t,n} - HC \cdot v_{t,n} - PC \cdot x_{t,n} + \mathcal{V}_{t+1}(z_t, \xi_t) \\
s.t.\ & v_{t,n} = v_{t-1,n} + x_{t-L,n} - s_{t,n} && n \in \mathcal{N} \\
& s_{t,n} \leq v_{t-1,n} && n \in \mathcal{N} \\
& s_{t,n} \leq D_{t,n}(\xi_t) && n \in \mathcal{N} \\
& x_{t,n} \leq C && n \in \mathcal{N} \\
& v_{t,n}, s_{t,n}, x_{t,n} \geq 0 && n \in \mathcal{N}
\end{aligned}
$$

The variables can be described as follows:

$z_{t,n}$: The state variable. This is a vector containing the on-hand inventory $v_{t,n}$ for product $n \in \mathcal{N}$, as well as the ordered quantities $x_{t-l,n}$ for $l = 1, ..., L$ with L being the lead time.

$s_{t,n}$: Variable representing the amount of sold items for product $n \in \mathcal{N}$.

The parameters can be described as follows:

L: The considered lead time.

P: The sales price.

HC: The inventory holding cost.

PC: the purchase cost.

$D_{t,n}$: The demand for product $n \in \mathcal{N}$.

C: The buying capacity.

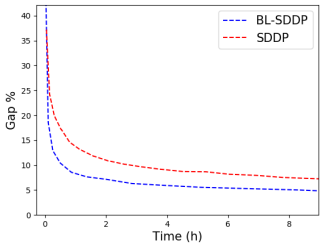
The sets are defined as:

$\mathcal{N}$: The set of possible products.

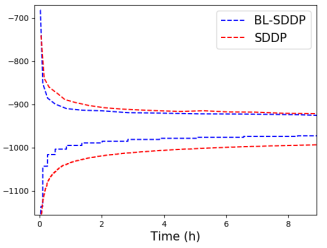
The instance that we consider is defined for 1 product and a lead time of 5 periods, thereby producing a 6-dimensional state space. The considered time

horizon is equal to 50 stages. We consider 100 realizations of uncertainty at each stage.

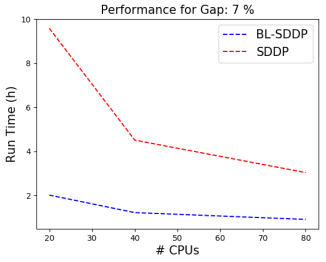
Following the experiments performed for the hydrothermal setting, Figure 3.15 presents the parallel performance of the BL-SDDP scheme. Panels (a) and (b) show the convergence evolution when using 20 CPUs, while Panels (c) and (d) present the performance when increasing the CPU count. As in the case of the hydrothermal setting, the BL-SDDP scheme is able to achieve better performance.



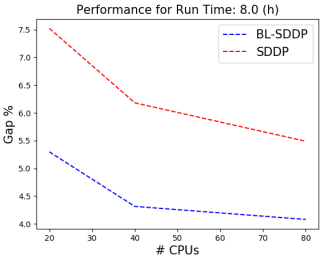
(a) Gap evolution.



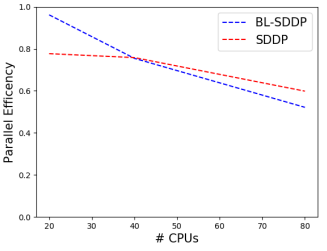
(b) Evolution of bounds.



(c) Performance for 7% optimality gap.



(d) Performance for 8 hours of run time.

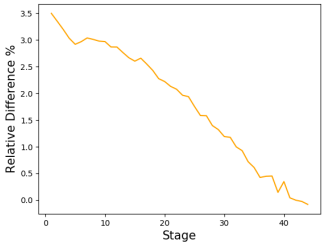


(e) Parallel efficiency.

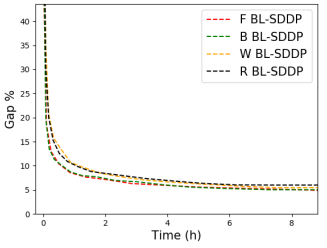
Figure 3.15: Parallel performance of the BL-SDDP algorithm. Panels (a) and (b) present convergence for the inventory test case using 20 CPUs, while panels (c) and (d) present the effect of CPU increase. Panel (e) presents the resulting parallel efficiency.

Figure 3.16 presents a summary of the variety of experiments that were

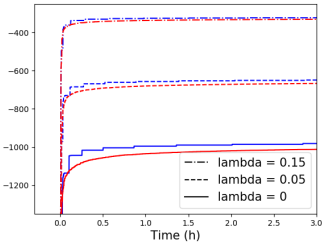
performed in this chapter, applied to the inventory test case. Panel (a) shows the value function comparison (BL-SDDP against SDDP). Panel (b) presents the results obtained when using different batch choice strategies. Finally, panel (c) shows the results when using the CVaR measure. These results are in accordance with the results shown in this chapter and further strengthen the case for the proposed methodology.



(a) Value function comparison.



(b) Batch choices.



(c) CVaR comparison.

Figure 3.16: Inventory test case experiments.

3.B Colombian Hydrothermal Scheduling Problem Formulation

The problem aims at minimizing the operational costs of thermal plants and curtailed demand by determining optimal water levels in the hydro reservoirs.

For the uncertainty model, we have considered stage-wise dependent inflows. The inflow model that we consider corresponds to the PAR inflow model used in the PSR SDDP software. Concretely, we have run the PSR software using the PAR(1) inflow model option. To make the comparison against PSR SDDP as fair as possible, we store the parameters of the PAR(1) model. Using these parameters, we run our SDDP algorithm. This implies that the inflow $z_{t,n}$ at

location n and stage t follows the equation:

$$\frac{z_{t,n} - M_{m,n}}{\Sigma_{m,n}} = \Phi_{m,n}\Big(\frac{z_{t-1,n} - M_{m-1,n}}{\Sigma_{m-1,n}}\Big) + A_{t,n}$$

Here, $M_{m,n}$ is the mean inflow of month m, $\Sigma_{m,n}$ is the standard deviation of inflow for month m, $\Phi_{m,n}$ is the auto-regressive parameter of order 1, and $A_{t,n}$ is a noise term with 0 mean and variance $\Sigma^2_{m,n}$. Further details regarding the uncertainty model can be found in [PSR].

Following [STdCS13], the inflow z_t is considered as a state variable, the previous equation is included in the optimization, and the noise A_t is discretized. Note that this inflow model was selected so as to have a model that is identical to PSR SDDP, and thus provide a comparison that is as fair as possible.

We can now formulate the problem. Let $\mathcal{V}_{t+1}(v_t, \xi_t)$ denote the expected value function. Each stage is described in terms of the value function $V_t(v_{t-1}, \xi_t)$, which is defined by solving the following problem:

$$\begin{aligned}
V_t(v_{t-1}, \xi_t) = \min\ & \sum_{n\in\mathcal{G}} C_n \cdot g_{n,t} + \text{VOLL} \cdot ls_t + \mathcal{V}_{t+1}(v_t, \xi_t) \\
s.t.\ & \sum_{n\in\mathcal{H}} P_n \cdot q_{t,n} + \sum_{n\in\mathcal{G}} g_{t,n} + ls_t = L_t \\
& v_t = v_{t-1} + z_t + M(q_t + s_t) \\
& \frac{z_{t,n} - M_{m,n}}{\Sigma_{m,n}} = \Phi_{m,n}\Big(\frac{z_{t-1,n} - M_{m-1,n}}{\Sigma_{m-1,n}}\Big) + A_{t,n}(\xi_t) \ \ \forall n \in \mathcal{H} \\
& g_t \le \bar{G} \\
& v_t \le \bar{V} \\
& q_t \le \bar{Q} \\
& g_t, v_t, q_t, s_t \ge 0
\end{aligned}$$

The sets are described as follows:

$\mathcal{H}$: The set of reservoirs.

$\mathcal{G}$: The set of thermal plants.

The variables are described as follows:

$v_{t,n}$: The storage level of reservoir $n \in \mathcal{H}$.

$z_{t,n}$: The inflow at location $n \in \mathcal{H}$.

$q_{t,n}$: Water turbined outflow of reservoir $n \in \mathcal{H}$.

$s_{t,n}$: Spilled volume of water at reservoir $n \in \mathcal{H}$.

$g_{t,n}$: Vector of thermal power from $n \in \mathcal{G}$.

ls_t: Variable which accounts for load shedding.

The parameters are given as follows:

C_n: The generation cost of thermal plant $n \in \mathcal{G}$.

P_n: The energy generation coefficient for the turbined outflow for hydro plant $n \in \mathcal{H}$.

L_t: Load at stage t.

A_t: The inflow vector.

M : Matrix representing the hydrological topology.

$\bar{G}, \bar{V}, \bar{Q}$: Upper limits on the variables.

$M_{m,n}$: Mean inflow of month m.

$\Sigma_{m,n}$: Standard deviation of inflow for month m.

$\Phi_{m,n}$: Auto-regressive parameter of order 1.

$A_{t,n}$: Noise term with 0 mean and variance $\Sigma^2_{m,n}$.

3.C Brazilian Hydrothermal Scheduling Problem Formulation

The formulation of the problem follows the mathematical description presented in Chapter 2. The difference relates to the uncertainty model, for which we have considered two alternatives: (i) the time series model presented in [STdCS13], and (ii) the Markov Chain model, as discussed in [LS19].

The former model fits the historical (1931 to 2008) log-inflows into a first-order periodic auto-regressive process, referred to as a geometric periodic auto-regressive process (GPAR). This leads to a model of the form:

$$\ln z_t = \mu_t + \Phi_t \ln z_{t-1} + \epsilon_t, \epsilon_t \sim \mathcal{N}(0, \Sigma_t)$$

Here, μ_t, Φ_t, Σ_t have a periodical behavior which spans 12 months and the error terms ϵ_t are stagewise independent.

One way to include this uncertainty model into the problem is to consider the inflow z_t as a state variable and add the previously described equation as a constraint of the optimization problem. Unfortunately, such an approach would destroy the overall structure of each subproblem (i.e. it would destroy the value function convexity). To deal with this, the authors of [STdCS13] propose a linearization of said equation by using a first-order approximation:

$$z_t = \exp^{\epsilon_t} \circ (A_t + \Phi_t z_{t-1}), \epsilon_t \sim \mathcal{N}(0, \Sigma_t)$$

Here, A_t, Φ_t, Σ_t have a periodical behavior which spans 12 months and the error terms ϵ_t are stagewise independent. The notation $\circ$ indicates that the operations are performed componentwise. The time series formulation proceeds by adding this last equation into the optimization problem. Our numerical experiments are run with the same parameters as in [STdCS13].

On the other hand, the Markov Chain model described in [LS19] proceeds by considering a large collection of inflow scenarios (over the whole planning period). These inflow scenarios are then partitioned, stage by stage, thus creating a lattice of inflows which serves as the uncertainty model for the stochastic problem. Note that, for this approach, a mere training set consisting of inflow paths is required. Thus, no linearization of the GPAR inflow model is required, as opposed to the previously discussed time series model. Furthermore, note that, in this approach, there is no need to consider the inflow as a state variable.

Part II

Parallel Computing and Two-stage Stochastic Programming: European Resource Adequacy Assessment

4 Applying High-Performance Computing to the ERAA study

4.1 Introduction

Within an uncertain world, measuring and analyzing the ability of the electric power system to react to adverse uncertain conditions has become increasingly important. The Clean Energy Package [Com19a] has recognized the importance of this task with Regulation (EU) 941/2019 [Com19b] and Regulation (EU) 943/2019 [Com19c]. The latter stipulates the need for a robust European resource adequacy assessment that provides an instrument for detecting and measuring adequacy concerns [erac]. In particular, resource adequacy concerns identified through the European Resource Adequacy Assessment (ERAA) are to become the basis for justifying the implementation of capacity mechanisms within European Member States. As required in Regulation (EU) 943/2019 [Com19c], Member States wishing to introduce capacity mechanisms must do so on the basis of an adequacy concern that is identified in the ERAA study, complemented possibly with a national resource adequacy study. Consequently, there is an institutional urge to develop a reliable and robust ERAA study at a pan-European level.

The European Network of Transmission System Operators for Electricity (ENTSO-E) is the body mandated by regulation to develop the methodology and conduct the study on an annual basis [erad]. The ERAA is a pan-European resource adequacy assessment for up to 10 years ahead, which aims at measuring the ability of the power system to react to a set of future uncertain events [erac]. The ERAA study covers the entire pan-European interconnected system, thus modeling 56 bidding zones in 37 countries. Adequacy concerns are identified with the help of adequacy indicators, with the Loss of Load Expectation (LOLE) being the most common indicator used among the EU Member States to define the respective reliability standards. The Expected Energy not Served (EENS) indicator is also computed in the ERAA study, in order to assess the depth (MW) of the curtailments. Such indicators are naturally linked to the installed capacity mix. For the purpose of determining the installed capacity mix, the ERAA study introduces the so-called Economic Viability

Assessment (EVA), which aims at modeling economic parameters that affect the available generation capacity within Europe. Unfortunately, due to the scale of the ERAA, incorporating an EVA that integrates the stochastic nature of future events has thus far been considered as being out of scope. Therefore, a deterministic approach has been followed in ERAA 2021 [erab], while in ERAA 2022 a tractable formulation is obtained by considering a reduced stochastic model. This simplification consists of using 3 of 35 uncertainty realizations. ENTSO-E is moving towards a stochastic programming formulation [erae], endorsed by ACER, but currently limited to three scenarios. This motivates the development of a framework which allows tackling the EVA in its stochastic version.

Our work aims at bridging this methodological gap by proposing a novel parallel computing algorithm, based on ideas from stochastic programming, and implemented on high-performance computing infrastructure. Our approach allows us to account for the stochastic nature of the EVA study in ERAA. Furthermore, we study the potential impact of ignoring the stochastic nature of the EVA on the capacity mix, and in turn, the consequences that this could have on adequacy indicators.

The EVA aims at determining capacity expansion and capacity retirement opportunities for the entire European network. As such, it relates to two streams of literature: (i) stochastic capacity generation expansion, with the added complexity of considering retirement opportunities; (ii) large-scale optimization, which is tackled with the aid of high-performance computing.

4.1.1 Stochastic capacity generation expansion

The stochastic capacity generation expansion literature presents a variety of strategies for addressing generation expansion[1]. When the size of the problem is manageable, the problem is solved directly by a commercial solver [BBC12, AABA14, GAC14, BAT21]. In order to decrease the computational burden, certain authors have followed scenario selection techniques, and report solving time improvements with corresponding losses in the quality of the solution [KS10, JRWW11, FR13, GAC14]. Note that these approaches typically still rely entirely on a commercial solver for solving the reduced problem. By relying on stochastic programming formulations [BL11], certain authors use Benders decomposition in order to tackle the problem [Blo82, GCCP93, STK07, RSW09]. Furthermore, modelling tools have been developed in order to solve the problem in practical applications. Examples of open-source software include the EMPIRE model [BSdG$^+$22] which solves the problem as a large-scale linear problem, while [RM18] has proposed extensions of EMPIRE where the problem is decomposed by using progressive hedging. Recent approaches [SSAG22], applied to the Brazilian power system, have proposed

[1] Note that transmission expansion is out of scope for the EVA, and thus we do not concern ourselves with this aspect in the present work.

extensions to Benders decomposition in order to speed up convergence. Examples of commercial tools include the PLEXOS software, which allows solving the problem as a large-scale linear problem or using Benders decomposition. It is worth emphasizing that the PLEXOS software has been the modelling tool used by ENTSO-E for their past studies. Certain heuristics have also been proposed, including a rolling-horizon scheme [PB16] and an hourly aggregation of time series [GBD21].

The modelling specifications of the EVA prevent us from solving the problem directly using a commercial solver. These specifications result in a large-scale problem, due to its wide geographical scope, combined with the time step chosen by ENTSO-E, as well as the endogenous representation of uncertainty. We highlight that a small set of uncertainty realizations are already enough to produce a problem of considerable size, thus preventing the use of a scenario reduction technique which allows tackling the problem in its extended for a reduced number of scenarios. On the other hand, a drawback of Benders decomposition is that its performance diminishes as the number of expansion / retirement candidates increases. The progressive hedging algorithm presented in [RM18] appears to deteriorate in performance as the size of each scenario subproblem increases, which renders it impractical for our purposes.

Some early theoretical work from 1986 [SY86] presented the possibility of employing subgradient schemes in deterministic capacity expansion problems. Furthermore, past research has proven the effectiveness of subgradient algorithms when dealing with large instances of stochastic unit commitment [POO11, AP20]. Consequently, our work aims at translating this success into the ERAA study. Due to the EVA modelling specifications, the calculation of each subgradient is time-consuming. Therefore, our work, in addition to putting forward a subgradient-based algorithmic framework, proposes a further algorithm that calculates approximations of the subgradients efficiently, thus providing notable computational benefits.

4.1.2 High performance computing

As highlighted in previous chapters, high performance computing has become critical in tackling large-scale power system optimization problems. The parallelization of decomposition techniques such as Benders decomposition or subgradient methods has been studied in the past [MPG87, TPPM90, BB03, POR14, AP21]. Nevertheless, the stochastic capacity expansion literature has not entirely exploited the benefits of parallel computing. In this work, we bridge this methodological gap by proposing a parallelization strategy for the considered algorithms, which allows us to arrive at solutions for the stochastic formulation of EVA within a few hours of computation.

4.1.3 Organization and contributions

(i) In terms of methodological contribution, we solve the stochastic EVA considering all available uncertainty conditions. (ii) Our algorithmic contribution amounts to proposing a parallel computing subgradient-based method and a novel parallel computing second-stage relaxation scheme, which decreases computational burden significantly. These algorithms are benchmarked against the commonly used Benders decomposition scheme (which is also the algorithm that the PLEXOS commercial software uses) and the progressive hedging scheme. (iii) Our policy analysis contribution amounts to testing the quality of the solution by comparing it against the ERAA 2021 deterministic approach.

The organization of the work is as follows. The EVA is formulated as a two-stage stochastic capacity expansion problem in section 4.3. This formulation leads to a large-scale stochastic programming problem. A customized algorithm for tackling such a problem is proposed in section 4.4. Section 4.5 proposes parallel schemes for the described algorithms. Finally, the results are discussed in section 4.6.

4.2 European resource adequacy assessment

The European resource adequacy assessment proposes a methodology for measuring the ability of the power system to react to uncertain events [EE21]. The overall methodology consists of two main blocks. The first block aims at determining investment and retirement opportunities (which we refer to as the expansion plan), the so-called Economic Viability Assessment (EVA). The second block uses these opportunities in order to measure adequacy indicators.

Figure 4.1 provides a visual representation of the overall ERAA methodology. During the first step (the EVA), a variety of uncertain climate conditions are considered (ENTSO-E considers a total of 35 climatic years for ERAA 2021 and ERAA 2022). The objective is to decide on investment and retirement that minimize the expected operational cost of the system. Note that this has to be decided before the realization of uncertainty, thus leading to a stochastic problem. During the second step, the adequacy indicators are calculated. The previously computed expansion plan is used for this purpose, and it is evaluated over an uncertainty set that consists of climate years as well as random outage patterns.

We highlight a critical difference between these steps. Step 2 can be decoupled into several independent problems, one for each climate year and outage pattern. This implies that it is computationally tractable. The situation regarding step 1 is considerably more involved, as it naturally links all climate years into a single problem, meaning that it is not possible to decouple the climatic years into independent problems. In our work, we focus on the first step. In order to tackle it, we propose a parallel computing algorithmic framework.

Due to increased computational complexity, ENTSO-E has considered simplified approaches in order to tackle step 1. For ERAA 2021 the simplification

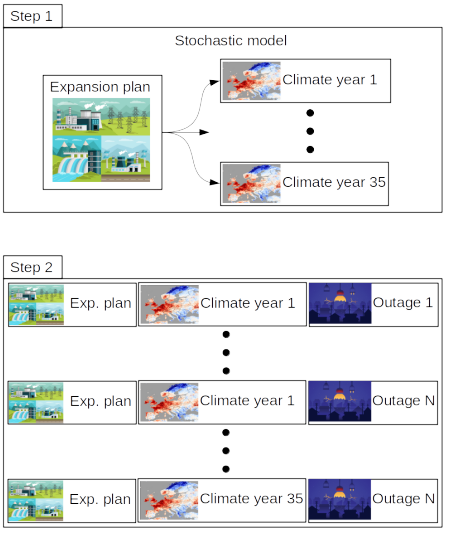


Figure 4.1: The ERAA methodology consists of 2 steps. The first step, the so-called economic viability assessment, is calculated using a set of uncertain climatic conditions. The second step computes adequacy indicators, which use the previously computed expansion plan over an uncertainty set that consists of climatic conditions as well as random outages.

has been two-fold. (i) On the one hand, a scenario reduction methodology is applied which selects 7 out of 35 climatic years. (ii) The optimal expansion plan for each one of the selected climatic years is calculated, thus obtaining 7 expansion plans. The average between these expansion plans is selected as the approximate solution of the stochastic problem. For ERAA 2022, a stochastic model is proposed. However, due to increased computational complexity, the model is reduced to 3 out of 35 climatic years, which are selected using a scenario reduction technique.

ENTSO-E has provided access to the ERAA 2021 input data for our team in the context of this work. For ERAA 2021, the EVA is modeled as a two-stage problem: the expansion plan is decided for a single target year. Note that ERAA 2022 adopts a multi-stage approach, namely there are consecutive years and the expansion is decided just before each year. In this work, we focus on ERAA 2021, which is the more natural step, since not even this model can be tackled in its stochastic form by state-of-the-art solvers.

The ERAA 2021 data considers the entire interconnected pan-European network, which is represented by considering 56 bidding zones that correspond to 36 countries. This is depicted in Figure 4.2. In terms of generators, the data contains the installed capacity mix per zone. We highlight that the generator data is aggregated per technology, per zone. A detailed overview of the installed capacity mix per zone can be found in [eraa]. The generator data also

Figure 4.2: The pan-European power system modelled in the ERAA 2021.

contains time series of planned maintenance of generators, de-rating factors, and must-run profiles of the generators. The data represents the transmission network as a transportation network, and includes transmission lines between zones, together with their limits. The data contains several candidate retirement opportunities, per zone, for thermal generators. Two thermal investment candidates, per zone, are considered. The uncertainty arises in the form of the so-called climatic years, each one being a time series (per zone) of demand, PV, wind, and inflow profiles. ENTSO-E has considered 35 climatic years for the EVA of the ERAA 2021 edition, which are depicted in Figure 4.3.

ENTSO-E uses the ERAA 2021 data as input for the PLEXOS modeling tool for compiling the EVA expansion problem. As this tool is unable to tackle the problem in its stochastic form, we instead use the exact same input data in order to put together an open-source Julia [BEKS17] version, which enables us to develop a decomposition scheme for solving the problem, using the full set of 35 climatic years. We highlight that both models are built with the same modelling specifications. In particular, similarly to the EVA PLEXOS model, ours is a linear model which takes into account the installed capacity mix per zone for proposing investment and retirement decisions (meaning that it is not a greenfield model).

4.3 Expansion problem

The stochastic capacity expansion problem is formulated as a two-stage stochastic program [BL11]. The first stage determines investments and retirements of technologies. The second stage solves an economic dispatch over a target year. The introduction of uncertainty takes place during the second stage. Each uncertainty realization corresponds to a so-called climatic year, which consists of a time series of demand, solar production, wind production, and hydro inflows.

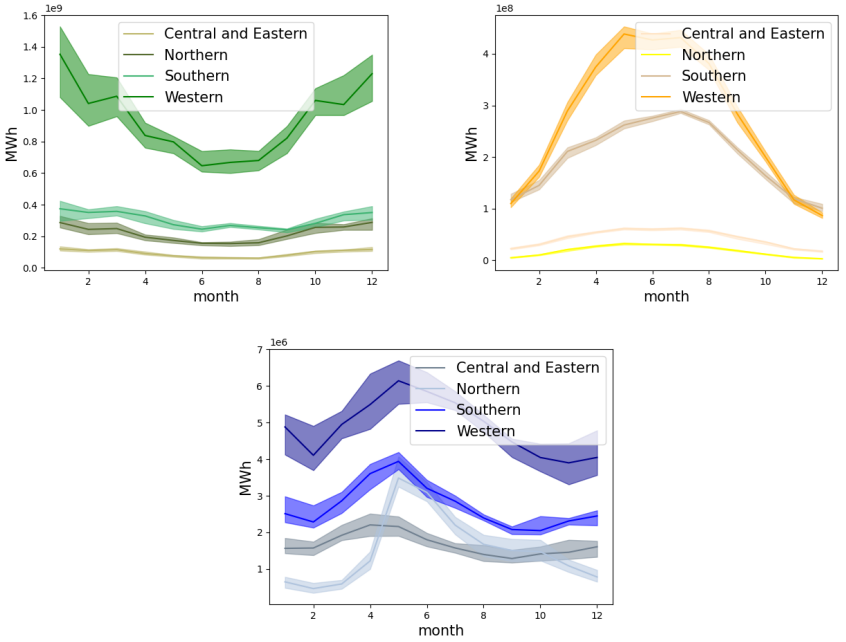


Figure 4.3: An overview of climatic years of the 4 main European regions. The upper and lower bounds represent the 0.75 and 0.25 quantiles. The upper left panel presents wind production, the upper right is PV production, while the lower figure corresponds to hydrological inflows.

In order to ease the exposition, the sets, variables, and parameters of the problem are presented in the appendix. Furthermore, parameters are denoted in upper case while optimization variables are in lower case.

4.3.1 Expansion constraints

There is a maximum amount of plausible invested/retired capacity. This is modeled with the following constraints:

$$x_{n,g} \leq X_{n,g} \text{ for all } n \in \mathcal{N}, g \in \mathcal{G} \tag{4.1}$$
$$x^*_{n,g} \leq X^*_{n,g} \text{ for all } n \in \mathcal{N}, g \in \mathcal{G}^* \tag{4.2}$$

4.3.2 Generator constraints

Given $\omega \in \Omega$, the minimum and maximum power generation capabilities of units are described by the following constraints:

$$p^*_{n,g,t,\omega} \leq x^*_{n,g} \text{ for all } n \in \mathcal{N}, g \in \mathcal{G}^*, t \in \mathcal{T}, \omega \in \Omega \tag{4.3}$$

$$p_{n,g,t,\omega} \leq P^{max}_{t,n,g} - x_{n,g} \tag{4.4}$$
$$P^{min}_{t,n,g} \leq p_{n,g,t,\omega} \tag{4.5}$$
$$\text{for all } n \in \mathcal{N}, g \in \mathcal{G}, t \in \mathcal{T}$$

Constraints 4.3, 4.4 model the power production of new and existing capacity. Constraint 4.5 models must-run obligations.

4.3.3 Transmission network

The transmission network is modeled as a transportation network. The constraints are then bounds on the transfer capacity. Given $\omega \in \Omega$, these can be described by the following constraints:

$$L^{min}_{n,l} \leq f_{n,l,t,\omega} \leq L^{max}_{n,l} \tag{4.6}$$
$$\text{for all } n \in \mathcal{N}, l \in \mathcal{L}(n), t \in \mathcal{T}$$

Transportation models fail to accurately represent the true physics of power flow in European network models. This has motivated the introduction of a zonal PTDF approximation in European market clearing models. This implies the addition of linear constraints, which do not affect the overall structure of the decomposition schemes that are proposed in this chapter.

4.3.4 Batteries

Batteries are modeled as energy storage resources available per zone. For a given $\omega \in \Omega$, they are modeled as follows:

$$bv_{n,t,\omega} = bv_{n,t-1,\omega} + BCE \cdot bc_{n,t,\omega} - BDE \cdot bd_{n,t,\omega} \tag{4.7}$$
$$bv_{n,t,\omega} \leq BV \tag{4.8}$$
$$bc_{n,t,\omega} \leq BC \tag{4.9}$$
$$bd_{n,t,\omega} \leq BD \tag{4.10}$$
$$\text{for all } n \in \mathcal{N}, t \in \mathcal{T}$$

Constraint 4.7 models the load balance of the battery. Constraints 4.8, 4.9, 4.10 model the maximum capacity, charge, and discharge capabilities of the battery.

4.3.5 Hydro power

Hydropower generation is modeled using four different hydro technologies: run-of-river, reservoir, pumped storage open loop, and pumped storage closed loop. The hydrology is simplified by ENTSO-E by considering the aggregated hydrological capabilities of each zone. The inflows are measured as equivalent energy inflows (MW). For a given $\omega \in \Omega$, each one of these technologies is modeled as

follows. For ease of exposition, a variable associated with a certain technology has an associated subscript: run-of-river R, reservoir S, pumped storage open loop O, and pumped storage closed loop C.

• Run-of-River. There is no storage capability, therefore the net inflows are considered as turbined water.

$$q_{n,R,t,\omega} = A_{n,R,t,\omega} \text{ for all } n \in \mathcal{N}, t \in \mathcal{T} \tag{4.11}$$

• Reservoir: There is storage capability, consequently the water can be turbined or stored.

$$v_{n,S,t,\omega} = v_{n,S,t-1,\omega} + A_{n,S,t,\omega} - q_{n,S,t,\omega} - s_{n,S,t,\omega} \tag{4.12}$$
$$v_{n,S,t,\omega} \leq V_{n,S} \tag{4.13}$$
$$q_{n,S,t,\omega} \leq Q_{n,S} \tag{4.14}$$
$$\text{for all } n \in \mathcal{N}, t \in \mathcal{T}$$

Constraint 4.12 describes the reservoir dynamics, while constraints 4.13, 4.14 describe the maximum storage and maximum power production capacities respectively.

• Pumped storage open loop: There are head and tail reservoirs. The head turbines water to the tail reservoir, thus producing power. In low-demand periods the tail pumps water back to the head reservoir, thus consuming power. The system is additionally exposed to natural inflows.

$$v^H_{n,O,t,\omega} = v^H_{n,O,t-1,\omega} + A_{n,O,t,\omega} + PE \cdot d_{n,O,t,\omega} - q_{n,O,t,\omega} - s_{n,O,t,\omega} \tag{4.15}$$
$$v^T_{n,O,t,\omega} = v^T_{n,O,t-1,\omega} - PE \cdot d_{n,O,t,\omega} + q_{n,O,t,\omega} \tag{4.16}$$
$$v^H_{n,O,t,\omega} \leq V_{n,O} \tag{4.17}$$
$$q_{n,O,t,\omega} \leq Q_{n,O} \tag{4.18}$$
$$d_{n,O,t,\omega} \leq D_{n,O} \tag{4.19}$$
$$\text{for all } n \in \mathcal{N}, t \in \mathcal{T}$$

Constraints 4.15, 4.16 are the water balance constraints of the head and tail reservoirs respectively. Note that the head reservoir is subject to rainfall uncertainty. Constraint 4.17 bounds the maximum storage, while constraints 4.18, 4.19 bound the turbined and pumped water respectively.

• Pumped storage closed loop: this is modeled in the same way as the open loop case, with the difference that no natural inflows are considered.

$$v^H_{n,C,t,\omega} = v^H_{n,C,t-1,\omega} + PE \cdot d_{n,C,t,\omega}$$

$$- q_{n,C,t,\omega} - s_{n,C,t,\omega} \tag{4.20}$$

$$v^T_{n,C,t,\omega} = v^T_{n,C,t-1,\omega} - PE \cdot d_{n,C,t,\omega} + q_{n,C,t,\omega} \tag{4.21}$$

$$v^H_{n,C,t,\omega} \leq V_{n,C} \tag{4.22}$$

$$q_{n,C,t,\omega} \leq Q_{n,C} \tag{4.23}$$

$$d_{n,C,t,\omega} \leq D_{n,C} \tag{4.24}$$

$$\text{for all } n \in \mathcal{N}, t \in \mathcal{T}$$

4.3.6 Load balance

The load balance constraint aims at satisfying the demand of each zone using the resources of each zone plus imports from neighbouring zones. For a given $\omega \in \Omega$, it is formulated as follows:

$$\begin{aligned} & p_{n,g,t,\omega} + p^*_{n,g,t,\omega} + bd_{n,t,\omega} + \sum_{h \in \mathcal{H}(n)} q_{n,h,t,\omega} + \sum_{l \in \mathcal{L}(n)} f_{n,l,t,\omega} \\ & + ls_{n,t,\omega} + \mathcal{PV}_{n,t,\omega} + \mathcal{W}_{n,t,\omega} \\ & = \mathcal{D}_{n,t,\omega} + ps_{n,t,\omega} + bc_{n,t,\omega} + \sum_{r \in \{C,O\}} d_{n,r,t,\omega} \\ & \text{for all } n \in \mathcal{N}, t \in \mathcal{T} \end{aligned} \tag{4.25}$$

We highlight that reserve requirements are modelled in the ERAA 2021 study as extra load. More accurate models for reserve requirements have been studied in the past [PO13]. Their inclusion implies the addition of linear constraints which do not disrupt the overall structure of the algorithms developed in this paper.

4.3.7 Objective function

For a given $\omega \in \Omega$, we define the following quantities:

$$\sum_{n \in \mathcal{N}, g \in \mathcal{G}^*} IC_g \cdot x^*_{n,g} + FOM_g \cdot x^*_{n,g} - \sum_{n \in \mathcal{N}, g \in \mathcal{G}} FOM_g \cdot x_{n,g} \tag{4.26}$$

$$\sum_{n \in \mathcal{N}, g \in \mathcal{G}, t \in \mathcal{T}} VOM_g \cdot p_{n,g,t,\omega} + FC_g \cdot p_{n,g,t,\omega} \tag{4.27}$$

$$\sum_{n \in \mathcal{N}, g \in \mathcal{G}^*, t \in \mathcal{T}} VOM_g \cdot p^*_{n,g,t,\omega} + FC_g \cdot p^*_{n,g,t,\omega} \tag{4.28}$$

$$\sum_{n \in \mathcal{N}, l \in \mathcal{L}(n), t \in \mathcal{T}} f_{n,l,t,\omega} \cdot WC \tag{4.29}$$

$$\sum_{n\in\mathcal{N},h\in\mathcal{H}(n),t\in\mathcal{T}} s_{n,h,t,\omega}\cdot SC \tag{4.30}$$

$$\sum_{n\in\mathcal{N},t\in\mathcal{T}} ls_{n,t,\omega}\cdot \text{VOLL} \tag{4.31}$$

Equation (4.26) is the first-stage cost. It consists of the investment cost plus the fixed maintenance cost of new capacity $x^*_{n,g}$, minus the fixed maintenance cost of retired capacity $x_{n,g}$. Equation (4.27) corresponds to the cost of producing $p_{n,g,t,\omega}$ units of power, similarly equation (4.28) corresponds to the generation cost of new capacity. Equation (4.29) corresponds to the cost of transporting power through the transmission network. Equation (4.30) corresponds to a penalty for water spillage. Finally, equation (4.31) is the cost of involuntary load shedding, which is penalized at VOLL. Putting these elements together leads to the stochastic capacity expansion problem, which is described as follows:

$$\begin{aligned}
&\min_{x,x^*}(4.26)+\mathbb{E}_\omega\Big[\min_{p_\omega,p^*_\omega,f_\omega,ls_\omega}(4.27)+(4.28)+(4.29)+(4.30)+(4.31)\Big]\\
&\text{s.t. } (4.1)-(4.2)\\
&\qquad (4.3)-(4.25) \text{ for all } \omega\in\Omega
\end{aligned} \tag{CEP}$$

Note that we have distinguished between two sets of constraints. The former refers to the so-called first-stage constraints, therefore they do not depend on $\omega \in \Omega$. The latter set of constraints constitutes the second-stage constraints, these depend on uncertainty, and there is one such set of constraints for each $\omega \in \Omega$. Note that this formulation implies that the first-stage variables $x_{n,g}, x^*_{n,g}$ are decided before uncertainty materializes, and thus these decisions do not depend on $\omega \in \Omega$.

4.4 Solution Strategy

The stochastic capacity generation expansion literature has often relied on Benders decomposition (also known as the L-Shaped method when uncertainty is introduced) [BL11]. This scheme performs poorly as the number of expansion / retirement possibilities increases. In view of this, we propose a subgradient algorithm that is better suited for such situations. This technique allows us to reduce the number of iterations. However, it can still be computationally costly, as the calculation of each subgradient is non-trivial. Consequently, we further propose a relaxation of the economic dispatch, which allows us to calculate subgradient approximations efficiently. The approximation is refined throughout iterations, thus ensuring convergence. Such a technique allows us to reduce the computational burden significantly. We begin by describing how we decompose the problem, followed by our algorithmic contributions.

Following standard procedures [BL11], we break down the overall problem into smaller subproblems, first by considering separate subproblems for the first and second stages, and secondly by considering a subproblem for each uncertainty realization for the second stage (recall the lattice description presented in Figure 1.1). Given an uncertainty realization ω and a first-stage decision x, x^*, the second-stage subproblem for a given ω is as follows:

$$\begin{aligned}\mathcal{V}(x, x^*, \omega) = \min_{p_\omega, p^*_\omega, f_\omega, ls_\omega} \; & (4.27) + (4.28) + (4.29) + (4.30) + (4.31) \\ \text{s.t. } & (4.3) \qquad (\lambda^*_{n,g,t,\omega}) \\ & (4.4) \qquad (\lambda_{n,g,t,\omega}) \\ & (4.5) - (4.25)\end{aligned}$$

where $\lambda^*_{n,g,t,\omega}, \lambda_{n,g,t,\omega}$ are dual multipliers of constraints (4.3), (4.4) respectively. These subproblems allow us to re-write the CEP problem as

$$\begin{aligned}&\min_{x,x^*} (4.26) + \mathbb{E}_\omega\Big[\mathcal{V}(x, x^*, \omega)\Big] \\ &\text{s.t. } (4.1) - (4.2)\end{aligned} \tag{CEP-R}$$

Using such a formulation, one could now employ, for instance, the L-Shaped method described in Alg. 1. However, as the computational experiments show subsequently, such a technique fails to converge in a reasonable amount of time, thus motivating the development of algorithmic alternatives.

4.4.1 Subgradient algorithm

A drawback of the L-Shaped scheme is that, as the dimensionality of x, x^* increases, additional supporting hyperplanes are required in order to describe $\mathbb{E}_\omega\Big[\mathcal{V}(x, x^*, \omega)\Big]$. Consequently, finding the optimal region may require many extra costly iterations. Instead, in this work we use a subgradient algorithm. Following the description presented in Alg. 2, given an initial expansion candidate $\widehat{x}, \widehat{x}^*$, we specifically calculate a subgradient of the objective function of the CEP-R problem around $\widehat{x}, \widehat{x}^*$ and update the candidate expansion plan along the direction of such a subgradient.

We decompose problem CEP by rewriting it as CEP-R, and so a subgradient of the objective function along the $x^*_{n,g}$ coordinate is:

$$\rho^*_{n,g} = IC_{n,g} + FOM_{n,g} + \mathbf{E}_\omega\Big[\sum_{t\in\mathcal{T}} \lambda^*_{n,g,t,\omega}\Big]$$

And a subgradient along the $x_{n,g}$ coordinate is:

$$\rho_{n,g} = -FOM_{n,g} + \mathbf{E}_\omega\Big[\sum_{t\in\mathcal{T}} \lambda_{n,g,t,\omega}\Big]$$

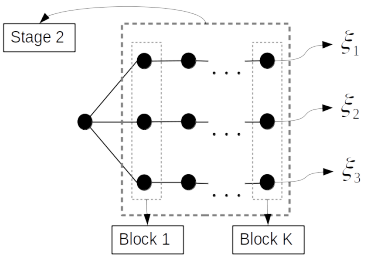


Figure 4.4: Chronological decomposition. The second stage is partitioned into several consecutive chronological blocks.

Note that the reformulation CEP-R decomposes the calculation of these slopes by calculating several subproblems, concretely by solving $\mathcal{V}(x, x^*, \omega)$ for each $\omega \in \Omega$. This decomposition allows us to apply the algorithmic scheme described in Alg. 2.

4.4.2 Second-stage relaxation algorithm

Each iteration of the subgradient scheme can be costly, because it carries the computational burden of solving each second-stage subproblem. For this reason, we propose a scheme that relaxes the second stage, namely the economic dispatch. Using such a relaxation, each iterate of a trial $\widehat{x}^i, \widehat{x}^{*i}$ becomes efficient, thus allowing us to increase the search speed. The relaxation can then be refined in order to tighten the search and ensure convergence.

Let us begin by describing the second-stage relaxation. To achieve this, we resort to dynamic programming [B$^+$11], and partition the second-stage horizon $1, \ldots, \mathcal{T}$ into K consecutive blocks: $\{1, t_1\}, \{t_1+1, t_2\}, \cdots, \{t_{K-1}+1, \mathcal{T}\}$. This leads to the representation shown in Figure 4.4, where the second stage has been partitioned into several blocks. The subproblems at each block are given by the dynamic programming equations. The equation at block k is given by:

$$
\begin{aligned}
&\mathcal{V}_k(x, x^*, bv_{t_{k-1},\omega}, v_{t_{k-1},\omega}, \omega) = \\
&\qquad \min \Big[(4.27) + (4.28) + (4.29) + (4.30) + (4.31)\Big]_{t=t_{k-1}+1}^{t_k} \\
&\qquad + \mathcal{V}_{k+1}(x, x^*, bv_{t_k,\omega}, v_{t_k,\omega}, \omega) \\
&\qquad \text{s.t. } \Big[(4.3) - (4.25)\Big]_{t=t_{k-1}+1}^{t_k}
\end{aligned}
$$

Here, the notation indicates that the objective function and the constraints are restricted to $t = t_{k-1} + 1, \ldots, t_k$. Note that, at the initial time boundary, i.e. at $t = t_{k-1} + 1$, constraint (4.7) requires the battery state of charge at $t = t_{k-1}$. Consequently, the notation indicates that $\mathcal{V}_k$ depends on $bv_{t_{k-1},\omega}$, the battery state of charge at stage t_{k-1}. Similarly, Eqs. (4.12), (4.15), (4.16),

(4.20), (4.21) require the reservoir level at stage t_{k-1}, and this information is summarized in the vector $v_{t_{k-1},\omega}$. At the final time boundary at $t = t_k$, the objective function includes $\mathcal{V}_{k+1}$, which captures the future costs of the system. Note that the subproblem of the first block satisfies $\mathcal{V}_1(x, x^*, \omega) = \mathcal{V}(x, x^*, \omega)$.

Each function $\mathcal{V}_{k+1}$ is piecewise linear convex in $x, x^*, bv_{t_k,\omega}, v_{t_k,\omega}$ [BL11], and so we can approximate it using supporting hyperplanes. Thus, given a collection of supporting hyperplanes $\{c_i(x, x^*, bv_{t_k,\omega}, v_{t_k,\omega})\}_{i=1}^N$, an approximation of the subproblem at the k-th block is given by:

$$\begin{aligned}
&\widehat{\mathcal{V}}_k(x, x^*, bv_{t_{k-1},\omega}, v_{t_{k-1},\omega}, \omega) = \\
&\quad \min \Big[(4.27) + (4.28) + (4.29) + (4.30) + (4.31)\Big]_{t=t_{k-1}+1}^{t_k} + \theta_{k+1,\omega} \\
&\quad \text{s.t. } \Big[(4.3) - (4.25)\Big]_{t=t_{k-1}+1}^{t_k} \\
&\qquad \theta_{k+1,\omega} \geq c_i(x, x^*, bv_{t_k,\omega}, v_{t_k,\omega}),\ i = 1 \dots N
\end{aligned}$$

In particular, the approximation of the first-block subproblem $\widehat{\mathcal{V}}_1$ is an approximation of the second-stage subproblem, namely the economic dispatch. Consequently, we can approximate problem CEP-R as follows:

$$\begin{aligned}
&\min_{x,x^*} (4.26) + \mathbb{E}_\omega\Big[\widehat{\mathcal{V}}_1(x, x^*, \omega)\Big] \\
&\text{s.t. } (4.1) - (4.2)
\end{aligned} \tag{CEP-A}$$

Calculating each $\widehat{\mathcal{V}}_1$ is straightforward, therefore problem CEP-A can be solved efficiently. Note, however, that, due to the approximation, the obtained solution is suboptimal. Consequently, the approximation is tightened throughout iterations, thus ensuring convergence. An initial approximation of $\mathcal{V}_1(x, x^*, \omega)$ is calculated at the beginning of the algorithm. The objective of doing so is to, similarly to the subgradient scheme, have an initial candidate expansion plan to start the search. These ideas are the basis of our algorithmic scheme, which is depicted in pseudo-code in algorithm 13.

The second-stage relaxation algorithm is illustrated graphically in Figure 4.5. The figure presents step 2) of the algorithm. For ease of exposition, step 1), where a warm-start is calculated, is described afterward. Step 2) is subdivided into two steps. Step 2.1) focuses on the first node and the first time step / nodes of the second stage. This corresponds to problem CEP-A. The algorithm uses the current approximation of $\mathcal{V}_1(x, x^*, \omega)$ to solve problem CEP-A (which approximates CEP) and obtain a candidate expansion plan $\widehat{x}^i, \widehat{x}^{*i}$. The algorithm proceeds with step 2.2), which aims at refining the approximation of $\mathcal{V}_1(x, x^*, \omega)$ around $\widehat{x}^i, \widehat{x}^{*i}$. To do so, the algorithm performs a forward pass (step 2.2.1) and a backward pass (step 2.2.2). The forward pass proceeds forward in the number of second-stage nodes, solving the first-node subproblem and proceeding until arriving at the last node. The algorithm continues with the backward pass. This step computes supporting hyperplanes around the

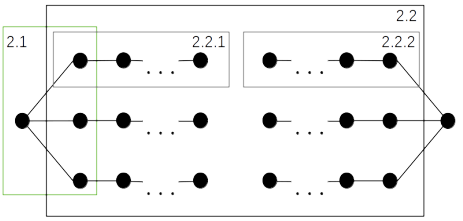


Figure 4.5: Step 2) of the second-stage relaxation algorithm. Step 2.1) solves the current approximation of problem CEP. Step 2.2) refines the approximation of problem CEP around the trial expansion plan found during step 2.1).

storages found during the forward pass and around the trial expansion plan $\widehat{x}^i, \widehat{x}^{*i}$. Starting from the last second-stage node, the subproblem is solved, and the dual multipliers that are computed at this stage are used for estimating a supporting hyperplane for the subproblem of the preceding node. The process is repeated until reaching the node associated with the first node of the second stage. The algorithm performs this procedure for all uncertainty realizations. Having described step 2), we can now describe the warm-start of step 1). Given an initial candidate expansion plan $\widehat{x}^0, \widehat{x}^{*0}$ the objective is to provide an approximation of $\mathcal{V}_1(x, x^*, \omega)$ around the given initial point. To do so, the warm-start performs step 2.2) over several iterations. These iterations are performed in order to ensure that the storages found during the forward pass provide a reasonable approximation.

Remark 4.1. *Note that both steps 2.1) and 2.2) can be solved efficiently. The former is a relatively small problem, which can be solved either by Benders decomposition or subgradient schemes. In this work, we use Benders decomposition. The latter step involves solving several small subproblems and thus does not pose a computational burden. As a consequence, this algorithmic approach is able to perform far more iterations than the subgradient algorithm.*

Upper and lower bounds can be obtained as follows. An upper bound is computed by using the optimal values found during step 2.1) and during step 2.2.1). A lower bound can be obtained by the solution of step 2.1). By comparing upper and lower bounds, one can measure the optimality gap, which can be used as a stopping criterion. The convergence of the decomposition algorithm is guaranteed due to the following lemma.

Algorithm 13: Second-stage relaxation algorithm

INPUT: Provide a lower bound for $\theta_{k,\omega}$ for $k = 1, \ldots, K$, $\omega \in \Omega$, and an initial trial action $\widehat{x}^0, \widehat{x}^{*0}$. A maximum number of iterations N.
OUTPUT: A cutting-plane approximation $\{c^i_{\xi_t}(x_t)\}_{i=1}^N$ for $t = 1, \ldots, T$, $\xi_t \in \Omega_t$.

1. WARM-START: Calculate an initial approximation of $\mathcal{V}_1$ for all $\omega \in \Omega$, around $\widehat{x}^0, \widehat{x}^{*0}$.

2. **for** $i = 1, \cdots, N$.

   (2.1) Solve CEP-A using the current approximation of $\mathcal{V}_1$. This produces a trial action $\widehat{x}^i, \widehat{x}^{*i}$.

   (2.2) **for** $\omega \in \Omega$:

      (2.2.1) **Forward Pass**. **for** $k = 1, \ldots, K$:

         Solve the approximated problem $\widehat{\mathcal{V}}_k(\widehat{x}^i, \widehat{x}^{*i}, \widehat{bv}^i_{t_{k-1},\omega}, \widehat{v}^i_{t_{k-1},\omega}, \omega)$ and get the optimal storage $\widehat{bv}^i_{t_k,\omega}, \widehat{v}^i_{t_k,\omega}$.

      (2.2.2) **Backward Pass**. **for** $k = K, \ldots, 2$:

         (2.2.2.1) Solve the approximation problem $\widehat{\mathcal{V}}_k(\widehat{x}^i, \widehat{x}^{*i}, \widehat{bv}^i_{t_{k-1},\omega}, \widehat{v}^i_{t_{k-1},\omega}, \omega)$.

         (2.2.2.2) Using the dual multipliers, compute a supporting hyperplane around $\widehat{x}^i, \widehat{x}^{*i}, \widehat{bv}^i_{t_{k-1},\omega}, \widehat{v}^i_{t_{k-1},\omega}$.

         (2.2.2.3) Add the supporting hyperplane to problem $\widehat{\mathcal{V}}_{k-1}(x, x^*, bv_{t_{k-2},\omega}, v_{t_{k-2},\omega}, \omega)$.

Lemma 4.1. *Algorithm 13 converges in a finite number of iterations to CEP.*

Proof. We prove that if no new cuts are added to $\widehat{\mathcal{V}}_1(x, x^*, \omega)$ for $\omega \in \Omega$, then we are at the optimal of CEP. Suppose that no new cuts are obtained after iteration i and consider the expansion plan at that iteration, denoted as x^i, x^{i*}. Note that, if no new cuts are added, then $\widehat{\mathcal{V}}_1(x^i, x^{i*}, \omega) = \mathcal{V}_1(x^i, x^{i*}, \omega)$ (otherwise during the $i+1$ iteration we would have found a new cut). Now let us assume that we have not converged, therefore there exists $\bar{x}, \bar{x}^*$ for which $(4.26) + \mathbb{E}_\omega\Big[\mathcal{V}_1(\bar{x}, \bar{x}^*, \omega)\Big] < (4.26) + \mathbb{E}_\omega\Big[\mathcal{V}_1(x^i, x^{*i}, \omega)\Big]$. CEP-A is an under-approximation of CEP. Thus, its maximum possible value is the left-hand side of the previous inequality. As a consequence,

$$\begin{aligned}(4.26) + \mathbb{E}_\omega\Big[\widehat{\mathcal{V}}_1(x^i, x^{*i}, \omega)\Big] &< (4.26) + \mathbb{E}_\omega\Big[\mathcal{V}_1(\bar{x}, \bar{x}^*, \omega)\Big] \\ &< (4.26) + \mathbb{E}_\omega\Big[\mathcal{V}_1(x^i, x^{*i}, \omega)\Big] \\ &= (4.26) + \mathbb{E}_\omega\Big[\widehat{\mathcal{V}}_1(x^i, x^{*i}, \omega)\Big]\end{aligned}$$

This leads to a contradiction, therefore we conclude that x^i, x^{*i} is optimal.

Finally, let us show that this can be achieved in finite iterations. Following the same idea as in [PG08], where it is shown that SDDP-type [PP91] algorithms converge, we start from the subproblem of the last block. Note that the set of dual multipliers of the subproblem associated to

$$\mathcal{V}_K(x, x^*, bv_{t_{K-1},\omega}, v_{t_{K-1},\omega}, \omega)$$

corresponds to the vertices of the feasibility set of the dual problem, and this set of dual multipliers does not depend on $x, x^*, bv_{t_{K-1},\omega}, v_{t_{K-1},\omega}$. Consequently, there are finitely many supporting hyperplanes for the last subproblem. Thus, after finitely many iterations, we arrive at the supporting hyperplane description of

$$\mathcal{V}_K(x, x^*, bv_{t_{K-1},\omega}, v_{t_{K-1},\omega}, \omega)$$

. We can now use such a description for the subproblem at block $K-1$, and apply the same argument. Inductively, we can proceed backward in the number of blocks. Note that, as opposed to the results provided in lemma 2.2, in the current situation we have convergence to the optimal solution. The main difference being that here we don't have the difficulty of dealing with incomplete information.

□

4.5 Parallel scheme

The present section aims at describing a parallel scheme for the algorithms presented in section 4.4. In this work, we have considered synchronous parallel implementations.

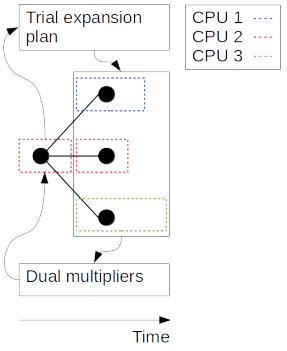


Figure 4.6: Parallel subgradient algorithm. The master CPU provides a trial expansion plan, which is sent to the subproblems. The subproblems are distributed among the CPUs and solved. The dual multipliers are sent to the master CPU.

• **Parallel subgradient algorithm:** The parallelization strategy followed for the subgradient algorithm is presented graphically in Figure 4.6. Each iteration proceeds as follows. The master CPU provides an initial candidate expansion plan. The second-stage subproblems are distributed among the available CPUs. Each CPU solves its associated subproblem, and the dual multipliers are collected and sent to the master CPU. At this point, the CPUs synchronize, i.e, a CPU stays idle until all other CPUs have finished their job. The master CPU uses the dual information in order to apply a projected subgradient and update the trial expansion plan.

• **Parallel L-shaped algorithm:** The parallelization follows a similar strategy as in the previous scheme. The difference is that, during each iteration, the master CPU calculates the master problem.

• **Parallel second-stage relaxation algorithm:** The parallel scheme for our subgradient-relaxation algorithm follows a similar procedure. Step 2.1) is parallelized using the strategy described in Figure 4.6. Step 2.2) is parallelized as follows. The uncertainty realizations are distributed among the available CPUs. Each CPU performs steps 2.2.1 and 2.2.2 of algorithm 13 (see Figure 4.5). At this point, the CPUs synchronize, so that the fastest CPU waits for the slowest CPU.

4.6 Case study

The EVA problem aims at determining the expansion / retirement plans that will occur, and our study considers the 2025 target year. We highlight that the choice of a single year is, as a first step, to emulate the modeling procedure that is performed for ERAA 2021, with the aim of tackling a multiyear setting in future research. Furthermore, within the modeling an amortization of fixed

cost is introduced and the model is equivalent to a multi-year model over the lifetime of the invested plants where each year is identical. We consider a total of 12 blocks per day.

Our algorithms are implemented in Julia v1.5 and JuMP v22. The chosen linear programming solver is Gurobi 9. The computational work is performed on the Lemaitre3 cluster of UCLouvain, which is hosted at the Consortium des Equipements de Calcul Intensif (CECI). The cluster consists of 80 compute nodes with two 12-core Intel SkyLake 5118 processors at 2.3 GHz and 95 GB of RAM (3970MB/core), interconnected with an OmniPath network (OPA-56Gbps).

4.6.1 Value of the stochastic solution

Due to the scale of the model, ENTSO-E has considered an approximate solution in ERAA 2021. This approximation proceeds by solving the so-called wait-and-see solution [BL11] of problem CEP. This leads to a candidate expansion plan x_ω, x^*_ω, for each $\omega \in \Omega$. The average expansion plan $\bar{x}, \bar{x}^*$ is used as an approximation of the stochastic expansion plan. A natural question is whether this is a good approximation. To measure this, we can use well-established bounds. The following relationship holds [BL11]:

$$\text{Wait-and-see} \leq \text{ CEP } \leq (4.26) + \mathbf{E}_\omega \Big[\mathcal{V}(\bar{x}, \bar{x}^*, \omega) \Big]$$

Note that the right-hand side is the objective of CEP when using the sub-optimal average expansion plan $\bar{x}, \bar{x}^*$. We refer to this as the average-W.S. solution. Note that these bounds do not require the calculation of the stochastic solution, and thus provide a reasonable way to measure if a stochastic solution is of interest for the problem. The wait-and-see solution has a cost of $5.0220e10$ €, while the average wait-and-see solution has a cost of $5.2298e10$ €. As one can observe, the relative difference between both quantities is approximately 4.13%, and thus the stochastic solution stands to improve the deterministic approximation by at most this quantity. This difference is of interest: adequacy studies such as ERAA aim at capturing adequacy metrics that involve curtailments. These curtailments, calculated as in equation (4.31), represent less than 2% of the total costs.

4.6.2 Stochastic solution

The previous subsection demonstrates the interest in computing a solution to the stochastic programming formulation. The present subsection discusses how to obtain such a solution using the previously discussed algorithms. We highlight that the extended formulation of this model, as a linear problem, does not fit in memory (95 GB). This motivates the need for employing decomposition approaches. The algorithms are run using 35 CPUs. In the case of the subgradient relaxation, we decompose the second stage into 92 blocks.

We begin by examining the convergence evolution of the L-shaped method. To do so, we examine the evolution of the optimality gap, which is presented in Figure 4.7. As one can observe, the L-shaped method struggles to close the optimality gap. In fact, after more than a day of computation, the obtained gap is not of practical use. On the other hand, our subgradient-relaxation scheme is able to provide an optimality gap near 1% after about 4 hours of computation. The reason behind the poor behaviour of the L-Shaped method is that only a few iterations of the algorithm can be performed. This in turn is due to the fact that each subproblem requires run time within the order of half an hour to be solved. Note that as the proposed subgradient algorithm does not provide a lower bound estimate, the calculation of an optimality gap is not presented. Instead, we will use the upper bound to measure convergence.

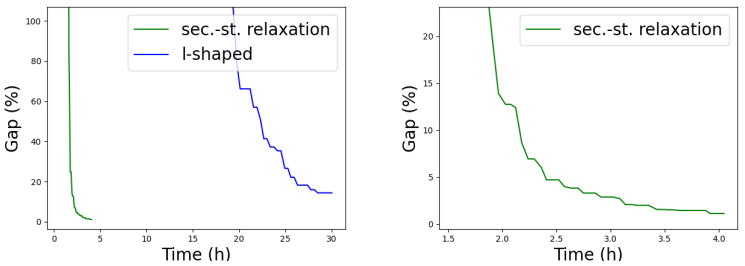


Figure 4.7: The evolution of the optimality gap of the L-shaped method and our subgradient-relaxation algorithm. The x-axis is the elapsed time, while the y-axis is the relative difference between the upper and lower bounds.

We also implement the progressive hedging algorithm[2] (described in subsection 1.2.3) for this problem, where the scenario subproblems correspond to each one of the 35 climate years. Figure 4.8 presents the evolution of the upper bound of the subgradient algorithm, the subgradient-relaxation scheme and the progressive hedging algorithm. Recall that the subgradient-relaxation scheme uses an approximation of the economic dispatch, thus it does not correspond to the true value of using the expansion plan. Consequently, in order to have comparable upper bound values, after running the subgradient-relaxation algorithm we evaluate the true cost of using the obtained expansion plan. This corresponds to the horizontal non-dashed green line. A similar procedure is performed with the expansion plans obtained by the progressive hedging algorithm. During each iteration, the true cost of using the expansion plan is evaluated (the computational time to do so is not considered, so as to have a fair comparison). These quantities and the upper bound of the subgradient

[2]It is well known that this algorithm is sensitive to the choice of the penalizatiion parameter ρ. In order to tune it, we consider a simplified version of the problem and test different values of $\rho = 1, 10, 100, 1000$. We observe that moving from $\rho = 1$ to $\rho = 100$ reduces the iteration count, while going beyond 100 seems to have an insignificant effect. Therefore, we consider $\rho = 100$.

algorithm are then comparable. As shown in Figure 4.8, we note that after 30 hours of computation the progressive hedging algorithm is unable to reach the solution that our proposed algorithms reach. Each subproblem of progressive hedging is harder than Benders as (i) it includes a quadratic term in the objective, and (ii) includes the optimization of the first and second stage. This renders each iteration prohibitively expensive. Regarding the subgradient scheme and the relaxation strategy, we note that both algorithmic schemes are able to converge. However, it can be observed that the subgradient-relaxation scheme converges considerably faster. In fact, for the 12-block formulation, after 30 hours of computation the subgradient scheme fails to achieve the bound that the subgradient-relaxation scheme is able to find in just 4 hours. We note that the obtained run times can be used in practice. ENTSO-E's experience with a stochastic model, which consists of climate years and is solved as a large LP for a variety of target years, is that it requires longer run times.

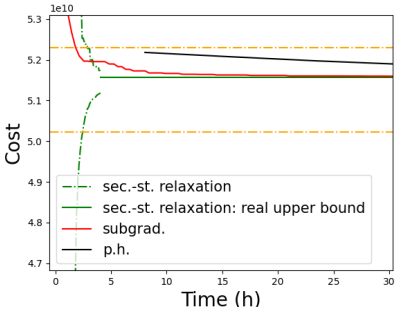


Figure 4.8: The convergence evolution of the subgradient algorithm and the subgradient-relaxation scheme. The x-axis is the elapsed time, while the y-axis is the cost. The yellow horizontal lines correspond to the wait-and-see solution (lower yellow horizontal) and ENTSO-E's solution (upper yellow horizontal).

To assess the scalability of the methods, we consider a system with a granularity of 24 blocks. The convergence is presented in Figure 4.9, where it is seen that after almost 2 days of computation, the subgradient scheme fails to attain the bound that the subgradient-relaxation scheme finds in 15 hours. This indicates that the subgradient-relaxation scheme is better suited as the size of the problem increases.

An alternative approach is considered for solving the problem, as a means of further comparison. This includes Benders Decomposition With Multiple Master Problems (BDMM) [SSAG22]. Unfortunately, this scheme is not suitable for our setting. The BDMM increases the amount of second-stage Benders subproblems. In the case of the problem that we are interested in, this is not tractable, as each Benders subproblem is large.

We highlight that the use of parallel computing has been crucial in order to obtain the solutions that are indicated here. In fact, as each CPU is handling

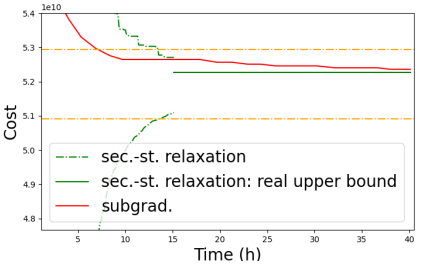


Figure 4.9: The convergence evolution of the subgradient algorithm and the subgradient-relaxation scheme. The model uses 24 blocks per day.

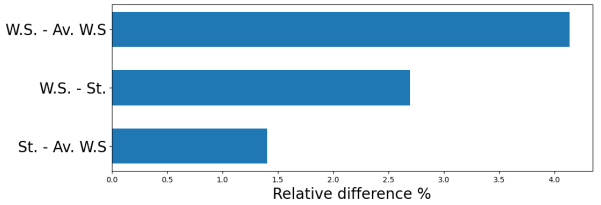


Figure 4.10: Comparison of relative total cost between average wait-and-see (Av. W.S.), Wait-and-See (W.S.), and Stochastic (St.) solutions.

one of the 35 climatic years, we expect a serial run to be approximately 35 times more time-consuming.

4.6.3 Analysis of the solution

The present subsection aims at studying the differences between the stochastic solution obtained in subsection 4.6.2 and the deterministic solution obtained in subsection 4.6.1. In terms of total cost, as presented in Figure 4.10, we observe a difference of nearly 2.7% between the stochastic solution and the wait-and-see solution. This difference justifies the effort in computing a stochastic solution. Furthermore, we observe that the stochastic solution provides an improvement of nearly 1.5% with respect to the average W.S. solution, implemented by ENTSO-E in ERAA 2021.

This difference in total costs is examined in Figure 4.11 by decomposing the total costs into operational costs, retired capacity savings, expansion plan costs, and curtailment costs. The upper panel presents the costs for both approaches, the stochastic solution, and the average W.S. solution, while the lower panel presents the relative difference between these solutions (a positive number indicates that the average W.S. solution has a higher value). Note that the stochastic solution tends to result in significantly fewer curtailments, which can be explained by the increased investments and fewer retirements, that is to say, we arrive at a more conservative solution. On the other hand, the lack

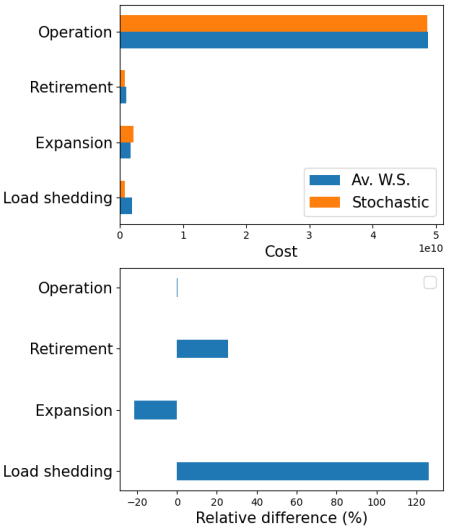


Figure 4.11: Cost breakdown. The upper panel presents the total costs, while the lower panel presents the relative difference between the quantities shown in the upper panel.

of information regarding the future possible outcomes in the average expansion plan approach makes the solution prone to over-retiring and underestimating the required investments. This implies that the power system is unable to satisfy the demand as effectively as in the stochastic solution.

4.6.4 Adequacy metrics

One objective of the ERAA study is to measure the ability of the system to maintain secure levels of supply. Two adequacy metrics are of particular interest to ENTSO-E. The first one is loss of load expectation (LOLE), defined as $\text{LOLE} = \mathbf{E}_\omega\left[\text{LOL}_\omega\right]$. Here, LOL_ω is the number of hours during which demand is not served for the climatic year $\omega \in \Omega$. The second metric is the expected value of energy not served (EENS), which is defined as $\text{EENS} = \mathbf{E}_\omega\left[\text{ENS}_\omega\right]$. Here, ENS_ω is the number of curtailments obtained in climatic year $\omega \in \Omega$. We consider the 35 climatic years that are used for formulating the stochastic model, thereby aiming to compare its performance against the approximate solution that is obtained by averaging the expansion plans (which is the approach used by ENTSO-E in ERAA 2021). Therefore, there is no out-of-sample testing. We calculate the metrics of interest and present the results in Figure 4.12. We consider both the LOLE and the EENS for the four main European regions: Central and Eastern Europe, Western Europe, Southern Europe, and Northern Europe. Figure 4.12 demonstrates how the use of a

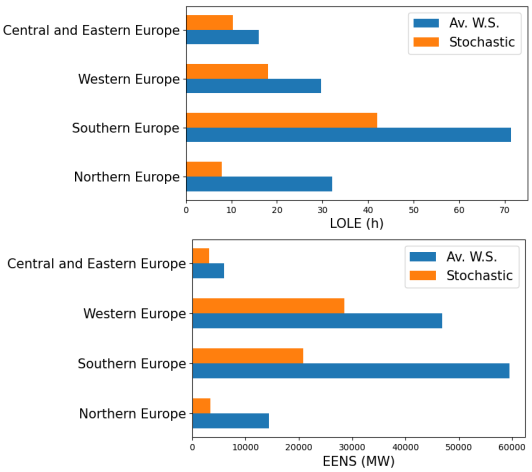


Figure 4.12: The LOLE and EENS metrics for different European regions.

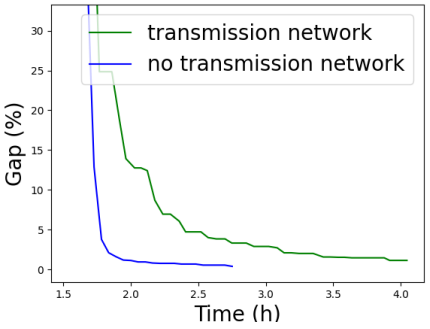


Figure 4.13: Convergence evolution when modeling and not modeling the transmission network.

stochastic solution is able to provide consistent and significantly more accurate metrics. This attribute is of particular interest to studies such as ERAA.

4.6.5 Impact of transmission congestion

In order to assess the impact of transmission congestion we have considered the following setting. The transmission network has been eliminated by increasing the network line limits. The stochastic model is then solved, and convergence is presented in Figure 4.13. In particular, we note that the model tends to be far easier to solve, in the sense that it converges faster. This is expected, since without a transmission network the algorithm no longer has to determine local expansions, but rather a global investment.

We observe an interesting behaviour when examining the value of the stochastic solution. Table 4.1 considers the optimal value of the stochastic solution (which has non-anticipative constraints) and the wait-and-see solution. In the wait-and-see solution, there is perfect foresight and thus the expansion plan decisions are obtained with advance knowledge of future events. We see that excluding the network largely diminishes the value of the stochastic solution [BL11]. Concretely, we move from a 2.7 % difference in total costs to a difference of approximately 1 % when excluding the network. This is in line with analogous observations in the literature [PO13], where the inclusion of network constraints has been shown to approximately double the value of the stochastic solution due to the fact that the endogenous representation of uncertainty allows us to determine not only what the optimal mix should be, but where it should be located, while ensuring delivery of the required energy over the network.

Table 4.1: Value of the stochastic solution.

Approach	stochastic solution (€)	wait-and-see solution (€)	%
Transp. model	5.157e10	5.022e10	2.7
No network	4.498e10	4.44e10	1.18

We also note that, in scarcity situations, the model with no congestion is able to serve the load more evenly among the available generators. This consequently results in lower EENS and LOLE, as shown in Figure 4.14.

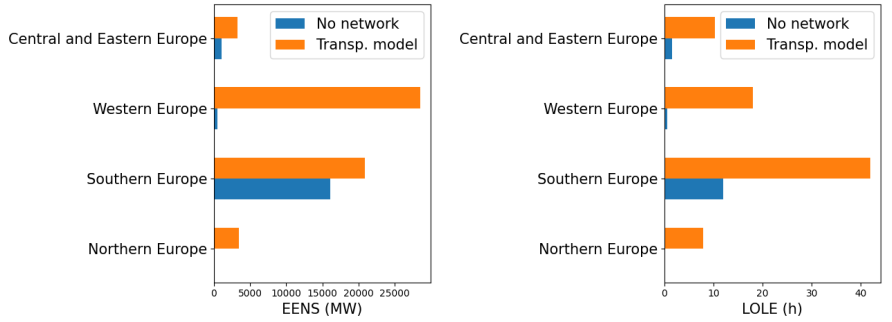


Figure 4.14: LOLE and EENS metrics when modeling and not modeling the transmission network.

These results highlight the relevance of transmission congestion in determining expansion decisions, and the impact of such an expansion on the adequacy of the power system.

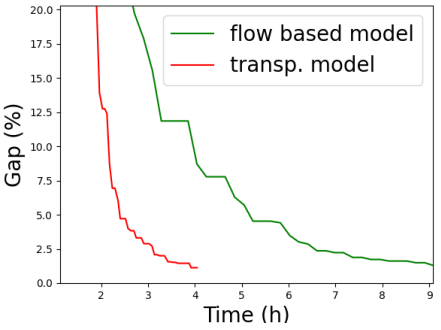


Figure 4.15: Convergence evolution of the flow-based model.

4.6.6 Modelling extensions

As described in subsection 4.3.3, for ERAA 2021 a transportation model has been considered for representing the transmission network. Transportation models result in an endemic misrepresentation of the true physics of power flow in European market models, and are related to the persistence of zonal pricing in Europe. However, since we are interested in tackling the 2021 ERAA, we limit ourselves to the same input data that ENTSO-E uses for its needs. Our methodology can be generalized to more general models based on DC power flow, and the essential decomposition steps are not affected. For instance, ERAA 2022 has included a flow-based methodology in its study. Using this flow-based data, we have developed a proof-of-concept algorithm which includes a PTDF approximation of the transmission network. We highlight that the decomposition algorithm remains unaffected. Figure 4.15 presents the convergence evolution of said approach, where it is seen that, due to the introduction of PTDF constraints, the problem becomes more intense computationally, nevertheless the decomposition technique is able to manage it.

Similarly as for the flow-based scheme, the methodology can include several extensions with relative ease, for example the algorithm remains unchanged if we add reserve requirements or binary investments. In terms of uncertainty modelling, additional sources of uncertainty may be included, for example gas price uncertainty or generators outages. Each one of these extensions increases the complexity the nature of the problem, thus likely leading to longer solution times. The extent to which the methodology can handle these extensions is out of scope for the present work, but opens the path for considering further modeling generalization.

4.7 Conclusions

In this work, we present a high-performance computing approach for obtaining a stochastic solution for the EVA of ERAA 2021. We specifically propose a subgradient algorithm implemented in parallel computing infrastructure, as well as a subgradient-relaxation approach that emerges as being practically attractive due to its capability to tackle large-scale problems efficiently. We further observe that the commonly used L-Shaped method is unable to provide a solution of practical use in a reasonable amount of time. Finally, we compare the stochastic solution against a deterministic approach implemented by ENTSO-E for the ERAA 2021 edition. A noticeable difference between both solutions in terms of commonly used adequacy metrics emerges, which highlights the practical value of the stochastic solution that we are able to compute.

Future ERAA studies aim at improving the EVA on two fronts. (i) On the one hand, a multi-year stochastic model is considered, that decides investments and retirements before the realization of each year. (ii) On the other hand, a more robust model is considered, which includes further sources of uncertainty, such as climate change patterns. These increased sources of uncertainty motivate the study of scenario reduction techniques.

Furthermore, assumptions on fuel prices can have a significant impact on the results. Fuel prices are quite volatile, as the recent European energy crisis suggests. The proposed approach can handle objective function uncertainty, thus allowing us to include these assumptions directly into the model. On the other hand, ENTSO-E's interaction with industry stakeholders indicates that this uncertainty can be considered as a sensitivity run of the model, as opposed to being modeled endogenously. Such a sensitivity analysis can provide insights on the impact of fuel price variations without introducing further computational complexity to the model.

4.A Nomenclature

- **Sets:**

  Ω: The set of uncertainty realizations.

  $\mathcal{N}$: The set of zones.

  $\mathcal{G}$: The set of existing generators.

  $\mathcal{G}^*$: The set of new generators.

  $\mathcal{T}$: The time horizon of the economic dispatch.

  $\mathcal{L}(n)$: The set of lines for zone n.

  $\mathcal{H}(n)$: The set of hydro technologies for zone n.

- **Parameters:**

  IC: The annualized investment cost (€/MW).

FOM: The fixed operational and maintenance cost (€/MW).

VOM: The variable cost (€/MW).

FC: The fuel costs (€/MW).

WC: The wheeling charge cost (€/MW).

SC: The spillage cost (€/MW).

VOLL: A high cost for curtailment (€/MW).

$X_{n,g}$: Retirement limit for $n \in \mathcal{N}, g \in \mathcal{G}$ (MW).

$X_{n,g}$: Expansion limit for $n \in \mathcal{N}, g \in \mathcal{G}^*$ (MW).

$P^{max}_{t,n,g}$: Maximum production, $t \in \mathcal{T}, n \in \mathcal{N}, g \in \mathcal{G}^*$ (MW).

$P^{min}_{t,n,g}$: Minimum production, $t \in \mathcal{T}, n \in \mathcal{N}, g \in \mathcal{G}^*$ (MW).

$X_{n,g}$: Expansion limit for $n \in \mathcal{N}, g \in \mathcal{G}^*$ (MW).

$L^{max}_{n,l}$: Transfer limit, $n \in \mathcal{N}, l \in \mathcal{L}(n)$ (MW).

$L^{min}_{n,l}$: Transfer minimum, $n \in \mathcal{N}, l \in \mathcal{L}(n)$ (MW).

BCE: Battery charge efficiency (%).

BDE: Battery discharge efficiency (%).

BV: Battery capacity (MW).

BC: Maximum charge capacity (MW).

BD: Minimum discharge capacity (MW).

$V_{n,h}$: Maximum storage for zone n, technology h (MWh).

$Q_{n,h}$: Turbine capacity for zone n, technology h (MW).

$D_{n,h}$: Pump capacity for zone n, technology h (MW).

PE: Pump efficiency (%).

- **Parameters with uncertainty:**

  $A_{n,h,t,\omega}$: Inflow for $n \in \mathcal{N}$, $h \in \mathcal{H}(n)$, $t \in \mathcal{T}$, $\omega \in \Omega$ (MW).

  $\mathcal{PV}_{n,t,\omega}$: PV production for $n \in \mathcal{N}$, $t \in \mathcal{T}$, $\omega \in \Omega$ (MW).

  $\mathcal{W}_{n,t,\omega}$: Wind production for $n \in \mathcal{N}$, $t \in \mathcal{T}$, $\omega \in \Omega$ (MW).

  $\mathcal{D}_{n,t,\omega}$: Demand for $n \in \mathcal{N}$, $t \in \mathcal{T}$, $\omega \in \Omega$ (MW).

- **Variables:**

  $x^*_{n,g}$: The invested capacity (MW).

  $x_{n,g}$: The retired capacity(MW).

  $p_{n,g,t,\omega}/p^*_{n,g,t,\omega}$: Power produced with existing/new generators (MW).

  $f_{n,l,t,\omega}$: Power transferred through lines (MW).

  $s_{n,h,t,\omega}$: The amount of spillage (MW).

$ls_{n,t,\omega}$: The load shedding (MW).
$bv_{n,t,\omega}$: The battery state of charge (MWh).
$bc_{n,t,\omega}$: Power charged into the battery (MW).
$bc_{n,t,\omega}$: The amount of charged power into the battery (MW).
$bd_{n,t,\omega}$: The battery discharged power (MW).
$q_{n,h,t,\omega}$: The hydro turbined produced power (MW).
$v_{n,h,t,\omega}$: The state of storage of the reservoir (MWh).
$d_{n,h,t,\omega}$: The pumped power (MW).

5 Applying scenario reduction to the ERAA study

5.1 Introduction

Within the European Resource Adequacy Assessment (ERAA), the computational complexity of the Economic Viability Assessment (EVA) has been a matter of concern. One of the fronts of modeling development aiming at simplifying the description of the problem is the selection of a representative set of climatic years, so as to reduce the overall size of the problem [erab]. This aspect is to become more relevant as new sources of uncertainty appear and current ones are expanded. On the one hand, ongoing discussions point towards a more robust model which considerably expands the number of climatic conditions. Despite the algorithmic improvements presented in chapter 4, these algorithmic improvements can not handle an arbitrarily large number of scenarios. On the other hand, and taking into account the methodological difference between steps 1 and 2 of the overall ERAA methodology (see Fig. 4.1), relevant sources of uncertainty such as random outage patterns, or fuel price uncertainty, may be considered as being relevant for the EVA in future ERAA versions. These additions imply a cross product between the different sources of uncertainty, thus leading to a stochastic problem with a prohibitively large set of uncertainty realizations, thus leading to an intractable problem. Inspired by these challenges, our work proposes a scenario reduction methodology which is still under development. The goal of this methodology is to select a representative set of uncertain realizations.

For ERAA 2021 a tailored scenario reduction methodology, based on statistical properties of the climatic years time series, was proposed and used by ENTSO-E. A relevant drawback of this approach is that, because it is based on statistical properties, it does not fully acknowledge the effect of scenarios on the optimization model. Given the emphasis of the adequacy assessment on load shedding, this suggests schemes that are based on assessing the impact of scenarios on the optimization problem, rather than the statistical properties of the scenarios alone. Of particular relevance is the fact that small perturbations in the total objective system cost may result in considerable perturbations in

the load shedding

The capacity generation expansion literature has proposed a variety of strategies for scenario reduction [KS10, JRWW11, FR13, GAC14], where the focus has rather been on statistical properties. Form the scenario reduction literature, a methodology that is often adopted in various applications, and has also been implemented in the GAMS software, is the so-called fast forward selection algorithm [DGKR03, HR03]. This algorithm requires a mapping from the uncertainty space into $\mathbb{R}^n$, in order to develop a measure between scenarios. In [MPCC09], the authors use the fast forward selection algorithm in combination with a metric based on the cost of the optimization problem.

Leveraging on the results presented in [Mor19], our analysis aims at developing a methodology, supported by parallel computing, which (i) is based on the effect of uncertainty on the optimization problem, rather than on statistical properties; (ii) proposes hierarchical criteria, in which the structure of the algorithm suggests natural stopping criteria (e.g. based on the distance between merged clusters).

5.1.1 Organization and contributions

(i) Supported by parallel computing, we introduce a scenario reduction scheme based on hierarchical clustering, which considers the optimization problem at hand. (ii) We empirically demonstrate the advantages of the proposed scheme to capture total costs of the system, which in turn resulted in the adoption of part of the methodology for ERAA 2022.

The present chapter is organized as follows. Section 5.2 describes the considered methodology, while section 5.3 discusses some preliminary results. Section 5.4 highlights open areas of study associated to the methodological scheme. The chapter ends with section 5.5 which presents the conclusions.

5.2 Scenario reduction

The considered scenario reduction scheme consists of two fundamental steps: (i) a definition of a distance between a pair of scenarios and (ii) a clustering strategy to merge similar pairs of scenarios. As discussed earlier, the motivation behind (i) is to account for the interplay between scenarios and the underlying optimization problem, while (ii) aims at being able to define a stopping criterion for selecting an optimal number of scenarios. We begin by introducing some notation followed by the description of the defintion of the distance metric and the clustering methodology.

Following the notation introduced in section 1.2, we consider a two-stage stochastic linear problem given by:

$$\min_{x_1, y_1} u_1^T x_1 + v_1^T y_1 + \mathbb{E}\left[\min_{x_2(\omega)} u_2^T(\omega) x_2(\omega) \right]$$

$$\begin{aligned} \text{s.t. } & A_1 x_1 + D_1 y_1 = b_1 \\ & x_1, y_1 \geq 0 \\ & B_2(\omega) x_1 + A_2(\omega) x_2(\omega) = b_2(\omega), \text{ for all } \omega \in \Omega \\ & x_2(\omega) \geq 0, \text{ for all } \omega \in \Omega \end{aligned}$$

Note that x_1 is coupled in time, while y_1 not. To ease notation, we write $\bar{x} = [x_1, y_1]$ and introduce the function $h(\bar{x}, \omega)$, which is defined as:

$$\begin{aligned} h(\bar{x}, \omega) = u_1^T x_1 + v_1^T y_1 + \min_{x_2(\omega)} & u_2^T(\omega) x_2(\omega) \\ \text{s.t. } & B_2(\omega) x_1 + A_2(\omega) x_2(\omega) = b_2(\omega) \\ & x_2(\omega) \geq 0 \end{aligned}$$

Using this notation, we can equivalently write the stochastic optimization problem as:

$$\min_{\bar{x}} \mathbb{E}\Big[h(\bar{x}, \omega)\Big]$$

Given a subset $\Omega_R \subset \Omega$, the reduced stochastic program can simply be formulated as $\min_{\bar{x}} \mathbb{E}_R\Big[h(\bar{x}, \omega)\Big]$, where $\mathbb{E}_R$ is the expectation operator taken just over the subset Ω_R, which is naturally easier to handle. Ideally, we want to obtain a subset Ω_R that approximates accurately the original stochastic problem.

5.2.1 Distance between scenarios

Motivated by the search for an optimization-based strategy, we consider the distance proposed in [HOR22]. We highlight that this distance definition is not a distance in the mathematical sense, as it doesn't satisfy the triangle inequality property. According to this definition, the distance between two scenarios is measured in terms of how far the scenarios are in arriving to a mutually acceptable first-stage decision. This naturally leads to a clustering which groups scenarios based on how close they are in terms of reaching mutually acceptable first-stage decisions.

For each $\omega \in \Omega$, let x^*_ω represent an optimal decision under the assumption that scenario ω will definitely occur. That is,

$$\bar{x}^*_\omega \in X^*_\omega := \operatorname*{argmin}_{\bar{x}} h(\bar{x}, \omega)$$

Now, define the opportunity cost of taking the optimal decision associated to scenario ω_1, when scenario ω_2 actually occurs, as:

$$\delta(\omega_1 | \omega_2) := h(x^*_{\omega_1}, \omega_2) - h(x^*_{\omega_2}, \omega_2) \geq 0.$$

We can now define a notion of distance[1] between scenarios ω_1 and ω_2 as:

$$c(\omega_1, \omega_2) = \delta(\omega_1|\omega_2) + \delta(\omega_2|\omega_1).$$

Note that it is possible that there exists more than one optimal solution $\bar{x}^*_\omega$ for the problem $\min_{\bar{x}} h(\bar{x}, \omega)$, so that this distance may not even be well defined, as the election of different optimal solutions may lead to different values of c. A workaround is to re-define the opportunity cost as:

$$\delta(\omega_1|\omega_2) := h(X^*_{\omega_1}, \omega_2) - h(X^*_{\omega_2}, \omega_2)$$

Here $h(X^*_{\omega_1}, \omega_2)$ equals the minimum value of $h(\bar{x}^*_{\omega_1}, \omega_2)$ over all possible optimal decisions $\bar{x}^*_{\omega_1}$. In practice, however, we cannot consider all the optimal decisions $\bar{x}^*_{\omega_1}$, thus we will just consider the optimal solution obtained by the solver when optimizing $\min_{\bar{x}} h(\bar{x}, \omega_1)$.

5.2.2 Hierarchical clustering methodology

Now that a notion of distance between scenarios has been introduced, we can introduce a notion of clustering. In this chapter we consider hierarchical clustering, which is motivated by the seek for a stopping criterion heuristic. The proposed procedure is presented in pseudo-code in Algorithm 14.

Here, a definition of a distance $d(C_i, C_j)$ between two pair of clusters is introduced. Some examples include single-linkage, average-linkage [ABDBL21] or optimal transport problem [DGKR03]. For this work we consider the latter.

Let C_i be a cluster. We can induce a probability measure over this set as $\mathcal{Q}_i(\omega) = \frac{P(\omega)}{P(C_i)}$ for $\omega \in C_i$. Given a pair of clusters C_1, C_2, we define its distance $d(C_1, C_2)$ in terms of the optimal transport problem between the associated probability distributions:

$$d(C_1, C_2) = \min_{\pi_{i,j} \geq 0} \Big\{ \sum_{i \in \Omega_1, j \in \Omega_2} \pi_{i,j} c(\omega_i, \omega_j) \Big| \sum_{i \in \Omega_1} \pi_{i,j} = \mathcal{Q}_2(\omega_j), \sum_{i \in \Omega_2} \pi_{i,j} = \mathcal{Q}_1(\omega_i) \Big\}$$

[1] Recall that this is not defined in the proper mathematical sense, as the triangle inequality may not be satisfied.

Algorithm 14: Scenario reduction

INPUT: A desired number K of clusters.

1. Each scenario in Ω will constitute a cluster, so that the array of clusters Z consists of $|\Omega|$ singletons.
2. **while** $|Z| > K$

   (2.1) Compute the $|Z| \times |Z|$ distance matrix M, defined by

   $$M(i,j) = d(C_i, C_j), \;\; C_i, C_j \in Z.$$

   (2.2) Choose C_{i^*} and C_{j^*} in Z such that

   $$d(C_{i^*}, C_{j^*}) \in \underset{i \neq j}{\operatorname{argmin}}\, M(i,j)$$

   (2.3) Remove the clusters C_{i^*} and C_{j^*} from Z and replace them with the cluster $C = C_{i^*} \cup C_{j^*}$. In this step, the size of Z is reduced by 1.

3. A representative element of each cluster is selected. From each cluster C_i choose ω^i such that $\omega^i \in \underset{\omega \in C_i}{\operatorname{argmin}} \sum_{\omega' \in C_i} c(\omega, \omega')$ and assign to it the aggregate probability of the cluster, given by $\pi_i = \sum_{\omega \in C_i} p_\omega$.

5.2.3 Parallelization scheme

The present subsection aims at describing the parallelization strategy proposed for the scenario reduction scheme. The most computationally expensive part of the methodology is the calculation of the distance $c(\omega_i, \omega_j)$ between a pair of scenarios, which takes place in step (2.1) of Alg. 14. As this distance has to be computed once, before running the scenario reduction methodology, we calculate the distance between all pairs of scenarios.

To calculate this distance, we first calculate the optimal strategy $h(x^*_\omega, \omega)$ for each scenario ω. This is accomplished by distributing the scenarios among the available cores. We next proceed by calculating the opportunity cost matrix δ, described as:

$$\delta = \begin{bmatrix} \delta(\omega_1|\omega_1) & \delta(\omega_1|\omega_2) & \dots & \delta(\omega_1|\omega_N) \\ & & \vdots & \\ \delta(\omega_N|\omega_1) & \delta(\omega_N|\omega_2) & \dots & \delta(\omega_N|\omega_N) \end{bmatrix}$$

Each one of the entries of the matrix corresponds to $\delta(\omega_i|\omega_j) = h(x^*_{\omega_i}, \omega_j) - h(x^*_{\omega_j}, \omega_j)$. We propose an embarrassingly parallel approach for calculating the

matrix, namely, the matrix entries are distributed among the available cores. As each entry is a non-trivial optimization problem, network latency is not problematic. Note that, in principle, one could use a considerably large amount of cores, as the bound of cores is given by the amount of entries in the matrix.

The other parts of the algorithm do not pose a computational burden and therefore are not parallelized in our implementation.

5.3 Preliminary analysis

The present subsection aims to present a preliminary analysis regarding the proposed scheme. For this purpose, we consider the expansion problem, based on the ERAA 2021 data, developed in chapter 4. We compare our methodology against the scenario reduction strategy proposed by ENTSO-E for the ERAA 2021 edition, which is based on statistical properties of the climatic years [erab]. We further compare the methodology against the widely known forward selection scenario reduction algorithm proposed in [DGKR03].

5.3.1 Comparison against ERAA 2021

We begin by describing the computational effort required to run the scenario reduction algorithm. Our algorithms are implemented in Julia v1.5 and JuMP v22. The chosen linear programming solver is Gurobi 9. The computational work is performed on the Lemaitre3 cluster of UCLouvain, which is hosted at the Consortium des Equipements de Calcul Intensif (CECI). The cluster consists of 80 compute nodes with two 12-core Intel SkyLake 5118 processors at 2.3 GHz and 95 GB of RAM (3970MB/core), interconnected with an Omni-Path network (OPA-56Gbps). As discussed earlier, the most expensive part of the algorithm is the calculation of the distance function $c(\omega_i, \omega_j)$. Using the embarrassingly parallel strategy described in subsection 5.2.3, we find that with 35 CPUs we require nearly 11 hours of computing time. We note that the bound on the number of CPUs that can be used for the computation of the matrix is large, therefore in principle we could have used a considerably larger set of CPUs.

For ERAA 2021, a total of 7 climatic years are selected. In order to allow for a fair comparison, we let our algorithm select 7 climatic years. For both selections of climatic years we consider the associated stochastic problem and solve it using the techniques developed in chapter 4. To test the quality of the obtained expansion plans, we let the full uncertainty unveil and evaluate the expansion plans over the whole set of 35 climatic years. We then compare the obtained results against the true solution of the problem (i.e. the stochastic solution using the full set of 35 climatic years, which is computed in chapter 4). Figure 5.1 compares costs relative to the cost of the true solution. The costs are divided into different categories: the expansion plan, the economic dispatch and the load shedding. We note that the methodology is able to capture more

accurately each item, with a particular interest in load shedding (since load shedding determines the adequacy metrics of the models).

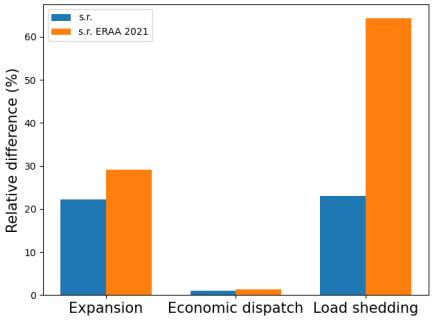


Figure 5.1: Cost comparison relative to the optimal solution.

In fact, Figure 5.2 presents the LOLE and EENS[2] errors obtained when comparing the optimal solution against the metrics obtained when using the approximate expansion plans. The metrics are calculated as the average over the 35 climatic years. The left panel presents the results for the LOLE while the right panel presents the EENS results. We note that both metrics present improvements as compared to the scenario reduction strategy followed by ENTSO-E. Furthermore, note that, while there are a few zones where the LOLE error of the proposed methodology is higher than the approach of ENTSO-E, overall the LOLE error improves substantially. This indicates that our proposed methodology is better suited for quantifying the adequacy metrics. Nonetheless, we highlight that in our study we have not considered the non-uniqueness of load shedding patterns.

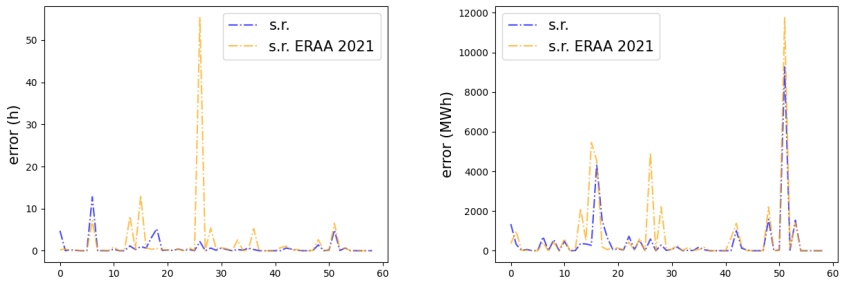


Figure 5.2: Adequacy comparison relative to the optimal solution for each zone. The x-axis corresponds to the different zones while the y-axis is the absolute error. The left panel presents the LOLE, while the right panel presents the EENS.

[2]The definition of LOLE and EENS is introduced in chapter 4, subsection 4.6.4.

The analysis is complemented by considering the LOL and ENS [3] for each individual climatic year, and comparing them against the true metric for each climatic year (which was found by solving the stochastic problem using all 35 climatic years). Figure 5.3 presents a boxplot of the results. The boxplot corresponds to the 35 error observations. The left panel presents the LOL, while the right panel presents the results for the ENS. We observe that the proposed approach reduces the width of the box, and the median error. We further observe that the highest value whiskers are reduced and the outlier error is also improved. This indicates that our proposed method achieves a consistent improvement across the different climatic conditions.

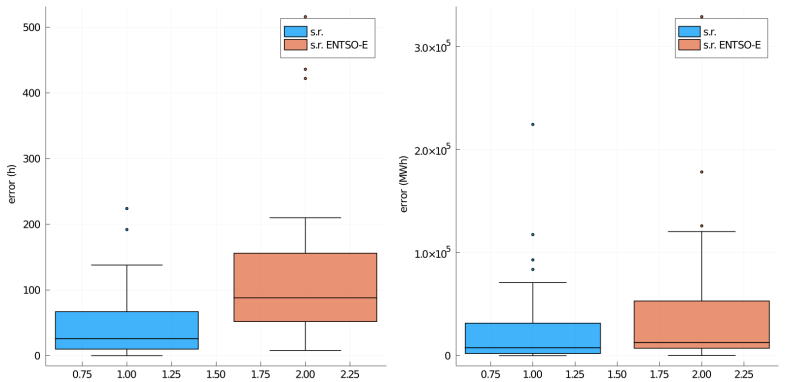


Figure 5.3: The boxplot considers 35 error observations, each one corresponding to a different climatic year. The y-axis corresponds to the absolute error. The left panel presents the LOL error while the right panel presents the ENS error.

From a practical point of view, developing a technique that selects an appropriate number of clusters is particularly relevant. In particular, given the size of the optimization models and their associated run times, it is challenging to perform experiments with a large variety of reduction sizes so as to find one that is most suitable. In this sense, thanks to the hierarchical structure of the methodology proposed in this chapter, it is possible to define certain heuristics. Following a similar idea as when reducing the dimensionality through Principal Component Analysis (PCA), we can plot the distance of the joined clusters (step 2.2 of Alg. 14) with respect to each reduction target, as presented in Figure 5.4. Ideally, the figure gives us an estimate on how big the distances become as the reduction proceeds. As a consequence, we can define a stopping criteria heuristic by stopping the algorithm at the point where a further reduction leads to a drastic increase in the measured distance, while less reduction does not provide remarkable benefits. In Figure 5.4 we observe that a choice of 7 scenarios seems as a reasonable strategy, while using less than 4 seems to decrease the quality of the solution considerably.

[3]The definition of LOL and ENS is introduced in chapter 4, subsection 4.6.4.

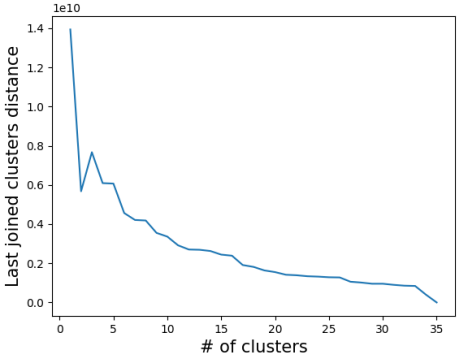


Figure 5.4: Optimal reduction target heuristic. The x-axis corresponds to the number of clusters, while the y-axis corresponds to the distance of the last two joined clusters.

For the present study, we consider 3 climatic years and, as for the case with 7 climatic years, we compare the solution against the optimal stochastic solution that considers the full set of 35 climatic years. The results are presented in Figure 5.5. We note that the choice of 3 climatic years worsens the quality of the solution. In particular, the load shedding error increases substantially. These results seem to be somewhat coherent with the heuristic presented in Figure 5.4. Further investigation is required in order to determine the quality of the heuristic and what can be considered as a good enough solution for the purposes of the ERAA study.

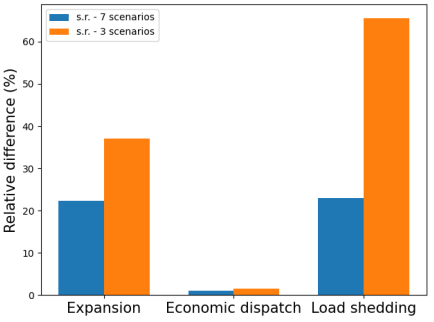


Figure 5.5: Cost comparison relative to the optimal solution.

In order to finalize this preliminary analysis, we highlight that different choices for measuring the distance between scenarios are possible. For instance, one could instead measure the distance between scenarios as the distance associated to their first-stage optimal decisions. Another possibility is to measure the distance as the difference between the optimal cost associated to each sce-

nario [MPCC09]. These different strategies are subject to future research.

5.3.2 Comparison against forward selection algorithm

The forward selection algorithm proposed in [DGKR03] is a common technique used to achieve scenario reduction, which is implemented in commercial software such as GAMS [sce]. The algorithm proceeds by selecting the scenario, from the set of non-selected scenarios, that minimizes the Kantorovich distance (i.e. the solution to the optimal transport problem) between the reduced set of scenarios and the original set. Despite being a heuristic, meaning that no theoretical guarantee is provided regarding its performance, empirical evidence shows that in practice the obtained reduced sets perform well [DGKR03].

For ERAA 2021 a total of 7 climatic years were selected. As in the previous subsection, we let our scheme and the forward selection algorithm select 7 of 35 climatic years. Next we solve the reduced stochastic problem, using the methods described in chapter 4. Once the stochastic solution is obtained, its performance is evaluated by solving the economic dispatch problem over the full uncertainty set, and we compare it against the true optimal solution. The results are presented in Figure 5.6, where it is observed that the proposed approach is able to capture more accurately the different cost components, in particular the cost of load shedding.

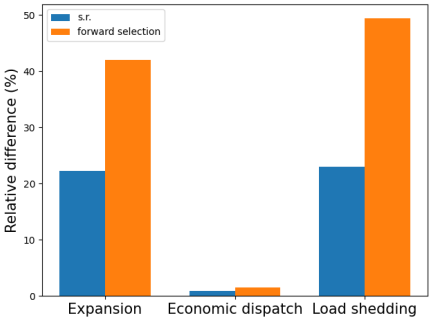


Figure 5.6: Cost comparison relative to the optimal solution. The scenario reduction selects 7 climatic years.

Figure 5.7 presents the LOLE and EENS errors per region. We observe that the LOLE tends to be better captured by the proposed scheme, while for the EENS the difference is less notable and large differences are observed only in specific regions.

Finally, in order to assess the quality of the reduction as the number of selected scenarios decreases, we allow both scenario reduction techniques to select 3 out of 35 climatic years. As before, we present the cost comparison relative to the true stochastic solution over the full set of uncertainty realizations. The results are presented in Figure 5.8, where we observe a considerable

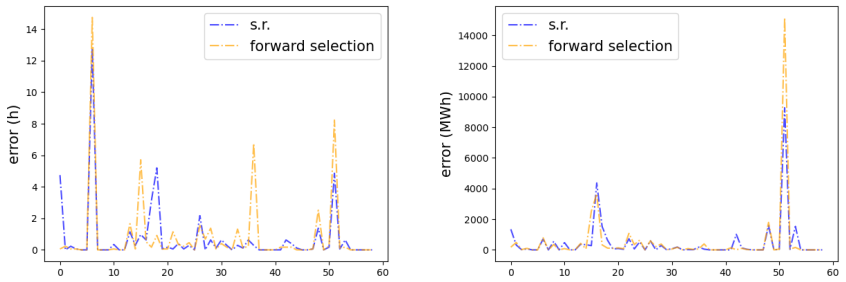


Figure 5.7: Adequacy comparison relative to the optimal solution for each zone. The x-axis corresponds to the different zones while the y-axis is the absolute error. The left panel presents the LOLE, while the right panel presents the EENS.

deterioration in performance for the forward scenario technique.

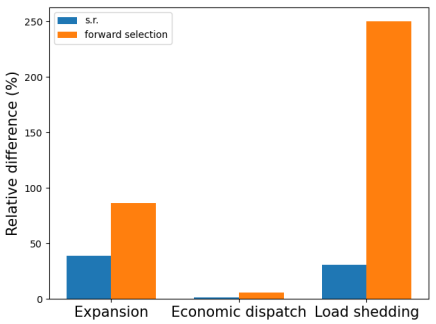


Figure 5.8: Cost comparison relative to the optimal solution. The scenario reduction selects 3 climatic years.

5.4 Open remarks

This section aims at summarizing relevant open questions and fronts of study, which the purpose of improving the overall methodology.

- Thanks to the hierarchical structure of the clustering, one could envision strategies for defining stopping criteria for scenario reduction. As discussed previously, this is a practically relevant question. The dissertation proposes a heuristic for achieving this. Nevertheless, further testing is required in order to understand the behaviour of the proposed heuristic.
- Theoretical results regarding the error bounds within each cluster are of interest and are within the scope of future research.

- The literature offers scenario reduction schemes that are not based on hierarchical clustering. A relevant strategy is the forward selection algorithm described in [DGKR03]. Although this approach requires mapping each uncertainty realization to a vector in $\mathbb{R}^n$ (how this is done in the current setting with composite uncertainty is unclear), other approaches are available such as [MPCC09]. The effect of considering a forward selection algorithm or a hierarchical clustering approach is unknown.

- The choice of the appropriate distance between scenarios is a matter of experimental study. The distance considered in the present chapter, which is proposed in [HOR22], has two major disadvantages: (i) it is computationally expensive, and (ii) it is not a distance in the proper mathematical sense, thus leading to situations where elements within a cluster are in fact not near. The literature offers other possibilities, for example the distance defined in [MPCC09].

- What should be considered as a good approximate solution? This is especially relevant, given that one of the most relevant parts of adequacy studies is load shedding, which represents a small portion of the total system cost.

- The fact that multiple solutions of load shedding may exist.

- Given that the ultimate objective is to properly quantify load shedding, one could define distances where the impact of load shedding is higher.

5.5 Conclusions

This chapter has introduced a scenario reduction scheme based on hierarchical clustering and where the impact of scenarios on the objective function of the underlying mode is accounted for. The preliminary analysis that we present in this chapter identifies advantages, relative to the approach followed by ENTSO-E for ERAA 2021. In particular, the approach seems to be better suited for capturing impacts in the total system cost, such as load shedding, which are of high relevance for the purpose of ERAA. The methodology proposed in this chapter has been implemented by ENTSO-E in the ERAA 2022 study [erae].

We identify several open problems that inspire further research, in order to develop a suitable strategy for reducing the complexity of the EVA study.

6 Conclusions and future perspectives

6.1 Summary of the contributions

Optimization under uncertainty has emerged within power systems, as a combination of tools for allowing decision makers to reach better informed decisions. Nonetheless, as the modelling specifications increase, standard methodologies become insufficient and the search for new algorithmic ideas becomes imperative. The present thesis proposes high performance computing algorithms and tailored solution methodologies for tackling relevant problems within power systems. We specifically focus on: (i) the long-term hydrothermal scheduling problem, and (ii) the European Resource Adequacy Assessment. This thesis is organized into four main chapters.

Chapter 2 considers the class of problems referred to as multistage stochastic linear problems. A famous example in this class of problems is the long-term hydrothermal planning problem. In this class of problems, one algorithm that has captivated the interest of both industry and academia is the SDDP algorithm. This chapter studies several parallelization strategies for such an algorithmic scheme, considering both synchronous as well as asynchronous computing. The proposed algorithmic schemes are tested on an instance of the Brazilian hydrothermal power system and an inventory problem. These case studies allow us to quantify the performance of the different schemes. Our results indicate that parallelization in its own may not be enough to achieve a scalable algorithmic strategy.

The findings of chapter 2 allow us to identify parallel computing issues behind SDDP. These revelations serve as an inspiration for the developments of chapter 3, where the focus is directed into providing more accurate value function approximations during each stage of decision making. This idea leads to the proposal of a novel algorithmic scheme, which we refer to as BL-SDDP. The proposed algorithm converges faster than the PSR SDDP software on various test instances, and we point out to connections between the proposed algorithm and ideas from the reinforcement learning framework. Several computational experiments are performed over two different instances of hydrothermal plan-

ning. One set of tests considers an instance of the Colombian power system, while another considers a known hard instance of the Brazilian power system. Further tests are performed using an inventory management problem with lead times.

Chapter 4 draws the attention to an ongoing European initiative, the European Resource Adequacy Assessment (ERAA). This is a problem of high institutional importance for the European market, since it aims at becoming the key instrument for detect adequacy concerns in the pan-European region. As such, it is likely to become a key ingredient for transmission system operators to apply for capacity mechanisms in the future European market. Within this a study, a critical step is to determine expansion plans that occur in the coming years, the so-called Economic Viability Assessment (EVA). The scale of the problem has revealed that industry grade tools are unable to tackle it. This dissertation proposes, as a first step, a way to tackle the two-stage formulation of the problem, the solution of which has not been achieved by state of the art methods so far, to the best of our knowledge.

Due to the increased computational challenges behind the EVA, ENTSO-E has also pursued scenario reduction strategies in order to decrease the computational burden of the model. Chapter 5 proposes a scenario reduction methodology for scenario resuction. A preliminary analysis demonstrates benefits relative to the method used by ENTSO-E for ERAA 2021. The proposed scheme is implemented by ENTSO-E for ERAA 2022.

6.2 Summary of the findings

Parallelization of the SDDP algorithm

- **Parallel scenario and parallel node schemes.** This dissertation proposes a novel parallelization strategy for SDDP, referred to as the parallel node strategy. This strategy is distinguished from the parallel scenario scheme (which is the most common strategy found in the literature) by the way in which the work for computing cuts is distributed. The parallel node scheme is observed to converge faster than the parallel scenario strategy at early steps of execution.

- **Scalability of the parallel node scheme.** The parallel node strategy presents desirable scalability properties as the CPU count increases. Unfortunately, such a behaviour is restricted to machines with shared memory. In a distributed memory setting, the solve time of each subproblem has to be considerably larger than the latency of the network in order for the scheme to work properly.

- **Scalability of the parallel scenario scheme.** The increase in forward samples may lead to a deterioration in the performance of the algorithm. This behaviour translates into poor scalability of the parallel scenario strategy, for certain instances, as the CPU count increases.

- **Synchronous and asynchronous computing.** The use of asynchronous computing presents limited benefits for the parallel scenario and node schemes.
- **Limitations of parallelization strategies for SDDP.** The different parallel strategies reach a point in time where their difference in terms of the obtained optimality gap is not significant.

Reinforcement Learning (RL) and SDDP

- **Connection between stochastic programming and reinforcement learning.** This dissertation has presented a mapping between multistage stochastic programming problems and Markov decision processes, which underlie the RL framework. We note that our view compares with previous work [AP18], in that we propose a map between Markov decision Processes and multistage stochastic problems, by providing an interpretation of Q-factors in terms of the standard cost-to-go functions of the multistage stochastic programming framework. We note that [AP18] does not provide an interpretation as a Markov Decision Process.
- **SDDP as an RL algorithm.** Using the bridge between multistage stochastic programs and Markov decision processes, the SDDP algorithm is described as an RL algorithm. SDDP exhibits a similar structure as double-pass algorithms in RL.
- **Batch learning SDDP.** This thesis introduces the Batch learning SDDP (BL-SDDP) scheme. Supported by the description of SDDP as an RL algorithm, this method uses batch learning through experience replay, a method inspired by the RL framework. The algorithm exhibits faster convergence than the industrial grade PSR SDDP commercial implementation. The parallel computing version of the algorithm is empirically observed to be superior than the standard parallel SDDP method on numerous test instanes.
- **Back-propagation of information.** This dissertation has empirically demonstrated that the SDDP algorithm may tend to be susceptible to back-propagating errors throughout stages. In turn, this undermines the algorithmic performance of SDDP. The BL-SDDP scheme tackles this issue by allowing the algorithm to arrive at a more accurate value function approximation at every stage, as well as allowing previously visited actions to have access to the value function updates carried out in upper stages.

European Resource Adequacy Assessment

- **EVA stochastic solution.** We use decomposition schemes and parallel computing to solve the Economic Viability Assessment (EVA) of the European Resource Adequacy Assessment (ERAA) 2021 edition.

- **Novel two-stage stochastic algorithms.** We consider a subgradient algorithm and a novel algorithmic relaxation scheme, capable of tackling the Economic Viability Assessment (EVA) in its stochastic form. In particular, the latter algorithm converges considerably faster than the subgradient algorithm. It is observed to be better suited for larger and more complex instances.

- **Impact on adequacy metrics.** We study the impact that approximate approaches, such as the one used for ERAA 2021, may have in the overall solution and in particular on adequacy metrics. Such adequacy metrics are highly relevant for the purposes of the ERAA study.

- **Scenario reduction.** The dissertation has proposed a scenario reduction methodology for reducing the complexity of the EVA. The proposed method is better suited for accounting for the impact of scenarios on total system cost. The method is also able to capture the interaction between scenarios and adequacy metrics, as compared to ENTSO-E's strategy for ERAA 2021.

6.3 Future perspectives

This subsection proposes a number of future research perspectives that are inspired by the developments of this dissertation.

ERAA: Multi-year formulation. The ERAA aims at delivering adequacy assessments for a time horizon of 10 years. As such, the Economic Viability Assessment model has the task of determining expansion and retirement opportunities in a multi-year setting. This naturally leads to a multi-stage stochastic problem. The increased complexity of such a model likely means that seeking for an exact solution is an intractable task. Therefore, a future front of study must determine a reasonable approximation strategy for tackling the problem. The hydrothermal scheduling framework inspires a workaround of interest. The hydrothermal literature often breaks the overall problem into long-term, medium-term and short-term problems. Each is of a finer granularity than the previous one. The long-term value functions are used as end conditions for the medium-term problem, while the medium-term value functions are used as boundary conditions for the short-term models. By doing this, instead of solving a intractable detailed model for a long-term horizon, several subproblems are solved, where each one is focused on a different front of the overall problem.

The success of such an approach may be translated to a feasible approach for the purposes of ERAA. Concretely, using the developments presented in chapter 3, a model with less granularity can be solved for a large horizon. The obtained value functions can then be used as end conditions for solving the

single-year problem, for which the developments carried out in chapter 4 can be used.

ERAA: Scenario reduction. As discussed in Chapter 5, several open fronts are left for investigation. In particular, a better understanding of the hierarchical structure of the clustering method would be of practical relevance as it would allow to develop stopping criteria strategies. Further testing against competitive strategies is required in order to strengthen the proposed approach. On the other hand, as discussed previously, the Economic Viability Assessment aims for a multi-year setting. Consequently, it remains an open question how to develop a disciplined methodology to select scenarios in such a framework.

ERAA: Modelling assumptions. (i) Due to the scale of the Economic Viability Assessment, strategies for simplifying the model are of interest as a way to cope with the computational burden. To do so, it is relevant to understand which modeling aspects play an important role: the network, the stochastic nature, the granularity, the multi-stage nature of the decisions, the hydrology etc. This thesis has demonstrated the importance of uncertainty and network constraints on the outcome of the adequacy analysis (the reader is referred to chapter 4 for additional details). (ii) The system cost minimization approach, followed in the Economic Viability Assessment, may yield undesirable results. Cost minimization and profit maximization are equivalent as long as the problem is convex, eg if an EENS target is set as the criterion for adequacy. If a LOLE criterion is used, then the problem is non-convex, and in general there does not exist a decentralized profit-maximizing equivalent model which leads to the same result as the centralized one. Which suggests a fundamental contradiction in an adequacy assessment which is performed as a means of simulating a decentralized market environment. in particular, it does not guarantee a unique pattern for shedding load. That is to say, there may be different optimal solutions with different load shedding profiles, and thus different adequacy indicators.

Further applications of RL within SDDP The presented map between stochastic programming and RL, and in particular the description of SDDP as an RL algorithm, allows us to envision applying further RL ideas within the SDDP framework. The RL literature offers a wide range of techniques, for example the TD-type algorithms. It would be interesting to understand how / if such techniques can be coordinated with SDDP in order to lead to improved algorithmic schemes.

Hyperplane selection rules. The BL-SDDP scheme proposes a scheme for re-visiting a batch of visited trial actions. An open front for speeding up the algorithmic procedure is to define priority rules over the set of visited trial

actions, so as to avoid the computational burden of re-visiting the full set of previous actions.

Asynchronous computing Both the BL-SDDP scheme and the algorithms developed in Chapter 4 are susceptible to known parallel computing synchronization issues. It remains open to investigate the effect of synchronization in overall algorithmic performance.

Bibliography

[AABA14] Jamshid Aghaei, Nima Amjady, Amir Baharvandi, and Mohammad-Amin Akbari. Generation and transmission expansion planning: Milp–based probabilistic model. *IEEE Transactions on Power Systems*, 29(4):1592–1601, 2014.

[ABDBL21] Margareta Ackerman, Shai Ben-David, Simina Brânzei, and David Loker. Weighted clustering: Towards solving the user's dilemma. *Pattern Recognition*, 120:108152, 2021.

[AP18] Tsvetan Asamov and Warren B Powell. Regularized decomposition of high-dimensional multistage stochastic programs with markov uncertainty. *SIAM Journal on Optimization*, 28(1):575–595, 2018.

[AP20] Ignacio Aravena and Anthony Papavasiliou. Asynchronous lagrangian scenario decomposition. *Mathematical Programming Computation*, pages 1–50, 2020.

[AP21] Ignacio Aravena and Anthony Papavasiliou. Asynchronous lagrangian scenario decomposition. *Mathematical Programming Computation*, 13(1):1–50, 2021.

[ÁPL21] D Ávila, A Papavasiliou, and N Löhndorf. Parallel and distributed computing for stochastic dual dynamic programming. *Computational Management Science*, pages 1–28, 2021.

[AR70] Nicolaos V Arvanitidits and Jakob Rosing. Composite representation of a multireservoir hydroelectric power system. *IEEE Transactions on Power Apparatus and Systems*, (2):319–326, 1970.

[B+11] Dimitri P Bertsekas et al. Dynamic programming and optimal control 3rd edition, volume ii. *Belmont, MA: Athena Scientific*, 2011.

[BAT21] Stian Backe, Mohammadreza Ahang, and Asgeir Tomasgard. Stable stochastic capacity expansion with variable renewables: Comparing moment matching and stratified scenario generation sampling. *Applied Energy*, 302:117538, 2021.

[BB03] Anastasios G Bakirtzis and Pandelis N Biskas. A decentralized solution to the dc-opf of interconnected power systems. *IEEE Transactions on Power Systems*, 18(3):1007–1013, 2003.

[BBC12] Grigorios A Bakirtzis, Pandelis N Biskas, and Vasilis Chatziathanasiou. Generation expansion planning by milp considering mid-term scheduling decisions. *Electric Power Systems Research*, 86:98–112, 2012.

[BEKS17] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017. URL: `https://doi.org/10.1137/141000671`.

[BL11] John R Birge and Francois Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[Blo82] Jeremy A Bloom. Long-range generation planning using decomposition and probabilistic simulation. *IEEE Transactions on Power Apparatus and Systems*, (4):797–802, 1982.

[Boy] Subgradient methods. `https://web.stanford.edu/class/ee364b/lectures/subgrad_method_slides.pdf`. Accessed: 2022-09-26.

[BSdG+22] Stian Backe, Christian Skar, Pedro Crespo del Granado, Ozgu Turgut, and Asgeir Tomasgard. Empire: An open-source model based on multi-horizon programming for energy transition analyses. *SoftwareX*, 17:100877, 2022.

[BT89] Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

[Com19a] European Commission. Clean energy package. `https://energy.ec.europa.eu/topics/energy-strategy/clean-energy-all-europeans-package_en`, 2019. Accessed: 2022-09-26.

[Com19b] European Commission. Regulation (eu) 2019/941 of the european parliament and of the council of 5 june 2019 on risk-preparedness in the electricity sector and repealing directive 2005/89/ec. *Off. J. Eur. Union*, 158:1–21, 2019.

[Com19c] European Commission. Regulation (eu) 2019/943 of the european parliament and of the council of 5 june 2019 on the internal market for electricity. *Off. J. Eur. Union*, 158:54–124, 2019.

[Com22] European Comission. Electricity price statistics . https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Electricity_price_statistics#Electricity_prices_for_household_consumers, 2022. Accessed: 2022-10-22.

[DB06] Christopher Donohue and John Birge. The abridged nested decomposition method for multistage stochastic linear programs with relatively complete recourse. *Algorithmic Operations Research*, 1(1):20–30, 2006.

[DGKR03] Jitka Dupačová, Nicole Gröwe-Kuska, and Werner Römisch. Scenario reduction in stochastic programming. *Mathematical programming*, 95:493–511, 2003.

[DHL17] Iain Dunning, Joey Huchette, and Miles Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017. doi:10.1137/15M1020575.

[DK17] O. Dowson and L. Kapelevich. SDDP.jl: a Julia package for stochastic dual dynamic programming. *Optimization Online*, 2017. URL: http://www.optimization-online.org/DB_HTML/2017/12/6388.html.

[DK21] Oscar Dowson and Lea Kapelevich. Sddp. jl: a julia package for stochastic dual dynamic programming. *INFORMS Journal on Computing*, 33(1):27–33, 2021.

[DMPF15] Vitor L De Matos, Andy B Philpott, and Erlon C Finardi. Improving the performance of stochastic dual dynamic programming. *Journal of Computational and Applied Mathematics*, 290:196–208, 2015.

[DMPFG10] V De Matos, Andrew B Philpott, Erlon C Finardi, and Ziming Guan. Solving long-term hydro-thermal scheduling problems. Technical report, Technical report, Electric Power Optimization Centre, University of Auckland, 2010.

[DPMD19] Oscar Dowson, Andy Philpott, Andrew Mason, and Anthony Downward. A multi-stage stochastic optimization model of a pastoral dairy farm. *European Journal of Operational Research*, 274(3):1077–1089, 2019.

[dSF03] Edson Luiz da Silva and Erlon Cristian Finardi. Parallel processing applied to the planning of hydrothermal systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(8):721–729, 2003.

[EE21] ENTSO-E. European resource adequacy assessment. 2021 edition. https://www.entsoe.eu/outlooks/eraa/2021, 2021. Accessed: 2022-09-26.

[eraa] European resource adequacy assessment. 2021 edition. annex 1 assumptions. https://eepublicdownloads.azureedge.net/clean-documents/sdc-documents/ERAA/ERAA_Annex_1_Assumptions.pdf. Accessed: 2022-09-26.

[erab] European resource adequacy assessment. 2021 edition. annex 3 methodology. https://eepublicdownloads.azureedge.net/clean-documents/sdc-documents/ERAA/ERAA_2021_Annex_3_Methodology.pdf. Accessed: 2022-09-26.

[erac] European resource adequacy assessment. 2021 edition. executive report. https://eepublicdownloads.azureedge.net/clean-documents/sdc-documents/ERAA/ERAA_2021_Executive%20Report.pdf. Accessed: 2022-09-26.

[erad] Methodology for the european resource adequacy assessment. https://www.acer.europa.eu/sites/default/files/documents/Individual%20Decisions_annex/ACER%20Decision%2024-2020%20on%20ERAA%20-%20Annex%20I_1.pdf. Accessed: 2022-09-26.

[erae] Methodology for the european resource adequacy assessment. https://eepublicdownloads.azureedge.net/clean-documents/sdc-documents/ERAA/2022/data-for-publication/ERAA2022_Annex_2_Methodology.pdf. Accessed: 2023-03-30.

[FBP10] BC Flach, LA Barroso, and MVF Pereira. Long-term optimal allocation of hydro generation for a price-maker company in a competitive market: latest developments and a stochastic dual dynamic programming approach. *IET generation, transmission & distribution*, 4(2):299–314, 2010.

[FR13] Yonghan Feng and Sarah M Ryan. Scenario construction and reduction applied to stochastic power generation expansion planning. *Computers & Operations Research*, 40(1):9–23, 2013.

[GAC14] Esteban Gil, Ignacio Aravena, and Raúl Cárdenas. Generation capacity expansion planning under hydro uncertainty using stochastic mixed integer programming and scenario reduction. *IEEE Transactions on Power Systems*, 30(4):1838–1847, 2014.

[GÁM+22] Céline Gérard, Daniel Ávila, Yuting Mou, Anthony Papavasiliou, and Philippe Chevalier. Comparison of priority service with multilevel demand subscription. *IEEE Transactions on Smart Grid*, 2022.

[GB19] Vincent Guigues and Michelle Bandarra. Single cut and multicut sddp with cut selection for multistage stochastic linear programs: convergence proof and numerical experiments. *arXiv preprint arXiv:1902.06757*, 2019.

[GBD21] Sebastian Gonzato, Kenneth Bruninx, and Erik Delarue. Long term storage in generation expansion planning models with a reduced temporal scope. *Applied Energy*, 298:117168, 2021.

[GCCP93] BG Gorenstin, NM Campodonico, JP Costa, and MVF Pereira. Power system expansion planning under uncertainty. *IEEE transactions on power systems*, 8(1):129–136, 1993.

[GLCP23] Joaquim Dias Garcia, Iago Leal, Raphael Chabar, and Mario Veiga Pereira. A multicut approach to compute upper bounds for risk-averse sddp. *arXiv preprint arXiv:2307.13190*, 2023.

[Gui17] Vincent Guigues. Dual dynamic programing with cut selection: Convergence proof and numerical experiments. *European Journal of Operational Research*, 258(1):47–57, 2017.

[HB15] Arild Helseth and Hallvard Braaten. Efficient parallelization of the stochastic dual dynamic programming algorithm applied to hydropower scheduling. *Energies*, 8(12):14287–14297, 2015.

[HOR22] Mike Hewitt, Janosch Ortmann, and Walter Rei. Decision-based scenario clustering for decision-making under uncertainty. *Annals of Operations Research*, 315(2):747–771, 2022.

[HP14] Magnus Hindsberger and AB Philpott. Resa: A method for solving multistage stochastic linear programs. *Journal of Applied Operational Research*, 6(1):2–15, 2014.

[HR03] Holger Heitsch and Werner Römisch. Scenario reduction algorithms in stochastic programming. *Computational optimization and applications*, 24:187–206, 2003.

[JRWW11] Shan Jin, Sarah M Ryan, Jean-Paul Watson, and David L Woodruff. Modeling and solving a large-scale generation expansion planning problem under uncertainty. *Energy Systems*, 2(3):209–242, 2011.

[KB97] Balho H Kim and Ross Baldick. Coarse-grained distributed optimal power flow. *IEEE Transactions on Power Systems*, 12(2):932–939, 1997.

[KS07] Shivaram Kalyanakrishnan and Peter Stone. Batch reinforcement learning in a complex domain. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, 2007.

[KS10] S Kamalinia and M Shahidehpour. Generation expansion planning in wind-thermal power systems. *IET generation, transmission & distribution*, 4(8):940–951, 2010.

[LGR12] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[Lin92] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

[LS19] Nils Löhndorf and Alexander Shapiro. Modeling time-dependent randomness in stochastic dual dynamic programming. *European Journal of Operational Research*, 273(2):650–661, 2019.

[LSLZ16] Alvaro Lorca, X Andy Sun, Eugene Litvinov, and Tongxin Zheng. Multistage adaptive robust optimization for the unit commitment problem. *Operations Research*, 64(1):32–51, 2016.

[LW20] Nils Löhndorf and David Wozabal. Gas storage valuation in incomplete markets. *European Journal of Operational Research*, 2020.

[LWM13] Nils Löhndorf, David Wozabal, and Stefan Minner. Optimizing trading decisions for hydro storage systems using approximate dual dynamic programming. *Operations Research*, 61(4):810–823, 2013.

[MDBB21] Felipe DR Machado, Andre Luiz Diniz, Carmen LT Borges, and Lilian C Brandão. Asynchronous parallel stochastic dual dynamic programming applied to hydrothermal generation planning. *Electric Power Systems Research*, 191:106907, 2021.

[MKS15] Volodymyr Mnih, Koray Kavukcuoglu, and et. al Silver. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[Mor19] Elvira Moreno. Decision-based scenario clustering in robust stochastic optimization, 2019.

[MPCC09] Juan M Morales, Salvador Pineda, Antonio J Conejo, and Miguel Carrion. Scenario reduction for futures market trading in electricity markets. *IEEE Transactions on Power Systems*, 24(2):878–888, 2009.

[MPG87] A Monticelli, MVF Pereira, and S Granville. Security-constrained optimal power flow with post-contingency corrective rescheduling. *IEEE Transactions on Power Systems*, 2(1):175–180, 1987.

[PB16] Heejung Park and Ross Baldick. Multi-year stochastic generation capacity expansion planning under environmental energy policy. *Applied energy*, 183:737–745, 2016.

[PBM13] Roberto J Pinto, CarmenL T Borges, and Maria EP Maceira. An efficient parallel algorithm for large scale hydrothermal system operation planning. *IEEE Transactions on Power Systems*, 28(4):4888–4896, 2013.

[PDM12] Andrew B Philpott and Vitor L De Matos. Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion. *European Journal of operational research*, 218(2):470–483, 2012.

[PG08] Andrew B Philpott and Ziming Guan. On the convergence of stochastic dual dynamic programming and related methods. *Operations Research Letters*, 36(4):450–455, 2008.

[PMCS17] Anthony Papavasiliou, Yuting Mou, Léopold Cambier, and Damien Scieur. Application of stochastic dual dynamic programming to the real-time dispatch of storage under renewable supply uncertainty. *IEEE Transactions on Sustainable Energy*, 9(2):547–558, 2017.

[PO13] Anthony Papavasiliou and Shmuel S Oren. Multiarea stochastic unit commitment for high wind penetration in a transmission constrained network. *Operations research*, 61(3):578–592, 2013.

[POO11] Anthony Papavasiliou, Shmuel S Oren, and Richard P O'Neill. Reserve requirements for wind power integration: A scenario-based stochastic programming framework. *IEEE Transactions on Power Systems*, 26(4):2197–2206, 2011.

[POR14] Anthony Papavasiliou, Shmuel S Oren, and Barry Rountree. Applying high performance computing to transmission-constrained stochastic unit commitment for renewable energy integration. *IEEE Transactions on Power Systems*, 30(3):1109–1120, 2014.

[Pow07] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[PP91] Mario VF Pereira and Leontina MVG Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1-3):359–375, 1991.

[PSR] PSR. Sddp - stochastic hydrothermal dispatch with network restrictions. URL: `https://www.psr-inc.com/softwares-en/?current=p4028`.

[Put14] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[PW16] Mathijs Pieters and Marco A Wiering. Q-learning with experience replay in a dynamic environment. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2016.

[RM18] Aksel Reiten and Mats Erik Mikkelsen. A stochastic capacity expansion model for the european power system-using a distributed progressive hedging approach to handle long-term uncertainty. Master's thesis, NTNU, 2018.

[RSW09] Jae Hyung Roh, Mohammad Shahidehpour, and Lei Wu. Market-based generation and transmission planning with uncertainties. *IEEE Transactions on Power Systems*, 24(3):1587–1598, 2009.

[RW91] R Tyrrell Rockafellar and Roger J-B Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1):119–147, 1991.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[sce] Scenario reduction gams. `https://www.gams.com/latest/docs/T_SCENRED.html`. Accessed: 2023-07-30.

[Sha11] Alexander Shapiro. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209(1):63–72, 2011.

[SSAG22] Alessandro Soares, Alexandre Street, Tiago Andrade, and Joaquim Dias Garcia. An integrated progressive hedging and benders decomposition with multiple master method to solve the brazilian generation expansion problem. *IEEE Transactions on Power Systems*, 37(5):4017–4027, 2022.

[STdCS13] Alexander Shapiro, Wajdi Tekaya, Joari Paulo da Costa, and Murilo Pereira Soares. Risk neutral and risk averse stochastic dual dynamic programming method. *European journal of operational research*, 224(2):375–391, 2013.

[STK07] Jiraporn Sirikum, Anulark Techanitisawad, and Voratas Kachitvichyanukul. A new efficient ga-benders' decomposition method: For power generation expansion planning with emission controls. *IEEE Transactions on Power Systems*, 22(3):1092–1100, 2007.

[SY86] Suvrajeet Sen and Diana Yakowitz. A primal-dual subgradient method for time staged capacity expansion planning. *European journal of operational research*, 27(3):301–312, 1986.

[TBL96] Samer Takriti, John R Birge, and Erik Long. A stochastic model for the unit commitment problem. *IEEE Transactions on Power Systems*, 11(3):1497–1508, 1996.

[TPPM90] MJ Teixeira, HJCP Pinto, MVF Pereira, and MF McCoy. Developing concurrent processing applications to power system planning and operations. *IEEE Transactions on Power Systems*, 5(2):659–664, 1990.