UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

Hierarchical Distribution System Optimization under Uncertainty

Dissertation presented by
Taku KANEDA

for obtaining the Master's degree in
Mathematical Engineering

Supervisor(s)
Anthony PAPAVASILIOU

Reader(s)
Yves SMEERS, Yasushi MASUDA

Academic year 2016-2018

Abstract

The integration of renewable energy resources (RES) leads to a significant change in the current state of the energy market in recent years. Although the high-voltage transmission system has been optimized with the integration of RES, the low-voltage distribution system has largely undeveloped due to its complexity. The difficulty of the distribution network operations can be characterized mainly by the following three parts: (i) Uncertainty, (ii) Scale, and (iii) Non-linearity of power flow constraints, and we tackle the first two challenges in this thesis. We propose a hierarchical approach which focuses on the radial structure of the distribution network that we can divide the network into small sub-networks (layers). In such a network, interactions of layers are weak in the sense that they exchange power only on a few interfaces. This fact implies that the possibility of solving a problem of each layer of the hierarchy, independently. Thus, the approach can be applicable to systems of arbitrary size. In the first proposed approach, we decompose the network and implement the stochastic dual dynamic programming (SDDP) at each layer in order to obtain the value functions which contain the information of future costs at a given stage. In the second approach, we combine the idea of the previous algorithm and Model Predictive Control (MPC) which is known as an iterative optimization algorithm with a feedback. We test our policies on a small-scale and a large-scale problem with several distribution system models as case studies. We demonstrate the superiority of proposed policies in terms of average performance and computation time. This study can be an important starting point for considering fully distributed systems or future electric market designs.

Acknowledgements

I would first like to thank my thesis advisor Professor Papavasiliou of the École polytechnique de Louvain at Université Catholique de Louvain. The door to Professor Papavasiliou was always open whenever I had a trouble or a question about my research or writing, and our meetings were the highlight of my week. I am grateful for having been his student.

I would also like to acknowledge my supervisor at Keio University, Professor Yasushi Masuda. He gave me a boost to my study abroad and helped me through three important years of my life. I learned an attitude towards research from him.

Besides my advisors, I would like to thank Professor Yves Smeers of the École polytechnique de Louvain at Université Catholique de Louvain for beeing the reader of this thesis.

My sincere thanks also goes to Ilyès Mezghani, who gave me the precious help and advice from the beginning to the end of this study. Without his passionate participation and input, my work could not have been successfully conducted.

Furthermore, I would like to express my sincere gratitude to all the people involved in the Double Degree program, especially to the Department Manager, Professor Absil, and to the coordinators: Ms Emmanuelle Brun and Ms Yuko Sakuma. Without their help, I could not complete the program. I hope that the friendly relationship between both universities will continue.

I thank all my friends in Belgium who spent pleasant time together and supported my stay for a year and a half. In particular, I am grateful to Bruno Losseau for working hard and encouraging each other. He gave me wonderful memories of the stay.

Finally, I must express my very profound gratitude to my family for providing me with unfailing support and continuous encouragement. This accomplishment would not have been possible without them. Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In recent years, the integration of renewable energy resources (RES) leads to a significant change in the current state of the energy market. The current paradigm of electric power system operations has focused on the high-voltage transmission networks, while there is less development on the operation of distribution networks. However, due to the existence of small-scale solar panels, demand response, or deferrable demand such as battery storage for electric vehicles, it is necessary that a considerable amount of intelligence will have to be integrated at the distribution level of electric power systems. As a result, distribution system operators (DSO) will assume a more important role in the future market.

The current paradigm of electric power system operations has focused on transmission networks. The high-voltage transmission system has been optimized with the integration of RES, whereas the low-voltage distribution system has not been considered in detail. The vast amount of flexible resources which are connected to the distribution network remain largely undeveloped. The flexibility of the distribution network originates mainly from a large amount of resources (e.g. household-level solar panels, and deferrable demand through battery storage). We will need to utilize these resources efficiently in order to keep the quality of electric service that we are now enjoying [6]. However, there are many challenges in effectively operating distribution networks.

The distribution network is characterized by significant complexity. This complexity can be mainly divided into the following three parts. (i) Uncertainty: The introduction of RES has been proceeded in household-level, and it becomes more important element of the power system. The problem is, however, the unpredictability of power generation since the renewable production cannot be perfectly forecasted. (ii) Scale: The number of resources we need to consider is also a problem. A typical transmission network is composed of a few hundreds or thousands of buses, while a distribution network possibly contains thousands to millions of resources that can be activated intelligently [7]. (iii) Non-linearity of power flow constrains: Since the distribution network is a low-voltage system, the linear approximation of Kirchhoff's laws is not reliable anymore. Voltage and reactive power constraints need to be expressed explicitly.

In this thesis, we will focus on the first two problems, (i) Uncertainty and (ii) Scale, as a starting point for future work. Our objective is to propose a new approach for dispatching distribution networks effectively. Our approach exploits the radial structure of the distribution network. Figure 1.1 provides a graphical depiction of the transmission and distribution network. We will exploit the radial structure by specifically proposing a *hierarchical approach* for dispatching distributed resources.

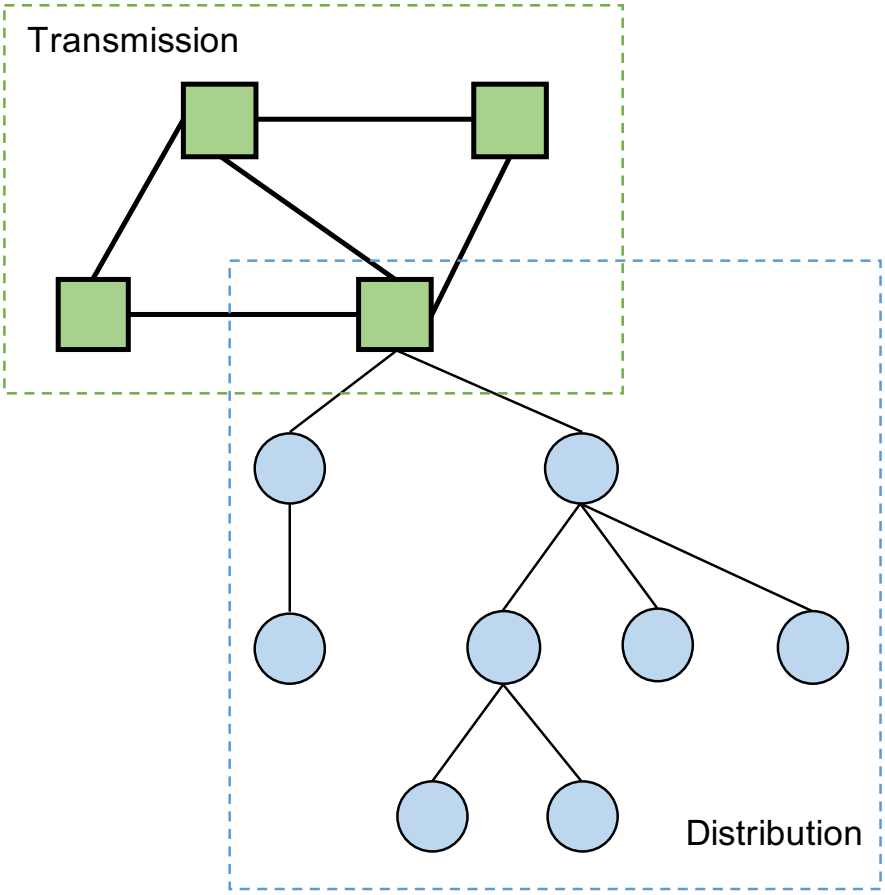


Figure 1.1: Transmission and distribution network. The square nodes represent the buses of transmission which possibly contains circle structure. While circular nodes represent resources of distribution network whose structure is radial.

1.2 State-of-the-Art

1.2.1 Short-Term Operation of Power Systems

We focus on the short-time operation (e.g. day-ahead and real-time) due to the following reasons. The first reason is that the majority of irrevocable decisions (e.g. unit commitment) regarding the operation of the system are determined in this time flame. Secondly, the short-term operations decide the real-time prices which involve the entire industry. In general, the system operator is required to perform decision making within a few minutes or an hour during real-time dispatch. Therefore, we have to adopt a dispatch policy that takes these time limits into consideration.

1.2.2 Uncertainty in Power System Operations

The short-term dispatch problem in power system operation of distribution networks can be expressed as a dynamic optimization problem under uncertainty. Multi-stage stochastic programming has been successfully applied to a wide range of planning problems, e.g. real-time operations [19], medium-term [20] and long-term planning [14]. Pinto and Perreira proposed an algorithm for solving a specific class of optimization problems under uncertainty, namely multi-stage stochastic linear programs [20]. The proposed algorithm is named stochastic dual dynamic programming (SDDP) and it has been shown that it performs well in medium-term multi-stage hydro-thermal scheduling under rainfall uncertainty for handling water levels of hydro reservoirs [20]. Furthermore, the SDDP algorithm has successfully been applied to real-time operations [19], [13]. As mentioned in the motivation, optimization under uncertainty is not

novel for high-voltage transmission networks. When we introduce the methods to the low-voltage network, however, certain challenges remain. The first one is that the power flow constraints are nonlinear and non-convex, and the second one is the scalability of the problem. Even if we ignore the first point in this thesis, the scale issue makes the problem unmanageable.

On the other hand, Model Predictive Control (MPC), also known as receding horizon control is a feedback control technique that naturally incorporates optimization [16]. The main advantage of MPC is the fact that it optimizes over a time horizon that looks beyond the current time stage, and applies the solution to the current stage. Hence it can take the future into account while preserving flexibility to cope with future unexpected incidents. MPC has been used in chemical process control [22], oil refineries [8], stochastic control in finance [11], revenue management [27], the control for hybrid vehicles [28], and spacecraft rendezvous and proximity maneuvering [5]. In recent years, it has also been applied to power system balancing problems [29]. In the context of energy management, the authors in [29] developed an open-source Python library implementing a dynamic energy management problem with robust MPC which optimizes the worst-case performance over scenarios which are provided as inputs to the optimization problem. However, there still exists the difficulty in the scale of the problem. Since MPC works in an online fashion where the frequency of decision making is a few minutes or an hour during real-time dispatch, a decision must be obtained quickly. However, if the network size becomes large, it is almost impossible to compute the problem with an adequate number of scenarios for robust MPC.

1.2.3 Scale of Power System Optimization Problems

In order to tackle the challenge that the distribution network includes a huge number of resources of uncertainty, we focus on the fact that the system has a radial structure. Indeed, the distribution network has a *self-similar* structure where a node of the network contains a sub-network of the system. Figure 1.2 provides a graphical illustration. Each node may contain RES (e.g. rooftop solar panels), household demand, battery storage, and electric vehicles, as well as a lower-level nodes with a self-similar structure. The important point is that this viewpoint can be exploited for solving the problem of scalability in the distribution network. Every node (or layer) connects to the adjacent layers with interfaces. That is to say, the power injected by a higher level of the hierarchy and the power that the corresponding low-level layer will receive are equivalent, and that is the only coupling relationship. In other words, the interaction of each layer in the distribution network is weak compared to a completely meshed structure. This fact implies that the possibility of solving a problem of each layer of the hierarchy, independently.

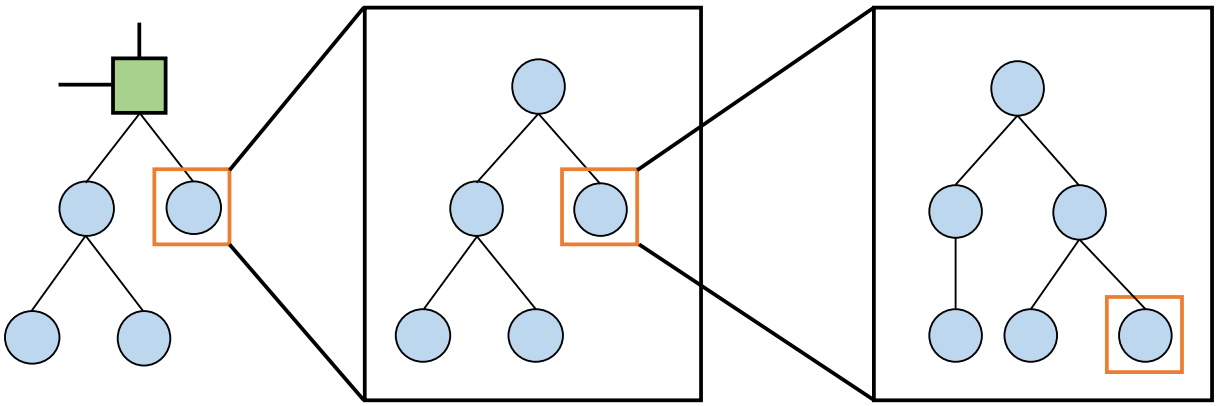


Figure 1.2: The self-similar structure of hierarchical distribution network. Red nodes can be expanded to the right-side sub-network and it contains a lower level self-similar node.

Hierarchical approaches have been applied for solving a multi-stage stochastic programming. Birge proposed an algorithm for solving a multi-stage stochastic linear programming by decomposing at each stage and each realization of random parameters (*nested decomposition*) [2]. The algorithm can be regarded as a hierarchical approach in the sense that the decomposition focuses on the *temporal* evolution of the problem whereby at a given stage, the cost of the following stages are cast as a piece-wise linear convex function. On the contrary, we focus on the *spatial* evolution of the distribution network. We consider that a layer nests the corresponding lower-level layers. The advantage of a hierarchical approach is the fact that, if it is possible, it becomes scalable to systems of arbitrary size.

1.2.4 Optimal Dispatch Problem

A general optimal dispatch problem can be described as follows:

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & Zx = v \end{aligned} \tag{1.1}$$

where x represents the set of state and action variables, and the objective function $f(x)$ is the cost induced by the operation. The set of linear equations $Zx = v$ expresses physical constraints of the system, e.g. power balance, capacities. In this thesis, we will consider a multi-stage problem with a finite horizon. Hence, x can be separable by time stage, i.e. $x^T = [x_1^T, \ldots, x_H^T]$ with the time horizon H, and the constraints $Zx = v$ may contain dynamical ones. A *policy* is a rule that determines the decision x_t given the available information for each stage $t = 1, \ldots, H$[21]. The information may consist of the variable x itself and a realization of uncertainty if the system is under uncertainty. We refer to the resulting variable x determined by a policy as the *dispatch*.

1.3 Overview of the Thesis

The rest of this thesis is organized as follows. In Chapter 2, we discuss general methods for managing the dispatch problem under uncertainty. Then, we introduce our proposed policies which rely on a hierarchical structure in Chapter 3. In Chapter 4, the distribution network models which we will investigate in this work are described. In Chapter 5, we introduce a stochastic modeling technique for generating random input for the problem (i.e. net demand). We also describe a method for generating a lattice model of uncertainty which is used in the algorithms. Chapter 6 proposes a scenario selection algorithm for obtaining suitable scenarios for optimization under uncertainty with small number of samples. The method is useful for reducing the computation time for evaluating the dispatch polices. We confirm our polices by case studies of a two-layer and a multi-layer model in Chapter 7. Finally, we conclude our work and state proposals of future research in Chapter 8.

Chapter 2

Uncertainty Management

In this chapter, we present general (traditional) dispatch policies under uncertainty in distribution networks. We extend the general optimal dispatch problem (1.1) to a multi-stage stochastic formulation.

2.1 Stochastic Dual Dynamic Programming

The optimal dispatch problem with discrete time steps $t \in \{1, \dots, H\} = T$ can be expressed as a multi-stage stochastic linear problem (MSLP). Ω_t denotes the discretized sample space at stage $t \in T$ and $\Omega_{[t]}$ is the set of histories up to t. Each history $\omega_{[t]} \in \Omega_{[t]}$ has a unique ancestor $a(\omega_{[t]}) \in \Omega_{[t-1]}$. We also assume that the realizations follow a Markov Process. This fact implies that a probability measure $\Pr(\omega_{[t]})$ can be defined in a recursive way: $\Pr(\omega_{[t]}) = \Pr(a(\omega_{[t]}))\Pr(\omega_{[t]}|a(\omega_{[t]}))$ with $\Pr(\omega_{[1]}) = 1$. Then, the MSLP can be written in extensive form as follows.

$$\begin{aligned} \min_{x} \quad & \sum_{t \in T} \sum_{\omega \in \Omega_{[t]}} \Pr(\omega_{[t]}) c_t^T x_{t,\omega_{[t]}} \\ \text{s.t.} \quad & W_t x_{t,\omega_{[t]}} = h_{t,\omega_t} - T_t x_{t-1,a(\omega_{[t]})}, t \in T, \omega_{[t]} \in \Omega_{[t]} \\ & x_{t,\omega_{[t]}} \geq 0, t \in T, \omega_{[t]} \in \Omega_{[t]} \end{aligned} \tag{2.1}$$

where $\omega_t \in \Omega_t$ is a realization, $x_{t,\omega_{[t]}}$ is the set of state and action variables at stage t, c_t^T are the cost coefficients, W_t are the coefficients of the current stage decision $x_{t,\omega_{[t]}}$, h_{t,ω_t} are the right-hand side parameters, and T_t is the coefficient matrix of previous-stage decisions $x_{t-1,a(\omega_{[t]})}$. Note that we assume right-hand side uncertainty, hence the cost coefficient c_t^T, and the coefficients of the current and the previous stage decisions W_t and T_t are deterministic at every stage. In practice, this problem is intractable due to the exponential growth of the size of the history of outcomes $\Omega_{[t]}$.

In [20], the authors developed an algorithm, named *stochastic dual dynamic programming* (SDDP), for solving MSLP (2.1) by decomposition and Monte-Carlo simulation. In order to describe the algorithm, we first describe the uncertainty model of the system in the form of a lattice.

2.1.1 Lattice Model of Uncertainty

A *lattice* is a economic way of describing uncertainties in a discrete time Markov process. A lattice has the following features: (i) there are $|\Omega_t|$ outcomes at each stage $t \in T$, (ii) each outcome $\omega_t \in \Omega_t$ is connected to all nodes of stage $t+1$ by edges which correspond to *transition probabilities*. More precisely, at each outcome $k \in \Omega_t$, the realization of the random vector $\xi_t = [h_t]$ is defined , and outcome k is connected to all outcomes of the next stage $j \in \Omega_{t+1}$.

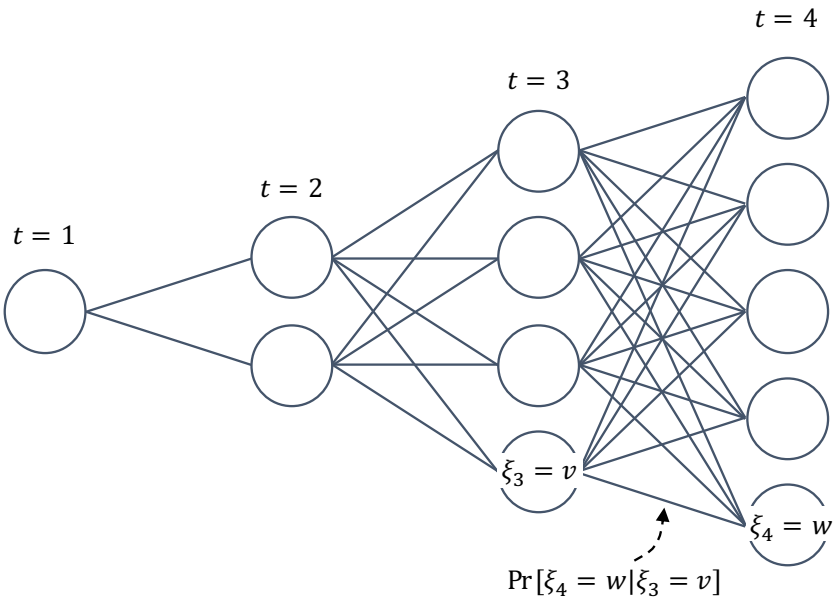


Figure 2.1: An illustration of a lattice. Each node is the realization of a stochastic process. Each edge is characterized by a transition probability.

The corresponding transition probabilities are described in the form of *conditional probabilities*: $\Pr(\xi_{t,j}|\xi_{t,k})$ for all $k \in \Omega_t$ and $j \in \Omega_{t+1}$. An illustration of a lattice is provided in Figure 2.1.

2.1.2 Nested L-shaped Decomposition Sub-problem

We first tackle the extended form of MSLP (2.1) using dynamic programming. At the final stage of the problem, we have,

$$\begin{aligned} Q_H(x_{H-1}, \xi_H) = \min_x \quad & c_H^T x \\ & W_H x = h_{t,\omega_H} - T_t x_{H-1} \\ & x \geq 0 \end{aligned} \tag{2.2}$$

where the function Q_H called a *Q-function*. Then, the *value function* at stage H can be defined as follows:

$$V_{H,\omega_{H-1}}(x_{H-1}) = \mathbb{E}_{\xi_H}[Q_H(x_{H-1}, \xi_H)|\omega_{H-1}]. \tag{2.3}$$

Recursively, given an input x_{t-1} and a realization ξ_t, the Q-function of stage t that expresses the cost of the given stage as a function of the expected cost of remaining stages can be described as follows:

$$\begin{aligned} Q_t(x_{t-1}, \xi_t) = \min_x \quad & c_t^T x + V_{t+1,\omega_t}(x) \\ & W_t x = h_{t,\omega_t} - T_t x_{t-1} \\ & x \geq 0 \end{aligned} \tag{2.4}$$

The value function at stage t for an input x_{t-1} is given as:

$$V_{t,\omega_{t-1}}(x_{t-1}) = \mathbb{E}_{\xi_t}[Q_t(x_{t-1}, \xi_t)|\omega_{t-1}]. \tag{2.5}$$

The value function has the following feature.

Theorem 2.1. *For $t \in T$ and $k \in \Omega_t$, $V_{t+1,k}(x)$ is a piecewise linear convex function.*

Proof. See page 94-96 of the textbook of Operations Research (LINMA 2491)[1]. □

[1]Webpage of the course: http://uclengiechair.be/operations-research-linma-2491/

We are now able to decompose the extended form of MSLP (2.1) into a set of sub-problems, which are defined at each stage and each outcome. The sub-problems are called *nested L-shaped decomposition sub-problems* (NLDS). Indeed, an NLDS can be defined at every node of the lattice. The $NLDS_{t,k}$ at stage $t \in T$ and outcome $k \in \Omega_t$ can be expressed as follows:

$$\begin{aligned} NLDS_{t,k}(\hat{x}_{t-1}) : \min_{x} \quad & c_t^T x + V_{t+1,k}(x) \\ & W_t x = h_{t,k} - T_t \hat{x}_{t-1} \\ & x \geq 0 \end{aligned} \tag{2.6}$$

where $\hat{x}_{t-1}$ is a trial decision at stage $t-1$, thus it is treated as a parameter in $NLDS_{t,k}$.

Since $V_{t+1,k}(x)$ is piecewise convex, we can express (2.6) equivalently in the following way[2]:

$$\begin{aligned} NLDS_{t,k}(\hat{x}_{t-1}) : \quad \min_{x} \quad & c_t^T x + \theta_{t,k} \\ (\pi) \quad & W_t x = h_{t,k} - T_t \hat{x}_{t-1} \\ (\rho) \quad & E_{t,k} x + \theta_{t,k} \geq e_{t,k} \\ & x \geq 0 \end{aligned} \tag{2.7}$$

where $\theta_{t,k}$ is a free variable, $E_{t,k}$ and $e_{t,k}$ are the coefficients and the right-hand-side parameters which express the shape of $V_{t+1,k}(x)$, and π and ρ are the dual multipliers of the corresponding constraints, respectively. The constraints $E_{t,k}x + \theta_{t,k} \geq e_{t,k}$ are named the *optimality cuts*. In general, the exact formulation of value functions can consist of a huge number of inequalities. However, our goal is to find the optimal solution, hence it is enough to know some "relevant" parts of the value functions. We do not have to spend time to figure out unnecessary parts of the function.

2.1.3 SDDP Algorithm

The idea of the SDDP algorithm is to solve the problem iteratively by: (i) generating favourable candidate decisions, and (ii) learning an approximation of the value functions by adding new cuts.

The *forward pass* generates candidate solutions by a user-defined number of Monte-Carlo samples, K. For each sample, we obtain a series of decisions over the entire horizon, $\hat{x}_i = (\hat{x}_{1,i}, \ldots, \hat{x}_{H,i})$, by solving (2.7) at each stage. We then obtain the corresponding total cost for each sample, z_i, and a statistical upper bound $\bar{z}$ can be computed as the sample average, $\bar{z} = \frac{1}{K}\sum_{i=1}^{K} z_i$.

On the other hand, the learning of value functions is achieved by a *backward pass*. The backward pass generates optimality cuts around the trial decisions obtained by the forward pass. The value function $V_{t,k}$ around $\hat{x}_{t,i}$ can be approximated by the dual multipliers $\pi_{t+1,j,i}$ and $\rho_{t+1,j,i}$, acquired from the solution of $NLDS_{t+1,j}$ for all $j \in \Omega_{t+1}$. The multipliers $\pi_{t+1,j,i}$ and $\rho_{t+1,j,i}$ are weighted according to the transition probability $\Pr(j|k)$ in order to obtain optimality cuts. These processes are illustrated in Figure 2.2. See [3] for a more detailed development of the theory.

For each Monte-Carlo sample, $H-1$ linear problems are solved in the forward pass. For each trial solution $\hat{x}_i$, the backward pass solves $\sum_{t=2}^{H} |\Omega_t|$ linear programs. This implies that with K Monte-Carlo samples, one needs to solve $K(H-2)+1$ linear problems in the forward pass (since the problem at $t=1$ is the same for all samples) and $K\sum_{t=2}^{H} |\Omega_t|$ linear problems in the backward

[2]In general, we also need to consider the feasibility cuts. However, since our model has production/load shedding as variables, which make the problem always feasible for any $\hat{x}_{t-1}$, feasibility cuts can be ignored.

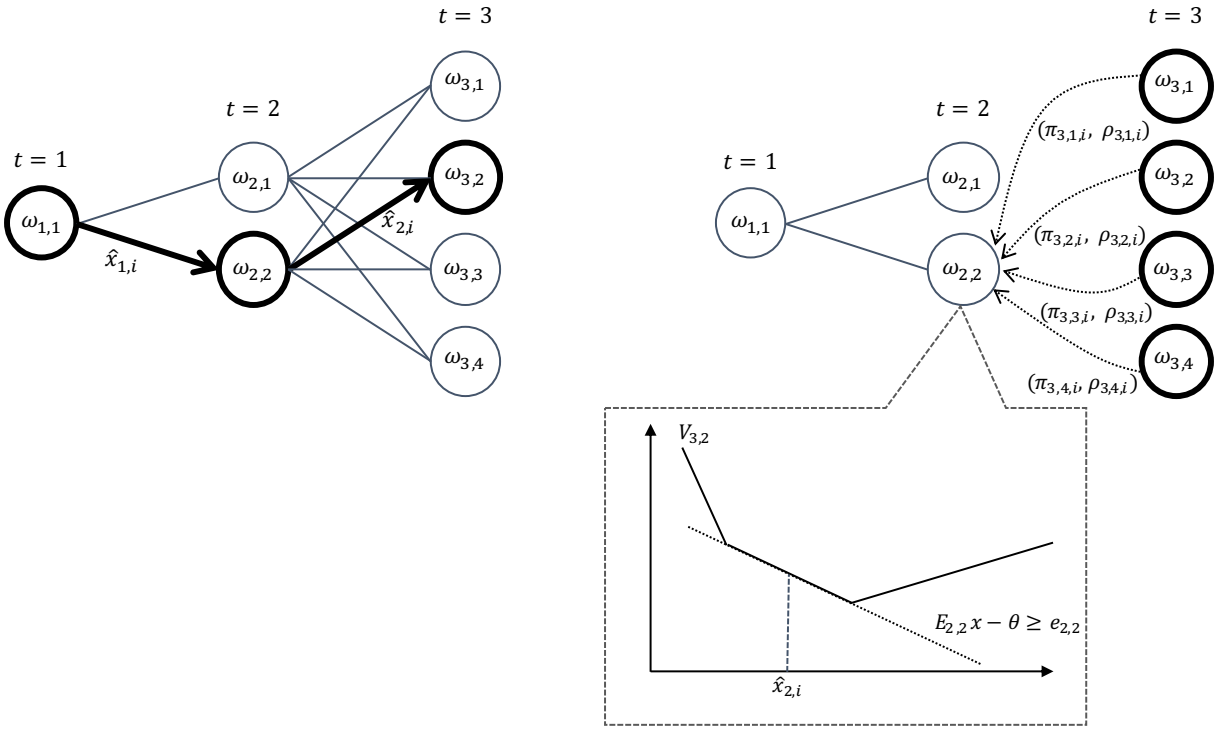


Figure 2.2: An example of forward pass (left panel) and backward pass (right panel). A trial solution is passed to the next $NLDS$ as an input. At the backward pass, the dual multipliers of $NLDS_{3,j}$ with the input $\hat{x}_{2,i}$ are computed for all $j \in \Omega_3$, and they generate the optimality cut $E_{2,2}x - \theta \geq e_{2,2}$. Note that at stage $H-1$, the optimality cuts are identical to the supporting hyper-plane of the value function, however, usually the cuts might be strictly below the value function due to the back propagation of underestimation of the value functions at the later stages.

pass. Therefore, the computational effect in SDDP is proportional to the number of Monte-Carlo sample K, which is an important parameter of the algorithm. Larger K guarantees a higher rate of learning, but restriction on increase in the number of linear problems that need to be solved, and an increase of size of each $NLDS$ (i.e. the number of cuts to be added to each $NLDS$).

Several termination criteria have been suggested for the SDDP algorithm, e.g. Pereira and Pinto [20], Shapiro [24], or a small standard deviation criterion whereby the algorithm is terminated only when the standard deviation of the mean cost estimated in the forward pass is small enough with respect to the lower bound[3]. However, here we apply the SDDP with a *fixed number of iterations* whereby the algorithm stops after M iterations of forward-backward passes with K Monte-Carlo samples per pass. In aggregate, the algorithm solves $MK\{(H-2) + \sum_{t=2}^{H} |\Omega_t|\} + M$ linear programs. After this learning process, we possess an approximation of the value functions, and this approximation can be used to obtain a dispatch which is feasible.

2.1.4 SDDP Features

SDDP is, in short, an algorithm which generates an estimation of value functions in an efficient way. Needless to say, we do not know the realization of uncertainties a priori when we implement SDDP. We run the algorithm with the discretized prediction of uncertainties in *offline* mode. For instance, suppose we want to develop a decision rule for tomorrow's optimal dispatch at every hour, from 0:00 to 23:00. Also suppose that at 12:00, we are provided with forecast data of

[3]See https://web.stanford.edu/∼lcambier/fast/tuto.php#std for more details

tomorrow's photovoltaic (PV) power and household demands, which are the stochastic parameters of the problem. On the day before (i.e. offline), we first create a lattice using this forecast information, and then implement the SDDP algorithm in order to obtain an approximation of the value functions. Then, being the following day, at every stage t (i.e. real-time, or *online*), we observe the realization of uncertainty $k \in \Omega_t$ and solve $NLDS_{t,k}$ with the previously obtained value functions.

The main part of the computation is performed offline, hence the online computation time will not be quite heavy in practical applications. However, the drawback of SDDP is that it is non-scalable with respect to the size of the network due to the increase of uncertainty and resources. In other words, it requires an unacceptable amount of time and memory in order to obtain promising approximations of value functions in a large-scale problem. Suppose one wants to solve an optimal dispatch problem in a distribution network with a few thousands of resources. Then, one might need to create a lattice with dozens or hundreds of outcomes at every stage in order to attain a reasonable result. Since the computational effort of the algorithm is proportional to the size of the lattice (i.e. the size of the horizon H and the number of outcomes at each stage, $|\Omega_t|$), the problem might never terminate within a time limit. In the previous example, the algorithm may need 30 hours to terminate, but one has to finish the computation at worst within 12 hours.

2.2 Model Predictive Control

Model Predictive Control (MPC), also known as receding horizon control is a feedback control technique that naturally incorporates optimization [16]. With MPC, an optimization problem is formulated and solved at each time stage in order to decide on a set of actions over a fixed time horizon. However, only the first stage (i.e. the current period) solution is applied to the system. Then, at the next time step, we repeat the planning process: we formulate a new optimization problem and solve it in order to execute the decision at that time. Concretely, at each time step $t \in T$ we

1. Predict: Predict the stochastic parameters for stages $t, \dots, H$[4], and build an optimization problem.
2. Optimize: Solve the optimization problem.
3. Execute: Apply the solution of stage t to the system.

Although this approach is not optimal, it can provide very good results in practice. There exist several formulations of MPC, in this work we adopt two formulations, *certainty-equivalent MPC* and *scenario-based robust MPC* [29].

2.2.1 Certainty-equivalent MPC

Certainty-equivalent MPC (ce MPC) is one of the simplest formulations of MPC. At each time step, ce MPC replaces the uncertainty by its prediction and solves an optimization problem over a predetermined horizon. Precisely, we repeat the following procedure at every stage $t \in T$:

1. Predict: Replace the stochastic parameters by the expected values, and build the following

[4]In general, one solves a problem for $t, \dots, H_t$ where H_t is a prediction horizon which can differ for each time stage t. In an infinite model, usually $H_{t+1} = H_t + 1$, i.e. the length of the prediction horizon is fixed. However, our interest is a fixed horizon model, in this work we set H as the horizon of MPC at any stage t, i.e. $H_t = H, t \in T$.

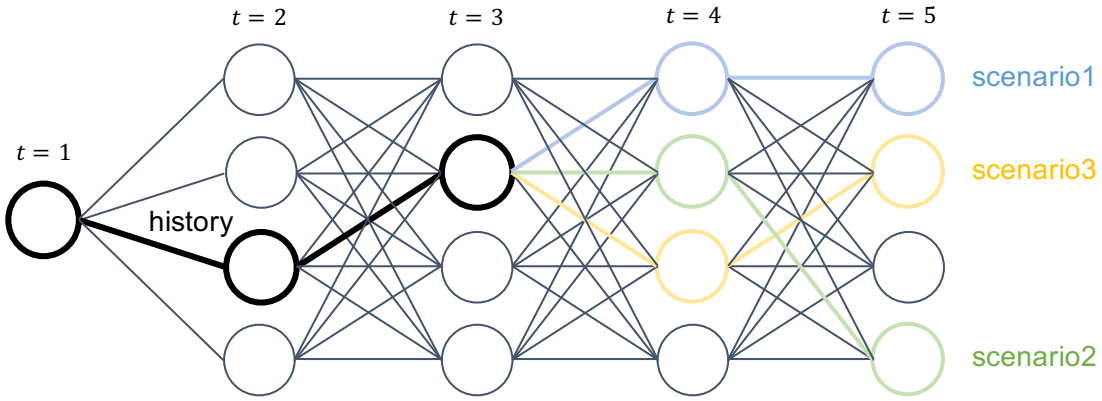


Figure 2.3: An example of scenario generation at stage $t = 3$ on a lattice with horizon $H = 5$. The history is depicted in black bold on the lattice and three scenarios are generated from $t = 3$ to $t = 5$. Obviously, at stage $t = 3$, all three scenarios must start from the the same outcome which has been just realized.

deterministic optimization problem.

$$\begin{aligned} \min_{x} \quad & \sum_{\tau=t}^{T} c_\tau^T x_\tau \\ \text{s.t.} \quad & W_\tau x_\tau = \tilde{h}_t - T_\tau x_{\tau-1}, \tau \in \{t, \dots, H\} \\ & x_\tau \geq 0, \tau \in \{t, \dots, H\} \end{aligned} \tag{2.8}$$

where $\tilde{\cdot}$ (tilde) represents the expectation of random parameters, and x_{t-1} is the previous-stage decision which is treated as a parameter in (2.8). Note that we keep the same assumption as before: of right-hand side uncertainty.

2. Optimize: Solve (2.8).

3. Execute: Apply the solution of stage t, x_t.

The advantage of MPC is that it inherently updates the information of the realization of random parameters. The expected value of a random parameter at stage $u(> t)$ is always more accurate at stage t than that of stage $t - 1$. Since (2.8) is a deterministic problem, the online computation time tends to be small. In contrast, ce MPC approximates the random parameters by their expected values, hence it might miss crucial information of some rare but "harsh" scenarios. The result of the algorithm can be poor if such scenarios materialize.

2.2.2 Scenario-based Robust MPC

Although we replace the uncertainty by its expectation in ce MPC, one might want to take into account some specific scenarios. This can be done by a *scenario generator*, which generates several scenarios according to the preference of the designer. Indeed, we can use the lattice model proposed in section 2.1.1 as a scenario generator. One can generate plausible scenarios by various techniques, e.g. Monte-Carlo sampling or uniform sampling, by sampling form the the lattice from the latest outcome. A graphical explanation with three scenarios is presented in Figure 2.3.

Given scenarios $s = 1, \dots, S$, the formulation of scenario-based robust MPC (sbr MPC) is

defined by minimizing the worst case scenario, i.e. solving the following optimization problem:

$$\begin{aligned} \min_{x} \quad & \max_{s=1,\dots,S} \sum_{\tau=t}^{H} \gamma^{\tau-t} (c_\tau^{(s)})^T x_\tau^{(s)} \\ \text{s.t.} \quad & W_\tau^{(s)} x_\tau = h_\tau^{(s)} - T_\tau^{(s)} x_{\tau-1}^{(s)}, s \in \{1,\dots,S\}, \tau \in \{t,\dots,H\} \\ & x_\tau^{(s)} \geq 0, s \in \{1,\dots,S\}, \tau \in \{t,\dots,H\} \\ & x_t^{(s)} \text{ is equal for all } s \in \{1,\dots,S\} \end{aligned} \tag{2.9}$$

where $\gamma \leq 1$ is a *discount factor* which discounts the cost of the future stages, superscript (s) represents the indexing of each scenario $s = 1,\dots,S$, and the last constraint assures that the solution at stage t, $x_t^{(s)}$, must be the same over all scenarios since $x_t^{(s)}$ needs to be executed as the current decision. The reason for introducing the discount factor γ is that sbr MPC might be too pessimistic in some problems. We can lessen this effect by setting a small γ in such cases. The full description of the algorithm is as follows. For each stage $t \in T$:

1. Predict: Generate S scenarios for stage $t,\dots,H$ on a lattice by a certain sampling technique, e.g. Monte Carlo sampling, uniform sampling.
2. Optimize: Solve (2.9).
3. Execute: Apply the solution of stage t, $x_t^{(1)}$. (Note that the solutions of other scenarios are also acceptable, i.e. $x_t^{(2)},\dots,x_t^{(S)}$.)

Increasing the number of scenarios S is more likely to produce conservative results. Meanwhile, increasing S leads to heavy online computation. In general, a real-time power system updates its dispatch every 15 minutes or one hour. One cannot wait for 20 minutes in order to obtain a result for the next operational interval which is going to be executed after 15 minutes. Hence, there is a trade-off: increasing the number of scenarios may provide a better solution, whereas it also escalates the computation time, and a system operator has to compromise between the two.

2.3 Discussion

In this section, we discuss advantages and disadvantages of each (traditional) uncertainty management policy. Table 2.1 provides a summary. Note that we assume that the horizon of the problem H is fixed, hence it is removed from the hyperparameters.

The SDDP algorithm is known to perform well in short-time operation problems e.g. [13]. The hyperparameters of SDDP are the number of Monte-Carlo sample K, the number of iterations M, and the size of lattice at each stage $|\Omega_t|$, each of these parameters affects the (mainly offline) computation time linearly.

On the other hand, MPC does not require offline computation, whereas it might need possibly heavy online computation. Ce MPC performs well in some small size problems, e.g. [29]. However, in a problem with a large number of resources and high volatility (such as the system that we will test in section 7.2), the result tends to be poor. Meanwhile, computational effort of sbr MPC is determined by the number of scenarios S to be generated (recall that we fix the prediction horizon to H for all stages). A small number of scenarios S requires less time to compute however it might produce a poor result due to a lack of information. The control of future importance can be done by setting of discount factor γ. $\gamma = 1$ implies that the future is treated in the same way as the current cost, whereas a small γ results in a greedier algorithm.

Table 2.1: Summary of each uncertainty policy

	Offline computation	Online computation	Hyperparameters	Scalability
SDDP	Heavy	Light	*K, the number of Monte Carlo sampling *M, the number of iteration *$\|\Omega_t\|$, the size of lattice (discretization precision)	No
ce MPC	-	Light	-	Yes
sbr MPC	-	Light-Heavy	*S, the number of scenarios *γ, the discount factor	No

Although SDDP and sbr MPC seem to be attractive policies, the common drawback is that neither is scalable to the size of the network. In spite of the fact that ce MPC scales linearly to problem size, the resulting dispatch can be poor. Therefore, it is necessary to consider smarter approaches which are scalable to arbitrary network size, while providing adequate results.

Chapter 3

Proposed Policy

In the previous chapter, we have confirmed that the traditional polices are not scalable to the size of distribution networks. In this chapter, we propose new dispatch policies which are applicable to systems of arbitrary size. The methods are based on the hierarchical approach as discussed in Chapter 1.

3.1 Decomposed SDDP

Due to the non-scalability of SDDP, we here consider a scheme that divides the radial distribution network into small pieces of sub-networks (we call them *layers*). The idea is originally inspired by an algorithm for solving a multi-stage stochastic linear programming (MSLP) by decomposition in time and each realization of random parameters (*nested decomposition*) [2]. The nested decomposition focuses on the fact that at a given stage, the cost of the future stages can be expressed in the value functions, hence the remaining stages are nested in the current stage problem. Therefore, the algorithm can be interpreted as a hierarchical approach in the sense that it nests the *temporal* evolution of MSLPs. On the contrary, we here extend the idea to the *spatial* evolution of the radial distribution network.

The advantage of decomposing the network is that in a radial network, each layer communicates only at a few interfaces, i.e. for every layer, there exist a single inflow from a higher layer (except the root layer) and a single outflow of power to possibly multiple lower layers (see Figure 3.1). The main idea of this policy is that at the offline step (e.g. at a day before), we first decentralize the network and build local lattices, in order to implement SDDP independently (and possibly in parallel) so as to obtain value functions. Then, at the online step, the decomposed network is reunited and we solve a centralized optimization problem with the previously obtained value functions.

3.1.1 Decomposition of the Network

We now describe our approach for decomposing the network and the global lattice. There are mainly three merits in our proposed approach: (i) we are able to solve a stochastic problem independently at each layer, (ii) layers communicate only at the interfaces, and (iii) this approach can be applied to network of arbitrary size.

Figure 3.1 illustrates the network decomposition. Due to the radial structure, the higher layer can treat the lower layer as one of the leaf nodes in its sub-network. In the figure, layer 1 corresponds to the higher level agent while layer 2 corresponds to the lower level agent. Although there might be some choices of how to decompose the network from the viewpoint of market

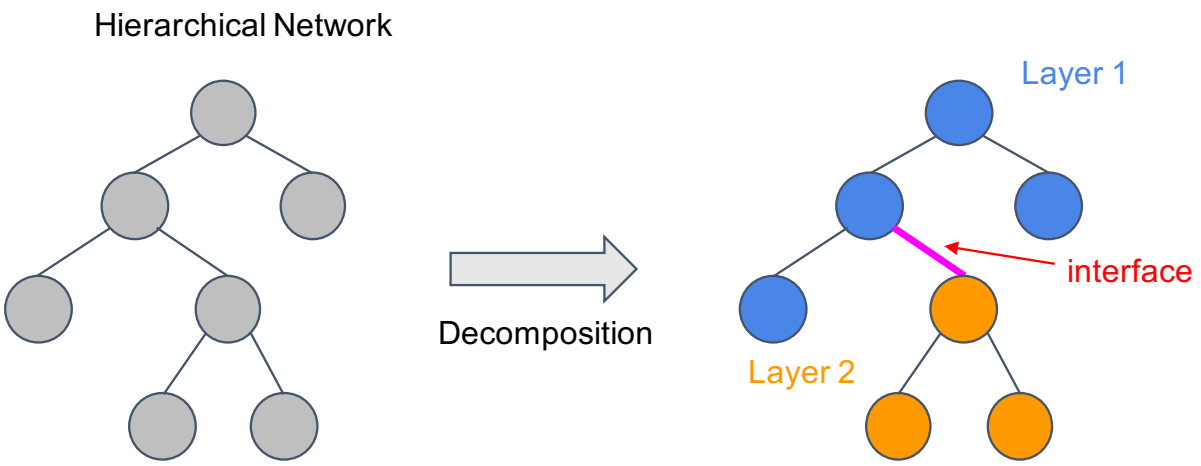


Figure 3.1: Decomposition of a radial network into two layers. The communication is done only at the interface, the line colored in pink.

design, we do not treat the topic here because it would be beyond the scope of this research. We assume that the decomposition itself is given in this work.

3.1.2 Decomposition of the Lattice

By decomposing the network, if a layer fixes the flow at its interfaces beforehand, then it can only consider the uncertainty of its own area. This fact leads to the decomposition of the lattice which describes the overall uncertainty in the network. Figure 3.2 explains the idea. The global lattice placed in the left side has 25 nodes at each stage $t \in \{2, ..., H\}$, from the pair $(\omega_t^1, \omega_t^2) = (1, 1)$ to (5,5), which is the Cartesian product of individual outcomes at every stage, i.e. $\Omega_t = \Omega_t^1 \times \Omega_t^2$. Instead the local lattices have 5 nodes at each stage as shown in the right panel of the figure.

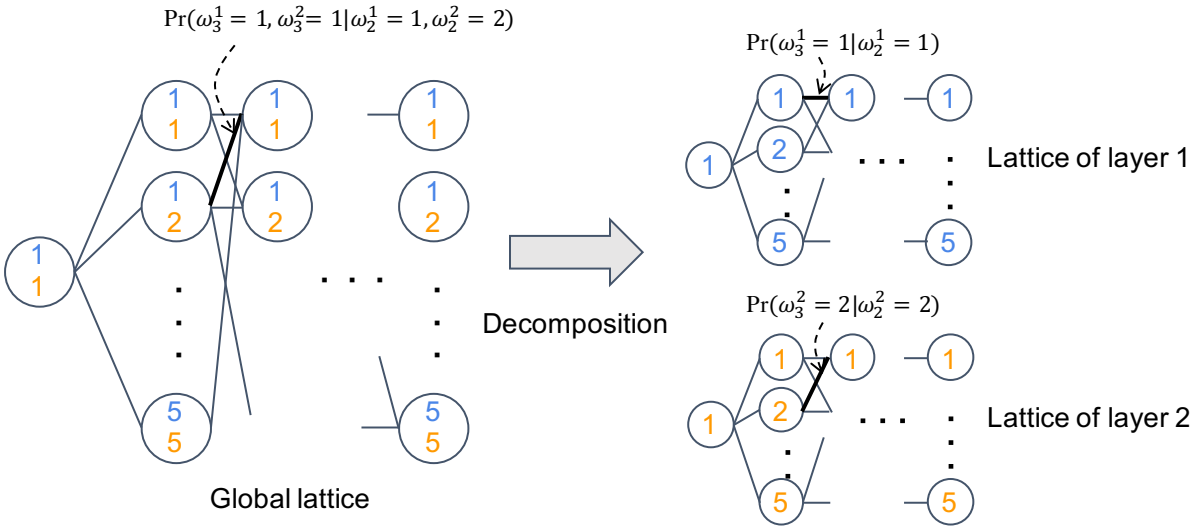


Figure 3.2: Decomposition of global lattice into a two-layer network. The global lattice has $25 (= 5^2)$ nodes, while each local lattice has 5 nodes at every stage.

Transition Probabilities. Since we approximate Ω with two sets, Ω^1 and Ω^2, there are "losses" of information. One of the losses of information corresponds to the transition probability of lattice.

The bold edge in the global lattice of Figure 3.2 corresponds to the transition from outcome $(1,2)$ at stage 2 to outcome $(1,1)$ at stage 3. Then, the corresponding transition probability can be described conditional on the previous outcome, i.e. $\Pr(\omega_3^1 = 1, \omega_3^2 = 1 | \omega_2^1 = 1, \omega_2^2 = 2)$ (note that subscripts represent the time indices while superscripts represent the layer indices). However, we cannot exactly use this joint probability but are only able to access its partial information in the local lattices. For the lattice of layer 1, this transition is $\omega_2^1 = 1 \to \omega_3^1 = 1$ and the agent no longer has the knowledge of transition in layer 2, $\omega_2^2 = 2 \to \omega_3^2 = 1$. Thus, the transition probability is defined as the marginal,

$$\begin{aligned}
\Pr(\omega_3^1 = 1 | \omega_2^1 = 1) = & \Pr(\omega_3^1 = 1, \omega_3^2 = 1 | \omega_2^1 = 1, \omega_2^2 = 1) \Pr(\omega_2^2 = 1) + \cdots \\
& + \Pr(\omega_3^1 = 1, \omega_3^2 = 5 | \omega_2^1 = 1, \omega_2^2 = 1) \Pr(\omega_2^2 = 1) \\
& + \cdots \\
& + \Pr(\omega_3^1 = 1, \omega_3^2 = 1 | \omega_2^1 = 1, \omega_2^2 = 2) \Pr(\omega_2^2 = 2) \\
& + \cdots \\
& + \Pr(\omega_3^1 = 1, \omega_3^2 = 1 | \omega_2^1 = 1, \omega_2^2 = 5) \Pr(\omega_2^2 = 5) + \cdots \\
& + \Pr(\omega_3^1 = 1, \omega_3^2 = 5 | \omega_2^1 = 1, \omega_2^2 = 5) \Pr(\omega_2^2 = 5) \\
= & \sum_{y_2 \in \Omega_2^2} \sum_{y_3 \in \Omega_3^2} \Pr(\omega_3^1 = 1, \omega_3^2 = y_3 | \omega_2^1 = 1, \omega_2^2 = y_2) \Pr(\omega_2^2 = y_2)
\end{aligned} \tag{3.1}$$

The true transition probability $\Pr(\omega_3^1 = 1, \omega_3^2 = 1 | \omega_2^1 = 1, \omega_2^2 = 2)$ is included in the aggregated probability. Similarly, the transition of layer 2, $\omega_2^2 = 2 \to \omega_3^2 = 1$, occurs with probability

$$\Pr(\omega_3^2 = 1 | \omega_2^2 = 2) = \sum_{x_2 \in \Omega_2^2} \sum_{x_3 \in \Omega_3^2} \Pr(\omega_3^1 = x_3, \omega_3^2 = 1 | \omega_2^1 = x_2, \omega_2^2 = 2) \Pr(\omega_2^1 = x_2) \tag{3.2}$$

In a multi-layer model, in which there exist dozens or hundreds of layers, it is impossible to define the global lattice due to the exponential growth of the cardinality of Ω. Nonetheless, we can build the local lattices if we know the joint distribution or have some historical data. For layer $l \in L$, the transition probability from outcome $j \in \Omega_t^l$ of stage t to outcome $k \in \Omega_{t+1}^l$ of stage $t+1$ can be expressed as follows:

$$\Pr(\omega_{t+1}^l = k | \omega_t^l = j) = \sum_{\omega_t^{-l} \in \Omega_t^{-l}} \sum_{\omega_{t+1}^{-l} \in \Omega_{t+1}^{-l}} \Pr(\omega_{t+1}^l = k, \omega_{t+1}^{-l} | \omega_t^l = j, \omega_t^{-l}) \Pr(\omega_t^{-l} = y_t^l) \tag{3.3}$$

where $\omega_\tau^{-i} \in \Omega_\tau^{-i}$ is the outcomes except for layer l at stage $\tau = t, t+1$. This process will be discussed in further detail in section 5.3.

Realization of Random Parameters. Another loss of information corresponds to the realization of random vector $\xi_t = h_t$ (as defined in section 2.1.1), which is originally defined on each outcome of the global lattice. Suppose that one can separate h_t by layers, i.e. $h_t^T = [(h_t^1)^T, \ldots, (h_t^{|L|})^T]$ for $t \in T$. Then, the realization of the random vector of the local lattice for layer $l \in L$ at stage $t \in T$, ξ_t^l, can be defined as follows:

$$\xi_t^l = \begin{bmatrix} h_t^l \\ \bar{b}_t^l \end{bmatrix} \tag{3.4}$$

where $\bar{b}_t^l$ refers to the estimation of the states of the other layers. We assume that the dimension of $\bar{b}_t^l$ is typically very small. That is to say, the original vector ξ_t is collapsed into a smaller size vector ξ_t^l consisting of (i) the realization in layer l, h_t^l, and (ii) vector $\bar{b}_t^l$, which contains condensed information of the other parts of ξ_t, $[(h_t^1)^T, \ldots, (h_t^{l-1})^T, (h_t^{l+1})^T, \ldots, (h_t^{|L|})^T]$. The

important fact is that the uncertainty that we need to consider in a given layer will be tractable since the size of the local random vectors ξ_t^l is much smaller than that of the original vector ξ_t.

In this study, the radial distribution network, the vector $\bar{b}_t^l$ refers to the *power flows at the interfaces*. The interface flows must be predetermined at every stage and every outcome of the local lattices. For instance, let us consider the local random vector for layer 1 of Figure 3.1, ξ_t^1. The layer has to deal with the uncertainty of the corresponding four nodes h_t^1, and the flow at the interface $\bar{b}_t^1$, i.e. $(\xi_t^1)^T = [(h_t^1)^T, (\bar{b}_t^1)^T]$. Thus, it can be regarded that the uncertainty of layer 2 is replaced by the flow at the interface. Note that the size of $\bar{b}_t^1$ is very small (just one dimension in this case).

The choice of $\bar{b}_t^l$ is crucial since a badly chosen $\bar{b}_t^l$ may lead poor results. Note that the right-hand side parameter h_t^l (e.g. RES production or demand) is not controllable, while $\bar{b}_t^l$ can be determined by an agent. For example, if there are adequate amounts of renewable energy in a layer, a given layer might want to inject the energy to its upper and lower layers, on the contrary, if the demand in a given layer is excessive, that layer may require its neighbors to supply energy. The value of $\bar{b}_t^l$ is user dependent, thus it is an important hyperparameter of the Decomposed SDDP.

3.1.3 Algorithm of the Decomposed SDDP

Offline Step. Here we assume that each layer $l \in L$ has an estimation of power flow at the interface edges $\bar{b}_t^l$ using some heuristics. This decouples every layer from the others, thus each SDDP can be implemented independently and in parallel. Note that each local lattice can have different interface flow values for each outcome of the lattice. The NLDS of layer $l \in L$, at each time $t \in T$ and each outcome $k \in \Omega_t^l$ can be described as follows:

$$\begin{aligned} NLDS_{t,k}^l(\hat{x}_{t-1}^l) : \min_{x^l} \quad & (c_{t,k}^l)^T x^l + \theta_{t,k}^l \\ \text{s.t.} \quad & W_t^l x^l = h_{t,k}^l - T_t^l \hat{x}_{t-1}^l \\ & A_t^l x^l = \bar{b}_{t,k}^l \\ & E_{t,k}^l x^l + \theta_{t,k}^l \geq e_{t,k}^l \\ & x^l \geq 0 \end{aligned} \tag{3.5}$$

where the first set of constraints express the dynamics and the linear conditions of the layer, while the second set of constraints guarantee that the flows at the corresponding interfaces of layer l, $A_t^l x^l$, must be fixed to predetermined values $\bar{b}_{t,k}^l$. This separation of constraints will be helpful for the online step description. Since we divide the network, a different value function (optimality cuts) is defined for each layer. This is indicated by the superscript l of $\theta_{t,k}^l$, the coefficients $E_{n,t,k}^l$ and the right-hand side parameters $e_{t,k}^l$.

Online Step. When dispatching the system in real-time, we use the optimality cuts which are obtained by (3.5). At this step, we solve a real-time dispatch problem with the all value functions in a centralized fashion. Thus, at stage $t \in T$, and each outcome of layer $l \in L$, $k^l \in \Omega_t^l$, the problem can be written as follows:

$$\begin{aligned} \min_{x} \quad & \sum_{l \in L} \left[(c_t^l)^T x^l + \theta_{t,k^l}^l \right] \\ \text{s.t.} \quad & W_t^l x = h_{t,k^l}^l - T_t^l \hat{x}_{t-1}^l, l \in L \\ & (A_t^l x^l)_i = (A_t^m x^m)_i, \text{ if layer } l \text{ and } m \text{ share interface } i \\ & E_{t,k^l}^l x^l + \theta_{t,k}^l \geq e_{t,k}^l, l \in L \\ & x^l \geq 0, l \in L \end{aligned} \tag{3.6}$$

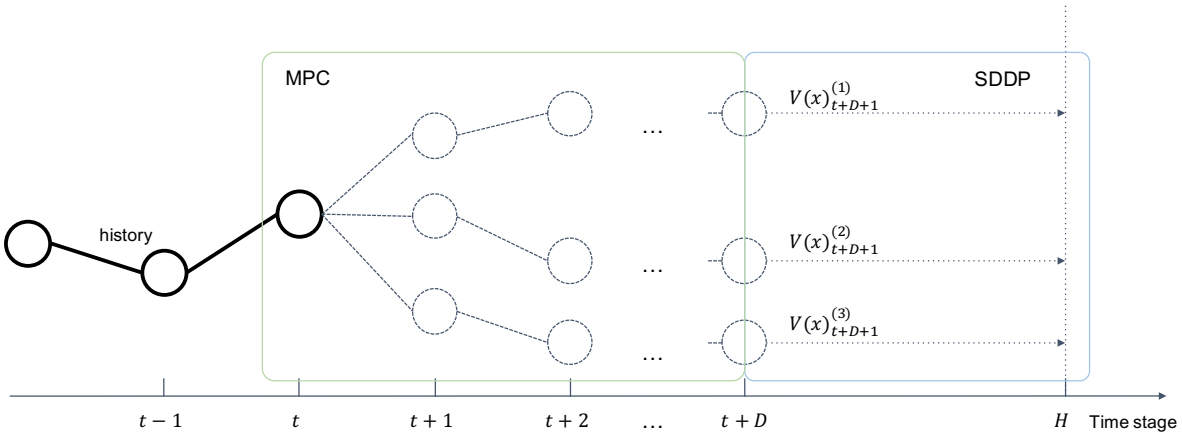


Figure 3.3: Illustration of Hybrid MPC. The three scenarios are generated until stage $t + D$ and each scenario has a corresponding value function which foresees the remaining stage cost until the end of the horizon H.

where $x^T = [(x_t^1)^T, \ldots, (x_t^{|L|})^T]$ are the variables combined for all layers, and the second set of constraints ensure that each interface flow must be the same for the layers which share the corresponding interface. That is to say, all interfaces flows are are no longer fixed, and each layer can exchange power since the system is reunited.

3.2 Hybrid MPC

The Decomposed SDDP algorithm accounts for uncertainty until the horizon H but there is some lack of global information caused by the lattice decomposition at the offline step. MPC can cope with uncertainty because it updates information at each online time step. However, as we discussed in section 2.3, the main drawback of sbr MPC is that it requires heavy online computation. By a proposed method we here describe, we try to overcome this problem by combining the two polices, sbrMPC and Decomposed SDDP. We refer to the resulting approach as the *Hybrid MPC* policy.

The main idea of Hybrid MPC is as follows. In an offline step, we first decompose the network and implement the Decomposed SDDP algorithm. Then, in online dispatch, at each time $t \in T$, we implement sbr MPC with a limited horizon size $D(< H)$ and S scenarios including the value functions at time $t+D$, that are obtained at the offline step. In this way, we tackle the uncertainty up to stage $t+D$ using an MPC, while the value functions are used for coping with the uncertainty from stage $t+D+1$ to the end of horizon H. Note that, at the ending stages ($t = H-D, \ldots, H$), since the value functions at the corresponding forecast horizon $t + D \geq H$ no longer exist, we switch to the usual sbr MPC without the value functions. The approach is illustrated in Figure 3.3.

An weakness of this approach is that the size of the value functions can increase substantially as the parameter S, the number of scenarios, grows. Even in a model with a small lookahead D and a small number of scenarios S, we might have to use a considerable amount of time in online computation. The reason is as follows. The value functions are defined at every stage and outcome. Thus, if two scenarios land to the same outcome at stage $t + D$, the value functions will be identical for both. However, since sbr MPC deals with each scenario independently, the number of variables and constraints will be proportional to the number of scenarios. Hence, the size of the dispatch problem also increases proportionally to the number of scenarios, even if some scenarios share a common outcome in the lattice. This facts implies that the possibility of a significant increase of online computation time. The parameters D and S might affect the

performance of Hybrid MPC and we will test it by the case studies in Chapter 7.

3.3 Discussion

Table 3.1 summarizes the proposed dispatch polices. Thanks to the decomposition of the network, both polices are now scalable to an arbitrary size of network. The Decomposed SDDP requires in addition to the usual hyperparameters of SDDP, the data of fixed flow at the interfaces $\bar{b}$. Note that the SDDP settings can be different for each individual SDDP, i.e. for a layer which has a small number of stochastic resources, one might require fewer iterations K and Monte-Carlo samples M compared to another layer which is composed of more sources of uncertainty. The Hybrid MPC uses the value functions that are obtained by Decomposed SDDP during online dispatch. Hence, it requires the same amount of offline computation as Decomposed SDDP. The online computation time is determined by the hyperparameters of sbr MPC, the foresight step size D and the number of scenarios S. Furthermore, the discount factor γ allows an operator to reduce the importance of future costs.

Table 3.1: Summary of the proposed dispatch policies. Note that Dec. SDDP refers to the Decomposed SDDP.

	Offline Computation	Online Computation	Hyperparameters	Scalability
Dec. SDDP	Heavy	Light	*K, the number of Monte Carlo sampling *M, the number of iterations *$\|\Omega_t^l\|$, the size of local lattice *$\bar{b}^l$, the fixed flow at interfaces	Yes
Hybrid MPC	Heavy	Moderate	*K, the number of Monte Carlo sampling *M, the number of iterations *$\|\Omega_t^l\|$, the size of local lattice *$\bar{b}^l$, the fixed flow at interfaces *γ, the discount factor *D, the foresight step size *S, the number of scenarios	Yes

Chapter 4

Distribution System Model Description

In the previous two chapters, we have discussed the general formulations of the dispatch polices. In this chapter, we introduce the detailed formulations of the storage model that we will apply in the case studies of Chapter 7. We adopt three different models which we refer to the Standard Storage model, the Injection Limit model, and the Battery Deadline model. We describe the stochastic program that we are interested in solving by the nested L-shaped decomposition sub-problem ($NLDS$) formulations. We consider the following problem for a given stage $t \in T$ and outcome $k \in \Omega_t$. We drop the t and k indices in order to lighten notation. The nomenclature is summarized in Appendix A.

4.1 Standard Storage Model

The first model, which we refer to the Standard Storage model, considers the simplest linear power flow. We denote the set of network nodes as N, the set of lines as I. We assume that at every node $n \in N$ there exists a battery storage, power demand, a renewable energy resource, and a set of generators G_n which is possibly empty at some nodes if there do not exist any generators.

The objective is to minimize the cost caused by the production of power and load shedding.

$$\min \sum_{n \in N, g \in G_n} MC_g \cdot p_g + \sum_{n \in N} VOLL \cdot ls_n + \theta \tag{4.1}$$

where p_g is the production of generator $g \in G_n$, ls_n is the amount of load shedding at node $n \in N$, MC_g is marginal cost of generator $g \in G$, $VOLL$ refers to the value of lost load, and θ is the value function which captures the (approximate) information of remaining stage costs.

The power balance constraints can be expressed as follows.

$$\sum_{g \in G_n} p_g + \sum_{m \in C_n} f_m + bd_n - bc_n + ls_n - ps_n - f_n = ND_n, n \in N \tag{4.2}$$

where C_n is the set of child nodes of node n, f_n/f_m is the power flow which arrives at the unique ancestor of node n/m, A_n/A_m (here $A_m = n$), bd_n represents the power supply from discharging a battery, bc_n is the power of battery charging, ps_n corresponds to production shedding[1], and

[1]The production shedding ps_n is introduced in order to make the problem always feasible. Indeed, not having the variable, the problem with the injection limit constraints (4.18) and (4.19) can be infeasible if there exists a large amount of renewable production (i.e. a huge negative net demand).

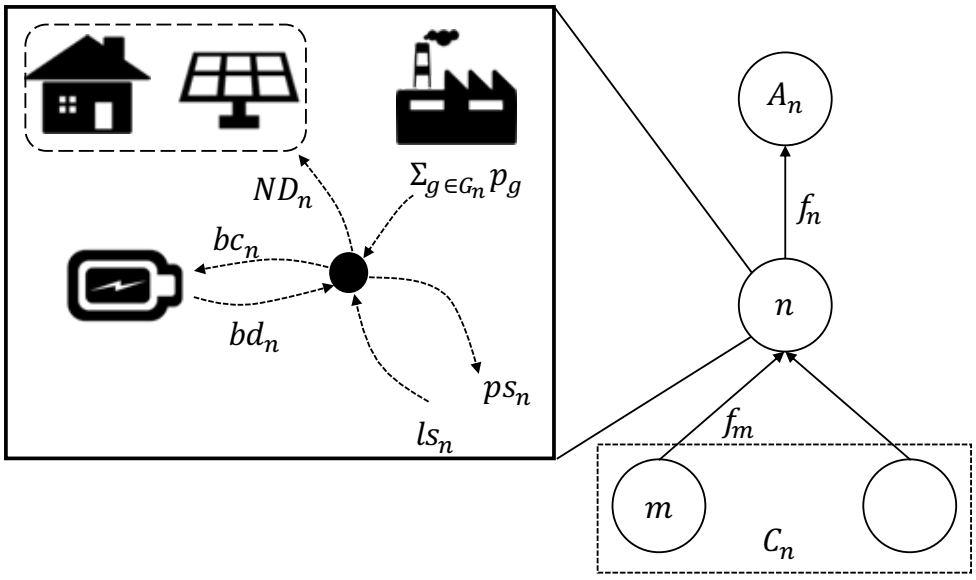


Figure 4.1: Power balance at a typical node.

ND_n is the net demand which is defined as the difference between demand and renewable energy production. Note that edge n is defined as the line between node n and the unique ancestor A_n. The graphical illustration is presented in Figure 4.1. The straight arrows indicate the power flows between adjacent nodes while the dotted line arrows denote the injection and withdrawal of power from a node.

The battery charge dynamics are expressed as follows.

$$s_n - \eta_n \cdot bc_n - \frac{bd_n}{\mu_n} = \hat{s}_{n,t-1}, n \in N \tag{4.3}$$

where s_n is the storage level of a battery at node n, whereas $\hat{s}_{n,t-1}$ refers to the previous-stage decision (or the initial condition at stage 1), hence it is treated as a parameter in this problem. $\eta_n (< 1)$ represents the efficiency of charging the battery while $\mu_n (< 1)$ is the efficiency of discharging. Note that the storage variable s_n is a *state variable* of this problem, hence it influences the evolution of the system.

The following inequalities are the physical capacity constraints for each variable.

$$f_i \leq T_i, i \in I \tag{4.4}$$
$$-f_i \leq T_i, i \in I \tag{4.5}$$
$$s_n \leq S_n, n \in N \tag{4.6}$$
$$bc_n \leq BC_n, n \in N \tag{4.7}$$
$$bd_n \leq BD_n, n \in N \tag{4.8}$$
$$p_g \leq PMax_g, g \in G_n, n \in N \tag{4.9}$$
$$-p_g \leq -PMin_g, g \in G_n, n \in N \tag{4.10}$$

where T_i refers to the flow capacity of line $i \in I$, S_n represents the maximum storage capacity of battery n, BC_n is the charging capacity of the battery, BD_n is the discharging capacity of the battery, $PMax_g$ is the production capacity of generator g, and $PMin_g$ is the technical minimum of generator g.

The optimality cuts are expressed in the following form.

$$\sum_{n \in N} E_{n,j} s_n + \theta \geq e_j, j = 1, \ldots, q \tag{4.11}$$

where $E_{n,j}$ is the coefficient of battery n in the jth optimality cut ($j = 1, \ldots, q$), and e_j is the corresponding right-hand-side parameter. Note that at stage H, the final stage, we do not have any optimality cuts, thus $q = 0$.

Finally, we impose non-negativity constraints.

$$s_n \geq 0, n \in N \tag{4.12}$$
$$bc_n \geq 0, n \in N \tag{4.13}$$
$$bd_n \geq 0, n \in N \tag{4.14}$$
$$p_g \geq 0, g \in G_n, n \in N \tag{4.15}$$
$$ls_n \geq 0, n \in N \tag{4.16}$$
$$ps_n \geq 0, n \in N \tag{4.17}$$

4.2 Injection Limit Model

In the Standard Storage model, we completely ignore the non-linearity of the distribution network. Although we are working with a linear model of power flow, we want to somehow approximate this feature. For this purpose, we add the following new constraints which are named injection limit constraints.

$$\sum_{g \in G_n} p_g + ls_n + bd_n - ps_n - bc_n - ND_n \leq IMax_n, n \in N \tag{4.18}$$

$$-\left(\sum_{g \in G_n} p_g + ls_n + bd_n - ps_n - bc_n - ND_n \right) \leq -IMin_n, n \in N \tag{4.19}$$

where $(\sum_{g \in G_n} p_g + ls_n + bd_n - ps_n - bc_n - ND_n)$ represents the injection at node n, $IMax_n$ is the upper bound on the injection, and $IMin_n$ is the lower bound on the injection. The injection can be regarded as the sum of all dotted line arrows in Figure 4.1. The constraints impose the limit on the amount of injection that a node can take in or out, and can weakly express the voltage constraints.

4.3 Battery Deadline Model

In the low voltage distribution network, e.g. at the household level, battery storage might correspond to an electric vehicle (EV). This fact motivates us to build another formulation of the problem. In the previous models, the storage is used to cover future demands. However, when the storage is an electric vehicle, it is necessary to be ready to leave at a given departure time. This feature introduces a new deferrable demand. The storage variable will disappear after the deadline, i.e. after the EV departs. Assume that each EV is characterized by a different departure time, and that we know the deadlines beforehand. Moreover, suppose that the injection limit constraints remain the same as before.

The following battery storage deadline constraints should be added to the $NLDS$.

$$s_n + ss_n = R_n, n \in N_{t,\text{deadline}} \tag{4.20}$$
$$ss_n \geq 0, n \in N_{t,\text{deadline}} \tag{4.21}$$

where R_n refers to the required storage level of electric vehicle n, ss_n is a non-negative slack variable that represents the lack of energy from a given required level, and $N_{t,\text{deadline}}$ is the set of nodes whose deadline correspond to stage t, i.e. the set of EVs which depart at this stage. Note that the constraints (4.20) and (4.21) only appear at stage t.

The objective function needs to be changed as follows.

$$\min \sum_{n\in N, g\in G_n} MC_g \cdot p_g + \sum_{n\in N} VOLL \cdot ls_n + \sum_{n\in N_{t,\text{deadline}}} VOLS \cdot ss_n + \theta \tag{4.22}$$

where $VOLS$ is the value of lost storage which corresponds to the cost induced by not covering the storage requirement.

The state variable of the problem, battery storage variable s_n, will decrease in time since each EV departs after the corresponding deadline. Therefore, the optimality cuts, which are composed of the state variables, must be modified as follows:

$$\sum_{n\in N_{t+1,\text{active}}} E_{n,j} s_n + \theta \geq e_j, j = 1, \ldots, r \tag{4.23}$$

where $N_{t+1,\text{active}}$ is the set of nodes where the battery storage is active at stage $t+1$ (i.e. set of nodes that there still exist EVs). The optimality cuts at stage t are composed of the storage variable s_n that will be available during stage $t+1$. That is to say, the batteries which disappear before stage t (EVs that have departed by stage t) do not affect the cost from stage $t+1$ onwards, and we do not have to keep track of those batteries.

Chapter 5

Stochastic Modeling of Net Demand in Distribution Network

In order to evaluate our polices by numerical experiments, it is necessary to use some data of stochastic parameters (e.g. RES production or demand) of the distribution network. A number of methods have been proposed for generating a stochastic model of RES, e.g. for PV power [23] and wind power [17]. However, generating the model from real data is challenging topic, and out of our scope. Therefore, we make some assumptions on a stochastic model in this work. In this chapter, we present the modeling of the stochastic process of net demand (defined as the difference between demand and PV production), which will be applied to the case study in Chapter 7. We need to account for the following features of net demand.

- The PV production at one location should be spatially correlated to that in another location.
- The PV production should be auto-correlated over time.

We will employ the idea of copula model in order to satisfy these requirements. In section 5.1, we introduce the theory of copulas, and in section 5.2 we propose an algorithm for generating the stochastic model of net demand. Finally, in section 5.3 we present a method for building lattices that are used in our simulation using the data.

5.1 Copula Model of Multi-Dimensional Stochastic Process

In order to generate samples from a multi-dimensional stochastic model, one generally requires a multi-dimensional cumulative distribution function (CDF) or probability density function (PDF). However, consider generating samples of 500 dimensional correlated data. In order to sample using Monte-Carlo or rejection sampling, a joint CDF with 1000 discretized values for each dimension requires storing 1000^{500} value. Obviously, it is impossible to store such a CDF in normal computer memory. Nevertheless, our interest is to analyze policies on a hierarchical distribution network which has a correlated multi-dimensional stochastic process. A *copula* is suitable for generating such data. In the following method, we decompose a multivariate CDF into: (i) a copula, and (ii) univariate marginal CDFs. In short, copulas are functions that describe dependencies among variables, and provide a way for creating distributions in order to model correlated multivariate data.

Let us consider a random vector $(X_1, \ldots, X_d)$ which follows the continuous d-dimensional CDF F. Assume that its marginal CDFs are denoted as $F_1, \ldots, F_d$. We exploit the following theorem.

Theorem 5.1 (Probability integral transform)**.** *If X is a random variable with continuous CDF F_X, then the random variable Y, defined as*

$$Y = F_X(X),$$

follows a uniform distribution $U(0,1)$, i.e.

$$\Pr(F_x(X) \leq y) = y, y \in [0,1].$$

Proof. See Appendix B. □

This theorem implies that *any given continuous distribution* can be converted to a random variable that has a standard uniform distribution. Applying this theorem to the marginal CDFs $F_1, \ldots, F_d$, we obtain the following new random vector

$$(U_1, \ldots, U_d) = (F_1(X_1), \ldots, F_d(X_d))$$

which has uniformly distributed marginals. The copula $C : [0,1]^d \to [0,1]$ of $(X_1, \ldots, X_d)$ is defined as the joint CDF of $(U_1, \ldots, U_d)$, i.e.

$$C(u_1, \ldots, u_d) = \Pr(U_1 \leq u_1, \ldots, U_d \leq u_d)$$

Note that copula C contains all the information on the dependence structure between the elements of the original random vectors $(X_1, \ldots, X_d)$, whereas the marginal CDFs $F_1, \ldots, F_d$ contain all the information on the marginal distributions.

Using this approach, we can generate samples from *any multivariate distribution function* by reversing the above procedure. Indeed, Sklar showed that there exists a copula C for any joint CDF F and its marginals $F_1, \ldots, F_d$, and the converse is also true [25]. That is to say, suppose we have a procedure to generate a sample $(U_1, \ldots, U_d)$ from the copula distribution, then a sample of d-dimensional process with the marginal distributions $F_1, \ldots, F_d$ can be generated as:

$$(X_1, \ldots, X_d) = (F_1^{-1}(U_1), \ldots F_d^{-1}(U_d))$$

where F_i^{-1} is the inverse of F_i.

One possible example of the application of copulas is as follows. Suppose that we want to generate a correlated sample (x_1, x_2), and that we have its marginal CDFs F_1 and F_2. Then, we first generate correlated samples z_1 and z_2 from a bivariate Gaussian distribution.We transform z_1 and z_2 using Gaussian CDF Φ_i, i.e. $u_1 = \Phi_i(z_1)$ and $u_2 = \Phi_i(z_2)$. Now, we possess a sample (u_1, u_2) supported on $[0,1] \times [0,1]$. Finally, we transform (u_1, u_2) with the inverse of marginal CDFs F_1 and F_2, and obtain the required correlated sample (x_1, x_2), i.e. $x_1 = F_1^{-1}(u_1)$ and $x_2 = F_2^{-1}(u_2)$. This last step is called *inverse transform sampling*. This process is an instance of applying a *Gaussian copula* since we use a Gaussian distribution for generating z_1 and z_2. This process is depicted in Figure 5.1. The copula of this example is the joint CDF which supports (u_1, u_2).

In the previous example, we decompose a multi-dimensional CDF which supports (x_1, x_2) by (i) the bivariate Gaussian distribution, and (ii) the marginal CDFs F_1 and F_2. A (multi-dimensional) Gaussian distribution can be characterized by a small number of parameters, mean and covariance matrix, and several numerical computation techniques have been proposed for generating a multi-dimensional Gaussian sample, e.g. [15]. On the other hand, the marginal CDFs only contain the corresponding one dimensional values, which are much smaller than the original CDF. Therefore, we no longer have to save the original function, and can overcome the memory storage problem.

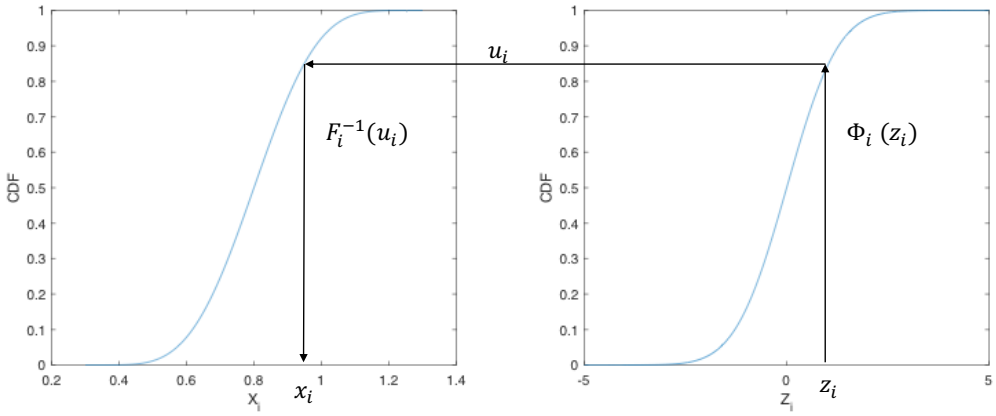


Figure 5.1: Process of applying a Gaussian copula to generate correlated samples.

5.2 Illustrative Example

5.2.1 Algorithm for Generating Net Demand Process

We use the copula method for modeling of net demand. The idea is inspired by [17]. The following algorithm generates S samples of net demand process, with the following assumptions:

- We apply the Gaussian copula. (Characterizes the geographical correlation.)
- PV power process follows an ARMA(1,1) model. (Characterizes the temporal correlation.)
- At a given stage and a given location, we know the upper and lower bound of PV power, and that the shape of the distribution of PV power follows a Beta distribution $I(\cdot; a, b)$ (a and b are the parameters of the distribution).
- We considers just two locations A and B.

It should be noted that the approach can easily be generalized to multiple locations. The concrete algorithm is as follows.

Step 1: For $t = 1$, let z_1^A and z_1^B be a sample at $t = 1$ from a joint Gaussian distribution $\mathcal{N}(0, \Sigma)$, with zero mean and the covariance matrix Σ:

$$\Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

where ρ is a correlation coefficient, with a value between from -1 and $+1$. $\rho = 1$ refers to the perfect positive correlation, 0 implies no correlation, and -1 represents perfect negative correlation. Note that the variances are assumed to be equal, i.e. $\sigma_A^2 = \sigma_B^2 = 1$ in this example, and $\rho = \mathrm{Cov}_{A,B} = \mathrm{Cov}_{B,A}$ by definition. Let $\epsilon_1 = z_1 = [z_1^A, z_1^B]$.

Step 2: For $t = 2, ..., H$, use an ARMA(1,1) model to generate z_t^A and z_t^B:

$$z_t^A = \phi^A z_{t-1}^A + \epsilon_t^A + \theta^A \epsilon_{t-1}^A$$
$$z_t^B = \phi^B z_{t-1}^B + \epsilon_t^B + \theta^B \epsilon_{t-1}^B$$

where ϕ and θ are the parameters of the ARMA model and ϵ_t is the error that is sampled from the previously defined distribution $\mathcal{N}(0, \Sigma)$. In this way, $z_t = (z_t^A, z_t^B)$ can exhibit temporal dependencies and spacial dependencies.

Step 3: Repeat step 1 and step 2 S times. We thus obtain S samples of the z process: $z^{i,A} = [z_1^{i,A}, ..., z_H^{i,A}]$ and $z^{i,B} = [z_1^{i,B}, ..., z_H^{i,B}]$ for $i = 1, ..., S$. Compute the mean and standard deviation of these processes at each location and time, μ_t^P and σ_t^P $(t \in T, P \in \{A, B\})$.

Step 4: (Probability integral transform.) z_t^P follows a Gaussian distribution $\mathcal{N}(\mu_t^P, (\sigma_t^P)^2)$. From Theorem 5.1, we can obtain samples of u_t^P, which follows a uniform distribution on [0,1]:

$$u_t^{i,P} = \Phi_t^P(z_t^{i,P})$$

where Φ_t^P is the CDF of a Gaussian distribution with mean μ_t^P and variance $(\sigma_t^P)^2$ for $t \in T, P \in \{A, B\}$.

Step 5: (Inverse transform sampling.) Obtain process x by transforming u using the inverse of the Beta distributions.

$$x_t^{i,P} = I_t^{P^{-1}}(u_t^{i,P}; a, b)$$

where I_t^P is the CDF of Beta distribution at time $t \in T$ at location $P \in \{A, B\}$, and a and b are parameters of the distribution.

Step 6: For $t = 1, ..., H$, denote lower bound and upper bound of net demand at time t at location P, by LB_t^P and UB_t^P, respectively. Then, the net demand ND can be computed as:

$$ND_t^{i,P} = x_t^{i,P} \times (UB_t^P - LB_t^P) + LB_t^P,\ t \in T, P \in \{A, B\}, i = 1, ...S.$$

5.2.2 Numerical Example

Here we describe a numerical example of the previously proposed algorithm. As before, we assume that there are two locations A and B to consider. Furthermore, we set the model parameters as follows.

- Sample size: $S = 10000$.
- Problem horizon: $H = 24$.
- Correlation coefficient ρ in Σ: $\rho = 0.6$.
- ARMA(1,1) model: $[\phi_A, \phi_B] = [0.6, 0.6]$ and $[\theta_A, \theta_B] = [0.8, 0.8]$
- Beta distribution: $a = b = 6$, identical at each time and location.
- Lower bound and upper bound of net demand are shown in Table 5.1:

Table 5.1: Upper bound and lower bound at each location over time

	A			B		
	stage 1-6	7-18	19-24	stage 1-6	7-18	19-24
UB (MW)	0.533	-0.020	0.800	0.356	-0.013	0.533
LB (MW)	0.267	-0.200	0.533	0.178	-0.133	0.356

It should be noted that the above parameters are used in the case study of Chapter 7 except for the upper bound and lower bound values.

Step 1-Step 3: Generation of z Process. By following steps 1 to 3 of section 5.2.1, we obtain the (Gaussian) process z. Figure 5.2 shows the histogram of 10000 samples of the z process at location A at four different time stages. Note that z follows a Gaussian distribution $\mathcal{N}(\mu_t^P, (\sigma_t^P)^2)$ with mean μ_t^P and variance $(\sigma_t^P)^2$ by definition, and the corresponding probability density functions (PDF) are the orange curves in the figure. Figure 5.3 presents the temporal and spatial dependency of z.

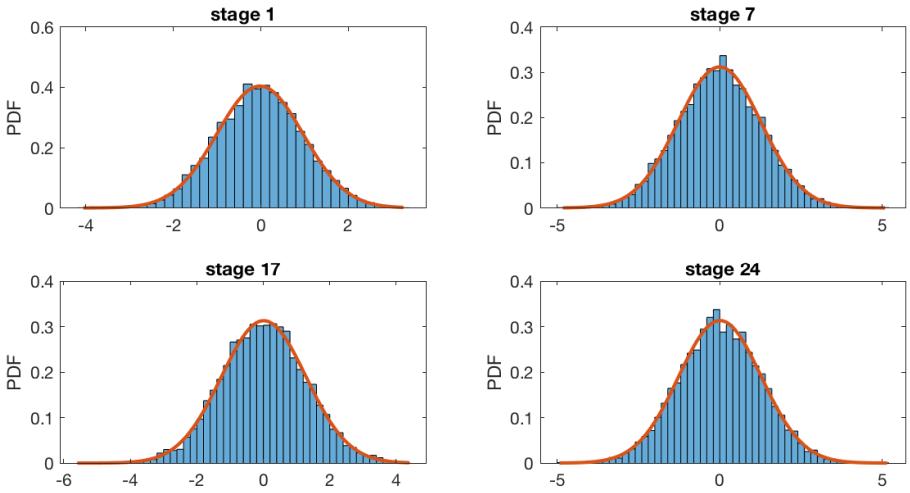


Figure 5.2: Histogram of z_t^A at $t = 1$, 7, 17 and 24. Orange curves are the PDFs of $\mathcal{N}(\mu_t^A, (\sigma_t^A)^2)$.

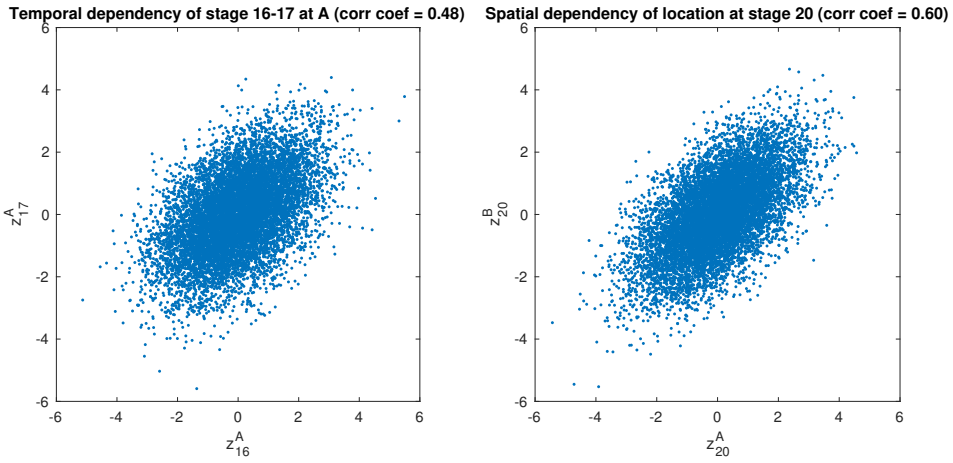


Figure 5.3: Left panel: Temporal dependency at location A, from stage 16 to stage 17. Right panel: Spatial correlation at stage 20.

Step 4: Probability Integral Transform of Process z. As discussed in Step 4 of section 5.2.1, using the marginal CDFs of the Gaussian distribution in order to transform process z, we obtain the samples of uniform distribution u. The distribution over u at each stage is the copula of this algorithm. It can be confirmed by Figure 5.4 which presents the transformed samples u at stage 1, 7, 17 and 24. Meanwhile, each dimension of u must follow a uniform distribution on [0,1] as shown by Theorem 5.1. This can be observed in Figure 5.5 which depicts the histogram of the transformed samples u at location A at stage 1, 7, 17 and 24. We note that we now posses the samples u from the copulas.

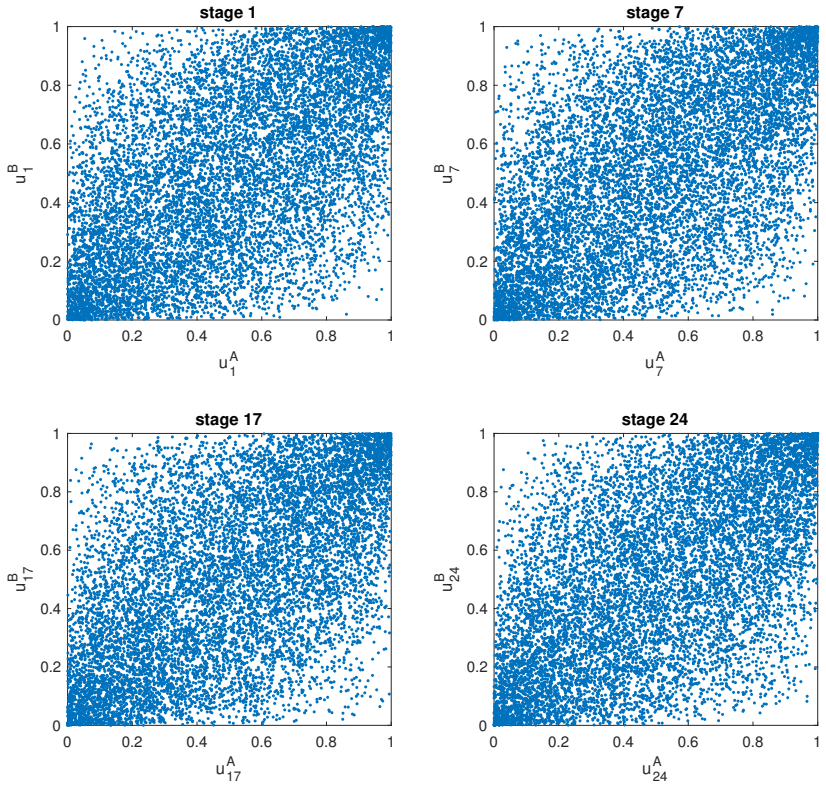


Figure 5.4: Gaussian copulas of (u_t^A, u_t^B) at stage 1, 7, 17 and 24.

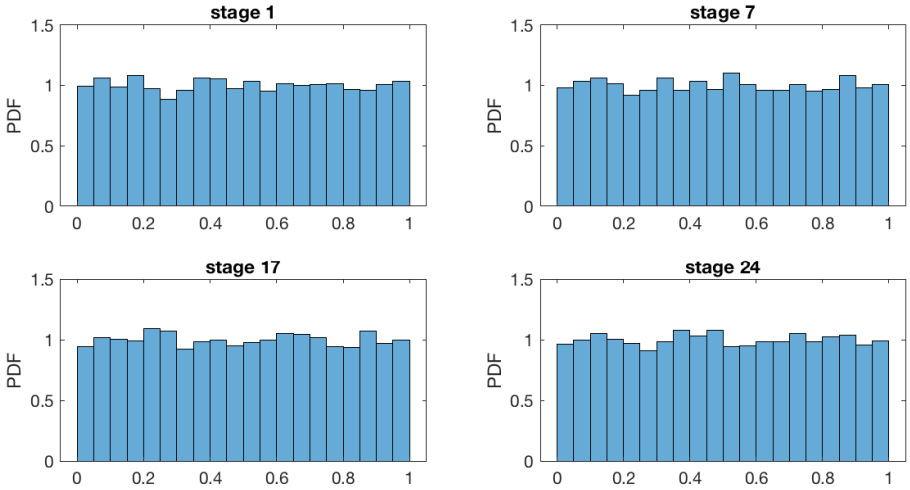


Figure 5.5: Histogram of u_t^A at stage 1, 7, 17 and 24.

Step 5-6: Inverse Transform Sampling with Scaling and Shifting. We are now able to apply the inverse transform sampling to process u. Taking the inverse CDF of $Beta(6, 6)$, we obtain process x from u. Then we scale and shift x in order to obtain net demand values, as shown in Step 6 of section 5.2.1. The histogram of net demand at location A is presented in Figure 5.6. The figure shows the bell curve of the Beta distribution which is bounded by

the upper and lower limit[1]. On the other hand, Figure 5.7 presents the auto-correlation and spacial correlation of net demand. The two correlation coefficients are sufficiently large and we can conclude that the process exhibits temporal and spacial dependency. Finally, Figure 5.8 illustrates 100 net demand samples at each location. The red lines refers to the means, and are situated just in the middle of the bounds by construction[2].

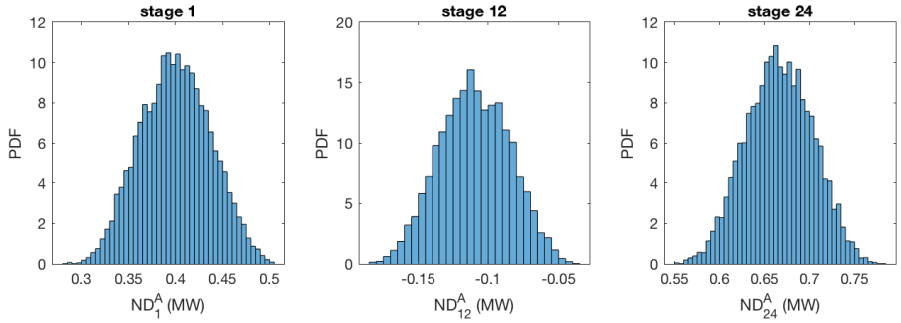


Figure 5.6: Histogram of net demand at location A at stage 1, 12, and 24.

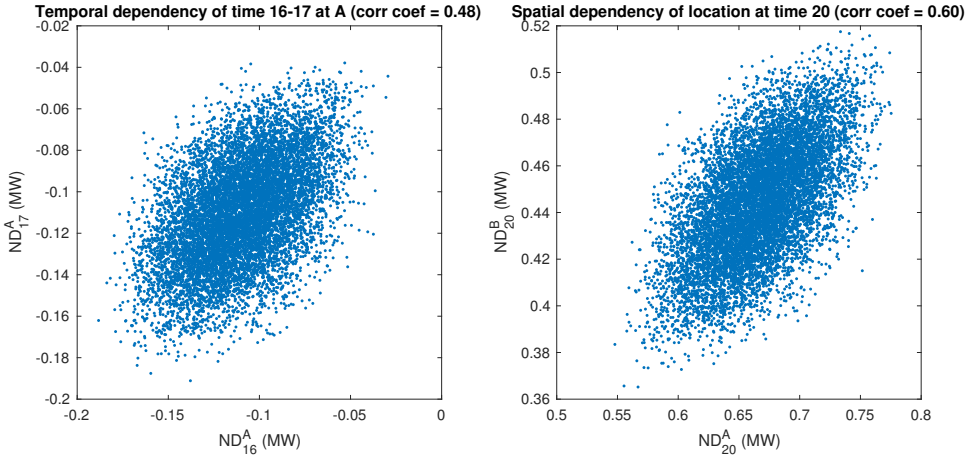


Figure 5.7: Auto-correlation of ND^A at stage 16 (left), correlation of ND^A and ND^B at stage 20 (right)

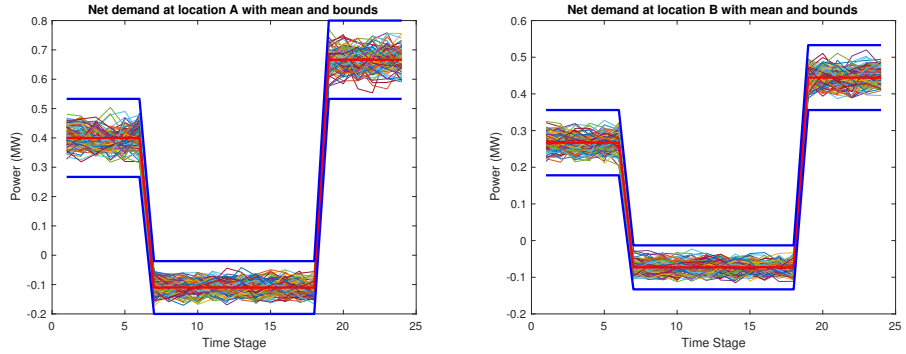


Figure 5.8: 100 net demand scenarios for location A (left) and B (right). Blue lines refers to the bounds while the red lines are the mean values.

[1]Note that these shapes are different from those of Gaussian which are supported on the whole real line by definition (see Figure 5.2).

[2]Symmetry of $Beta(6, 6)$

5.3 Building Local Lattices of Net Demand Process for Decomposed SDDP

Once we obtain the data, we can now build the lattice of net demand ND. Let us denote S as the number of samples of net demand data obtained by the previously proposed algorithm. As discussed in section 3.1, each layer possess a local lattice which is composed of the values of net demand of its own nodes and fixed flows at the corresponding interfaces. Recall that L is the set of layers, and N_l refers to the set of nodes in layer $l \in L$. Denote V as the number of outcomes at each time step except the first stage for every local lattice. We determine the value of net demand associated to every outcome of local lattice $l \in L$ using the following process.

1. For $t = 1$, take the average value of S samples of ND_1^n for each location $n \in N_l$.
2. For every time stage $t = 2, \ldots, H$,
   (a) Sort the data by the sum of net demand data $\sum_{n \in N_l} ND_t^n$.
   (b) Discretize the sorted data by V ranges so as to make each range have the same amount of data S/V.
   (c) Compute the mean of net load at each range for every node $n \in N_l$.

The transition probabilities are computed by counting the ratio of transition of the S samples from one range to another over consecutive time steps.

Chapter 6

Scenario Selection

In this chapter we discuss the sampling methods for the evaluation of the different dispatch policies. In order to evaluate each policy, we generate several samples from the stochastic model that we developed in Chapter 5. Since certain policies require large computation time in online application, the policies need to be evaluated by a reasonable number of samples. We first describe Monte-Carlo sampling. However, this can lead to an unreliable result. Therefore, we consider importance sampling in order to reduce variances. The basic theories are presented in [12] and [4].

6.1 Monte-Carlo Sampling

Let us denote Ω as the sample space, p as the probability measure on Ω, and $\omega \in \Omega$ as an outcome. Define C as a random variable which models the cost of a sample. The expected value of C, denoted as z, can be described as:

$$\begin{aligned} z &= \mathbb{E}[C] \\ &= \sum_{\omega \in \Omega} p(\omega) C(\omega) \end{aligned} \tag{6.1}$$

Suppose that we have statistically independent samples ω^i for $i = 1, \ldots, K$ generated from the distribution p. Note that $C(\omega^i)$ refers to the outcome of the cost of sample ω^i for $i = 1, \ldots, K$. Moreover, define $\bar{z}$ as the unbiased estimate of the expected value of cost, z. Then, $\bar{z}$ can be computed as:

$$\bar{z} = \frac{1}{K} \sum_{i=1}^{K} C(\omega^i) \tag{6.2}$$

From the central limit theorem, the variance of z is:

$$\mathrm{Var}(\bar{z}) = \frac{\sigma^2}{K} \tag{6.3}$$

where σ^2 is the variance of C, which can be estimated by $\bar{\sigma}^2 = \frac{1}{K} \sum_{i=1}^{K} (z_i - \bar{z})^2$ with $z_i = C(\omega^i)$. Therefore the standard deviation of $\bar{z}$ decreases by the square root of sample size, $\sqrt{K}$.

6.2 Importance Sampling

Although Monte-Carlo sampling is a simple technique, it is inefficient in certain situations where only a small number of samples determine an important part of z. In order to prevent a huge amount of sampling, we would like to effectively select the samples where the impact on the

expected value, $p(\omega)C(\omega)$, is large.

Importance sampling is a technique for estimating $z = \mathbb{E}[C]$ by using a different distribution from the original distribution p. By definition, the expected value of C, z, can be expressed as:

$$\begin{aligned} z &= \sum_{\omega \in \Omega} p(\omega)C(\omega) \\ &= \sum_{\omega \in \Omega} q(\omega)\frac{p(\omega)C(\omega)}{q(\omega)} \end{aligned} \tag{6.4}$$

where we introduce q, which can be interpreted as a new probability measure for sampling a random variable $p(\omega)C(\omega)/q(\omega)$. Furthermore, the value $p(\omega)/q(\omega)$ can be interpreted as a correction for the bias induced by the change in distribution q. Note that we can design q as we like. Indeed, the best choice of q in terms of estimating z, q^*, can be defined as follows:

$$\begin{aligned} q^*(\omega) &= \frac{p(\omega)C(\omega)}{z} \\ &= \frac{p(\omega)C(\omega)}{\sum_{\omega \in \Omega} p(\omega)C(\omega)} \end{aligned} \tag{6.5}$$

since we can obtain the exact expected value by just one sample, i.e. $p(\omega)C(\omega)/q^*(\omega)$ is a random variable with zero variance. This fact can be confirmed as follows. Suppose that we have K samples generated by the ideal biased distribution q^*. A new estimator $\bar{z}_q$ is expressed as

$$\begin{aligned} \bar{z}_q &= \frac{1}{K}\sum_{i=1}^{K} \frac{p(\omega^i)C(\omega^i)}{q^*(\omega^i)} \\ &= \frac{1}{K}\sum_{i=1}^{K} \frac{p(\omega^i)C(\omega^i)}{\frac{p(\omega^i)C(\omega^i)}{z}} \\ &= \frac{1}{K}\sum_{i=1}^{K} z \\ &= z \end{aligned} \tag{6.6}$$

We see that the cost of each sample $C(\omega^i)$ is unbiased by $p(\omega^i)/q^*(\omega^i)$ and it will be the true expected value z. Furthermore, the variance of $\bar{z}_q$ can be written as

$$\begin{aligned} \mathrm{Var}(\bar{z}_q) &= \frac{\bar{\sigma}_q^2}{K} \\ &= \frac{1}{K^2}\sum_{i=1}^{K} \left(\frac{p(\omega^i)C(\omega^i)}{q^*(\omega^i)} - \bar{z}_q \right)^2 \\ &= 0 \end{aligned} \tag{6.7}$$

where $\bar{\sigma}_q^2$ is the estimation of variance of the new random variable $p(\omega)C(\omega)/q^*(\omega)$ with the measure q^*. Of course, this is practically useless since in order to obtain the ideal new measure q^*, we need the exact value of z, which is indeed what we want to estimate. However, we use the insight from this analysis in order to apply this method with some estimate of q that is nearly proportional to $p(\omega^i)C(\omega^i)$. The intuition behind this is that the new measure q places more "importance" on a scenario that has a very high impact on z even if the probability of its occurrence is quite low.

6.3 Importance Sampling Algorithm

As discussed in the previous section, we can estimate q by some heuristic, and thus obtain a reliable estimation of the expected cost with a smaller number of samples. In this work we approximate q by the cost of the *perfect foresight policy*, whereby the stochastic parameters of the problem are assumed to be known a priori therefore the policy will provide the best possible solution (even if this solution is not implementable in practice). It is reasonable to do so because a sample that results in high cost in the perfect foresight policy also highly affects the cost of other policies.

Step 1: Computation of the cost of perfect foresight: for $i = 1, \ldots, N$,

(a) Generate a scenario ω^i from the true distribution (using the transition probability of the lattice).

(b) Implement the perfect foresight policy and denote the cost as $C^{\text{perfect}}(\omega^i)$.

Step 2: Define a new measure q as follows:

$$q_i = \frac{C^{\text{perfect}}(\omega^i)}{\sum_{j=1}^{N} C^{\text{perfect}}(\omega^j)}, i = 1, \ldots, N \tag{6.8}$$

Step 3: Select M (usually $M << N$) scenarios among the N previously generated scenarios by sampling from the q distribution.

Step 4: Implement a policy: for $k = 1, \ldots, M$,

(a) Implement the policy on the scenario ω^k, denote the cost as $C^{\text{policy}}(\omega^k)$

(b) Unbias the cost by p/q, i.e.

$$\begin{aligned} C^{\text{policy}}_{\text{unbiased}}(\omega^k) &= C^{\text{policy}}(\omega^k)\frac{p_k}{q_k} \\ &= C^{\text{policy}}(\omega^k)\frac{\frac{1}{N}}{\frac{C^{\text{perfect}}(\omega^k)}{\sum_{j=1}^{N} C^{\text{perfect}}(\omega^j)}} \end{aligned} \tag{6.9}$$

Step 5: Compute the mean $\bar{z}_q$:

$$\bar{z}_q = \frac{1}{M}\sum_{k=1}^{M} C^{\text{policy}}_{\text{unbiased}}(\omega^k) \tag{6.10}$$

and the variance of the mean $\text{Var}(\bar{z}_q)$:

$$\begin{aligned} \text{Var}(\bar{z}_q) &= \frac{\bar{\sigma}_q^2}{M} \\ &= \frac{1}{M}\left[\frac{1}{M}\sum_{k=1}^{M}\left(C^{\text{policy}}_{\text{unbiased}}(\omega^k) - \bar{z}_q\right)^2\right] \end{aligned} \tag{6.11}$$

Note that $\bar{\sigma}_q^2$ refers to the estimation of variance of the unbiased cost $C^{\text{policy}}_{\text{unbiased}}$ with the measure q.

We store the selected samples at Step 3 and use them in order to evaluate the policies (Step 4-5). This leads to economical computation time and usage of memory.

Chapter 7

Case Study

In this chapter, we will compare the performance of the policies with case studies. First, we perform tests in a small network with two layers using the three different formulations defined in Chapter 4. The stochastic data is generated by hand since the model size is compact enough, and we can implement the global SDDP on the model. Then, we will move to a larger network with multiple layers using the Injection Limit formulation. The data is generated by the method introduced in Chapter 5. The network is not tractable anymore, thus it is impossible to implement SDDP directly. All problems are implemented by `AMPL`[1] using the solver: `Gurobi`[2] and `CPLEX`[3].

7.1 Two-Layer Model

7.1.1 Problem Settings

Parameters and Topology. The topology of the model is shown in Figure 7.1. There are 15 nodes in total and we name the upper layer with nodes 0-6, 12-14 as Layer 1 and the lower layer with nodes 7-11 as Layer 2. The conventional generators are only available at the root node, node 0. Each node has battery storage and net demand (difference between demand and PV production) which is one of the stochastic parameters of the problem. The other stochastic parameter, the generator capacities, vary in time and are negatively correlated to the amount of net demand. This setting is based on the assumption that when net demand is high (solar power generation is small or demand is high) in an area, the net demand of other areas is also likely to increase as well, hence the demand for power generation from power plants located in the high-voltage grid will rise[4].

Table 7.1 presents the data which we employ in the model. There exist three generators with different marginal costs. Line and battery capacities are identical everywhere, but the quality of battery is different at some nodes. We consider a one-day operating horizon with hourly time step thus, $H = 24$. The level of discretization of the lattice at each layer is five, hence the number of outcomes of the overall lattice for $t = 2, \dots, H$ is $|\Omega_t| = 5^2 = 25$, and each local lattice has $|\Omega_t^l| = 5$ outcomes for $l = 1, 2$.

[1] `AMPL` web page: https://ampl.com/

[2] `Gurobi` web page: http://www.gurobi.com/

[3] `CPLEX` web page: https://www.ibm.com/analytics/cplex-optimizer

[4] Indeed, we assume that node 0 is an interface node with the transmission network, thus net demand affects the generator capacities.

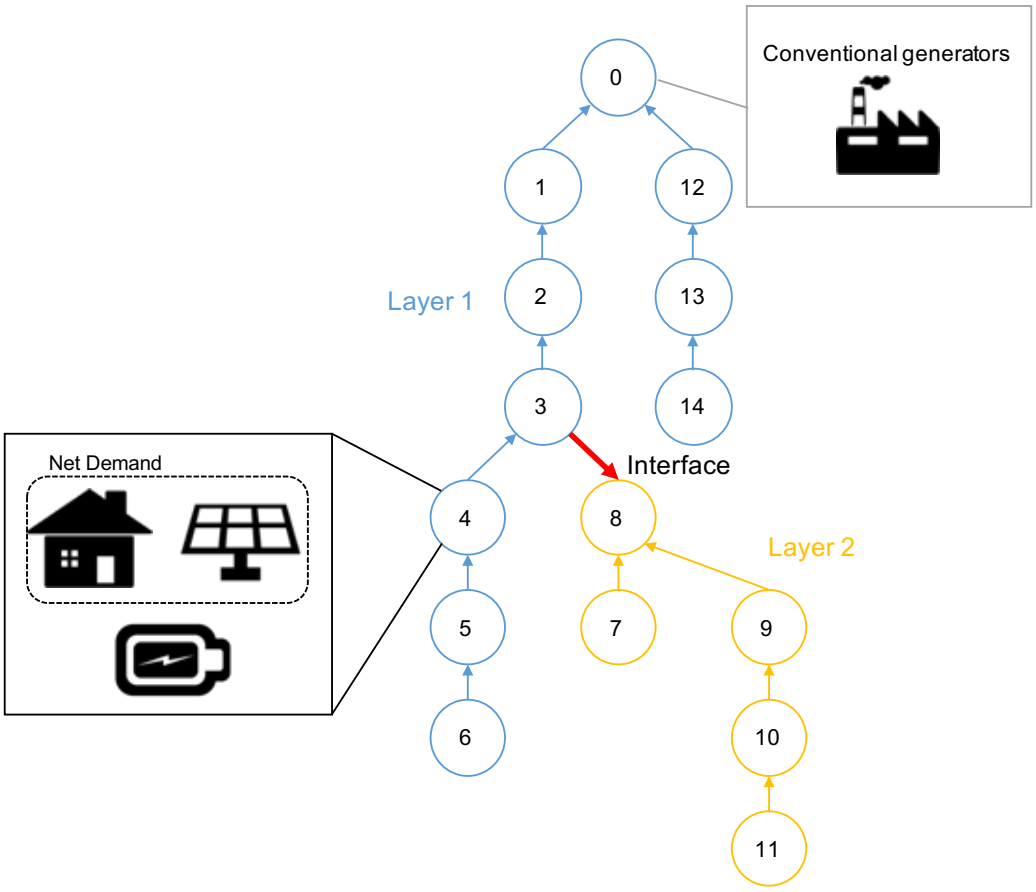


Figure 7.1: Topology of the two-layer model. Blue nodes represent Layer 1 and yellow nodes represent Layer 2. Conventional generators are only available at the root node, node 0. The interface corresponds to the edge between node 3 and node 8.

Table 7.1: Problem parameters of the two-layer model.

Parameter	Notation	Value	Unit
Marginal cost of generator 1	MC_1	10	\$/MWh
Marginal cost of generator 2	MC_2	20	\$/MWh
Marginal cost of generator 3	MC_3	50	\$/MWh
Value of lost load	$VOLL$	5000	\$/MWh
Value of lost storage[a]	$VOLS$	2500	\$/MWh
Line capacity i	T_i	1.85	MW
Battery capacity n	S_n	1	MWh
Storage requirement n[a]	R_n	1	MWh
Battery charge efficiency of node 0-6,8-14	η_n	0.9	
Battery charge efficiency of node 7	η_7	0.95	
Battery discharge efficiency of node 0-6,8-14	μ_n	0.9	
Battery discharge efficiency of node 7	μ_7	0.95	
Battery charge rate of node 1-14	BC_n	0.6	MW
Battery charge rate of node 0	BC_0	0.9	MW
Battery discharge rate of node 1-14	BD_n	0.6	MW
Battery discharge rate of node 0	BD_0	0.9	MW

[a] These values are only applied to the third formulation, the Battery Deadline model.

Stochastic Data. As stated above, the stochastic data is the net demand at each node and the capacity of generators of the root node with the two being negatively correlated. The model crosses four regimes which change in time: these are referred to as Initial ($t = 1$), Morning ($t = 2, ..., 6$), Daytime ($t = 7, ..., 18$), and Night ($t = 19, ..., 24$). In the Daytime regime, there is low net demand and enough capacity of generators, hence there is sufficient production by solar panels. On the other hand, the system is in risk of load shedding due to high demand and limited production in the Night regime. The Morning regime is an intermediate regime between the Daytime and Night regime. See Appendix C.1 for more details.

Policy Settings. The hyperparameters of the policies are set as follows:

- SDDP:
  - $M = 20$: The number of forward-backward SDDP iterations.
  - $K = 50$: The number of Monte-Carlo samples at each iteration.
  - $|\Omega_t| = 5^2 = 25$: The size of the (global) lattice for $t = 2, \ldots, H$.
- sbr MPC:
  - $S = 5$: The number of scenarios (generated by uniform sampling).
  - $\gamma = 0.9$: The discount factor.
- Decomposed SDDP:
  - $M = 15$: The number of forward-backward SDDP iterations.
  - $K = 50$: The number of Monte-Carlo samples at each iteration.
  - $|\Omega_t^l| = 5$: The size of the local lattice for $t = 2, \ldots, H$ for $l = 1, 2$.
  - $\bar{b}^l$: The fixed flow at the interface is the average of 10000 solutions obtained by the perfect foresight policy.
- Hybrid MPC:
  - The value functions are the same as in the case of Decomposed SDDP.
  - $D = 4$: The foresight step size.
  - $S = 5$: The number of scenarios (generated by uniform sampling).

Note that ce MPC does not have any specific hyperparameters.

7.1.2 Performance Comparison

Results of SDDP and Decomposed SDDP algorithms

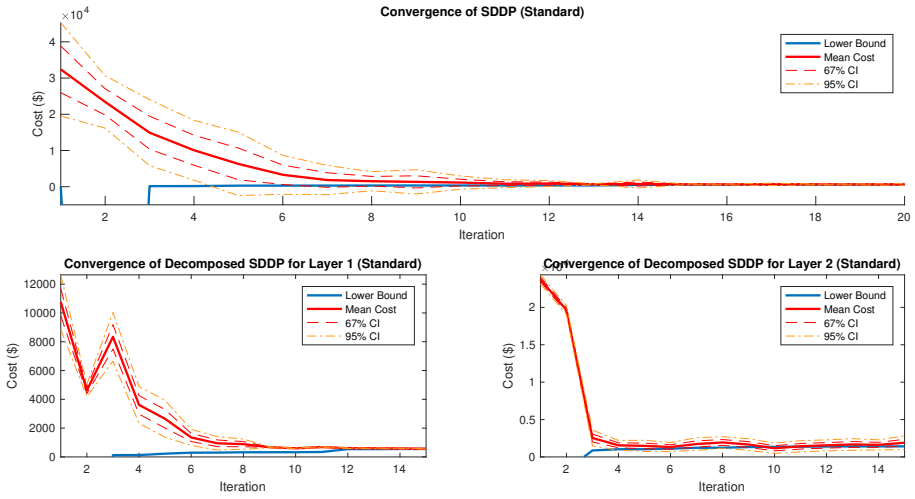
Figure 7.2 presents the evolution of the upper and lower bound generated by the SDDP and Decomposed SDDP algorithm for each formulation. The probabilistic upper bound (i.e. mean cost) is indicated in the red solid line, and the lower bound is depicted in the blue line. The two dashed lines are the 67% and 95% confidence intervals of the mean cost. We observe that all the upper and lower bounds do not change much after certain iterations. In Table 7.2, we present the offline computation time of the two algorithms. `_total_solve_time` refers to the sum of system CPU seconds used by all `solve` commands and user CPU seconds used by all `solve` commands, while `_ampl_time` is the sum of system CPU seconds used by the `AMPL` process itself and user CPU seconds used by the `AMPL` process itself[5]. That is to say, `_total_solve_time` refers to the computation time for solving each $NLSD$ at forward-backward iterations, while `_ampl_time`

[5]See Table A-14 of the reference appendix of the AMPL book for more information.

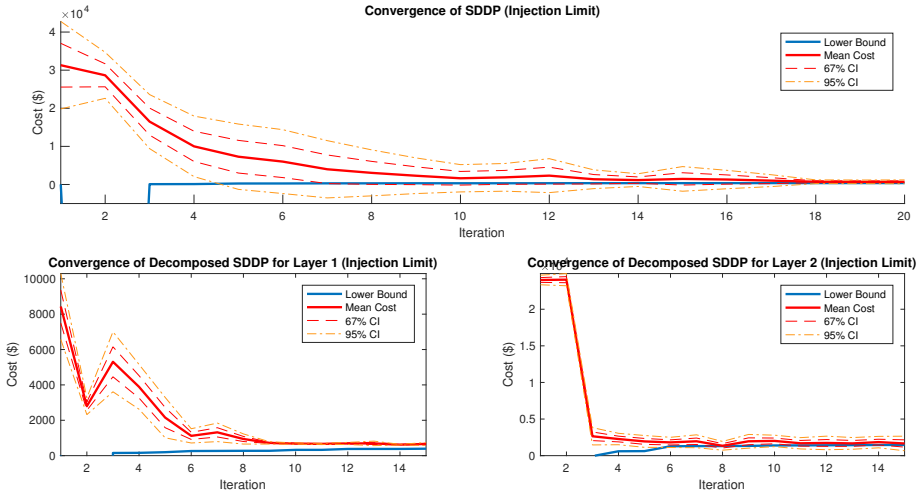
Table 7.2: Computation time of SDDP and Decomposed SDDP algorithm for each formulation.

	Standard		Injection Limit		Battery Deadline	
	SDDP	Dec. SDDP	SDDP	Dec. SDDP	SDDP	Dec. SDDP
`_total_solve_time` (s)	4694	1492	4885	1482	4808	1498
`_ampl_time` (s)	15732	1696	16646	1676	12418	1659

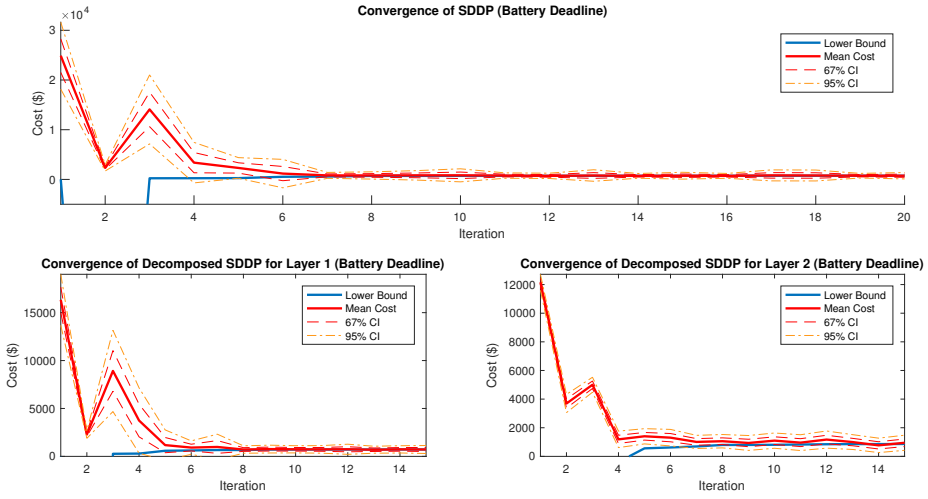
represents the time used by other operations, e.g. loading/whiting a file or generating the optimality cuts. We observe that the SDDP requires a lot of time for generating and checking the optimality cuts (i.e. `_ampl_time` is larger than `_total_solve_time`). Note that the computation time of Decomposed SDDP is the sum of two individual SDDP algorithm for layer 1 and 2, hence it can be reduced to about half of the time if one uses two computers in parallel. The SDDP algorithm terminates within a few hours, whereas Decomposed SDDP terminates within one hour for all formulations. Hence, the run times are acceptable because the two polices are solved in the day-ahead time frame, and the results are stored for real-time (i.e. online) applications.



(a) Standard



(b) Injection Limit



(c) Battery Deadline

Figure 7.2: Evolution of the SDDP and Decomposed SDDP algorithm for each formulation. The red solid line represents the mean cost (stochastic upper bound), the blue line refers to the lower bound, and the two dashed red lines are the 67% and 95% confidence intervals, respectively.

Table 7.3: Average cost and its standard deviation for each policy against 1000 online samples. Note that "SD of mean ($)" represents the standard deviation of the mean as defined in (6.11) in section 6.3.

	Standard		Injection Limit		Battery Deadline	
	Mean ($)	SD of mean ($)	Mean ($)	SD of mean ($)	Mean ($)	SD of mean ($)
ceMPC	2838	68.55	2529	56.44	1895	43.49
sbrMPC	710	9.58	862	11.18	842	6.10
SDDP	687	8.81	736	5.23	768	3.74
Dec. SDDP	713	10.94	812	9.97	847	2.56
Hybrid MPC	731	13.52	828	15.33	802	4.45

Online Applications

We test the performance of all policies on three problem formulations, the Standard Storage, Injection Limit, and Battery Deadline model. We use 1000 samples generated by importance sampling. Note that the costs of these samples are unbiased by the cost of the perfect foresight policy as explained section 6.3[6], hence the exact cost on each sample can be different. The average cost and its standard deviation are summarized in Table 7.3. For all formulations, SDDP provides the best performance in terms of mean cost, while ce MPC yields quite poor results. Indeed, this result is expected since SDDP is designed to minimize the average cost and the policy has all the information of the problem (i.e. the global lattice). The other policies, sbr MPC, Decomposed SDDP, and Hybrid MPC, seem to achieve similar relative performance in every formulation.

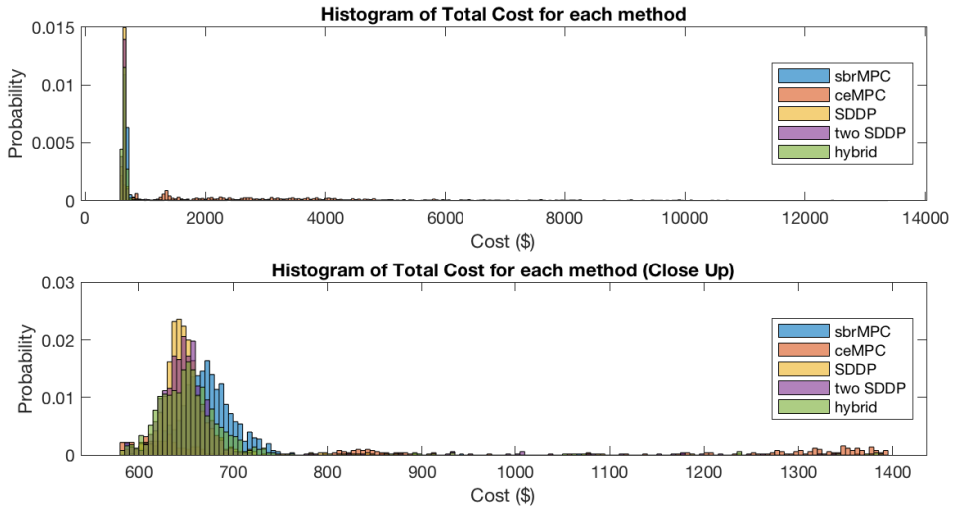
Standard Storage Formulation. In Figure 7.3, we see the histograms of each policy for the three formulations. In the Standard model, SDDP, Decomposed SDDP and Hybrid MPC result in a nearly identical peak around 650 $, meanwhile the peak of sbr MPC is placed at slightly further to the right. Ce MPC performs poorly, and does not seem to have any specific peaks. Indeed, in the upper panel of Figure 7.3a, the wide scope of horizontal axis (total cost) is caused by the poor performances of ce MPC in some samples. The height of the summits is the highest in SDDP, which implies that SDDP has less probability of working poorly, and this fact can be confirmed by the small standard deviation of the mean cost in Table 7.3. However, the difference between the four faith policies that perform well is not very significant.

Injection Limit Formulation. In the Injection Limit model (Figure 7.3b), we observe more interesting results. Although the average performance of SDDP surpasses sbr MPC, Decomposed SDDP and Hybrid MPC as shown in Table 7.3, significant amount of mass of the histogram of SDDP is further to the right. This implies that SDDP is quite risk-averse and that it stores energy even in some "easy" scenarios where there is plentiful renewable production (i.e. less net demand), whereas the three policies, sbr MPC, Decomposed SDDP and Hybrid MPC, are able to manage the system with lower cost. We must insist that Hybrid MPC has the great part of mass on the leftmost side among all policies, i.e. the policy achieves to manage the system with the lowest cost in the majority of samples. We also note that sbr MPC has a second peak around 1050 $. This highlights a weakness of sbr MPC for this formulation. We will investigate this phenomenon in the next section, 7.1.3.

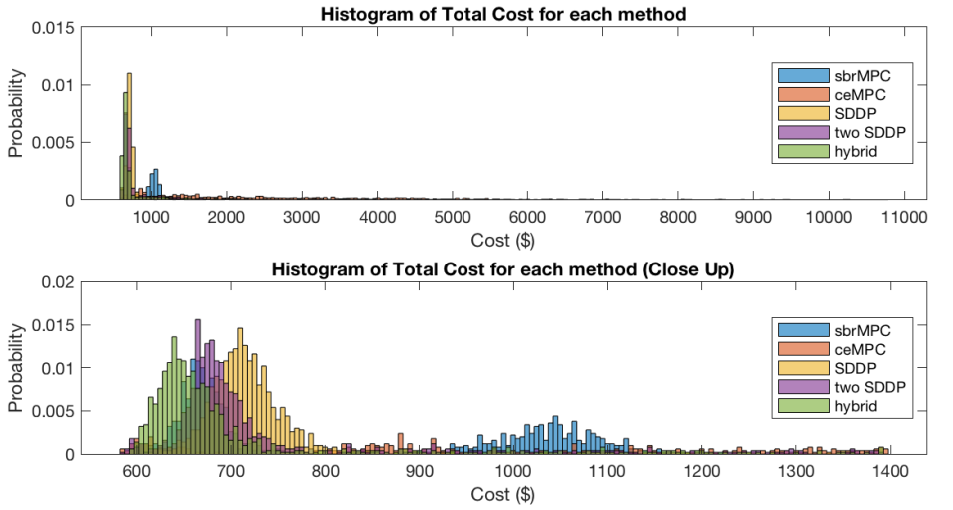
Battery Deadline Formulation. In the last formulation (Figure 7.3c), the Battery Deadline model, SDDP performs quite well, with respect to both the average and the mass of histogram.

[6] As defined in (6.9) in section 6.3, the unbiased cost is computed by dividing the cost of a policy by the cost of the perfect foresight policy. Since importance sampling tends to select disastrous samples, we here plot the unbiased cost.

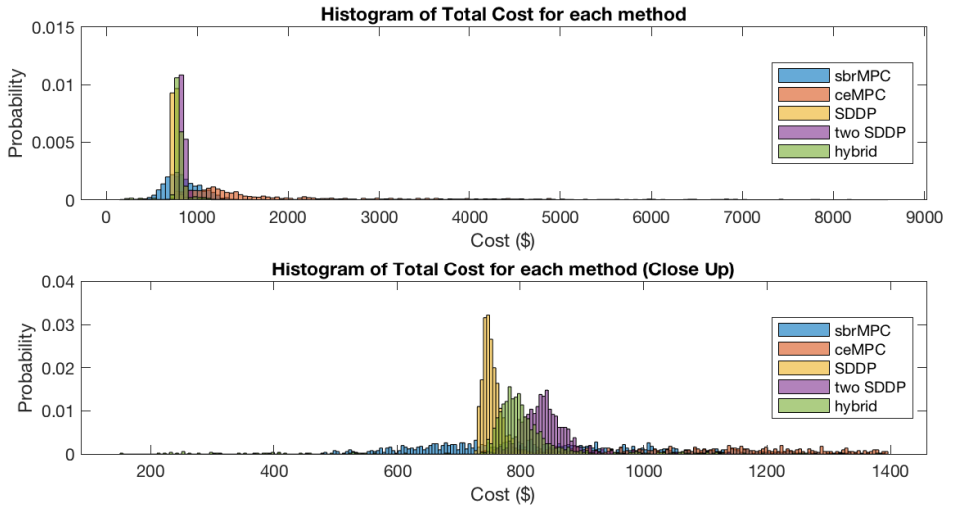
As stated above, the histogram of unbiased costs is generated by dividing the cost of a policy by the cost of perfect foresight. Therefore, the SDDP is strongly correlated to the perfect foresight (the costs are centered), while sbr MPC seems to be independent of the perfect foresight policy (the costs are widely spread). Comparing Decomposed SDDP with Hybrid MPC, the shapes of the histograms appear similar however the latter policy expends less in most cases. Although the standard deviation of Hybrid MPC is slightly greater than that of Decomposed SDDP, Hybrid MPC performs better than Decomposed SDDP in this formulation.



(a) Standard



(b) Injection Limit



(c) Battery Deadline

Figure 7.3: Histograms of unbiased costs against 1000 online samples generated by importance sampling. Note that "twoSDDP" refers to the Decomposed SDDP.

Table 7.4: Operation costs of each policy on the selected sample in Injection Limit formulation.

	Generation ($)	Load Shedding ($)	Total ($)
ceMPC	579.8	662.2	1242.0
sbrMPC	641.5	364.0	1005.5
SDDP	723.8	0.0	723.8
Decomposed SDDP	674.8	0.0	674.8
Hybrid	617.8	0.0	617.8
Perfect Foresight	553.2	0.0	553.2

7.1.3 Relative Comparison in the Injection Limit Model

In the previous section, we observed that sbr MPC got into difficulties in the Injection Limit formulation. Hence, we here present an analysis on a typical single sample in the formulation[7]. The operation cost over time and its breakdown is presented in Table 7.4. Note that the cost of the perfect foresight policy is the best possible cost for this sample. We see that load shedding occurs in both MPC methods, indeed this is the case belonging to the second peak of the cost histogram for sbr MPC as depicted in Figure 7.3b. Since we have observed that ce MPC performs poorly, it will not be investigated in the following analysis.

Behaviors on the Selected Sample. The behavior of each policy at every time stage is shown in Figure 7.4. The rows of the figure indicate (1) demand side: net demand, battery charge, production shedding, and their sum (total demand), (2) supply side: generation, battery discharge, load shedding and their sum (total supply), (3) total storage level of batteries, and (4) the cost induced by generators and load shedding. On the other hand, the columns of the figure represent the polices, perfect foresight, sbr MPC, SDDP, Decomposed SDDP, and Hybrid MPC. We observe that the SDDP and Decomposed SDDP use a lot of generators at the first period, stage 1. It should be noted that this utilization is independent of samples, i.e. we always solve the same problem ($NLDS$) at stage 1. Load shedding in sbr MPC occurs at the final time step, stage 24[8]. Indeed, this happens at Node 1 (see Figure 7.1), and we will investigate the node in the following paragraphs.

[7] See Appendix C.2 for detailed data of the sample.

[8] This result of sbr MPC is problem setting dependent, i.e. the net demand is high at the ending stages (Night regime). Hence, this does not mean that sbr MPC performs poorly at the end of horizon.

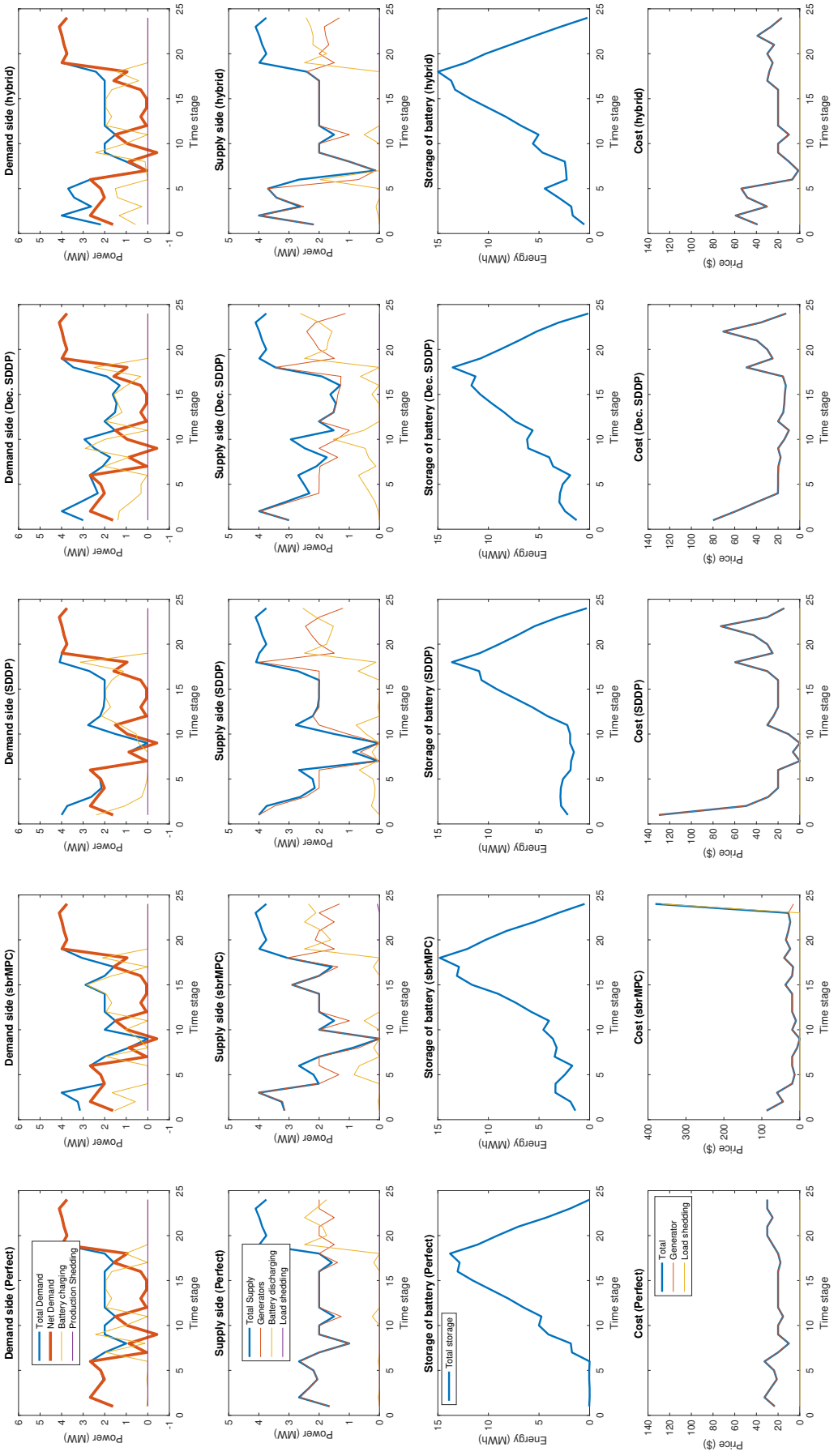


Figure 7.4: Behaviors of each policy on a typical sample. Rows: demand side, supply side, total storage level, and cost. Columns: Perfect Foresight, sbr MPC, SDDP, Dec. SDDP, and Hybrid MPC.

Energy Management at Node 1. In the previous paragraph, we observed that load shedding occurs at Node 1 in sbr MPC. Hence, let us now consider the behavior at the node in detail. The upper panel of Figure 7.5 illustrates the net demand at every stage at Node 1 and the corresponding injection limits. When net demand goes below the limit, from stage 19 to 24 in this sample, it is necessary to cover the difference between the lower limit and net demand using battery (i.e. battery discharge), otherwise the demand must be spilled (i.e. load shedding). This can be confirmed at the bottom panel of Figure 7.5. The figure presents the battery discharge power for each policy during stage 19-24. The minimum required power refers to the difference, which is required to be satisfied the battery discharge in order to fulfill the injection limit constraints. We observe that sbr MPC fails to achieve the requirement at stage 24, while all other polices manage to cover the demand by the battery discharge. Moreover, it seems that sbr MPC overuses the stored energy during stage 21. Hence, let us examine the battery storage level at Node 1.

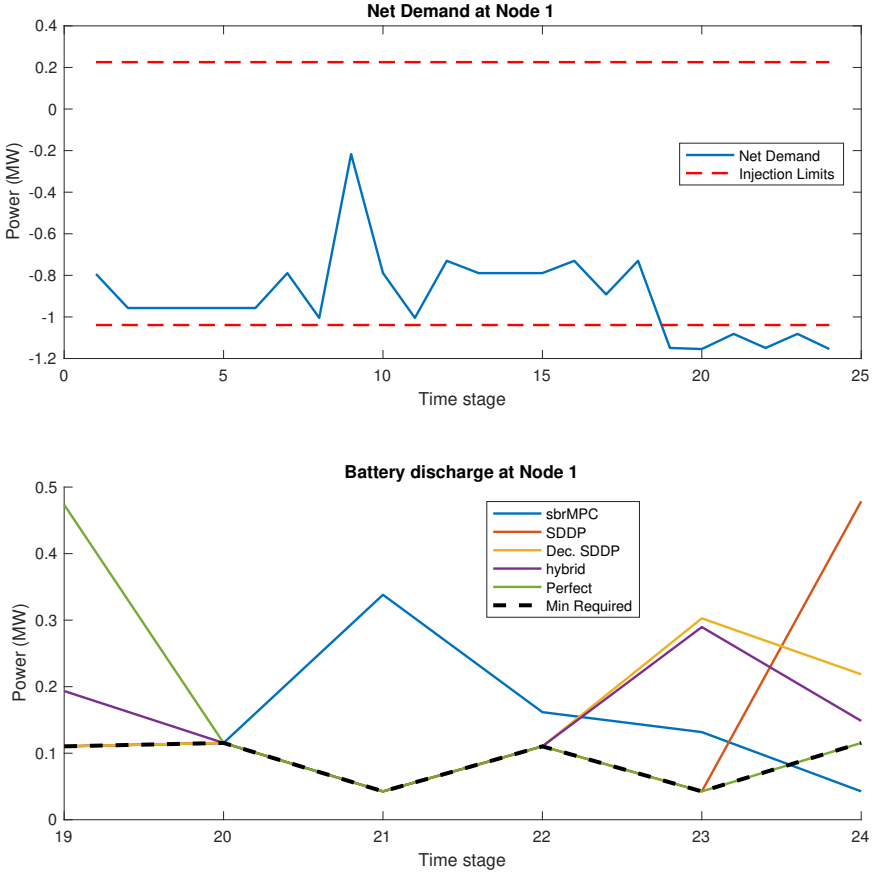


Figure 7.5: Operation of each policy in the sample at Node 1. Upper panel: Net demand with the injection limits. Bottom panel: Battery discharge from stage 19 to 24 with the minimum required power to satisfy the injection limit constraints.

The battery storage level at Node 1 from stage 18 to stage 23 can be seen in Figure 7.6. The bars indicate the storage level of every policy, whereas the orange horizontal line represents the minimum required storage level in order to satisfy the injection limit constraint[9]. Note that the bars refer to the stored levels at the ending of each stage, i.e. the storage level at stage 18 is

[9]We note that we cannot know these required levels a priori when we are actually implementing the polices because the values are computed by the future value of net demand and the generator capacities.

equivalent to that of the beginning of stage 19. As can be seen from the figure, all policies store energy at the maximum level at the end of stage 18. However, sbr MPC discharges too much energy by stage 23, hence the storage level falls below the requirement at the end of stage 23. Consequently, the policy cannot avoid load shedding at stage 24. On the other hand, the other policies keep energy over the required level, and succeed to avert load shedding. By introducing injection limit constraints, it is necessary to consider where energy is stored and how to use it. The total storage level over the entire network at stages 23 and 24 is summarized in Figure 7.7. Interestingly, in total, sbr MPC leaves the most unused energy at stage 24, at the end of the horizon. It implies that sbr MPC is not effective in managing the placement and utilization of energy due to the lack of scenarios to be considered. Sbr MPC is sensitive at the ending stages since the effect of generated scenarios becomes significant for its performance. In this sample, the policy fails to anticipate the actual scenario where net demand of Node 1 is relatively high. On the other hand, the policies with value functions anticipate locational need for storage well.

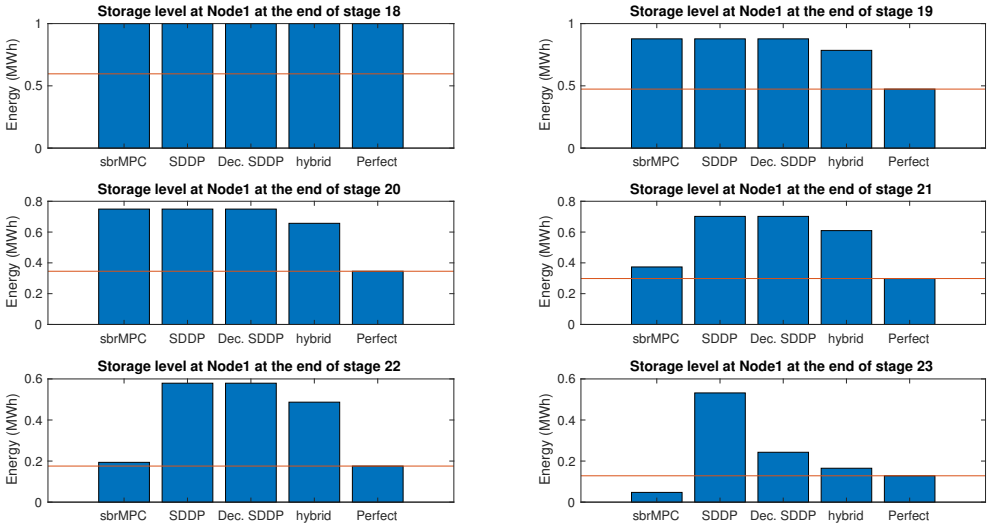


Figure 7.6: Battery storage level at Node 1 from stage 19 to 24 for each policy. The horizontal orange line represents the minimum required level in order to satisfy the injection limits.

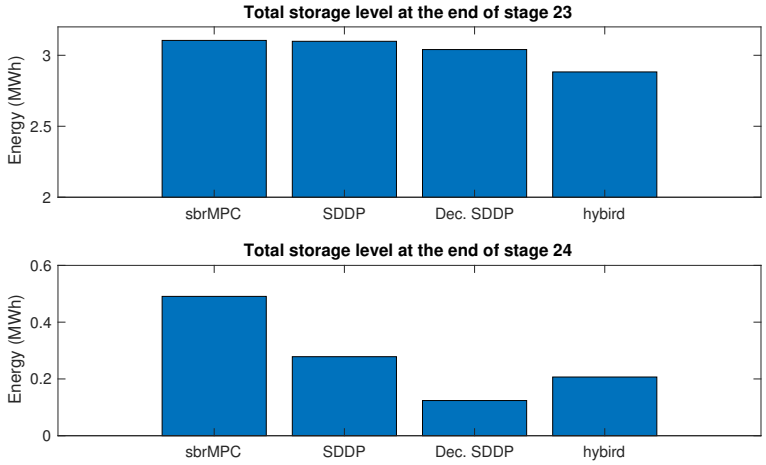


Figure 7.7: Total storage level over the entire network at stage 23 and 24 for each policy.

Table 7.5: Total generation over the entire network in the sample for each policy.

	Generation (MW)				
	sbr MPC	SDDP	Dec. SDDP	Hybrid MPC	Perfect
Total	47.171	47.226	47.414	47.116	46.160
Generator 1	34.029	31.685	34.003	33.828	37.000
Generator 2	11.864	12.336	11.192	12.828	9.160
Generator 3	1.278	3.205	2.220	0.460	0.000

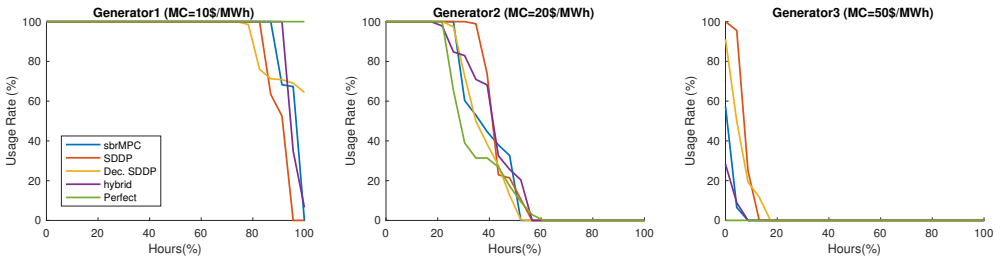


Figure 7.8: Production duration curves for every generator. A usage rate of 100 % implies that production reaches the maximum capacity of generators, whereas 0 % corresponds to the technical minimum of a generator.

Usage of Generators. We now proceed to consider the operation of generators in the selected sample. Table 7.5 presents the total amount of energy produced by the conventional generators over the full horizon for every policy. We observe that Hybrid MPC saves the most generation, while Decomposed SDDP produces the most. Comparing the table with Table 7.4, although the generation cost of SDDP is higher than the others, its total production is less than that of Decomposed SDDP. This feature is due to the efficiency of using the generators. Figure 7.8 depicts the usage rate of each generator (production/capacity of generator), versus its operation rate (occurrence in 24 hours) during the scenario. As can be seen, the usage time of Generator 3, the most expensive generator, is the smallest for Hybrid MPC, while the policy activates cheaper generators more. We also observe the efficient performance of sbr MPC which uses Generator 3 less and use Generator 1 more compared to SDDP and Decomposed SDDP.

7.2 Multi-Layer Model

In this section we investigate the policies on a multi-layer model. We only consider the second formulation, the Injection Limit formulation.

7.2.1 Problem Settings

Parameters and Topology. The topology of the model is shown in Figure 7.9. There are 569 nodes in total and they are grouped into 50 layers. Each layer is composed of 4-28 nodes. The depth of the network is 10. As in the case of the two-layer model, the horizon of the problem is one day, $H = 24$, with an hourly time step, and every node has net demand and a battery storage. Furthermore, we assume that the physical features of the batteries are identical for all nodes, as shown in Table 7.6, and that the initial storage is empty. Five conventional generators with different marginal costs are available at the root node.

Stochastic Data. The stochastic data of the problems are net demand and generator capacities. The net demand is generated by the process described in Chapter 5. Since the size of network is relatively large, it is impossible to generate the global lattice. Therefore, we cannot implement the SDDP in this model, and need to work with local lattices. The local lattices are built by the algorithm of section 5.3 with the level of discretization $|\Omega_t^l| = 5$ for $t = 2, \ldots, H$ and $l = 1, \ldots, 50$.

Table 7.6: Problem parameters of the multi-layer model.

Parameter	Notation	Value	Unit
Marginal cost of generator 1	MC_1	10	\$/MWh
Marginal cost of generator 2	MC_2	20	\$/MWh
Marginal cost of generator 3	MC_3	30	\$/MWh
Marginal cost of generator 4	MC_4	50	\$/MWh
Marginal cost of generator 5	MC_5	100	\$/MWh
Value of lost load	$VOLL$	5000	\$/MWh
Battery capacity n	S_n	1	MWh
Battery charge efficiency n	η_n	0.9	
Battery discharge efficiency n	μ_n	0.9	
Battery charge rate n	BC_n	0.6	MW
Battery discharge rate n	BD_n	0.6	MW

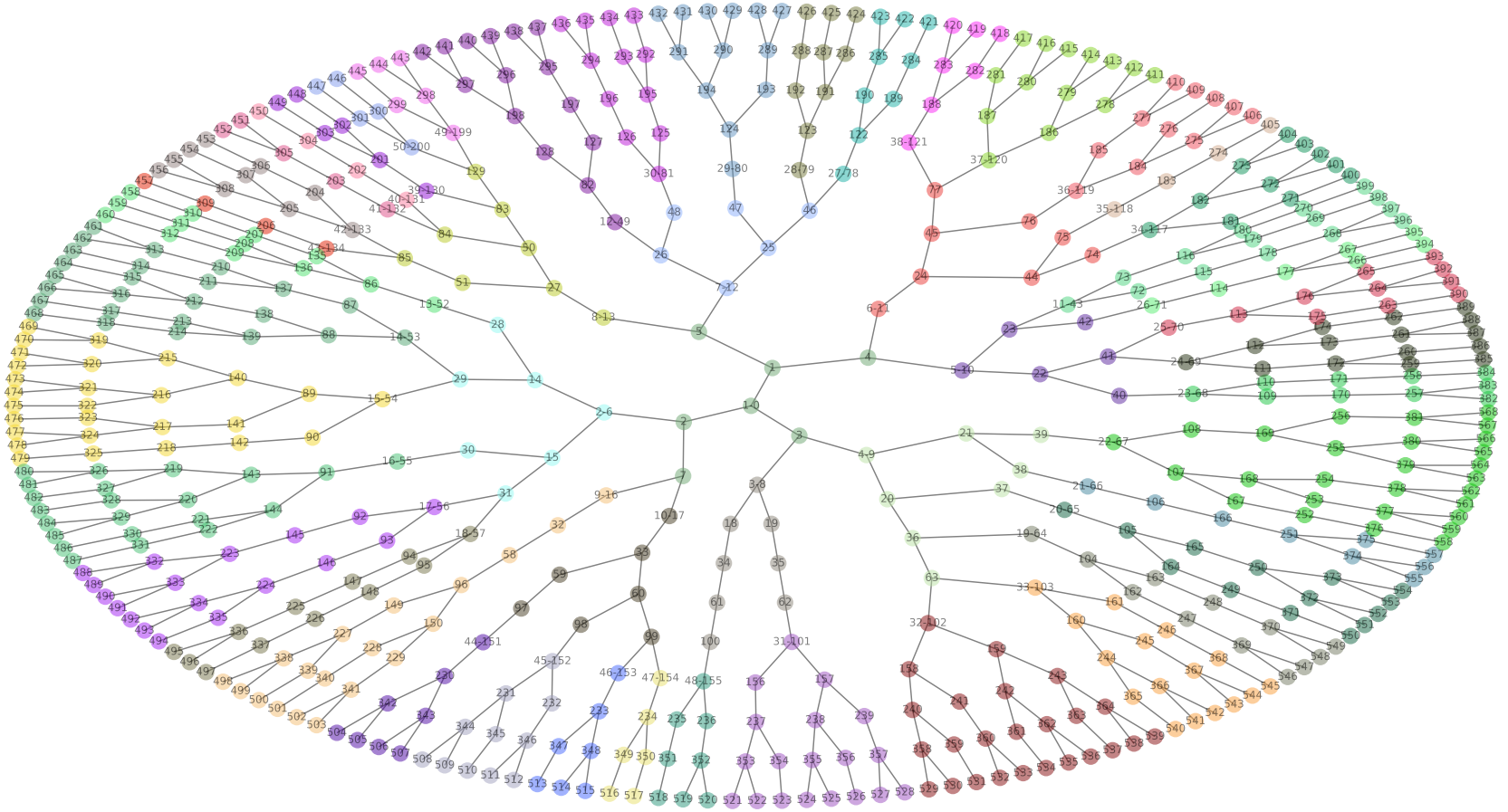


Figure 7.9: Topology of the multi-layer model. Coloring represents the grouping of layers. The numbers on nodes indicate their indices, while the root nodes of each layer have the two numbers, with the first one referring to the index of that layer.

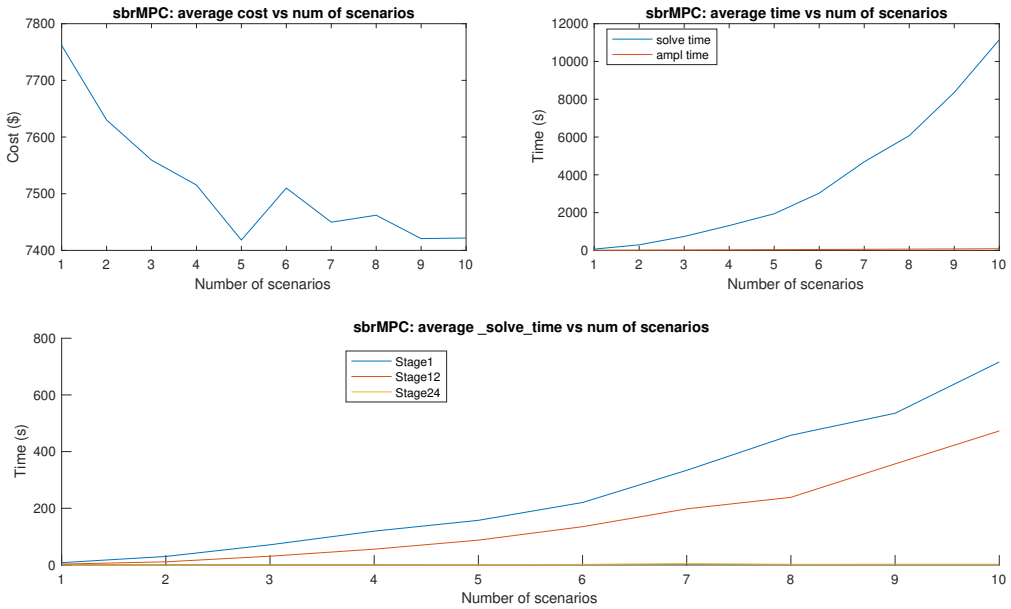


Figure 7.10: Upper left: Average cost performance against 10 Monte-Carlo samples as a function of the number of scenarios S used by sbr MPC. Upper right: Average `_solve_time` and `_ampl_time` over samples. Bottom: Average `_solve_time` at stage 1, 12 and 24.

7.2.2 Computation Time of Sbr MPC

Before proceeding to examine all policies, it is important to examine the scalability of sbr MPC. The upper left panel of Figure 7.10 presents the average cost of 10 samples as a function of the number of scenarios $S = 1, \ldots, 10$ for sbr MPC. The discount factor of the algorithms is set to $\gamma = 0.9$. Due to randomness, the curve is not monotonically decreasing. Nonetheless, it seems that $S = 5$ is the smallest number that gives a promising result. In the upper right panel, we see the average `_solve_time` and `_ampl_time`[10] over samples, i.e. stage 1 to 24. As can be seen, the average time increases faster than linear. In the bottom panel, the average `_solve_time` at stage 1, 12 and 24 are depicted. As stated in section 2.2.2, early stages require a lot of time to compute. In a large network which we are interested in, the scalability of the algorithm appears to be a significant problem. It is required to reach a compromise between computation time and performance quality.

7.2.3 Performance Comparison

Policy Settings. Hyperparameters of each policy are set as follows:

- sbr MPC:
  - $\gamma = 0.9$: The discount factor.
  - $S = 5$: The number of scenarios (generated by uniform sampling).
- Decomposed SDDP:
  - $M = 20$: The number of forward-backward iterations.
  - $K = 50$: The number of Monte-Carlo samples at each iteration

[10]Since sbr MPC requires online computation and the optimization problem itself is large, `_solve_time` will be significant compared to the SDDP where `_ampl_time` requires more time.

  - $|\Omega_t^l| = 5$: The size of the local lattice for $t = 2, \ldots, H$ for $l = 1, \ldots, 50$.
  - $\bar{b}^l$: The fixed flow at the interface is the average of 10000 solutions obtained by the perfect foresight policy.

- Hybrid MPC 1:
  - The value functions are the same as in the case of Decomposed SDDP.
  - $\gamma = 0.9$: The discount factor.
  - $D = 3$: The foresight lookahead.
  - $S = 3$: The number of scenarios (generated by uniform sampling).

- Hybrid MPC 2:
  - The value functions are the same as in the case of Decomposed SDDP.
  - $\gamma = 0.9$: The discount factor.
  - $D = 4$: The foresight lookahead.
  - $S = 5$: The number of scenarios (generated by uniform sampling).

Note again that ce MPC does not have any specific hyperparameters. The number of scenarios for sbr MPC is set to 5, inspired by the result of the previous section. We analyze two implementations of Hybrid MPC, the first one is characterized by a smaller number of scenarios S and step size D (Hybrid MPC 1), and the other is composed of larger values for the hyperparameters (Hybrid MPC 2). This implies that the first algorithm is strongly influenced by the value functions, whereas the latter is influenced more by MPC.

Result of Decomposed SDDP Algorithm. Decomposed SDDP and Hybrid MPC require offline computations. In fact, the advantage of this algorithm is that one can implement local SDDPs independently, hence the computation can be done in parallel. We implement 50 local SDDPs for all the layers with the previously designed hyperparameters. The maximum `_total_solve_time` is 787 seconds, and the maximum `_ampl_time` is 2529 seconds. Therefore, we can say that the offline step of the algorithm can be adopted in practice.

Online Applications

Average Performance. We test each policy against 40 samples generated by importance sampling. Figure 7.11 presents the histogram of unbiased costs for every policy. As can be seen, sbr MPC, and two Hybrid MPCs have their mass on the leftmost side, while the Decomposed SDDP has a peak at about 5000 $ higher than them. The average cost and its breakdown is summarized in Table 7.7. Although sbr MPC has the smallest cost peak, under 10000 $ in the figure, its average cost is relatively higher than the proposed policies. A significant amount of mass of the histogram of Hybrid MPC 2 (designed to be strongly influenced by MPC) is placed slightly to the left compared to that of Hybrid MPC 1 (designed to be strongly influenced by the value functions). However, in terms of mean cost and its standard deviation, Hybrid MPC 1 performs better than Hybrid MPC 2 (see Table 7.7).

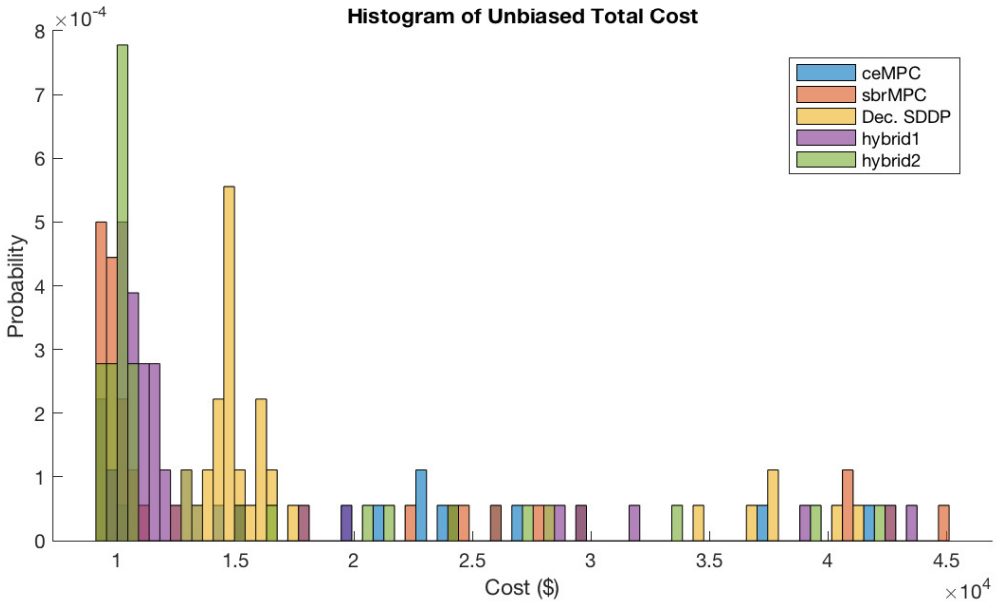


Figure 7.11: Histogram of unbiased total costs for each policy against 40 online samples generated by importance sampling.

Table 7.7: Mean cost and its breakdown, and the standard deviation of the mean against 40 samples. Note that "SD of mean" refers to the standard deviation of the mean.

		Generation	Load Shedding	Total	SD of mean
ce MPC	Cost ($)	9203	30096	39299	4291
	Percentage	23%	77%		
sbr MPC	Cost ($)	7407	16191	23599	3526
	Percentage	31%	69%		
Dec. SDDP	Cost ($)	13142	6132	19274	1336
	Percentage	68%	32%		
Hybrid MPC 1	Cost ($)	9310	5124	14434	1217
	Percentage	65%	35%		
Hybrid MPC 2	Cost ($)	8385	7221	15606	1708
	Percentage	54%	46%		

Online Computation Time. In a large-scale network, the online computation time becomes a crucial factor for policy selection. Figure 7.12 illustrates the average `_solve_time` at every stage with its 95 % confidence interval for each policy. We observe that ce MPC and Decomposed SDDP can be implemented in few seconds at all stages. On the other hand, sbr MPC and Hybrid MPCs require some minutes to compute. While sbr MPC drops after stage 7 since the size of the problem becomes smaller and smaller, the computation time of Hybrid MPCs remains non-negligible until around stage 20. This is due to the value functions that are included in Hybrid MPC. Since the lookahead of the algorithm D is fixed, the size of the optimization problem at every stage t depends on the number of optimality cuts at the prediction horizon $D + t$. At the ending stages, there no longer exist value functions, thus the size of the problem will be quite small (indeed, the algorithm will become a usual sbr MPC of that stage).

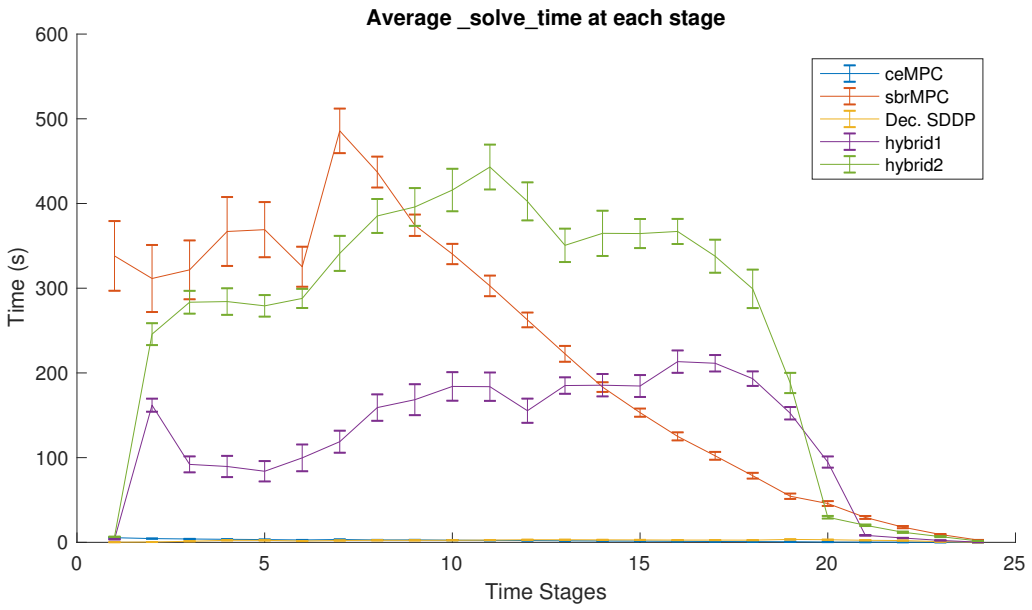


Figure 7.12: Average `_solve_time` at every stage. Vertical bars represent 95% confidence intervals.

Chapter 8

Conclusion

8.1 Summary of Contributions

The objective of this thesis was to propose a new policy for the short-term dispatch operation problem in the distribution network where operations remain largely passive. The complexity of distribution network operations relates to three major factors: (i) Uncertainty, (ii) Scale, and (iii) Non-linearity of power flow constraints. In this study, we focused on the first two challenges, (i) Uncertainty, and (ii) Scale.

In Chapter 2, we introduced the current approaches for solving the dispatch problem under uncertainty. The dispatch problem can be cast as a multi-stage stochastic linear program (MSLP). Stochastic dual dynamic programming (SDDP) is an algorithm for solving the MSLP by decomposition (nested L-shaped decomposition sub-problems) and Monte-Carlo simulation. On the other hand, model predictive control (MPC) is a feedback control technique that naturally incorporates optimization and repeats the process of prediction, optimization and execution, and there exist various designs. Certainty-equivalent MPC (ce MPC) predicts the uncertainty by its expected value while scenario-based robust MPC (sbr MPC) optimizes the worst-case performance over scenarios which are provided as inputs to the optimization problem. All these methods can cope with the uncertainty, however, the difficulty of scalability remains.

In Chapter 3, we propose an approach in order to resolve the scalability problem. In our strategy, a *hierarchical approach*, we focused on the fact that the distribution network has a radial structure which can be decomposed into small sub-networks (layers). The advantage of this approach is that each layer interacts in a weak sense whereby exchange of power only takes place in a few interfaces. Therefore, if a layer fixes the flow at the corresponding interfaces beforehand, the uncertainty of other layers does not need to be considered. This fact suggests that the possibility of solving a problem at each layer of the hierarchy, independently. In our first approach, the *Decomposed SDDP*, we first decompose the network into layers and apply the SDDP to each layer in order to obtain the value functions which contain the information on where and how much energy should be stored (offline step). Then, during real-time operation, we solve a real-time dispatch problem using all the value functions in a centralized fashion (online step). On the other hand, the second approach, the *Hybrid MPC*, is the combination of the sbr MPC and Decomposed SDDP in the sense that we run sbr MPC with a limited horizon size and the remaining uncertainties are dealt with by using the value functions obtained by Decomposed SDDP. Both policies are scalable to an arbitrary size of network.

The distribution system models that we considered in this work were presented in Chapter 4. We adopted three different types of formulations: the Standard Storage model, the Injection Limit model, and the Battery Deadline model.

In Chapter 5, we proposed a stochastic model of net demand (defined as the difference between PV power and demand) in the distribution network in order to evaluate our policies. In the proposed procedure, we applied the copula method for generating the multi-dimensional net demand process samples which are correlated in space and time.

Since certain policies require substantial computation time in online application, we had to select the samples in an efficient way. For this purpose, we proposed an importance sampling technique in Chapter 6. The algorithm generated a new probability measure which places importance on a scenario that has a high impact on the expected cost even if its occurrence is quite low.

Finally, in Chapter 7, we compared the performance of our policies with case studies. We first conducted tests on a small size network where the current approach (the SDDP) is implementable. We observed that the Decomposed SDDP and the Hybrid MPC performed well in easy scenarios where adequate amounts of renewable energy are available. We then moved to a multi-layer network where the SDDP is no longer applicable. We observed that the Hybrid MPC outperformed other policies in terms of the mean cost and its standard deviation. However, in order not to use excessive online calculation time, we had to select the hyperparameters of the Hybrid MPC carefully.

8.2 Future Directions

8.2.1 Detailed Modeling of Distribution Network

In this work, we focused on the two complexities of the distribution network, (i) Uncertainty, and (ii) Scale. However, as discussed in Chapter 1, the non-linearity of the power flow constrains cannot be ignored due to the role of reactive power and voltage constraints. Fortunately, in the radial network, it has been shown that if power flow is balanced then the second-order conic (SOC) relaxation of power flow equations is exact under reasonable assumptions [9]. This fact implies the possibility of extending our analysis to a setting of multi-stage stochastic second-order cone programming.

8.2.2 Completely Decentralized System

Our proposed approaches were centralized schemes in the sense that (i) we applied the perfect foresight policy in order to create local lattices, and (ii) all the value functions obtained by every local SDDP were collected and used in a real-time dispatch problem. For the first point, the perfect foresight policy is a centralized method since it requires the centralization of information (i.e. solving the dispatch problem of the entire network). However, if the system becomes decentralized, we can no longer rely on such a policy. Regarding the second fact, even though the real-time problem is now much smaller than the original multi-stage problem, the size of the value functions that need to be stored can be huge in a large-scale distribution network. For these reasons, a decentralized approach may be required. Several algorithms that can be applicable to the real-time dispatch problem have been proposed, e.g. the alternating direction method of multipliers (ADMM)[1].

8.2.3 Market Design

As described in Chapter 3, we did not discuss the choice of how to decompose the distribution network in this thesis. However, this subject will take an important role in electricity market design due to the deregulation of power system that has occurred in recent years. In this

[1]See http://web.stanford.edu/ boyd/admm.html for more details.

thesis, we focused on the centralized system where each layer works cooperatively so as to maximize expected social welfare (minimize the operation cost in this study). However, in a completely distributed system, the objective of each layer (or more concretely, each agent) will become maximizing its own profit. Hence, the behaviors in the market can be different from our investigation. One possible starting point is an analysis of TSO-DSO coordination schemes [18]. In this paper, the authors focus on a centralized coordination scheme. An alternative direction of research could involve the application of Generalized Nash Equilibrium (GNE) [26] in order to analyze non-cooperative interactions of DSOs, as for proposed in [1], [10].

Bibliography

[1] Smartnet. `http://smartnet-project.eu/`, n.d.

[2] J. R. Birge. Decomposition and partitioning methods for multi–stage stochastic linear programs. *Operations Research*, 33:989–1007, 1985.

[3] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer Series in Operations Research and Financial Engineering. Springer, 2010.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] S. D. Cairano, H. Park, and I. Kolmanovsky. Model predictive control approach for guidance of spacecraft rendezvous and proximity maneuvering. *International Journal of Robust and Nonlinear Control*, 22(12):1398–1427, 2012.

[6] D. S. Callaway and I. A. Hiskens. Achieving controllability of electric loads. *Proceedings of the IEEE*, 99(1):184–199, January 2011.

[7] M. Caramanis, E. Ntakou, W. Hogan, A. Chakrabortty, and J. Schoene. Co-optimization of power and reserves in dynamic T&D power markets with non-dispatchable renewable generation and distributed energy resources. *Proceedings of IEEE*, 104(4):807–836, April 2016.

[8] E. G. Cho, K. A. Thoney, T. J. Hodgson, and R. E. King. Supply chain planning: Rolling horizon scheduling of multi-factory supply chains. In *Proceedings 35th Conference Winter Simulation: Driving Innovation*, page 1409–1416, 2003.

[9] M. Farivar and S. H. Low. Branch flow model: Relaxations and convexification-part I. *IEEE Transactions on Power Systems*, 28(3):2554–2564, 2013.

[10] H. Gerard, E. I. R. Puente, and D. Six. Coordination between transmission and distribution system operators in the electricity sector: A conceptual framework. *Utilities Policy*, 2017.

[11] F. Herzog. *Strategic portfolio management for long-term investments: An optimal control approach*. PhD thesis, ETH, Zurich, 2005.

[12] G. Infanger. Monte carlo (importance) sampling within a benders decomposition algorithm for stochastic linear programs. *Annals of Operations Research*, 39(1):69–95, 1992.

[13] T. Kaneda, B. Losseau, A. Papavasiliou, D. Scieur, L. Cambier, P. Henneaux, and N. Leemput. Optimal management of storage for offsetting solar power uncertainty using multistage stochastic programming. In *20th Power Systems Computation Conference*, 2018.

[14] F. V. Louveaux and Y. Smeers. Optimal investments for electricity generation: a stochastic model and a test problem. *Numerical Techniques for Stochastic Optimization*, page 33–64, 1988.

[15] G. Marsaglia and W. W. Tsang. A fast, easily implemented method for sampling from decreasing or symmetric unimodal density functions. *SIAM Journal of Scientific and Statistical Computing*, 5:349–359, 1984.

[16] J. Mattingley, Y. Wang, and S. Boyd. Receding horizon control. *IEEE Control Systems*, 31(3):52–65, 2011.

[17] J. M. Morales, R. Minguez, and A. J. Conejo. A methodology to generate statistically dependent wind speed scenarios. *Applied Energy*, 87:843–855, March 2010.

[18] A. Papavasiliou and I. Mezghani. Coordination schemes for the integration of transmission and distribution system operations. In *20th Power Systems Computation Conference*, 2018.

[19] A. Papavasiliou, Y. Mou, L. Cambier, and D. Scieur. Application of stochastic dual dynamic programming to the real-time dispatch of storage under renewable supply uncertainty. *IEEE Transactions on Sustainable Energy*, 2017.

[20] M. V. F. Pereira and L. M. V. G. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52:359–375, 1991.

[21] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality (2nd Edition)*. John Wiley and Sons, 2011.

[22] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003.

[23] Z. Ren, W. Yan, X. Zhao, W. Li, and J. Yu. Chronological probability model of photovoltaic generation. *IEEE Transactions on Power Systems*, 29(3):1077–1088, May 2014.

[24] A. Shapiro. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209:63–72, 2011.

[25] A. Sklar. Fonctions de répartition aǹ dimensions et leurs marges. *Publications de l'Institut de statistique de l'Université*, 8:229–231, 1959.

[26] Y. Smeers, G. Oggioni, E. Allevi, and S. Schaible. Generalized nash equilibrium and market coupling in the european power system. *EPRG Working Paper Series*, 2010.

[27] K. T. Talluri and G. J. V. Ryzin. The theory and practice of revenue management. In *New York: Springer-Verlag*, 2004.

[28] M. J. West, C. M. Bingham, and N. Schofield. Predictive control for energy management in all/more electric vehicles with multiple energy storage units. In *Proceedings IEEE International Electric Machines and Drives Conference*, volume 1, page 222–228, 2003.

[29] M. Wytock, N. Moehle, and S. Boyd. Dynamic energy management with scenario-based robust mpc. In *Proceedings of the American Control Conference*, pages 2042–2047, May 2017.

Appendix A

Notations for Distribution System Models

- Sets
  - N : Set of all nodes
  - I : Set of all lines
  - L : Set of all layers
  - N_l: Set of nodes of layer l
  - I_l: Set of lines of layer l
  - C_n: Children of node n
  - A_n: Ancestor of node n (contains a single node)
  - $T = \{1, 2, ..., H\}$: Set of stages
  - G_n: Set of generators at node n
  - $N_{t,\text{deadline}}$: Set of nodes whose deadline corresponds to stage t (For the Battery Deadline formulation)
  - $N_{t,\text{active}}$: Set of nodes where the battery storage is active at stage t (For the Battery Deadline formulation)
- Variables (without time indices)
  - f_i: Flow along line i, direction: to ancestor.
  - p_g: Production of generator g
  - s_n: Battery storage of node n
  - bc_n: Battery charge
  - bd_n: Battery discharge
  - ls_n: Load shedding
  - ps_n: Production shedding
  - ss_n: Non-negative slack variable which represents the lack of energy (For the Battery Deadline formulation)
- Parameters (without time indices):
  - H: Horizon of the problem
  - MC_g: Marginal cost of generator g
  - $VOLL$: Value of lost load

- $\Pr(\omega)$: Probability of scenario $\omega \in \Omega$
- $ND_n(\omega)$: Net demand
- $PMax_g(\omega)$: Production capacity of generator g
- $PMin_n(\omega)$: Technical minimum of generator g
- η_n/μ_n: Battery charging / discharging efficiency of node n
- T_l: Flow limit of line l
- S_n: Battery storage capacity of node n
- BC_n/BD_n: Battery charging / discharging rate
- $IMax_n/IMin_n$: Upper / Lower bound on the injection at node n
- $VOLS$: Value of lost storage (For the Battery Deadline formulation)
- R_n: Required storage level of electric vehicle n (For the Battery Deadline formulation)

Appendix B

Proof of Theorem 5.1

Proof. Given a random continuous variable X, let define $Y = F_X(X)$. Then,

$$\begin{aligned} F_Y(y) &= \Pr(Y \leq y) \\ &= \Pr(F_X(X) \leq y) \\ &= \Pr(X \leq F_X^{-1}(y)) \\ &= F_X(F_X^{-1}(y)) \\ &= y \end{aligned}$$

Hence, F_Y is the CDF of a standard uniform random variable. Therefore, Y has a uniform distribution $U(0, 1)$. □

Appendix C

Data of the Two-Layer Model

C.1 Stochastic Data

Capacity of Generators Figure C.1 presents the capacity of generators at each regime. Note that a smaller index of outcome represents easy scenario where there exist plenty of renewable production and vice versa.

		Outcome 1	Outcome 2	Outcome 3	Outcome 4	Outcome 5
stage 1 Initial	Generator 1	1				
	Generator 2	1				
	Generator 3	2				
stage 2-6 Moring	Generator 1	2	2	1	1	0.5
	Generator 2	2	2	2	1	1
	Generator 3	3	2	2	1	1
stage 7-18 Daytime	Generator 1	2	2	2	1	1
	Generator 2	3	2	2	1	1
	Generator 3	3	3	2	2	1
stage 19-24 Night	Generator 1	1	1	1	0.5	0.5
	Generator 2	2	1	1	1	0.5
	Generator 3	2	2	1	1	1

Figure C.1: Production capacity of generators which are located at the root node, node 0.

Transition Probabilities of the Global Lattice For the global lattice, there are $5 \times 5 = 25$ outcomes at every stage $t = 2, \ldots, H$. These outcomes are indexed as shown in Figure C.2. Outcome 1 of the global lattice refers to a state that plentiful amounts of PV power are available while in outcome 25, the layers require huge demand. The coloring represents the *state* of each outcome. The blue area (outcome 1, 2, 3, 6 and 11) is a state where there is a low net demand, and refer to the state as Low. The deep orange blocks (outcome 7, 8, 9, 12, 13, 14, 17, 18 and 19) represent states where there exists moderate amount of net demand, while light orange area (outcome 4, 5, 10, 16, 21 and 22) is another moderate state. The two states are named as Mid 1 and Mid 2, respectively. Finally, red blocks (outcome 15, 20, 23, 24 and 25) represent states with high demand, and we call High state.

Lattice1\2	outcome 1	outcome 2	outcome 3	outcome 4	outcome 5
outcome 1	1	2	3	4	5
outcome 2	6	7	8	9	10
outcome 3	11	12	13	14	15
outcome 4	16	17	18	19	20
outcome 5	21	22	23	24	25

Figure C.2: Indexing of outcomes for the global lattice.

The transition probabilities of the global lattice is defined as shown in Figure C.3. Each value of row n and column m corresponds to the probability of the transition from outcome n to

outcome m. Note that the transition probabilities are identical from stage 2 to 24.

stage1->2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	0.06	0.04	0.02	0.008	0.004	0.04	0.11	0.09	0.04	0.008	0.02	0.09	0.15	0.09	0.02	0.008	0.04	0.09	0.03	0.004	0.004	0.008	0.02	0.004	0.002

stage t->t+1 (t>1)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	0.1	0.05	0.02	0.008	0.004	0.05	0.15	0.09	0.04	0.004	0.02	0.09	0.116	0.08	0.008	0.008	0.04	0.08	0.02	0.002	0.004	0.004	0.008	0.002	0.002
2	0.1	0.05	0.02	0.008	0.004	0.05	0.15	0.09	0.04	0.004	0.02	0.09	0.116	0.08	0.008	0.008	0.04	0.08	0.02	0.002	0.004	0.004	0.008	0.002	0.002
3	0.1	0.05	0.02	0.008	0.004	0.05	0.15	0.09	0.04	0.004	0.02	0.09	0.116	0.08	0.008	0.008	0.04	0.08	0.02	0.002	0.004	0.004	0.008	0.002	0.002
4	0.07	0.04	0.01	0.004	0.0001	0.04	0.1	0.058	0.04	0.004	0.01	0.058	0.131	0.058	0.01	0.004	0.04	0.058	0.1	0.04	0.0001	0.004	0.01	0.04	0.07
5	0.07	0.04	0.01	0.004	0.0001	0.04	0.1	0.058	0.04	0.004	0.01	0.058	0.131	0.058	0.01	0.004	0.04	0.058	0.1	0.04	0.0001	0.004	0.01	0.04	0.07
6	0.1	0.05	0.02	0.008	0.004	0.05	0.15	0.09	0.04	0.004	0.02	0.09	0.116	0.08	0.008	0.008	0.04	0.08	0.02	0.002	0.004	0.004	0.008	0.002	0.002
7	0.06	0.04	0.02	0.008	0.004	0.04	0.11	0.09	0.04	0.008	0.02	0.09	0.15	0.09	0.02	0.008	0.04	0.09	0.03	0.004	0.004	0.008	0.02	0.004	0.002
8	0.06	0.04	0.02	0.008	0.004	0.04	0.11	0.09	0.04	0.008	0.02	0.09	0.15	0.09	0.02	0.008	0.04	0.09	0.03	0.004	0.004	0.008	0.02	0.004	0.002
9	0.06	0.04	0.02	0.008	0.004	0.04	0.11	0.09	0.04	0.008	0.02	0.09	0.15	0.09	0.02	0.008	0.04	0.09	0.03	0.004	0.004	0.008	0.02	0.004	0.002
10	0.07	0.04	0.01	0.004	0.0001	0.04	0.1	0.058	0.04	0.004	0.01	0.058	0.131	0.058	0.01	0.004	0.04	0.058	0.1	0.04	0.0001	0.004	0.01	0.04	0.07
11	0.1	0.05	0.02	0.008	0.004	0.05	0.15	0.09	0.04	0.004	0.02	0.09	0.116	0.08	0.008	0.008	0.04	0.08	0.02	0.002	0.004	0.004	0.008	0.002	0.002
12	0.06	0.04	0.02	0.008	0.004	0.04	0.11	0.09	0.04	0.008	0.02	0.09	0.15	0.09	0.02	0.008	0.04	0.09	0.03	0.004	0.004	0.008	0.02	0.004	0.002
13	0.06	0.04	0.02	0.008	0.004	0.04	0.11	0.09	0.04	0.008	0.02	0.09	0.15	0.09	0.02	0.008	0.04	0.09	0.03	0.004	0.004	0.008	0.02	0.004	0.002
14	0.06	0.04	0.02	0.008	0.004	0.04	0.11	0.09	0.04	0.008	0.02	0.09	0.15	0.09	0.02	0.008	0.04	0.09	0.03	0.004	0.004	0.008	0.02	0.004	0.002
15	0.02	0.04	0.02	0.004	0.0001	0.04	0.15	0.125	0.04	0.004	0.02	0.125	0.175	0.07	0.01	0.004	0.04	0.07	0.02	0.004	0.0001	0.004	0.01	0.004	0.001
16	0.07	0.04	0.01	0.004	0.0001	0.04	0.1	0.058	0.04	0.004	0.01	0.058	0.131	0.058	0.01	0.004	0.04	0.058	0.1	0.04	0.0001	0.004	0.01	0.04	0.07
17	0.06	0.04	0.02	0.008	0.004	0.04	0.11	0.09	0.04	0.008	0.02	0.09	0.15	0.09	0.02	0.008	0.04	0.09	0.03	0.004	0.004	0.008	0.02	0.004	0.002
18	0.06	0.04	0.02	0.008	0.004	0.04	0.11	0.09	0.04	0.008	0.02	0.09	0.15	0.09	0.02	0.008	0.04	0.09	0.03	0.004	0.004	0.008	0.02	0.004	0.002
19	0.06	0.04	0.02	0.008	0.004	0.04	0.11	0.09	0.04	0.008	0.02	0.09	0.15	0.09	0.02	0.008	0.04	0.09	0.03	0.004	0.004	0.008	0.02	0.004	0.002
20	0.02	0.04	0.02	0.004	0.0001	0.04	0.15	0.125	0.04	0.004	0.02	0.125	0.175	0.07	0.01	0.004	0.04	0.07	0.02	0.004	0.0001	0.004	0.01	0.004	0.001
21	0.07	0.04	0.01	0.004	0.0001	0.04	0.1	0.058	0.04	0.004	0.01	0.058	0.131	0.058	0.01	0.004	0.04	0.058	0.1	0.04	0.0001	0.004	0.01	0.04	0.07
22	0.07	0.04	0.01	0.004	0.0001	0.04	0.1	0.058	0.04	0.004	0.01	0.058	0.131	0.058	0.01	0.004	0.04	0.058	0.1	0.04	0.0001	0.004	0.01	0.04	0.07
23	0.02	0.04	0.02	0.004	0.0001	0.04	0.15	0.125	0.04	0.004	0.02	0.125	0.175	0.07	0.01	0.004	0.04	0.07	0.02	0.004	0.0001	0.004	0.01	0.004	0.001
24	0.02	0.04	0.02	0.004	0.0001	0.04	0.15	0.125	0.04	0.004	0.02	0.125	0.175	0.07	0.01	0.004	0.04	0.07	0.02	0.004	0.0001	0.004	0.01	0.004	0.001
25	0.02	0.04	0.02	0.004	0.0001	0.04	0.15	0.125	0.04	0.004	0.02	0.125	0.175	0.07	0.01	0.004	0.04	0.07	0.02	0.004	0.0001	0.004	0.01	0.004	0.001

Figure C.3: Transition probabilities of the global lattice at every stage. Upper panel: form stage 1 to 2. Lower panel: from stage t to $t+1$ ($t = 2, \ldots, H$).

C.2 Selected Sample Description

We here give an explanation the sample that we use in section 7.1.3. The sample passes the outcomes shown in Figure C.4. Note that each outcome of the figure corresponds to that of the global lattice. We observe that the sample goes through both Low and High state once, and the majority of the outcome state is Mid 1. This fact implies that this sample is a typical one.

	outcome	state
stage1	1	initial
stage2	9	mid1
stage3	8	mid1
stage4	6	low
stage5	7	mid1
stage6	9	mid1
stage7	7	mid1
stage8	18	mid1
stage9	4	mid2
stage10	9	mid1
stage11	19	mid1
stage12	12	mid1
stage13	8	mid1
stage14	7	mid1
stage15	7	mid1
stage16	13	mid1
stage17	23	high
stage18	14	mid1
stage19	18	mid1
stage20	7	mid1
stage21	13	mid1
stage22	18	mid1
stage23	14	mid1
stage24	7	mid1

Figure C.4: Outcomes and the corresponding states of the selected sample.

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**