HEC MONTRÉAL

A Comparative Analysis of Reinforcement Learning and Stochastic Dual Dynamic Programming for Optimizing Energy Dispatch in Québec's Hydroelectric Grid

par

Iheb Hajji

Michel Denault	**Pierre-Olivier Pineau**
HEC Montréal	**HEC Montréal**
Directeur de recherche	**Co-Directeur de recherche**

Sciences de la gestion
(Spécialisation Financial Engineering)

Mémoire présenté en vue de l'obtention
du grade de maîtrise ès sciences
(M. Sc.)

December 2024

Résumé

Ce mémoire examine l'utilisation de la Programmation Dynamique Stochastique Duale (SDDP) et de l'Apprentissage par Renforcement (RL) pour la gestion dynamique de la production d'énergie dans des systèmes multi-réservoirs hydroélectriques, dans le but de minimiser les coûts opérationnels tout en gérant les incertitudes liées aux apports d'eau. En utilisant une sous-région du réseau hydroélectrique du Québec comme étude de cas, SDDP a démontré de solides performances dans la résolution de problèmes d'optimisation stochastique multi-étapes, bien que son temps de calcul augmente exponentiellement avec la complexité du problème. En revanche, les algorithmes de RL offre une évolutivité nettement meilleure pour les problèmes à plusieurs étapes. REINFORCE donne une solution très proche à la solution optimale de SDDP tout en nécessitant un temps de calcul considérablement réduit. Ces résultats soulignent le potentiel de l'apprentissage par renforcement comme alternative efficace sur le plan computationnel aux méthodes d'optimisation traditionnelles dans les scénarios de gestion dynamique et incertaine de l'énergie.

Mots-clés

Gestion des systèmes multi-réservoirs hydroélectriques; Optimisation stochastique; Programmation Dynamique Stochastique Duale; Apprentissage par Renforcement; Algorithme REINFORCE

Abstract

This thesis investigates the use of Stochastic Dual Dynamic Programming (SDDP) and Reinforcement Learning (RL) for dynamic energy dispatch in multi-hydro-reservoir systems, with the goal of minimizing operational costs while managing uncertain water inflows. Using a sub-region of Quebec's hydroelectric grid as a case study, SDDP demonstrated strong performance in solving multi-stage stochastic optimization problems, though its computational time increased exponentially with problem complexity. To address this limitation, the RL algorithms was evaluated for its scalability and efficiency. REINFORCE closely approximated SDDP's optimal solutions while requiring significantly less computational time. These results emphasize RL's potential as a computationally efficient and scalable alternative to traditional optimization methods for managing dynamic and uncertain energy systems.

Keywords

Energy Dispatch; Multi-Hydro-Reservoir Management; Stochastic Optimization; Stochastic Dual Dynamic Programming; Reinforcement Learning; REINFORCE Algorithm

Contents

List of Tables

List of Figures

List of Acronyms

ED	Energy Dispatch
SDDP	Stochastic Dual Dynamic Programming
RL	Reinforcement Learning
DP	Dynamic Programming
MC	Monte Carlo
MSSP	Multi-Stage Stochastic Programming
MSLP	Multi-Stage Linear Programming
OOS	Out-Of-Sample
NBD	Nested Benders Decomposition
ROR	Run-of-the-river
IDR	Intra-day

Acknowledgements

I would like to express my deepest gratitude to my supervisors Prof. *Michel Denault* and Prof. *Pierre-Olivier Pineau*, for their invaluable guidance, encouragement, and expertise throughout this research journey. Their insights and constructive feedback have been instrumental in shaping the direction and depth of this thesis.

A special thanks goes to my colleagues and fellow researchers, especially *Soroosh Sabaei*, whose collaboration and intellectual discussions have been a constant source of motivation and inspiration. I extend my heartfelt appreciation to my family and friends for their unwavering support, understanding, and patience during this challenging yet fulfilling endeavor. I am especially grateful to my fiancée, *Ichrak*, for her continuous support, encouragement, and belief in me.

Introduction

The energy sector is crucial in combating climate change, as it accounts for approximately three-quarters of current greenhouse gas emissions (Climate Watch [11]). The transition to low-carbon and renewable energy sources is critical for mitigating global warming and achieving net-zero CO2 emissions. To limit global warming to 2°C or below, almost all electricity by 2050 should come from zero or low-carbon sources such as renewable energies or fossil fuels with carbon capture and storage (IPCC [19]). Given the excessive burning of fossil fuels and the global rise in temperatures (which is a major threat to climate, and in turn has many consequences, notably on public health, economic stability, agriculture, etc.), reducing greenhouse gas emissions is essential to limiting global warming.

Human-caused climate change brought new weather extremes and drought seasons affecting particularly the hydro-energy usage. On one hand, the atypically rising temparatures in the spring accelerated snowmelt which led to a significant reduction in hydropower ressources. On the other hand, various regions encountered drought conditions, with British Columbia, Canada's second-largest hydropower-producing province, experiencing especially severe droughts (International Energy Agency [18]). In fact, during the winter season of 2023/2024 (December through February), Canada's national average temperature was recorded at 5.2°C above the baseline average, which is established based on the 1961-1990 reference period. This temperature rise represents the highest nationwide winter temperature since records began in 1948. Moreover, the 2023/2024 winter's average temperature exceeded the previous record, set in the 2009/2010 winter, by 1.1°C (Environment and Climate Change Canada [16]). This unprecedented warmth underscores significant deviations from historical climate norms, illustrating a marked increase in winter temperatures.

The generation of electricity in Canada in 2023 was dominated by hydropower, which represented 58.0% of the total electricity generation, as shown in Figure 0.1. Nuclear energy was the second largest contributor, accounting for 14.3% of total generation, followed closely by natural gas at 13.6%. Wind energy represented 6.0% of the generation mix, while coal's contribution was minimal at 4.3%. Other sources, such as biofuels, solar energy, and oil, played a smaller role. The energy sources of Canada in 2023 highlight its heavy reliance on renewable energy, particularly hydropower, with notable contributions from nuclear and wind energy.

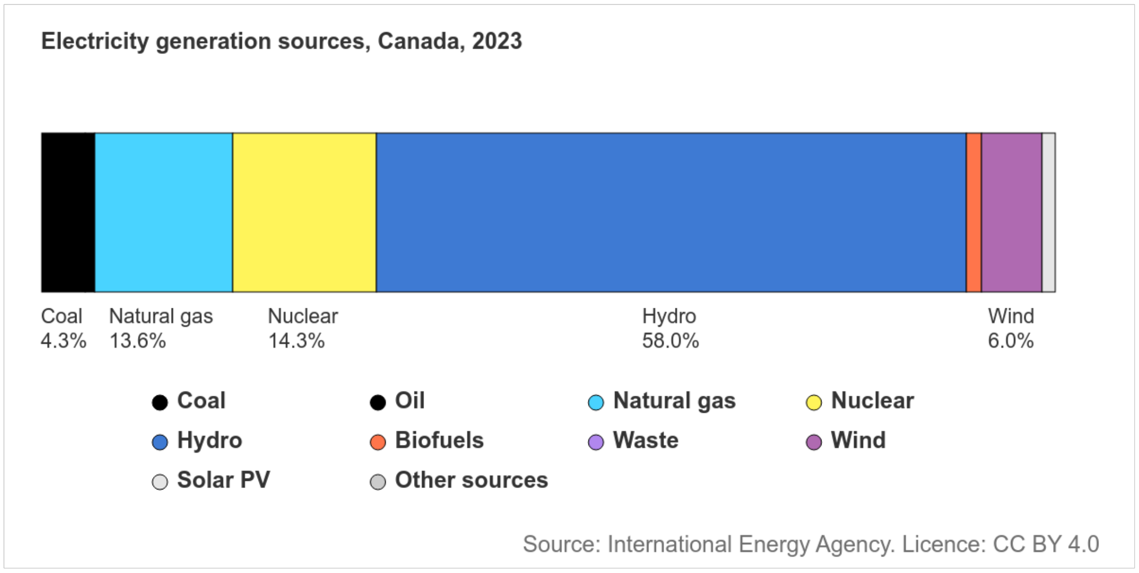


Figure 0.1: Generation Mix for Canada for electric utilities, 2023.

In Quebec, hydroelectric power is even more predominant, accounting for about 71% of the province's total energy production, with wind and biomass contributing roughly 14.5% each (Whitmore et al. [52]). The critical importance of managing the energy dispatch problem in Quebec arises from its heavy reliance on hydroelectric power, where energy dispatch involves determining the optimal allocation of water resources to meet electricity demand while minimizing costs and ensuring reliability.

Poor energy dispatch decisions, such as overestimating water availability or underestimating demand, can lead to wasted resources, increased costs, or even electricity shortages.

To address the dispatching problem in energy planning, we primarily utilize two approaches: Stochastic Dual Dynamic Programming (SDDP) and Reinforcement Learning (RL). SDDP is widely recognized as the state-of-the-art algorithm for solving multistage stochastic programming problems, particularly in the context of hydropower management. It effectively handles the uncertainty inherent in future water inflows by considering multiple scenarios over a planning horizon. On the other hand, Reinforcement Learning is an emerging method that deals with sequential decision-making problems, where the decisions made at each step influence the future state of the environment. RL is particularly useful for problems where the model of the environment is not fully known and needs to be learned through interaction.

This thesis conducts a comparative analysis of Stochastic Dual Dynamic Programming and Reinforcement Learning algorithms for energy planning in Québec, with a particular focus on optimizing hydropower generation. Québec's energy system benefits from its vast network of hydroelectric reservoirs, which enable flexible and controllable energy dispatch. This flexibility is critical in managing the uncertainties associated with water inflows, which are highly variable and influence generation capacity. Unlike renewable sources such as wind, which cannot be stored, hydropower plays a central role in balancing supply and demand by adjusting water levels in the reservoirs which can be seen as storage facilities.

Energy dispatch in Québec involves navigating multiple factors, including local electricity generation, imports and exports of power through interconnections, and the activation of demand response programs. These elements must be carefully coordinated to meet demand reliably while minimizing costs and ensuring system stability. However, this thesis primarily focuses on the reservoirs management for electricity generation because it serves as the foundation for all other dispatch decisions. Efficient generation planning is critical for managing hydroelectric reserves, ensuring cost-effective energy production, and minimizing the reliance on external sources. Furthermore, generation decisions have a cascading impact on the feasibility and efficiency of imports, exports, and demand response strategies. Suboptimal generation planning could result in inefficient resource utilization, higher operational costs, or an inability to meet demand reliably.

By evaluating the performance of SDDP and RL under diverse scenarios, this thesis aims to determine which approach offers more robust and efficient solutions. The findings are expected to contribute to improved strategies for managing Québec's renewable energy system, ensuring both economic and operational efficiency in the face of uncertainties.

This thesis is structured as follows. We start by a literature review, which sets the groundwork for the research conducted in this study. Chapter 1 includes an in-depth look at SDDP, covering its underlying principles and related techniques like stochastic programming and dynamic programming. The chapter also introduces Reinforcement Learning, offering a broader perspective on alternative approaches beyond SDDP. Chapter 2 shifts to the hydropower management problem in Québec, outlining the specific challenges faced in managing hydroelectric generation. It explains the system's constraints, how uncertainties like water inflows are modeled, and the mathematical structure of the problem. The chapter also discusses how RL is structured to address these challenges and the criteria used to ensure both RL and SDDP methods converge to a solution. In Chapter 3, the computational results are presented. This chapter provides a detailed overview of the test cases, including the specific characteristics of Québec's hydro-power grid and how the RL and SDDP methods perform under various scenarios. It highlights key findings from a 12-stage problem and compares the two methods in terms of efficiency. The thesis concludes by summarizing the insights gained and their implications for energy dispatch in Québec, focusing on the contributions of this research to hydro-power management.

Literature Review

Energy dispatch

Economic energy dispatch refers to the process of determining how much electricity each available power generation resource, such as hydropower plants, thermal plants, or renewable energy sources, should produce at any given time to meet electricity demand. The goal is to minimize the total cost of production while respecting operational constraints, such as generator capacity limits, system reliability requirements, and environmental considerations. In the context of hydroelectric power, this involves allocating water resources across various reservoirs and turbines to balance demand and supply efficiently. This problem is pivotal in power system management, especially given the growing integration of renewable energy sources, such as hydro-power, which is a primary focus of this thesis. The increasing complexity of modern power grids has spurred the development of diverse methodologies to address the energy dispatch problem effectively.

One of the earliest and most widely adopted techniques was *Linear Programming* (LP), which provides a framework for solving energy dispatch problems with linear cost functions and constraints. Its computational efficiency has made it well-suited for systems with well-defined parameters. For instance, Mixed Integer Linear Programming (MILP) has been successfully applied in studies such as Pan et al. [33], Tenfen et al. [50], and Pan et al. [32] to optimize energy dispatch.

As the scale and complexity of ED problems grew, *metaheuristic* methods gained traction. For instance, *Particle Swarm Optimization* (PSO) (Kennedy et al. [22]),

which is inspired by the social behavior of swarms, has been applied in Coelho et al. [12], Park et al. [35], and Chen et al. [9], *Whale Optimization Algorithm* (WOA) (Mirjalili et al. [27]), which mimics the social behavior of humpback whales, was used in Azizivahed et al. [2] and Nazari-Heris et al. [31], *Genetic Algorithms* (Chun et al. [10]) was used in Kalakova et al. [21] and Yeh et al. [56]
Another solution is the use of *Dynamic Programming* (Bellman [3]) and its stochastic extension *Stochastic Dynmamic Programming* (SDP) (Bertsekas et al. [6]) which is a widely recognized approach for solving multi-stage decision-making problems under uncertainty. SDP divides the problem into stages and leverages the principle of optimality to optimize decisions at each stage based on current states and stochastic inputs. For instance, SDP was used in Moazeni et al. [30] to solve the multi-stage optimization problem of energy hub optimal dispatch, with an approach that allows a risk-sensitive energy hub operator to consider a non-differentiable risk measure and various constraints, such as minimum uptime and downtime requirements. Similarly, Saadat et al. [44] present a Feasibility Improved Stochastic Dynamic Programming (FISDP) model for optimizing reservoir operation. The authors address the common issue of infeasibility in SDP by adjusting reservoir volume interval indices to transform infeasible policies into feasible ones. Despite its success, SDP faces the "curse of dimensionality" (Bellman [3]), which limits its applicability to high-dimensional, long-horizon problems. To address this challenge, *Stochastic Dual Dynamic Programming* (SDDP) was developed as a scalable alternative.

In the context of Québec's hydro-power-dominated energy dispatch system, optimally managing reservoirs is critical. Québec benefits from the ability to import and export electricity to neighboring regions, and studies such as Mitjana et al. [28] have explored the role of cooperative management among NPCC members to meet carbon emission goals. Meanwhile, Côté et al. [13] evaluated a Sampling Stochastic Dynamic Programming algorithm for managing Hydro-Québec's hydroelectric power plants on the Manicouagan and Outardes Rivers. Similarly, Séguin et al. [46] investigated short-term planning of three power plants in Quebec owned by Rio

Tinto under uncertain water inflows using scenario trees generated from historical and weather data to optimize water usage and turbine operations.

Stochastic Dual Dynamic Programming

Stochastic Dual Dynamic Programming (SDDP) (Pereira et al. [36]) is widely regarded as the preeminent algorithm for power management, particularly in the domain of hydro-power systems. SDDP was specifically developed to address the inherent complexities of hydro-power optimization, including multi-stage decision-making under uncertainty and the challenge of managing reservoirs over extended temporal horizons. By employing piecewise linear approximations of the value function, SDDP effectively mitigates the curse of dimensionality that traditionally hampers Stochastic Dynamic Programming. While SDP is constrained by its computational intractability when the dimensionality of the state space increases, SDDP capitalizes on problem-specific decompositions and iterative solution techniques to achieve tractability in high-dimensional settings. SDDP has been applied to short-term energy dispatch, as demonstrated in Chabar et al. [8], Ding et al. [14], and Papavasiliou et al. [34], and to medium-term operational planning for horizons spanning one to two years, as in Philpott et al. [39] and Rebennack [42]. Notably, its principal strength lies in addressing long-term operational challenges, where planning horizons extend over multiple years, necessitating the simultaneous consideration of immediate and long-term trade-offs in generation and dispatch decisions. For example, SDDP has been utilized in Maceira et al. [26] and Pinto et al. [40] for the Brazilian hydrothermal system, in Rotting et al. [43] for the Norwegian power system, and in Philpott et al. [38] for the New Zealand hydropower system.

The major drawback of SDDP is its exponential growth in computational complexity with respect to both the planning horizon and the state dimension (Füllner et al. [17]). This limitation becomes particularly challenging in large-scale, high-dimensional problems. Many solutions such as the Batch Learning and Experience Replay (Ávila et al. [1]) and Alternative Upper/Lower bound computation tech-

niques (Lan et al. [24]) have been proposed to speed up SDDP computation time, but the computation time remains considerable. Consequently, exploring the potential of AI-based approaches to overcome the computational bottlenecks of SDDP is the primary goal of this thesis.
For a comprehensive review of SDDP and its applications, we refer the reader to Füllner et al. [17].

Reinforcement Learning

Reinforcement Learning (Sutton et al. [49] and Bertsekas et al. [6]) introduces a data-driven approach to energy dispatch, enabling the derivation of optimal policies without the need for explicit system modeling. This methodological shift has broadened the scope of optimization techniques in energy management by offering flexibility and adaptability to complex, dynamic systems. RL can be seen as an extension to the traditional Dynamic Programming by using a new approach to approximate value functions even when we do not have an explicit model of the environemnt.

RL has been applied in various energy dispatch contexts, highlighting its versatility. For instance, Kuznetsova et al. [23] utilized Q-learning to optimize the energy management of a microgrid comprising a consumer, a wind turbine, battery storage, and a grid connection. Their approach focused on improving operational efficiency by leveraging learned policies. Similarly, Yang et al. [55] employed a deep reinforcement learning (DRL) technique to address the dynamic dispatch of a hydro-photovoltaic-pumped hydro storage (PHS) integrated power system. This approach aimed to maximize economic benefits while minimizing fluctuations in power supply and grid connection points, showcasing the potential of DRL in handling integrated renewable energy systems. Furthermore, Xu et al. [54] proposed a multi-agent reinforcement learning framework for optimal reactive power dispatch in power systems. They introduced a consensus-based global information discovery algorithm to compute global reward signals, enabling distributed Q-learning agents to learn and optimize their actions effectively.

The growing interest in RL stems from its ability to scale efficiently with problem complexity and its capacity to discover policies in systems where explicit modeling is challenging. This flexibility positions RL as a promising alternative to traditional methods in energy dispatch, particularly for addressing the limitations of algorithms like SDDP.

Chapter 1

Theoretical framework

1.1 Theory behind SDDP

1.1.1 Sequential decision making under uncertainty

Sequential decision problems involve a continuous cycle of observations and decisions, where each observation informs the subsequent decision. When the sequence of decisions ends after a fixed number of steps, the problem is referred to as a *Finite Horizon* problem, otherwise, it is known as an *Infinite Horizon* problem.

The sequential decision problems are assumed to be temporally-dependent and therefore, the decision maker gathers information through observations over time, using it to make informed decisions, aiming to achieve a determined objective.

When future observations are uncertain or cannot be predicted from the initial time, the problem is referred to as a *Stochastic* problem. Such problems present a significant challenge for researchers, as constructing an accurate model of the *environment* is often difficult or even impossible. The uncertainty is typically modelled as a stochastic process, with its values revealed over time.

To address this, it is essential to formally define the underlying probabilistic framework: we consider a filtered probability space $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω denotes the sample space containing all possible realizations of the stochastic processes under consid-

eration. The filtration $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_T)$ is a family of σ-algebras such that $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \ldots \subset \mathcal{F}_T$, modelling the accumulation of information over time. Specifically, $\mathcal{F}_t$ represents the information available up to and including time t. The measure $\mathbb{P}$ is a probability measure defined on the measurable space $(\Omega, \mathcal{F})$, assigning probabilities to events in a manner consistent with the probability theory.
Intuitively, the stochastic process that models the uncertainty should be $\mathcal{F}$-adapted. Additionally, we assume that the random variables within the stochastic process are independent of one another.

We can, now, write formally the optimization problem as the minimization of the expected value of the future costs:

$$\min_{u \in \mathcal{U}_T} \mathbb{E}\left[\sum_{t=0}^{T-1} L_t(x_t, u_t) + K(x_T)\right] \tag{1.1}$$

subject to the linear transition function of some state variable $x_t \in \mathcal{X}$:

$$x_{t+1} = f_t(x_t, u_t, \xi_t) \tag{1.2}$$

and a number of linear constraints for each $t \leq T$:

$$A_t x_t \leq b_t \tag{1.3}$$

Where $u_t \in \mathcal{U}$ is the control, ξ_t represents the random disturbance, L_t is the stage cost, K is the terminal cost, A_t for all $t \leq T$ is the constraints matrix and b_t is a constant vector whose elements represent the upper bound for the corresponding constraint.

Assumption 1 *The functions L_t and K are assumed to be convex, and f_t is affine in x_t and u_t.*

1.1.2 Nested Benders decomposition

Benders decomposition (Benders [4]) is one of the widely used exact algorithms in stochastic programming. Its core idea is to split large problems into easier subproblems that are much quicker to solve. The *Nested Benders Decomposition* (NBD)

(Birge [7]) method extends the Benders Decomposition approach by applying it recursively to multi-stage stochastic problems.

We shall give here a brief idea about NBD because it will be used in defining the SDDP algorithm. We will give a description of the approach for a two-stage optimizaproblem but the result of NBD is general to any finite number of periods. Consider a two-stage version of the optimization problem (1.1)-(1.3). The NBD algorithm involves the following steps:

1. First Stage Decomposition: The first stage problem involves making decisions u_0 based on the expected future costs. This can be formulated as:

$$\min_{u_0 \in \mathcal{U}_0} \mathbb{E}\left[L_0(x_0, u_0) + \min_{u_1 \in \mathcal{U}_1} \mathbb{E}\left[L_1(x_1, u_1) + K(x_1) \mid x_0, u_0\right]\right] \tag{1.4}$$

   where x_1 is the state variable at the second stage, dependent on x_0 and u_0, and the expectation is over the random disturbances ξ_0 and ξ_1.

2. Second Stage Subproblem: For each fixed u_0, solve the second stage subproblem:

$$\min_{u_1 \in \mathcal{U}} \mathbb{E}\left[L_1(x_1, u_1) + K(x_1) \mid x_0, u_0\right] \tag{1.5}$$

   subject to:

$$x_1 = f_1(x_0, u_0, \xi_0) \tag{1.6}$$

$$A_1 x_1 \leq b_1 \tag{1.7}$$

3. Recursion: The NBD algorithm iteratively refines the first stage decisions u_0 based on the solutions of the second stage subproblems. As new information about ξ_0 and ξ_1 becomes available, the solutions of the second stage problem adjust the first stage problem, updating the cost estimates and constraints.

1.1.3 Stochastic Dynamic Progamming

Stochastic Dynamic Programming (SDP) proposes another approach to solve multi-stage stochastic optimization problems where decisions are made sequentially over time under uncertainty that is revealed over time. Recall that we are using a filtred probability space $(\Omega, \mathcal{F}, \mathbb{P})$ where the evolution of the information available until time t is modelled as $\mathcal{F}_t$. Therefore, the problem can be rewritten dynamically using the *value functions* V_t for all $t \leq T$ defined by

$$V_t(x_t) := \min_{u \in \mathcal{U}_t} L_t(x_t, u_t) + \mathbb{E}\left[\sum_{s=t+1}^{T-1} L_t(x_s, u_s) + K(x_T) \mid \mathcal{F}_t\right] \tag{1.8}$$

subject to the state transition dynamics:

$$x_{t+1} = f_t(x_t, u_t; \mathcal{F}_t) \tag{1.9}$$

That is knowing the observations up to time t, we want to find the best current control u_t that minimizes the current cost plus the expected coming cost.
We can rewrite the equation 1.8 by integrating V_{t+1}:

$$\begin{aligned}
V_t(x_t) &= \min_{u \in \mathcal{U}_t} L_t(x_t, u_t) + \mathbb{E}\left[L_t(f_t(x_t, u_t, \xi_t), u_{t+1}) + \sum_{s=t+2}^{T-1} L_t(x_s, u_s) + K(x_T) \mid \mathcal{F}_t\right] \\
&= \min_{u \in \mathcal{U}_t} L_t(x_t, u_t) + \mathbb{E}\left[L_t(f_t(x_t, u_t, \xi_t), u_{t+1}) + \mathbb{E}\left(\sum_{s=t+2}^{T-1} L_t(x_s, u_s) + K(x_T) \mid \mathcal{F}_{t+1}\right) \mid \mathcal{F}_t\right] \\
&= \min_{u \in \mathcal{U}_t} L_t(x_t, u_t) + \mathbb{E}\left[V_{t+1}(f_t(x, u, \xi_t)) \mid \mathcal{F}_t\right]
\end{aligned} \tag{1.10}$$

Now we define the final condition to complete the definition of the SDP reformulation:

$$V_T(x_T) = K(x_T) \tag{1.11}$$

Putting equations 1.11 and 1.10 together defines the *Bellman Equation*

$$V_t(x) = \begin{cases} K(x) & \text{if } t = T, \\ \\ \min_{u \in \mathcal{U}} \{L_t(x, u) + \mathbb{E}\left[V_{t+1}(f_t(x, u, \xi_t))\right]\} & \text{if } t < T. \end{cases} \tag{1.12}$$

The Bellman equation constitutes the core of the Stochastic Dynamic Programming. The way SDP solves the problem is through state discretization.

State discretization involves dividing the state space into a finite set of discrete states to make the problem computationally manageable. SDP computes the value functions at each point of the grid starting from the final condition. It moves backwards to constitute a discrete representation of the value function.
Under certain compactness and *Lipschitz* continuity assumptions, Bertsekas [5] proved that the discrete schemes converges to the solution of the continuous problem.

Remark 1 *We know from the theory of numerical analysis that the finer the discretization, the better the approximation of the value functions.*

A policy is constructed by moving backward and using the Bellman principle in the equation 1.12. We use interpolation and extrapolation techniques to find the value function at points that do not coincide with points of the grid.

As the number of state variables increases, the number of discrete states grows exponentially, leading to an explosion in the computational and memory requirements. This known as the *curse of dimensionnality*, as stated by *Bellman* himself, and many approaches have been introduced in order to solve this issue (Powell [41], Luus [25]).

1.1.4 SDDP

Stochastic Dual Dynamic Programming addresses the computational infeasibility of classical dynamic programming for large-scale problems by breaking them into smaller subproblems. Each stage of the problem represents a time period, with decisions made sequentially under uncertainty. The dual decomposition technique facilitates the iterative solution of these subproblems, combining them to form a global solution. If the support of the random events is finite, also known as the *Finitely Supported Noise* assumption, SDDP converges almost-surely in a finite number of iterations (Philpott et al. [37]).

SDDP effectively combines the principles SDP and NBD. Specifically, it approximates the value functions using a set of linear functions, or "cuts," generated through

NBD, which are derived iteratively through forward and backward passes.
The algorithm proceeds as follows

Initialization Initialize the value function approximations $V_t^{(0)}(x)$ with a lower bound for each stage t.

Forward Pass The forward pass involves simulating sample paths of the uncertainty and computing the corresponding state and control trajectories. For a given scenario of the uncertainty, denoted as $\left\{\xi_t^{(k)}\right\}_{t=1}^{T}$, the forward problem at each stage t is to determine the control $u_t^{(k)}$ that minimizes the sum of the immediate cost and the approximated future cost-to-go, subject to the system dynamics. Mathematically, this is formulated as:

$$u_t^{(k)} = \arg\min_{u\in\mathcal{U}_t}\left[L_t\left(x_t^{(k)}, u\right) + V_{t+1}^{(k)}\left(f_t\left(x_t^{(k)}, u, \xi_t^{(k)}\right)\right)\right] \tag{1.13}$$

Backward Pass In the backward pass, for each stage t (from $T-1$ to 0), and for each state $x_t^{(k)}$ visited during the forward pass, linear subproblems are solved to update the value function approximations. These subproblems generate optimality and feasibility cuts that refine the value function approximations.
Formally, starting for the stage $T-1$ to the stage 0, for each state $x_t^{(k)}$ and scenario $\xi_t^{(k)}$, we determine the expected cost-to-go $Q_t^{(k)}(x_t)$ given by:

$$Q_t^{(k)}(x_t) = \min_{u\in\mathcal{U}_t}\left\{L_t(x_t, u) + \mathbb{E}\left[V_{t+1}^{(k)}(f_t(x_t, u, \xi_t))\right]\right\} \tag{1.14}$$

Then, we update the value function approximation $V_t^{(k+1)}(x)$ by adding a new cut:

$$V_t^{(k+1)}(x) = \max\left\{V_t^{(k)}(x), Q_t^{(k)}(x) + \lambda_t^{(k)}(x - x_t^{(k)})\right\} \tag{1.15}$$

where $\lambda_t^{(k)}$ represents the dual variables associated with the subproblem, defining the slope of the new cut. This process iteratively improves the approximation of $V_t(x)$.

Remark 2 *(Importance of Cuts) Cuts play a crucial role in SDDP by approximating the value functions. There are two main types of cuts:*

- ***Feasibility Cuts****: These ensure that the solution remains feasible with respect to the constraints of the problem. They prevent the state from violating any constraints by adding necessary bounds.*
- ***Optimality Cuts****: These are derived from the dual solutions of the subproblems and help approximate the value function more accurately. They ensure that the value function is a lower bound to the true value function and improve the solution's optimality.*

Formally, the algorithm can be written as follows:

Algorithm 1 Stochastic Dual Dynamic Programming (SDDP) Algorithm

```
1: **Initialization:** Set $k = 0$. Initialize $V_t^{(0)}(x)$ for all $t$.
2: **repeat**
3:     **Forward Pass:**
4:     **for** each scenario $\xi_t^{(k)}$ **do**
5:         Solve forward problem (1.13) to get $\{x_t^{(k)}, u_t^{(k)}\}$ for all t.
6:     **end for**
7:     **Backward Pass:**
8:     **for** each stage $t$ from $T-1$ to 0 **do**
9:         **for** each state $x_t^{(k)}$ from forward pass **do**
10:            Solve linear subproblem to compute $Q_t^{(k)}(x_t)$.
11:            Update $V_t^{(k+1)}(x)$ with new cut.
12:        **end for**
13:    **end for**
14:    Increment $k$.
15: **until** convergence criterion is met
```

We note that we use the Julia implementation of SDDP in the SDDP.jl package (Dowson et al. [15])

1.2 Reinforcement Learning: beyond SDDP

1.2.1 Introduction to Reinforcement Learning

Reinforcement Learning (RL) is a computational framework designed to model and solve sequential decision-making problems under uncertainty. Unlike supervised learning, where the model learns from labeled input-output pairs, RL focuses on learning an optimal policy through interactions with an environment. This is achieved by observing the consequences of actions and receiving scalar feedback, referred to as rewards.

The core components of RL are defined by the Markov Decision Process (MDP) framework, which comprises a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition probability function $P(s'|s, a)$, and a reward function $R(s, a)$. A critical feature of the MDP framework is the Markov property, which states that the future state s' depends only on the current state s and action a, and is conditionally independent of all previous states and actions. Formally, the Markov property is expressed as:

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0) = P(s_{t+1}|s_t, a_t).$$

This property is particularly important for the theoretical formalism of Reinforcement Learning because it allows academics to use the classical theory of Dynamic Programming (Bellman [3]) (discussion in Chapter 1) and Stochastic processes (Sutton [48], Watkins et al. [51]). Furthermore, Sutton et al. [49] argue that it is in practice useful for some problems to consider that the current state is sufficient to summarize the information needed to take actions.

The agent is an autonomous decision-maker that learns to select actions based on observations of the environment, with the objective of maximizing cumulative rewards over time. The *environment* represents the system or process the agent interacts with, and its dynamics evolve in response to the agent's actions.

At any given time t, the environment is described by a *state* $s_t \in \mathcal{S}$, where $\mathcal{S}$

is the set of all possible states. A state encapsulates all relevant information about the system necessary for decision-making. The agent observes the current state and selects an *action* $a_t \in \mathcal{A}$, where $\mathcal{A}$ is the set of feasible actions available to the agent. The selected action influences the state of the environment, leading to a transition to a new state s_{t+1}, determined by the environment's dynamics, often modeled as a stochastic process governed by the transition probability function $P(s_{t+1}|s_t, a_t)$.

After the transition, the environment provides feedback to the agent in the form of a scalar *reward* $r_t \in \mathbb{R}$, computed using a *reward function* $R(s_t, a_t)$. The reward quantifies the immediate utility of the chosen action in the current state, guiding the agent toward its objective. The interaction between the agent and environment continues over a sequence of time steps, resulting in a trajectory of states, actions, and rewards:

$$\{(s_0, a_0, r_0), (s_1, a_1, r_1), \ldots, (s_T, a_T, r_T)\}.$$

The goal of the agent is to learn a policy $\pi(a|s)$, which maps states to actions, such that the expected cumulative reward, known as the return, is maximized. The return is typically expressed as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k},$$

where $\gamma \in [0, 1]$ is the discount factor that determines the relative importance of future rewards compared to immediate rewards.

A fundamental challenge in Reinforcement Learning is balancing *exploration* and *exploitation*. Exploration involves the agent selecting actions that may not yield immediate rewards but can help discover potentially better policies by gathering new information about the environment. In contrast, exploitation focuses on selecting actions that maximize the expected reward based on the agent's current knowledge. Striking the right balance is crucial: excessive exploration may lead to inefficient learning, while premature exploitation may cause the agent to converge to suboptimal policies.

The classical method for model-based algorithms is Dynamic Programming (DP) (Bertsekas et al. [6]), widely applied when the environment dynamics are fully known. DP relies on *Bellman equations* (Bellman [3]) to iteratively compute the value functions and derive an optimal policy. The Bellman equation for the state-value function $V(s)$ is expressed as:

$$V(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s') \right], \tag{1.16}$$

where $R(s,a)$ represents the immediate reward for taking action a in state s, γ is the discount factor, and $P(s'|s,a)$ denotes the transition probability to state s' from state s given action a.
While DP provides a theoretical foundation for optimal decision-making, its practical applicability is limited due to its reliance on a perfect model and its computational infeasibility in large state spaces (curse of dimensionality).

In contrast, *model-free algorithms* estimate value functions and/or policies using sampled episodes, making them suitable for environments where a model is unavailable or too complex to construct. These methods include *Monte Carlo (MC)* and *Temporal Difference (TD)* learning. MC methods estimate value functions by averaging the returns G_t over multiple episodes, where:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{1.17}$$

TD learning, on the other hand, updates value estimates incrementally using both observed returns and predictions from previously learned values. A common TD update rule for the state-value function $V(s)$ is:

$$V(s) \leftarrow V(s) + \alpha \left[R_{t+1} + \gamma V(s') - V(s) \right], \tag{1.18}$$

where α is the learning rate, and $R_{t+1} + \gamma V(s')$ is the *TD target*.

Within this framework, model-free algorithms can be categorized into *value-based methods* and *policy-based methods*. *Value-based methods*, such as Deep Q-Learning

(Mnih et al. [29]) and SARSA (Sutton et al. [49]), combine DP principles with TD learning. These methods aim to approximate the optimal action-value function $Q(s,a)$, which satisfies the Bellman optimality equation:

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a'). \tag{1.19}$$

By using TD updates, value-based methods can efficiently learn directly from raw experience without needing a model. For instance, SARSA updates $Q(s,a)$ using:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R_{t+1} + \gamma Q(s',a') - Q(s,a)\right], \tag{1.20}$$

where a' is the action taken in the next state s'. While these methods are data-efficient, they can exhibit higher variance and bias, potentially affecting learning stability.

1.2.2 Policy Gradient methods

Policy gradient methods are a class of algorithms in RL that optimizes policies directly by adjusting the parameters of a policy function to maximize cumulative rewards. Unlike value-based methods, which estimate value functions to guide decision-making, policy gradient methods parameterize the policy itself and adjust its parameters to improve performance.

The Policy Gradient Theorem provides the foundation for policy gradient methods. It states that the gradient of the expected return $J(\theta)$ with respect to the policy parameters θ can be expressed as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) Q^\pi(s_t,a_t)\right] \tag{1.21}$$

Here, $Q^\pi(s_t,a_t)$ is the action-value function, representing the expected return starting from state s_t, taking action a_t, and thereafter following the policy π. This theorem is fundamental because it allows the direct computation of the policy gradient without needing to differentiate the state distribution under the policy.

REINFORCE (Williams [53]), the simplest policy gradient method, updates the policy parameters in the direction that increases expected returns. While straightforward, REINFORCE can suffer from high variance in gradient estimates. A solution to the high variance was the introduction of a baseline. Natural Policy Gradient (NPG) (Kakade [20]) methods improve upon REINFORCE by incorporating second-order information, leading to more stable and efficient updates. Deep Deterministic Policy Gradient (DDPG) (Silver et al. [47]) combines the strengths of policy gradients and deterministic policies, enabling the learning of continuous actions in high-dimensional spaces. Trust Region Policy Optimization (TRPO) (Schulman et al. [45]) further enhances stability by ensuring policy updates stay within a trust region, preventing drastic policy changes that can destabilize learning.

1.2.3 The REINFORCE Algorithm

REINFORCE is one of the most fundamental policy gradient methods. The core idea behind REINFORCE is to optimize the expected return of a policy by following the gradient of the expected return with respect to the policy parameters.

Given a policy $\pi_\theta(a|s)$ parameterized by θ, the objective is to maximize the expected return $J(\theta)$:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right] \tag{1.22}$$

where $\gamma \in [0, 1)$ is the discount factor, r_t is the reward at time step t, and T is the episode length.

The REINFORCE algorithm estimates the gradient of the objective function using the following expression:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t\right] \tag{1.23}$$

Here, G_t represents the return, or the sum of discounted rewards, from time step t onwards:

$$G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k \tag{1.24}$$

Theory Behind REINFORCE

The theory behind REINFORCE is grounded in the likelihood ratio method, which allows the gradient of the expected return to be expressed in terms of the policy's log-probability. This is achieved by recognizing that the expectation of the return can be differentiated using the log-likelihood trick:

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\pi_\theta} \left[G_t \right] = \mathbb{E}_{\pi_\theta} \left[G_t \nabla_\theta \log \pi_\theta(a_t|s_t) \right] \tag{1.25}$$

This gradient is then used to perform stochastic gradient ascent on the policy parameters:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \tag{1.26}$$

where α is the learning rate.

The key insight of REINFORCE is that by increasing the probability of actions that yield high returns, the policy is improved over time. Conversely, the probability of actions leading to lower returns is decreased. This process, although simple, is effective in a wide range of reinforcement learning problems.

Variance Reduction and Baselines

One of the challenges with REINFORCE is the high variance in gradient estimates, which can lead to unstable learning. To mitigate this, a baseline function $b(s)$ can be subtracted from the return without introducing bias, resulting in a variance reduction. The gradient update then becomes:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(G_t - b(s_t) \right) \right] \tag{1.27}$$

A common choice for the baseline is the value function $V^{\pi}(s_t)$, which represents the expected return from state s_t under the current policy π.

Chapter 2

The hydro-power management problem

2.1 Electricity generation in Québec

We present here the optimization problem we studied. We consider minimizing the cost of electricity generation while satisfying the demand at each time step. We present the different constraints of the model as well as the mathematical formulation of the problem. Our model is a simplification of the model presented in [28] where they studied a multi-stage stochastic optimization problem to determine cost-effective, net-zero emission expansion plans for the power sector by 2050 for five Northeastern North American regions (Québec, Ontario, New York, New England and the Atlantic provinces of Canada) using more than one source of electricity (Hydropower, Nuclear, Solar, ..etc). In our case, we only consider electricity generation problem in one region (Québec) using one source which is hydropower.

2.1.1 Constraints

The optimization model for energy dispatch in the hydrogird system is subject to several physical constraints that ensure feasibility and adherence to system reliability. These constraints govern the production of energy, water management, and resource utilization, and are critical for obtaining a feasible solution. We present at first the sets, variables and parameters

We consider three types of hydroplants in our analysis: large reservoir hydroplants, intra-day reservoir hydroplants, and run-of-the-river hydroplants. Large reservoir hydroplants are equipped with significant storage capacities, enabling long-term water management and providing flexibility to adapt to seasonal demand and inflow variations. In contrast, intra-day reservoir hydroplants have smaller reservoirs, allowing for limited water storage within a single day to address short-term demand fluctuations. Run-of-the-river hydroplants, on the other hand, have negligible or no storage capacity and depend directly on the immediate river flow for energy generation.

Remark 3 *For simplicity, we treat run-of-the-river and intra-day reservoir hydroplants equivalently in our analysis, assuming that water storage is only feasible in large reservoirs. We also treat ROR hydroplants as intra-day reservoirs with no storage capacity.*

Releases of water from reservoirs occur through two distinct processes: turbining or non-turbining releases. In the first process, water is passed through turbines, generating electricity before being released downstream. In the second process, water is released without being turbined. This is done either to manage reservoir levels, adhere to operational constraints, or support downstream needs. However, this non-turbined water remains within the hydro system and can potentially be utilized by downstream reservoirs for power generation.

A key feature of the system is the time lag in water movement between reservoirs. Specifically, water released from an upstream reservoir at stage t does not instantly reach the next reservoir; instead, it arrives in the downstream reservoir during period $t + 1$.

Sets

LR	Large reservoirs hydropower plants
$IDRoR$	Intra-day reservoirs hydropower plants and Run-of-the-river hydropower plants
$ResRiv$	Set of hydroplants directly associated with rivers
$Prev(i)$	Reservoirs upstream of reservoir i, $\forall i \in LR \cup IDRoR$
$r(i)$	The rivers associated with the hydroplant i, $\forall i \in ResRiv$

Exogenous Variables

$demand_t$	Energy demand at time t [MW]
$inflow_{i,t}$	Water inflow to reservoir i, $\forall i \in LR \cup IDRoR$ at time t [m^3]

Decision Variables

$hq_{i,t}$	Energy produced by hydroplant i, $\forall i \in LR \cup IDRoR$ at time t [MWh/hour]
$ww_{i,t}$	Water released but not turbined from reservoir i, $\forall i \in LR$ at time t [m^3/s]
$wd_{i,t}$	Turbined water in m^3 from reservoir i, $\forall i \in LR \cup IDRoR$ at time t [m^3/s]
$wav_{i,t}$	Available water in m^3 in reservoir i, $\forall i \in LR \cup IDRoR$ at time t [m^3/s]
ll_t	Lost load at time t [MWh]
$ws_{i,t}$	Water level in large reservoir i, $\forall i \in LR$ at time t [hm^3]

Parameters

c_{ll}	Cost of unsatisfied energy demand (lost load) [\$/MWh]
c_i	Unit production cost for hydroplant i, $\forall i \in LR \cup IDRoR$ [\$/MWh]
$Prod_i$	Production factor for hydroplant i, $\forall i \in LR \cup IDRoR$ [MWh per m^3/s]
$\underline{WS}_i, \overline{WS}_i$	Lower and upper bounds for water levels in large reservoir i, $\forall i \in LR$ [hm^3]
$ws_{i,0}$	Initial water level in large reservoir i, $\forall i \in LR$ [hm^3]
$wav_{i,0}$	Initial water for reservoir i, $\forall i \in LR \cup IDRoR$ [m^3/s]
$\overline{WW}_i$	Upper limit for non-turbined water release from large hydroplant i, $\forall i \in LR$ [m^3/s
CF	Conversion coefficient for flows from m^3/h to hm^3
Cap_i	The turbine capacity for the hydroplant i, $\forall i \in LR \cup IDRoR$ [MW]
k	Penalty coefficient [\$/MWh]
N	Total number of hydroplants
M	Number of rivers
T	Number of stages

Remark 4 *The ratio* $\dfrac{Cap_i}{Prod_i}$ *is reffered to as the maximum turbining capacity (or discharge) of the hydroplant i* $\quad \forall i \in LR \cup IDRoR$

Below, we present and explain the key constraints that govern the behavior of the system:

Energy production We approximate the production function that converts the water turbined to energy. The production function is approximated as the product of the water turbined and a production factor specific for each hydroplant:

$$hq_{i,t} = wd_{i,t} \times Prod_i \quad \forall\ i\ \in LR \cup IDRoR,\ \forall t \leq T \tag{2.1}$$

IDR and ROR Hyroplants not directly connected to rivers production For intra-day reservoirs hydroplants and run-of-the-river (ROR) reservoirs hydroplants that are not connected directly to rivers, the amount of water turbined must not exceed the available water coming from the upstream hydroplants at the previous stage:

$$wd_{i,t} \leq wav_{i,t} \quad \forall i \in IDRoR - ResRiv, \forall t \leq T \tag{2.2}$$

IDR and ROR Hyroplants directly connected to rivers production Water turbined from IDR and ROR reservoir hydroplants that are directly linked to rivers must not exceed the sum of the inflows coming for the upstream reservoirs at the previous stage and the water inflows coming from the river associated to the hydroplant:

$$wd_{i,t} \leq wav_{i,t} + inflow_{r(i),t} \quad \forall i \in IDRoR \cap ResRiv, \forall t \leq T \tag{2.3}$$

Water transition between reservoirs Water dynamics between upstream and downstream reservoirs are modeled by this constraint, which ensures that the available water at any reservoir depends on the water turbined and released by previous reservoirs:

$$\begin{aligned} wav_{i,t+1} = & \sum_{j \in Prev(i) \cap LR} wd_{j,t} + ww_{j,t} \\ & + \sum_{j \in Prev(i) \cap (ROR - ResRiv)} wav_{j,t} \\ & + \sum_{j \in Prev(i) \cap (ROR \cap ResRiv)} wav_{j,t} + inflow_{j,t} \quad \forall\ i\ \in LR \cup IDRoR,\ \forall t \leq T \end{aligned} \tag{2.4}$$

In particular, the water inflows might come from large reservoirs which are characterized by their storing capacity, and from run-of-river reservoirs which pass all the water available to the downstream reservoirs.

Upper limit for non-turbined water Each reservoir has a maximum allowable quantity of water that can be released without generating electricity (non-turbined water). This constraint is expressed as:

$$ww_{i,t} \leq \overline{WW}_i \quad \forall i \in LR,\ \forall t \leq T \tag{2.5}$$

Remark 5 *We need an upper bound for water released but not turbined because we use it as a component in defining the states (see section 2.3) and we need all the inputs of the neural networks to be between 0 and 1 (see section 2.3).*

Water level dynamics in large reservoirs The dynamics governing the water levels in large reservoir hydroplants are represented by the following constraint, which ensures that water levels are updated based on inflows, outflows, and the capacity:

$$ws_{i,t+1} \leq ws_{i,t} + CF\left(wav_{i,t} + inflow_{r(i),t} - wd_{i,t} - ww_{i,t}\right) \quad \forall i \in LR, \forall t \leq T \tag{2.6}$$

The gap between the RHS and LHS can be interpreted as water that exceeds the reservoir's capacity and is thus "out of the system". To prevent overflow and ensure storage capacity is met, excess water that surpasses the storage capacity is treated as being redirected out of the system (release capacity for water to stay in the system is limited).

Remark 6 *If the option to redirect water out of the system were not considered, scenarios could arise where water accumulates in a reservoir such that, at a certain stage, it becomes impossible to release sufficient water to satisfy the reservoir's storage capacity constraints. Typically, this issue is resolved by adding feasibility cuts to the optimization process. However, since SDDP.jl does not provide this functionality, we addressed the problem by allowing excess water to be redirected out of the system as needed. This approach ensures compliance with the relatively complete recourse assumption while enabling the algorithms to learn strategies that minimize water losses from the system.*

Energy demand balance The total energy generated by all reservoirs at each time step must meet the energy demand to ensure system reliability. If the energy production falls short of the demand, the unmet portion is referred to as the lost load. This represents the amount of energy demand that cannot be satisfied due to generation constraints, leading to potential interruptions in power supply.

$$ll_t + \sum_{i=1}^{N} hq_{i,t} = demand_t \quad \forall t \leq T \tag{2.7}$$

For SDDP, both ll_t and $hq_{.,t}$ are decision variables. However, for RL, we only decide the energy we generate from each hydroplant (see section 2.3 for the definition of the actions of the RL agent). Overproduction occurs when $\sum_{i=1}^{N} hq_{i,t} \geq demand_t$. In such cases, the constraint (2.7) cannot hold, as it would imply $ll_t \leq 0$, violating the non-negativity condition (2.11) on ll_t. To prevent this issue, we should manually set ll_t to be 0 when there is an overproduction.

$$ll_t \leftarrow \max\{demand_t - \sum_{i=1}^{N} hq_{i,t}, 0\} \quad \forall \leq T \tag{2.8}$$

Generation upper bound of the reservoirs The total energy produced by each reservoir must be limited by its maximum generating capacity:

$$hq_{i,t} \leq Cap_i \quad \forall\, i \;\in LR \cup IDRoR,\ \forall t \leq T \tag{2.9}$$

Water level bounds in large reservoirs The water level in each large reservoir is subject to upper and lower bound physical constraints, which must be respected at every time step:

$$\underline{WS}_i \leq ws_{i,t} \leq \overline{WS}_i \quad \forall i \in LR,\ \forall t \leq T \tag{2.10}$$

Non-negativity constraints All decision variables, including the amount of water turbined, released, and available in each reservoir, must be non-negative. This constraint ensures that there is no negative water flow or energy production:

$$hq \geq 0,\ wd \geq 0,\ wav \geq 0,\ ww \geq 0,\ ll \geq 0 \tag{2.11}$$

2.1.2 Mathematical formulation of the energy transition problem

The optimization problem under study addresses the management of a multi-reservoir hydropower system over a multi-period finite horizon, taking into account uncertain water inflows. This multi-stage stochastic optimization framework aims to balance energy production, demand satisfaction, and long-term reservoir sustainability. The objective is to minimize the total cost over the planning horizon. This includes two components:

1. The cost of unsatisfied energy demand (or lost load) at each period t, weighted by the penalty c_{ll}, to ensure demand is met where possible.

2. The cost of energy production by each reservoir i, weighted by the unit production cost c_i..

The cost of lost load is, intuitively, higher than the cost of electricity production to prioritize demand satisfaction and minimize lost load.

At the terminal period T, the function incentivizes a desirable final state:

- A reward/penalty is assigned based on the final water levels $ws_{.,T}$ compared to initial levels $ws_{.,0}$, weighted by the reservoir-specific production factors $Prod$.

- This ensures reservoirs are left in a state beneficial for future operations where the productivity of the water available at the large reservoirs is close to the productivity of the initial state.

The optimization framework is based on Dynamic Programming, where the decision-making process is decomposed over discrete time periods t. The objective function uses the *Bellman Equation*, which represents the minimum expected cost from period t onward, reflecting the temporal nature of reservoir operations.

Formally, the problem can be represented as follows:

$$V_t = \begin{cases} \min \left\{ c_{ll} \times ll_t + \sum_{i=1}^{N} c_i hq_{i,t} + V_{t+1}; \ \text{ s.t. constraints (2.1)-(2.11)} \right\}, \forall t < T \\ \\ k \times \langle ws_{.,T} - ws_{.,0}, Prod \rangle & t = T \end{cases} \tag{2.12}$$

where:

- V_t is the value function at time t, representing the minimum cost from time t onward.
- V_T is the terminal value function, rewarding favorable reservoir states.
- k is a scaling factor to adjust the importance of the terminal reward as defined previously.
- $\langle ., . \rangle$ denotes the scalar product.

The reward at T prevents over-depletion of reservoirs, encouraging strategic reserve of water for future energy production.

2.2 Uncertainty modelling

In modelling the uncertainty of water inflows, we employ a *Multivariate Gaussian* distribution. For simplicity, we assume that there is no correlation between the inflows of different rivers, an assumption that allows each river's inflow to be treated as an independent variable.

We analyzed a dataset (obtained from an internal ressource) of historical daily average water inflows, measured in m^3/s for each river in Québec's hydro-grid. The dataset represents time series of daily inflows for each river over multiple years.

To model the variability of inflows of each river, we divided each river's time series into groups based on the calendar day. Specifically, we aggregated all historical inflow observations of each river for each day of the year (e.g., all January 1st inflows, all January 2nd inflows, etc.), resulting in 366 batches of data to account for leap years. Next, we fitted a multivariate normal distribution to the inflow data for each calendar day for each river which provided a probabilistic estimate of the distribution of daily average inflows for each river for each specific day of the year. The result of this process is a stochastic process $\boldsymbol{inflow}$, which can be described as

$$\boldsymbol{inflow} = (inflow_{.,1}, inflow_{.,2}, \ldots, inflow_{.,365}), \tag{2.13}$$

where, for each $0 \leq t \leq 366$, the random variable $inflow_{.,t}$ is represented by the vector

$$\boldsymbol{inflow}_{.,t} = \begin{pmatrix} inflow_{1,t} \\ inflow_{2,t} \\ \vdots \\ inflow_{M,t} \end{pmatrix} \sim \mathcal{N}(\mu_t, \Sigma_t) \tag{2.14}$$

with M denoting the number of rivers under consideration, μ_t is the mean vector and Σ_t denotes the covariance matrix. Each component of this vector represents the inflow for a particular river at time t, with the overall process capturing the joint distribution of inflows across all rivers.

Remark 7 *It is important to note that this thesis does not consider inter-temporal dependencies in the inflow dynamics. While such relationships could potentially enhance the accuracy of the inflow models, they are beyond the scope of this study. The primary focus here is on the optimization aspects of the problem, for which the simplified assumption of independence across time and rivers is sufficient.*

2.3 Reinforcement Learning Structure

This section details the structure of the Reinforcement Learning framework used in this study. The RL framework is designed to solve the energy dispatch problem by modeling it as a sequential decision-making process. The primary components of the RL framework are the environment, states, actions, and reward function, which are described below.

States The state space includes all variables that describe the current status of the hydroelectric system at a given time step. These variables are critical for making informed decisions. The state s_t at time t includes:

- Stage t represented by $cos(2\pi \frac{t}{T})$ and $sin(2\pi \frac{t}{T})$
- Water levels in each reservoir in stage t $ws_{.,t}$ that are normalized by min-max normalization
- Water inflows to the reservoirs in stage t: $inflow_{.,t}$, that are normalized by min-max normalization.
- Available water at each reservoir in stage t: $wav_{.,t}$ that is normalized by dividing it by the maximum water that we can have at the reservoir i. Specefically:

$$\frac{wav_{i,t}}{\sum_{j \in Prev(i)} Cap_j/Prod_j + \sum_{j \in Prev(i) \cap LR} \overline{WW}_j} \quad \forall t \leq T \; \forall i \in LR \cup IDRoR$$

  This normalization reflects the sum of the maximum water that can be turbined ($Cap_j/Prod_j$) and the maximum water that can be released without being turbined ($\overline{WW}_j$) for each uptream reservoir of the reservoir i.

Actions The action space defines the set of possible decisions the agent can make at each time step. At time t, the action $a_t = (wd_{.,t}\ , ww_{.,t})$ is a vector consisting of the following components:

- $wd_{i,t}$, the amount of water turbinated by reservoir i at time t, and
- $ww_{i,t}$, the amount of water released without generating electricity for each large reservoir hydroplant $i \in LR$ at time t.

Each component of the action vector corresponds to an independent decision variable. To enable the agent to learn the optimal decisions while maintaining exploration, we represent each component of the action vector as a random variable. Specifically, the mean of the random variable encodes the optimal action, and its variance controls the level of exploration.
The action space is continuous because the amounts of water release can take any value within the system's operational constraints.
We model the policy $\pi(a_t|s_t)$ as a multivariate normal distribution parameterized by a mean vector $\mu(s_t;\theta)$ and a diagonal covariance matrix $\sigma^2(s_t;\theta)$. These parameters depend on the current state s_t and the policy's learned parameters θ. Formally, the action at each time step is sampled as:

$$a_t \sim \mathcal{N}(\mu(s_t;\theta), \mathrm{diag}(\sigma^2(s_t;\theta))) \tag{2.15}$$

where $\mu(s_t;\theta)$ represents the most likely action (or the "greedy" decision), and $\sigma(s_t;\theta)$ determines the exploration scale.

As training progresses, the mean vector $\mu(s_t;\theta)$ converges to the optimal action for each state, while the covariance matrix $\sigma^2(s_t;\theta)$ gradually shrinks. This reduction in variance decreases the randomness in the agent's decisions over time, ensuring that the policy becomes increasingly deterministic. This dynamic trade-off between exploration and exploitation is essential for learning an effective policy while avoiding suboptimal local solutions.

Taking actions using (2.15) cannot ensure that constraints (2.2)-(2.11) can be satisfied. Therefore an alternative method is used. At first, we consider the unconditional action $\tilde{a}_t$ by sampling from

$$\tilde{a}_t \sim \mathcal{N}(\mu(s_t;\theta), \mathrm{diag}(\sigma^2(s_t;\theta))) \tag{2.16}$$

Then, we transform $\tilde{a}_t$ to have a value between 0 and 1 by applying a sigmoid function and obtain

$$(\varepsilon^{wd}_{\cdot,t}, \varepsilon^{ww}_{\cdot,t}) = Sigmoid(\tilde{a}_t) \tag{2.17}$$

Next, we determine the upper bound for the discharge for each hydroplant denoted $\overline{TURB}_{i,t}$ for all reservoirs $i \in LR \cup IDRoR$.

For larger reservoir hydroplants, the upper bound is the difference between the water that can be discharged and the lower level of the reservoir that cannot be surpassed. The water that can be discharged is the sum of the water already in the reservoir, the water inflows coming from the river directly associated with the reservoir, and the water coming from the upstream reservoir. To ensure that water level in the large reservoirs does not go below the lower level we subtract the minimum water that must stay in the reservoir which is given by $\frac{\underline{WS}_i}{CF}$[1]. This yields to the following formula for the upper bound for the discharge for large reservoirs:

$$\overline{TURB}_{i,t} = \frac{ws_{i,t}}{CF} + inflow_{r(i),t} + wav_{i,t} - \frac{\underline{WS}_i}{CF} \quad \forall i \in LR \tag{2.18}$$

For IDR and ROR reservoir hydroplants that are not directly connected to a river, the upper bound is given by the water coming from the upstream reservoirs:

$$\overline{TURB}_{i,t} = wavi,t \quad \forall i \in IDRoR - ResRiv \tag{2.19}$$

For IDR and ROR reservoir hydroplants that are directly connected to a river, the upper bound is given by the water coming from the upstream reservoirs and the water inflows:

$$\overline{TURB}_{i,t} = wavi,t + inflow_{r(i),t} \quad \forall i \in ResRiv \tag{2.20}$$

We can compute the water discharged then as follows:

$$wd_{i,t} = \varepsilon^{wd}_{i,t} \ \overline{TURB}_{i,t} \quad \forall i \in LR \cup IDRoR \tag{2.21}$$

This quantity must also satisfy the constraints (2.9) and (2.1) therefore the discharge must not exceed the turbine capacity. We must modify the previous equation as follows:

$$wd_{i,t} = \min \left\{ \frac{\varepsilon^{wd}_{i,t} \ \overline{TURB}_{i,t}}{Prod_i}, \ Cap_i \right\} \quad \forall i \in LR \cup IDRoR \tag{2.22}$$

[1]We need to convert the lower bound for water levels to m^3

For the amount of water released without generating electricity ($ww_{i,t}$), we need to determine the upper bound of the release $\overline{WW}_i$. To determine the maximum feasible release for the each large reservoir hydroplant, we compute the available water that can be released as the difference between the water that can be discharged and the water that has been turbined

$$\overline{WW}_{i,t} = \frac{ws_{i,t}}{CF} + inflow_{r(i),t} + wav_{i,t} - \frac{WS_i}{CF} - wd_{i,t} \qquad \forall i \in LR \tag{2.23}$$

Then, we determine the water released from each large reservoir as

$$ww_{i,t} = \varepsilon^{ww}_{i,t} \ \overline{WW}_{i,t} \qquad \forall i \in LR \tag{2.24}$$

We sum up how we sample feasible actions in the following algorithm

Algorithm 2 Taking actions

```
1: Sample $\tilde{a}_t$ from $\mathcal{N}(\mu(s_t;\theta), \mathrm{diag}(\sigma^2(s_t;\theta)))$
2: Determine $(\varepsilon^{wd}_{i,t},\ \varepsilon^{ww}_{i,t})$ using (2.17)
3: Determine the upper bounds for the water that can be turbined $\overline{TURB}_{.,t}$ using
   (2.18)-(2.21)
4: Determine water turbined $wd_{.,t}$ using (2.22)
5: Determine the upper bound for water released (non turbined) using $\overline{WW}_{.,t}$ (2.23)
6: Determine the water released (non turbined) $ww_{.,t}$ using (2.24)
```

Reward Function The reward function quantifies the immediate benefit or cost of an action taken by the agent at time t. It is designed to guide the agent toward optimal decision-making by balancing energy production, operational costs, and system reliability. The reward R_t is defined as:

$$R_t = \begin{cases} -\left(c_{ll} \cdot ll_t + \sum_{i=1}^{N} c_i \cdot hq_{i,t}\right), \forall t < T \\ \\ k \times \langle ws_{.,T} - ws_{.,0}, Prod\rangle \qquad t = T \end{cases} \tag{2.25}$$

The negative sign ensures the maximization of the reward aligns with minimizing costs. The reward function encourages the agent to satisfy energy demand while minimizing production costs and penalties for lost load.

2.4 Convergence criterion

2.4.1 Reinforcement Learning convergence

For the RL training phase, we use a fixed number of gradient steps and evaluate the Out-of-Sample (OOS) performance every 100 episodes. This approach is motivated by the need to balance computational efficiency with effective performance monitoring. Evaluating at regular intervals allows us to track the agent's progression without incurring excessive computational overhead. Furthermore, since the policy is unlikely to change significantly within a small number of gradient steps, selecting an interval strikes a tradeoff between capturing meaningful updates and maintaining efficiency. A fixed number of gradient steps also facilitates consistent comparisons across experimental setups while ensuring training completes within a reasonable timeframe. This methodology offers a practical balance between thorough evaluation and computational feasibility.

2.4.2 SDDP

As in [15], we avoid using the conventional convergence criteria based on the gap between the *deterministic bound* and the *simulation bound.* We combine two criterion and stop when the first of them is met:

1. We give SDDP.jl a maximum time to generate cuts and converge

2. We use the default SDDP.jl stopping rule based on the stabilization of the deterministic bound and the oos simulations convergence within a specified tolerance.

For more explanation on why we do not use the conventional criteria, we refer the reader to SDDP.jl documentation (`https://sddp.dev/stable/apireference/#SDDP.OutOfSampleMonteCarlo`)

To speed up computation, we use parallel computing on 40 cores, leveraging the inherently parallelizable nature of SDDP.

Chapter 3

Computational results

3.1 Test case presentation

3.1.1 Québec's hydro-grid considered in the study

We consider a fraction of the hydro-grid in Québec given in the figure (3.1). Our systme is composed of multiple reservoirs and multiple hydroplants. We have 3 types of hydroplants: Large Reservoir, Intra-Day and Run-of-the-River.

The hydroplants P1, P5 and P9-P10 are associated with large reservoirs. The hydroplants P3 and P4 are associated with intra-day reservoirs. The rest of the hydroplants (P2 and P11) are run-of-the-river. The run-of-the-river hydroplant P11 is not directly connected to a river but rather receives water coming from the P9-P10. The run-of-the-river P2 is directly connected to the river R2. The hydroplant P4 is associated with an intra-day reservoir which directly connected to the river R3. However, the plant P3 isassociated with an intra-day reservoir which is not directly connected to a river but rather we have inflows coming to P2 then to P3.

Therefore, the sets defined in chapter 1 can be determined as follows:

- $LR = \{P1, P5, P9 - P10\}$
- $IDRoR = \{P2, P3, P4, P11\}$

- $ResRiv = \{P2, P4\}$

Remark 8 *The names of each reservoir and river is in the Appendix A*

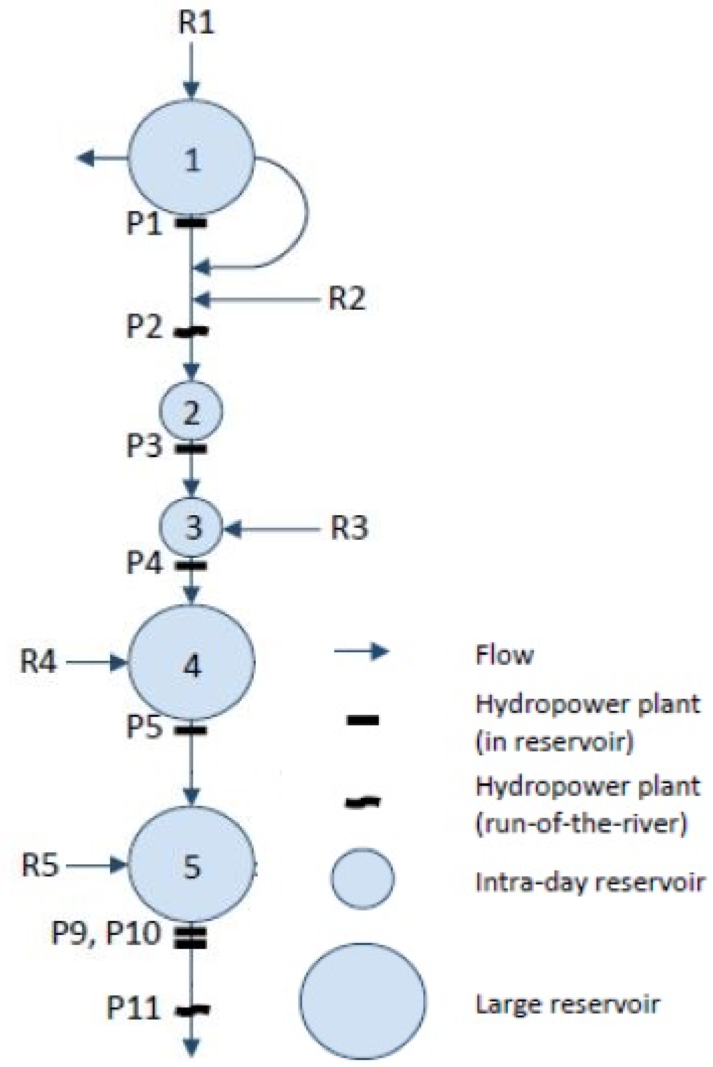


Figure 3.1: Québec's hydro-grid considered

3.1.2 RL structure

We consider the REINFORCE algorithm with a baseline, where the baseline is implemented as a neural network. In this setup, the neural network approximates the value function, providing an estimate of the expected return from a given state. The neural network is trained alongside the policy, adjusting its parameters to improve the accuracy of the baseline and, consequently, the performance of the policy gradient updates.

Policy Network The policy network takes the state of the environment as input and outputs the parameters of a Gaussian distribution, specifically the mean and standard deviation, which are then used to sample actions. The network consists of three layers: the input layer, the hidden layer, and the output layer. The input to the network is the state of the environment, represented by a vector of the same dimensionality as the state vector. The hidden layer has 48 units with a *tanh* activation function. The output layer is divided into two parts: the first half of the output corresponds to the logarithm of the standard deviation of the Gaussian distribution, which is then transformed using an exponential function to obtain the actual standard deviation. The second half of the output represents the mean of the Gaussian distribution.

Value Network The value network is designed to estimate the expected return, or value, of a given state, and serves as a baseline to reduce the variance of the policy gradient estimates. This network also takes the state representation as input, similar to the policy network. The hidden layer consists of 32 units with a *ReLU* activation function. The output layer provides a scalar value that estimates the expected return for the given state.

Remark 9 *We studied multiple architectures of the neural networks considered. In particle, we tried architectures with more layers and neurons but it we found that the current architecture ensures good performance and computational effiency.*

Learning rate We try multiple learning rates for both the value network and the policy network and choose the one ensuring the best performance and the fastest convergence. Figure (3.2) shows the convergence of the REINFORCE algorithm under different learning rates values for both the value network and the policy network. Notably, a high policy network learning rate results in a loss of convergence and exhibits chaotic behavior. Conversely, while reducing the policy network learning rate enhances stability, it also slows convergence when the value network learning rate is held constant. The impact of the value network learning rate on convergence is comparatively less pronounced, primarily influencing the rate of convergence rather than stability.

Through this analysis, we found that the best learning rate for the policy network is $5e-4$ and the best learning rate for the value network is 0.5.

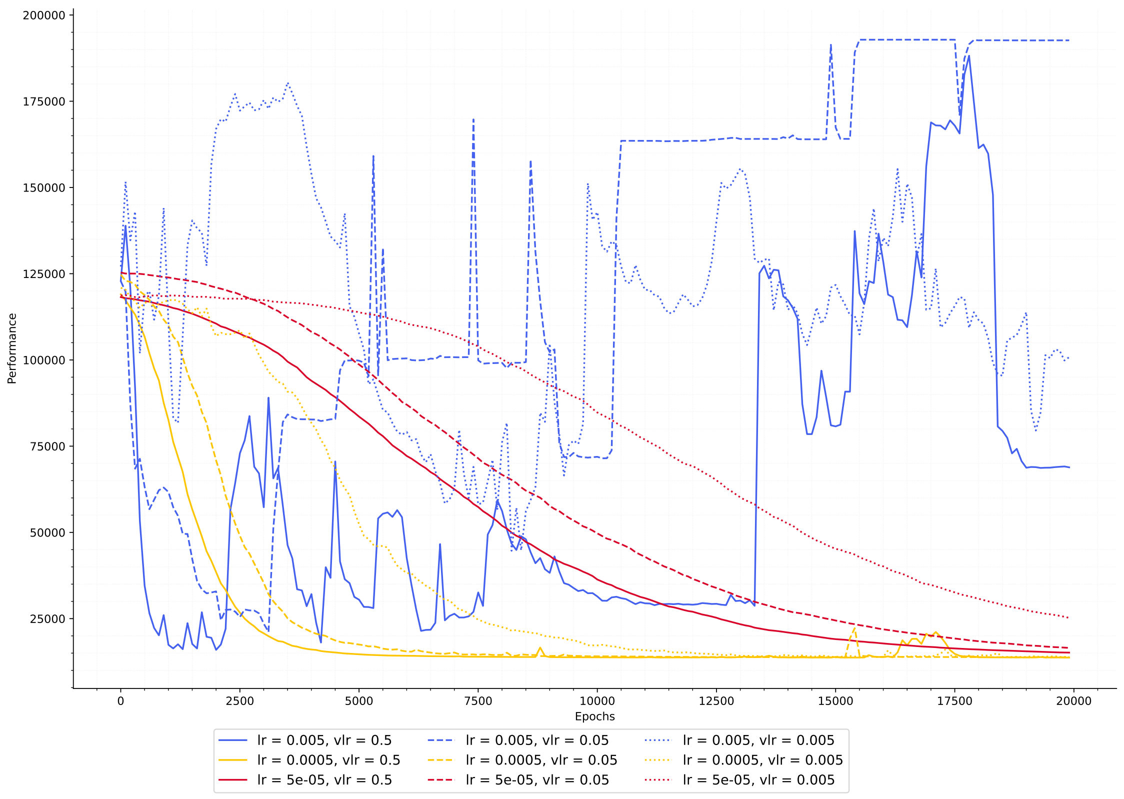


Figure 3.2: Convergence of the REINFORCE algorithm under different combinations of value learning rates (vlr) and policy learning rate (lr)

3.2 Results

In this section, we evaluate and compare the performance of SDDP and RL applied to the energy dispatch problem described in the previous chapter. Both methods are assessed using the parameter configurations outlined earlier. The evaluation focuses on two main aspects. First, we analyze a specific instance of the problem with 12 time periods (stages) to gain insights into the policies generated by the two algorithms and their operational characteristics. Second, we compare the two methods across multiple planning horizons, highlighting differences in computational efficiency and achieved optimal values.

In the following experiments, we use the following numerical values:

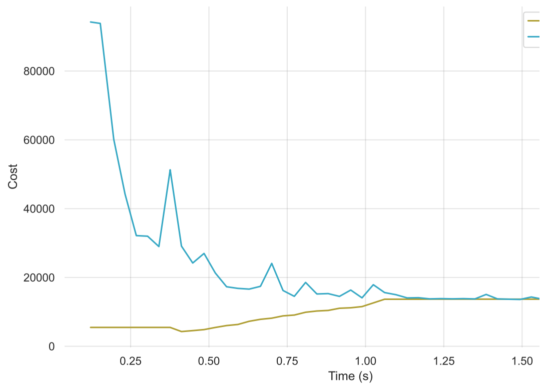
- $c_{ll} = 1\$/MWh$
- $c_i = 0.001\$/MWh \quad \forall i \in LR \cup IDRoR$
- $k = 1\$/MWh$
- $ws_{i,0} = 0.8\overline{WS}_i + 0.2\underline{WS}_i \quad \forall i \in LR$
- $wav_{i,0} = 0.8\dfrac{Cap_i}{Prod_i} \quad \forall i \in LR \cup IDRoR$
  This means that the available water already in the system initially is 80% of the maximum turbine capacity
- $\overline{WW}_i = 60\ 000 MWh$
- All other numerical values are available in the Appendix A

3.2.1 12-stage Problem

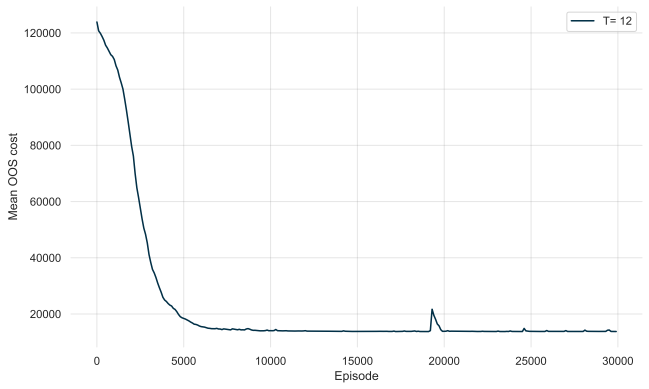
We study the problem (2.12) with 12-stage to illustrate how SDDP and RL generate policies and handle the underlying decision-making process. This simplified problem size allows for a more granular examination of the policy decisions at each stage and helps to isolate specific behavioral differences between the two methods. We analyze the water releases, energy production of both SDDP and RL. Through this case study, we aim to offer an intuitive understanding of the decision processes of each algorithm under a finite planning horizon.

Remember that uncertainty comes from water inflows to the system via the 5 rivers considered. The training phase simulates scenarios of water inflows and lets the agent explore various decisions by interacting with the simulated environment. As mentionned in the chapter 2, we allow the agent to explore for high number of iterations and observe the oos performance on simulated data.

We observe in figure 3.3 the convergence of both algorithms. For SDDP, con-



(a) Convergence of SDDP



(b) Convergence of RL

Figure 3.3: A figure with two subfigures

vergence is observed when the gap between the upper and lower bounds diminishes over time. Initially, the upper bound fluctuates significantly, reflecting uncertainty in the cost estimates during the early stages. However, as the iterations progress, these oscillations reduce, and the upper bound approaches the lower bound, indicating the algorithm's ability to stabilize around a near-optimal policy. On the other hand, the convergence of RL is demonstrated by the evolution of the mean oos cost over episodes. At the start of training, the oos cost is high, showcasing the initial inefficiency of the policy. Over time, this cost decreases steadily and stabilizes

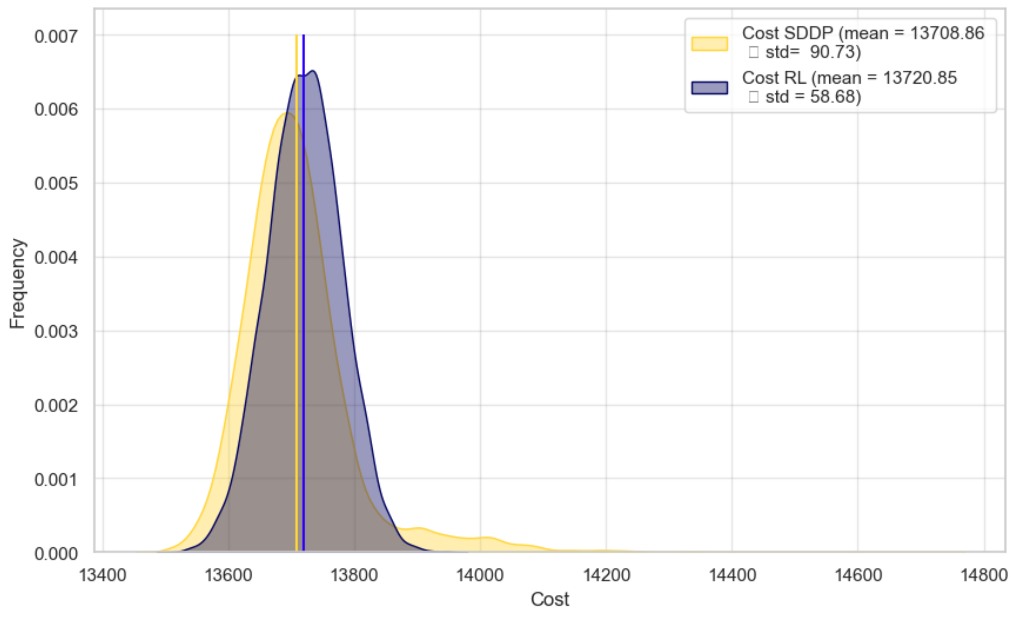


Figure 3.4: Out-of-sample tests RL vs SDDP. The relative gap is $Relgap = 0.3\%$

after approximately 10,000 episodes, suggesting that the RL model has learned a policy that generalizes well to unseen scenarios. Although a spike in the oos cost is observed around 20,000 episodes, it resolves quickly, with no long-term impact on convergence.

Now, we compare both algorithms in terms of oos mean cost to evaluate their performance under unseen scenarios. Figure (3.4) illustrates the distribution of oos costs obtained for both approaches. It is evident that SDDP slightly outperformed RL in terms of mean cost, with a relative performance gap, defined as

$$\text{RelGap} = \frac{\text{Cost}_{\text{RL}} - \text{Cost}_{\text{SDDP}}}{\text{Cost}_{\text{SDDP}}}, \tag{3.1}$$

being less than 1%. This indicates that while the average costs for both methods are nearly equivalent, SDDP holds a marginal advantage. One improvement brought by RL lies in the reduction of the cost variability. The standard deviation of oos costs was significantly reduced from 90.73 under SDDP to 58.68 under RL. This reduction implies that RL produces more consistent and reliable policies, making it better suited for scenarios where minimizing risk and variability is crucial.

Let's analyze the operational policies of the two algorithms through a specific hydropower scenario. Figures (3.6) and (3.5) show a consistent trend in energy production across the hydroplants over the time horizon. However, Figure (3.7) highlights a distinct difference in the management of non-turbined water releases between the two policies.

Both algorithms demonstrate a shared strategy to maximize energy output by taking advantage of the cascading structure of the hydropower system. Specifically, they prioritize water accumulation in the most downstream reservoir, *La Grande 2*, to capitalize on the opportunity to generate energy multiple times as the same water volume passes through upstream turbines before reaching the final reservoir.

To ensure that the reservoirs reach a state comparable to their productive initial conditions by the end of the planning horizon, both algorithms direct significant volumes of water to *La Grande 2*, which is filled to its maximum capacity at the terminal period. The two other large reservoirs—*La Grande 3* and *La Grande 1*—are gradually depleted during the process, reflecting their role in prioritizing multi-stage water utilization.

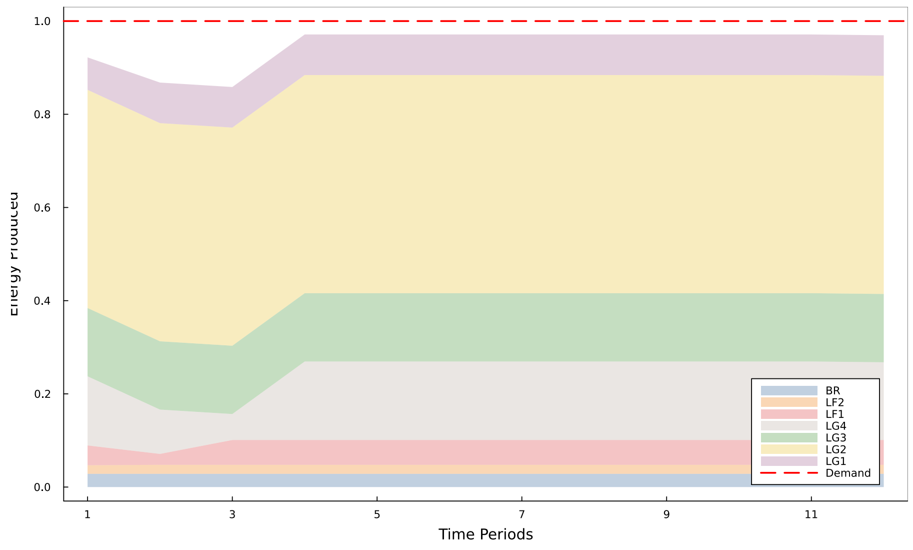


Figure 3.5: SDDP: Contribution of each hydroplant in satisfying the demand

The primary difference between RL and SDDP lies in how water is managed dur-

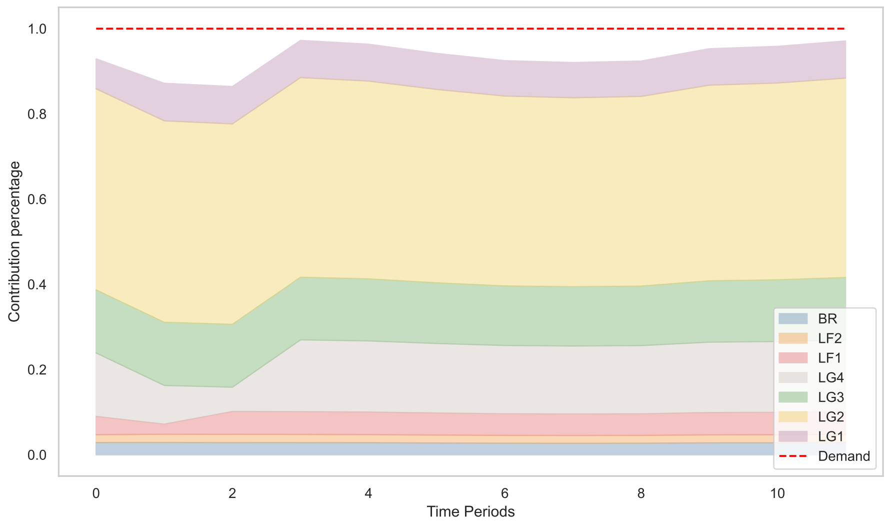


Figure 3.6: RL: Contribution of each hydroplant in satisfying the demand

ing intermediate periods. RL exhibits a policy of consistent water release throughout the planning horizon, maintaining stable reservoir dynamics and evenly distributing energy production. In contrast, SDDP adopts a more concentrated release strategy by temporarily accumulating water in *La Grande 3* (directly upstream of *La Grande 2*) before releasing it into *La Grande 2* in the final time step. This behavior indicates that SDDP places a higher emphasis on maximizing terminal-period flexibility and energy generation at the expense of steadier production throughout the earlier periods.

The divergence in water release patterns shows the difference in how each algorithm optimizes operational objectives. RL aims for balanced temporal dynamics, while SDDP, emphasizes end-of-horizon objectives, particularly through pre-emptive water accumulation upstream.

3.2.2 RL and SDDP comparison

We compare the performance of the algorithms across planning horizons of $\{4, 12, 20, 28, 36, 44, 52\}$ periods, where each period corresponds to one hour of planning. These horizons are chosen to capture a range of short- to long-term planning scenarios,

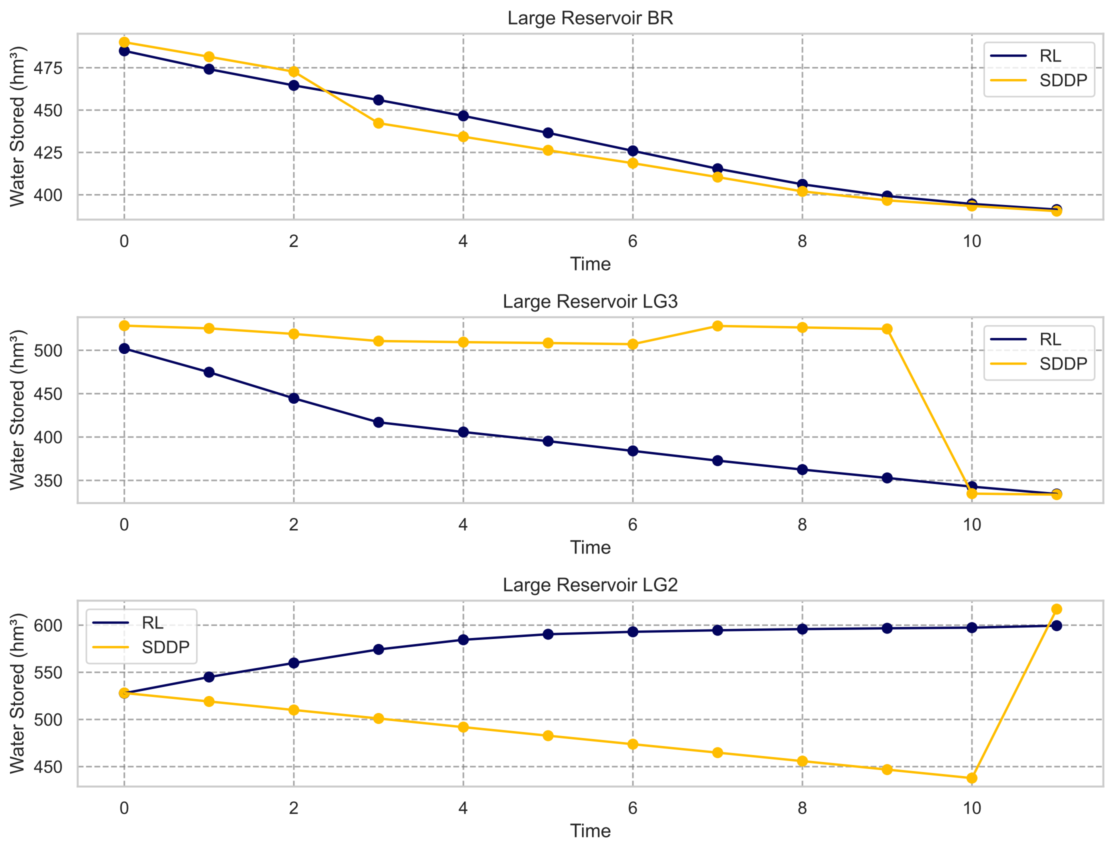


Figure 3.7: Comparison of the evolution of the water level in the large reservoirs between both algorithms

enabling an assessment of how the algorithms adapt to varying time scales and the computational challenges associated with each. This comparison highlights the trade-offs between computational efficiency and solution quality as the problem size increases.

To determine the convergence time of RL, we do two steps. At first, we let the agent train for a high number of episodes and we observe the oos performance every 100 training episodes. This will help determining the lowest number of training episodes needed for the convergence. The second step, is to remove the oos tests at every 100 training episodes and redo the training (from scratch). This will help determining the minimal number of training episodes needed and the time the training

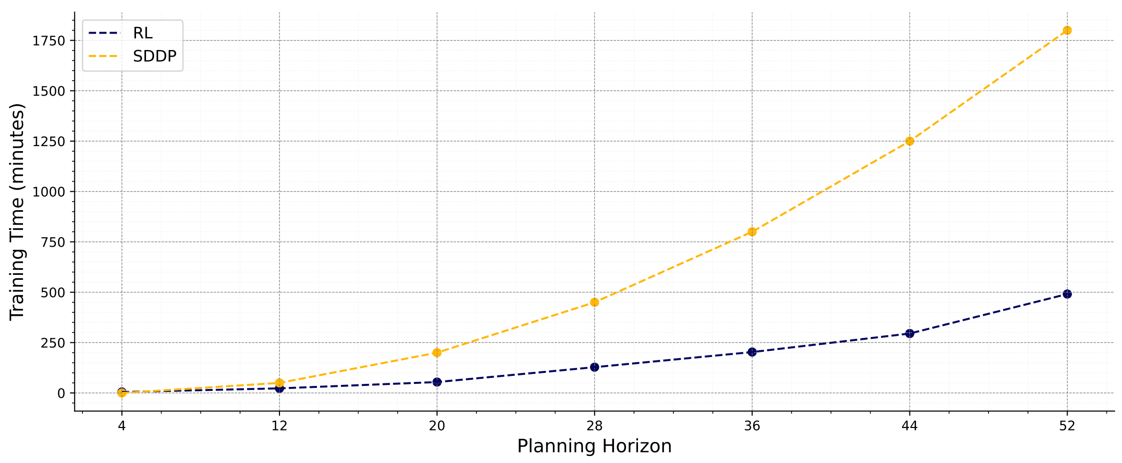


Figure 3.8: Training time for both algorithms for different problems

phase takes.

Figure (3.8) illustrates the training times for SDDP and RL across the problem set, which includes problems up to 52 periods. As expected, SDDP exhibits an exponential increase in training time as the problem size grows, consistent with its theoretical computational complexity. This exponential growth arises from its iterative backward-forward process, where an increasing number of state and decision variables demand greater computational resources. In contrast, RL demonstrates considerably lower training times even as problem complexity increases. This makes RL a computationally efficient alternative, particularly for scenarios where time and resources are constrained.

Figure (3.9) and figure (3.10) compare the relative performance of SDDP and RL in terms of the optimal value achieved. While SDDP generally attains slightly better solutions, the relative gap between the two algorithms remains within 3%, even for the largest problems considered. Notably, RL's ability to maintain this level of performance with significantly lower training time becomes especially advantageous in large-scale problems. The results suggest that RL can provide solutions close to those of SDDP, but with a fraction of the computational effort, making it a compelling choice for tackling large-scale problems efficiently.

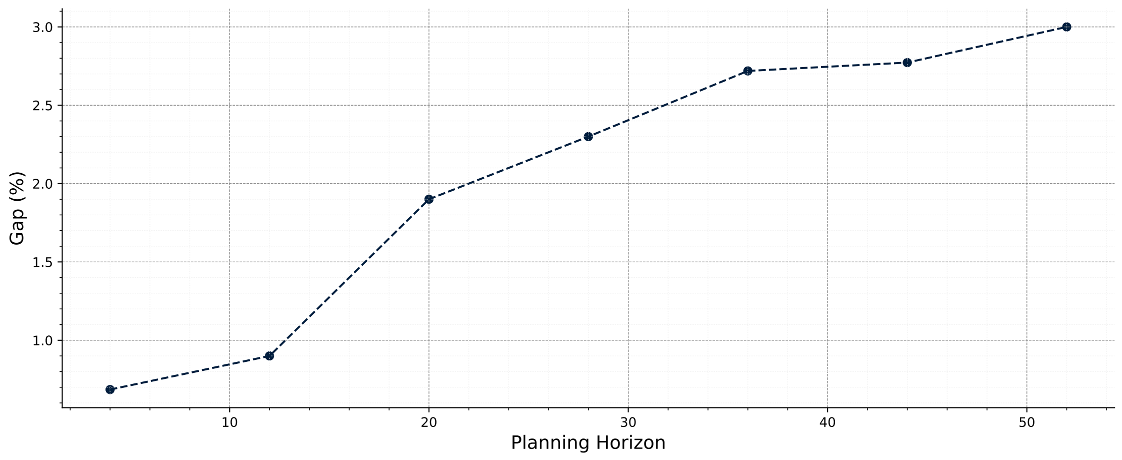


Figure 3.9: Relative gap of SDDP and RL for different problems

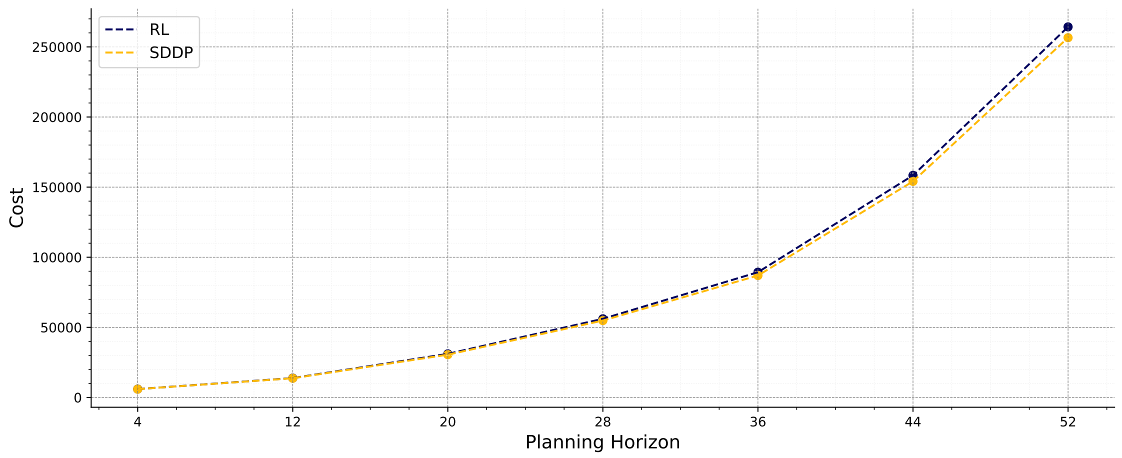


Figure 3.10: Mean value of the oos tests for SDDP and RL for different problems

Conclusion

In this thesis, we presented a detailed comparison between Stochastic Dual Dynamic Programming (SDDP) and Reinforcement Learning (RL), specifically the REINFORCE algorithm, for addressing a Dynamic Energy Dispatch problem in the context of multi-hydro-reservoirs management. Using a sub-region of the hydroelectric grid in Quebec as a case study, we analyzed the performance of both algorithms in managing energy dispatch with stochastic water inflows.

SDDP, recognized as the state-of-the-art algorithm for multi-stage stochastic optimization in hydropower management, demonstrated a slight advantage over the REINFORCE algorithm in terms of achieving the theoretical optimal solution. This is consistent with the inherent design of SDDP, which is optimized to converge to the theoretical solution by exploiting the stochastic structure of the problem. However, REINFORCE, a simple policy gradient algorithm within the reinforcement learning framework, showed its significant advantage in computational efficiency. Despite requiring less training time, REINFORCE was able to approximate the optimal value provided by SDDP, even in large-scale problems, suggesting its potential as a practical alternative when computational resources are constrained.

It is important to note that we tested both algorithms under the assumption of intertemporally independent water inflows, meaning that a high inflow at time t does not necessarily imply a high inflow at time $t+1$. This assumption excludes the phenomenon of seasons of high rainfall and low rainfal. Our findings confirm that RL, particularly the REINFORCE algorithm, is indeed capable of approximating the solution provided by SDDP, posing an interesting challenge to the traditional approach in hydropower optimization.

A natural extension of this work would involve relaxing the assumption of independent water inflows by considering a more realistic model incorporating *Markovian Noise*, which acknowledges the temporal correlation between inflows at different time steps. Future work could explore how this adjustment impacts the performance of SDDP and whether its advantage over RL remains under these more complex conditions. This exploration would further validate the potential of RL as a viable alternative to SDDP in dynamic energy dispatch problems and contribute to the ongoing evolution of optimization techniques for renewable energy systems. A further extension direction is the inclusion of additional exogenous variables in the state definition for Reinforcement Learning (RL). By incorporating these variables, RL algorithms can gain valuable insights and adapt to complex dynamics more effectively. This flexibility is a significant advantage of RL, as it enables the algorithm to infer relationships and dependencies from data without requiring explicit specification of the informational structure. In contrast, SDDP relies on predefined models of exogenous variables, and its performance is contingent on the accuracy of this explicit modeling. This limitation of SDDP makes it less adaptable in scenarios where the precise informational dependencies of exogenous variables are unknown or challenging to define.

Bibliography

[1] Daniel Ávila, Anthony Papavasiliou, and Nils Löhndorf. “Batch Learning SDDP for Long-Term Hydrothermal Planning”. In: *IEEE Transactions on Power Systems* 39.1 (2023), pp. 614–627.

[2] Ali Azizivahed, Roozbeh Karandeh, Valentina Cecchi, Ehsan Naderi, Li Li, and Jiangfeng Zhang. “Multi-area dynamic economic dispatch considering water consumption minimization, wind generation, and energy storage system”. In: *2020 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE. 2020, pp. 1–5.

[3] Richard Bellman. “The theory of dynamic programming”. In: *Bulletin of the American Mathematical Society* 60.6 (1954), pp. 503–515.

[4] J. F. Benders. “Partitioning procedures for solving mixed-variables programming problems”. In: *Numerische Mathematik* 4.1 (1962), pp. 238–252.

[5] Dimitri Bertsekas. “Convergence of discretization procedures in dynamic programming”. In: *IEEE Transactions on Automatic Control* 20.3 (1975), pp. 415–419. DOI: 10.1109/TAC.1975.1100984.

[6] Dimitri Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming.* Athena Scientific, 1996.

[7] John R Birge. “Decomposition and partitioning methods for multistage stochastic linear programs”. In: *Operations research* 33.5 (1985), pp. 989–1007.

[8] R. M. Chabar, M. V. F. Pereira, S. Granville, L.A. Barroso, and N. A. Iliadis. “Optimization of Fuel Contracts Management and Maintenance Scheduling for Thermal Plants under Price Uncertainty”. In: *2006 IEEE PES Power Systems Conference and Exposition*. 2006, pp. 923–930. DOI: `10.1109/PSCE.2006.296437`.

[9] Xu Chen and Guowei Tang. “Solving static and dynamic multi-area economic dispatch problems using an improved competitive swarm optimization algorithm”. In: *Energy* 238 (2022), p. 122035.

[10] Jang-Sung Chun, Hyun-Kyo Jung, and Song-Yop Hahn. “A study on comparison of optimization performances between immune algorithm and other heuristic algorithms”. In: *IEEE transactions on magnetics* 34.5 (1998), pp. 2972–2975.

[11] Climate Watch. *Climate Watch. 2024. Washington, DC: World Resources Institute (WRI).* Available online at: https://www.climatewatchdata.org. 2024.

[12] Leandro dos Santos Coelho and Viviana Cocco Mariani. “Economic dispatch optimization using hybrid chaotic particle swarm optimizer”. In: *2007 IEEE International Conference on Systems, Man and Cybernetics*. 2007, pp. 1963–1968. DOI: `10.1109/ICSMC.2007.4414152`.

[13] Pascal Côté, Didier Haguma, Robert Leconte, and Stephane Krau. “Stochastic optimisation of Hydro-Quebec hydropower installations: a statistical comparison between SDP and SSDP methods”. In: *Canadian Journal of Civil Engineering* 38.12 (2011), pp. 1427–1434.

[14] Tao Ding, Xiaosheng Zhang, Runzhao Lu, Ming Qu, Mohammad Shahidehpour, Yuankang He, and Tianen Chen. “Multi-stage distributionally robust stochastic dual dynamic programming to multi-period economic dispatch with virtual energy storage”. In: *IEEE Transactions on Sustainable Energy* 13.1 (2021), pp. 146–158.

[15] Oscar Dowson and Lea Kapelevich. "SDDP. jl: a Julia package for stochastic dual dynamic programming". In: *INFORMS Journal on Computing* 33.1 (2021), pp. 27–33.

[16] Environment and Climate Change Canada. *Climate trends and variations bulletin - Winter 2023-2024*. Series. 2024. URL: `https://publications.gc.ca/collections/collection_2024/eccc/En81-23-2024-1-eng.pdf`.

[17] Christian Füllner and Steffen Rebennack. *Stochastic dual dynamic programming and its variant-a review*. 2023. URL: `https://optimization-online.org/?p=16920`.

[18] International Energy Agency. *CO2 Emissions in 2023*. Licence: CC BY 4.0. Paris, 2024. URL: `https://www.iea.org/reports/co2-emissions-in-2023`.

[19] IPCC. "Climate Change 2023: Synthesis Report. Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change". In: (2023). Ed. by H. Lee Core Writing Team and J. Romero, pp. 35–115.

[20] Sham M Kakade. "A natural policy gradient". In: *Advances in neural information processing systems* 14 (2001).

[21] Aidana Kalakova, H. S. V. S. Kumar Nunna, Prashant K. Jamwal, and Suryanarayana Doolla. "A Novel Genetic Algorithm Based Dynamic Economic Dispatch With Short-Term Load Forecasting". In: *IEEE Transactions on Industry Applications* 57.3 (2021), pp. 2972–2982. DOI: `10.1109/TIA.2021.3065895`.

[22] James Kennedy and Russell Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. ieee. 1995, pp. 1942–1948.

[23] Elizaveta Kuznetsova, Yan-Fu Li, Carlos Ruiz, Enrico Zio, Graham Ault, and Keith Bell. "Reinforcement learning for microgrid energy management". In: *Energy* 59 (2013), pp. 133–146.

[24] Yu Lan, Qiaozhu Zhai, Xiaoming Liu, and Xiaohong Guan. “Fast stochastic dual dynamic programming for economic dispatch in distribution systems”. In: *IEEE Transactions on Power Systems* 38.4 (2022), pp. 3828–3840.

[25] Rein Luus. “Iterative dynamic programming: From curiosity to a practical optimization procedure”. In: *Control and intelligent systems* 26.1 (1998), pp. 1–8.

[26] MEP Maceira and JM Damázio. “Use of the PAR (p) model in the stochastic dual dynamic programming optimization scheme used in the operation planning of the Brazilian hydropower system”. In: *Probability in the Engineering and Informational Sciences* 20.1 (2006), pp. 143–156.

[27] Seyedali Mirjalili and Andrew Lewis. “The whale optimization algorithm”. In: *Advances in engineering software* 95 (2016), pp. 51–67.

[28] Florian Mitjana, Michel Denault, and Pierre-Olivier Pineau. *Power Decarbonation in Northeastern America: the benefits of multi-stage planning and regional cooperation.* Oct. 2023. DOI: 10.13140/RG.2.2.28713.57445.

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. “Human-level control through deep reinforcement learning”. In: *nature* 518.7540 (2015), pp. 529–533.

[30] Somayeh Moazeni, Amir H. Miragha, and Boris Defourny. “A Risk-Averse Stochastic Dynamic Programming Approach to Energy Hub Optimal Dispatch”. In: *IEEE Transactions on Power Systems* 34.3 (2019), pp. 2169–2178. DOI: 10.1109/TPWRS.2018.2882549.

[31] Morteza Nazari-Heris, Mehdi Mehdinejad, Behnam Mohammadi-Ivatloo, and Gevork Babamalek-Gharehpetian. “Combined heat and power economic dispatch problem solution by implementation of whale optimization method”. In: *Neural Computing and Applications* 31 (2019), pp. 421–436.

[32] Shanshan Pan, Jinbao Jian, Huangyue Chen, and Linfeng Yang. “A full mixed-integer linear programming formulation for economic dispatch with valve-point effects, transmission loss and prohibited operating zones”. In: *Electric Power Systems Research* 180 (2020), p. 106061.

[33] Shanshan Pan, Jinbao Jian, and Linfeng Yang. “A hybrid MILP and IPM approach for dynamic economic dispatch with valve-point effects”. In: *International Journal of Electrical Power & Energy Systems* 97 (2018), pp. 290–298.

[34] 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”. In: *IEEE Transactions on Sustainable Energy* 9.2 (2017), pp. 547–558.

[35] Jong-Bae Park, Yun-Won Jeong, Woo-Nam Lee, and Joong-Rin Shin. “An improved particle swarm optimization for economic dispatch problems with non-smooth cost functions”. In: *2006 IEEE Power Engineering Society General Meeting*. 2006, 7 pp.-. DOI: 10.1109/PES.2006.1709300.

[36] Mario VF Pereira and Leontina MVG Pinto. “Multi-stage stochastic optimization applied to energy planning”. In: *Mathematical programming* 52 (1991), pp. 359–375.

[37] Andrew B Philpott and Ziming Guan. “On the convergence of stochastic dual dynamic programming and related methods”. In: *Operations Research Letters* 36.4 (2008), pp. 450–455.

[38] Andrew B Philpott, Vitor L de Matos, and Lea Kapelevich. “Distributionally robust SDDP”. In: *Computational Management Science* 15 (2018), pp. 431–454.

[39] Andy Philpott, Vitor de Matos, and Erlon Finardi. "On solving multistage stochastic programs with coherent risk measures". In: *Operations Research* 61.4 (2013), pp. 957–970.

[40] Roberto J Pinto, CarmenL T Borges, and Maria EP Maceira. "An efficient parallel algorithm for large scale hydrothermal system operation planning". In: *IEEE Transactions on Power Systems* 28.4 (2013), pp. 4888–4896.

[41] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality.* Vol. 703. John Wiley & Sons, 2007.

[42] Steffen Rebennack. "Combining sampling-based and scenario-based nested Benders decomposition methods: application to stochastic dual dynamic programming". In: *Mathematical Programming* 156 (2016), pp. 343–389.

[43] TA Rotting and A Gjelsvik. "Stochastic dual dynamic programming for seasonal scheduling in the Norwegian power system". In: *IEEE Transactions on Power Systems* 7.1 (1992), pp. 273–279.

[44] Mohsen Saadat and Keyvan Asghari. "Feasibility improved stochastic dynamic programming for optimization of reservoir operation". In: *Water Resources Management* 33.10 (2019), pp. 3485–3498.

[45] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. *Trust Region Policy Optimization.* 2017. arXiv: `1502.05477 [cs.LG]`. URL: `https://arxiv.org/abs/1502.05477`.

[46] Sara Séguin, Stein-Erik Fleten, Pascal Côté, Alois Pichler, and Charles Audet. "Stochastic short-term hydropower planning with inflow scenario trees". In: *European Journal of Operational Research* 259.3 (2017), pp. 1156–1168.

[47] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. "Deterministic Policy Gradient Algorithms". In: *Proceedings of the 31st International Conference on Machine Learning.* Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Re-

search 1. Bejing, China: PMLR, 22–24 Jun 2014, pp. 387–395. URL: `https://proceedings.mlr.press/v32/silver14.html`.

[48] Richard S Sutton. “Learning to predict by the methods of temporal differences”. In: *Machine learning* 3 (1988), pp. 9–44.

[49] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[50] Daniel Tenfen and Erlon Cristian Finardi. “A mixed integer linear programming model for the energy management problem of microgrids”. In: *Electric Power Systems Research* 122 (2015), pp. 19–28.

[51] Christopher JCH Watkins and Peter Dayan. “Q-learning”. In: *Machine learning* 8 (1992), pp. 279–292.

[52] J. Whitmore and Pierre-Olivier Pineau. *État de l’énergie au Québec 2023.* Préparé pour le gouvernement du Québec. Chaire de gestion du secteur de l’énergie, HEC Montréal, 2023.

[53] Ronald J Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine learning* 8 (1992), pp. 229–256.

[54] Yinliang Xu, Wei Zhang, Wenxin Liu, and Frank Ferrese. “Multiagent-Based Reinforcement Learning for Optimal Reactive Power Dispatch”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.6 (2012), pp. 1742–1751. DOI: `10.1109/TSMCC.2012.2218596`.

[55] Jingxian Yang, Jichun Liu, Yue Xiang, Shuai Zhang, and Junyong Liu. “Data-driven Optimal Dynamic Dispatch for Hydro-PV-PHS Integrated Power Systems Using Deep Reinforcement Learning Approach”. In: *CSEE Journal of Power and Energy Systems* 9.3 (2023), pp. 846–858. DOI: `10.17775/CSEEJPES.2021.07210`.

[56] Wei-Chang Yeh, Min-Fan He, Chia-Ling Huang, Shi-Yi Tan, Xianyong Zhang, Yaohong Huang, and Li Li. “New genetic algorithm for economic dispatch of stand-alone three-modular microgrid in DongAo Island”. In: *Applied Energy* 263 (2020), p. 114508.

Appendix A

Code	Name	Abbreviation
R1	Caniapiscau	Can
R2	Laforge	Laf
R3	La Grande	Lag
R4	De Pontois	Dep
R5	Sakami	Sak

Table 1: List of the rivers considered in this thesis

Code	Name	Abbreviation
P1	Brisay	BR
P2	Laforge 2	LF2
P3	Laforge 1	LF1
P4	La Grande 4	LG4
P5	La Grande 3	LG3
P9	La Grande 2-A	**
P10	La Grande 2	LG2
P11	La Grande 1	LG1

Table 2: List of hydroplants and their corresponding code and abbreviation used in this thesis

Remark 10 *P9 and P10 are treated as one unique hydroplant with capacity equals to the sum of both hydroplants' capacities.*

Code	**Name**	**Discharge (m^3/s)**	**Generation (MW)**
P1	Brisay	1130	469
P2	Laforge 2	784	319
P3	Laforge 1	1693	878
P4	La Grande 4	2783	2779
P5	La Grande 3	3439	2417
P9	La Grande 2-A	1620	2106
P10	La Grande 2	4300	5616
P11	La Grande 1	5950	1436

Table 3: Discharge and generation capacity of represented hydropower plants

Number	**Name**	**Min. (Billion m^3)**	**Max. (Billion m^3)**
1	Caniapiscau	39.0	52.6
4	La Grande 3	25.2	60.0
5	Robert Bourassa	19.4	61.7

Table 4: Storage capacity of large reservoirs considered in this thesis