Supervised Learning for Sequential and Uncertain Decision Making Problems

Application to Short-Term Electric Power Generation Scheduling

Thesis by

Bertrand Cornélusse

Université de Liège
Faculté des sciences appliquées
October 2010

Contents

Résumé

Cette thï¿½se a pour origine une classe de problèmes pratiques de prise de décisions séquentielles dans le contexte de la gestion de la production d'électricité sous incertitudes. Ces problèmes sont généralement traités comme des problèmes d'optimisation déterministes résolus périodiquement (horizon glissant) et/ou comme des problèmes de programmation stochastique. La programmation stochastique permet de déterminer des décisions optimales compte tenu des évolutions possibles de l'environnement du parc de production (de diffiï¿½rents scï¿½narios) et d'ajuster ces décisions – s'il existe des possibilités de recours – après avoir collecté de l'information sur l'évolution réelle de l'environnement du système.

Bien que des techniques de décomposition existent, la programmation stochastique n'est actuellement pas réellement exploitable dans le cadre de la production d'électricité journalière ou infra journalière et, d'autre part, ne permet pas d'obtenir des stratégies de recours explicites. Cela rend également difficile l'ï¿½valuation de la valeur de cette approche sur des scénarios indépendants de ceux utilisï¿½s pour l'optimisation.

Nous proposons une méthodologie fondée sur l'apprentissage supervisé afin de dï¿½terminer une stratégie de recours explicite pour un plan de production donnï¿½, sur base d'ajustements optimaux du parc de production calculï¿½s pour une série de conditions perturbées de l'environnement du système. Cette méthode peut être considï¿½rï¿½e comme complémentaire à une approche par programmation stochastique. Dans le contexte d'une optimisation à horizon glissant, notre méthode permet de rï¿½aliser une partie significative des calculs en avance, tout en offrant la possibilité d'infï¿½rer rapidement des décisions d'ajustement durant l'exploitation du parc de production. Notre mï¿½thode peut facilement ï¿½tre validï¿½e hors ligne sur un ensemble de scénarios indépendants. En outre, comme la stratégie de recours généralise les informations contenues dans les décisions relatives à des instances perturbées du problème d'optimisation du plan de production, elle fournit une certaine robustesse par rapport à l'incertitude sur les paramètres du problème. À cet égard, nous bénéficions des propriétés de régularité de certains algorithmes d'apprentissage supervisé.

Sur une instance réaliste du problème d'ajustement de la production d'électricité en mode infra journalier, nous montrons comment générer des scénarios de perturbations, comment calculer les ajustement optimaux, comment formuler le problème d'apprentissage supervisé pour obtenir une stratégie de recours, comment restaurer la faisabilité des ajustements prédits et enfin com-

ment évaluer la stratégie de recours sur un ensemble de scénarios indépendants. Nous analysons diverses formulations du problï¿½me d'apprentissage, à savoir soit le problï¿½me consistant ï¿½ prï¿½dire l'ajustement détaillé de toutes les unités de production, soit le problï¿½me consistant ï¿½ prédire des variables qualitatives qui permettent d'accélérer la procédure de calcul d'ajustement du plan de production en facilitant le problème d'optimisation sous-jacent.

Notre approche est intrinsèquement adaptable ï¿½ des problèmes de gestion de production de grande échelle, et peut en principe traiter toutes sortes d'incertitudes et de contraintes d'ordre pratique. Nos résultats montrent la faisabilité de notre mï¿½thode et sont également très prometteurs en termes d'efficacité économique des stratégies d'ajustement. Notre approche peut également fournir des informations interprétables sous la forme de mesures d'influence de différents paramètres sur les stratégies de d'ajustement.

Les solutions du problème d'optimisation de la production doivent satisfaire des contraintes de diverses natures. Bien que ces contraintes ne soient pas prises en compte par les algorithmes d'apprentissage classiques, un algorithme d'apprentissage qui ignore ces contraintes parvient tout de mï¿½me à modï¿½liser la sensibilité de la solution aux paramètres du problème de maniï¿½re satisfaisante. Cela a néanmoins attirï¿½ notre attention sur un aspect particulier de la relation entre les algorithmes d'apprentissage et les algorithmes d'optimisation. Lorsqu'on applique un algorithme d'apprentissage supervisï¿½ qui cherche dans un espace d'hypothèses ï¿½ partir de données qui satisfont un ensemble connu de contraintes, peut-on garantir que l'hypothèse sélectionnï¿½e prï¿½dira des sorties qui satisfont les contraintes ? Pouvons-nous au moins profiter de notre connaissance des contraintes afin d'éliminer certaines hypothèses durant l'apprentissage, et par consï¿½quent espï¿½rer que l'hypothèse choisie a une meilleure capacitï¿½ en généralisation ?

Dans la deuxième partie de cette thèse, oï¿½ nous tentons de rï¿½pondre ï¿½ ces questions, nous proposons une extension générique des méthodes ï¿½ base d'ensembles d'arbres de rï¿½gression qui permet d'incorporer des données incomplètes, mais aussi des connaissances a priori sur le problème. Cette approche consiste ï¿½ rï¿½soudre un problème d'optimisation convexe qui a pour but de régulariser un ensemble d'arbres en ajustant les étiquettes assignï¿½es aux feuilles de l'ensemble d'arbres de régression, et/ou les sorties des observations de l'échantillon d'apprentissage. Notre formulation permet d'incorporer des informations supplémentaires telles que de l'information partielle sur les étiquettes de sortie (ce qui arrive en pratique quand les données sont censurées ou en apprentissage semi-supervisé), des observations dont la prï¿½cision est variable, ou encore un a priori fort sur la structure du modèle recherché.

En plus d'améliorer la précision en exploitant des informations qui ne peuvent ï¿½tre exploitï¿½es par les mï¿½thodes classiques, l'approche proposée peut être utilisée pour produire des modèles qui, naturellement, se conforment mieux aux contraintes de faisabilité devant être satisfaites dans de nombreux problï¿½mes de prise de décisions, spécialement dans des contextes oï¿½ l'espace de sortie est de grande dimension et/ou structuré par des invariances, des symétries et d'autres types de contraintes.

Abstract

Our work is driven by a class of practical problems of sequential decision making in the context of electric power generation under uncertainties. These problems are usually treated as receding horizon deterministic optimization problems, and/or as scenario-based stochastic programs. Stochastic programming allows to compute a first stage decision that is hedged against the possible futures and – if a possibility of recourse exists – this decision can then be particularized to possible future scenarios thanks to the information gathered until the recourse opportunity.

Although many decomposition techniques exist, stochastic programming is currently not tractable in the context of day-ahead electric power generation and furthermore does not provide an explicit recourse strategy. The latter observation also makes this approach cumbersome when one wants to evaluate its value on independent scenarios.

We propose a supervised learning methodology to learn an explicit recourse strategy for a given generation schedule, from optimal adjustments of the system under simulated perturbed conditions. This methodology may thus be complementary to a stochastic programming based approach. With respect to a receding horizon optimization, it has the advantages of transferring the heavy computation offline, while providing the ability to quickly infer decisions during online exploitation of the generation system. Furthermore the learned strategy can be validated offline on an independent set of scenarios. Furthermore as the recourse strategy generalizes the information contained in decision sequences relative to perturbed instances of the generation scheduling problem, it provides some robustness with respect to uncertainty on problem parameters. In this respect, We benefit of the regularity properties of some supervised learning algorithms.

On a realistic instance of the intra-day electricity generation rescheduling problem, we explain how to generate disturbance scenarios, how to compute adjusted schedules, how to formulate the supervised learning problem to obtain a recourse strategy, how to restore feasibility of the predicted adjustments and how to evaluate the recourse strategy on independent scenarios. We analyze different settings, namely either to predict the detailed adjustment of all the generation units, or to predict more qualitative variables that allow to speed up the adjustment computation procedure by facilitating the "classical" optimization problem.

Our approach is intrinsically scalable to large-scale generation management problems, and may in principle handle all kinds of uncertainties and practical

constraints. Our results show the feasibility of the approach and are also very promising in terms of economic efficiency of the resulting strategies. Our approach may also provide interpretable information in the form of measures of influence of different parameters on the decision strategies.

The solutions of the optimization problem of generation (re)scheduling must satisfy many constraints. However, a classical learning algorithm that is (by nature) unaware of the constraints the data is subject to may indeed successfully capture the sensitivity of the solution to the model parameters. This has nevertheless raised our attention on one particular aspect of the relation between machine learning algorithms and optimization algorithms. When we apply a supervised learning algorithm to search in a hypothesis space based on data that satisfies a known set of constraints, can we guarantee that the hypothesis that we select will make predictions that satisfy the constraints? Can we at least benefit from our knowledge of the constraints to eliminate some hypotheses while learning and thus hope that the selected hypothesis has a better generalization error?

In the second part of this thesis, where we try to answer these questions, we propose a generic extension of tree-based ensemble methods that allows incorporating incomplete data but also prior knowledge about the problem. The framework is based on a convex optimization problem allowing to regularize a tree-based ensemble model by adjusting either (or both) the labels attached to the leaves of an ensemble of regression trees or the outputs of the observations of the training sample. It allows to incorporate weak additional information in the form of partial information about output labels (like in censored data or semi-supervised learning) or – more generally – to cope with observations of varying degree of precision, or strong priors in the form of structural knowledge about the sought model.

In addition to enhancing the precision by exploiting information that cannot be used by classical supervised learning algorithms, the proposed approach may be used to produce models which naturally comply with feasibility constraints that must be satisfied in many practical decision making problems, especially in contexts where the output space is of high-dimension and/or structured by invariances, symmetries and other kinds of constraints.

Acknowledgements

First of all I would like to express my gratitude to my advisor Prof. Louis Wehenkel. He was at the origin of this research, identified my wish to work on an applied problem and connected me with the research environment of EDF. He transmitted me the taste for research, especially in machine learning and optimization and encouraged me all along the past four years. He impressed me by the quality of his guidance and his reactivity despite his very busy agenda.

I sincerely thank the members of the jury for devoting themselves to the reading of this manuscript and to all the related administrative duties.

I am grateful to Damien Ernst, Pierre Geurts (also for the source code of the Extra-Trees) and Quentin Louveaux for their valuable advices on various aspects of my research, as well as the members of the OSIRIS team from EDF and especially to (in chronological order) Yannick Jacquemart, Patrick Pruvot, Gérald Vignal, Jérome Quenu, Vincent Grellier, Céline Le Goazigo, Jérome Collet and Ala Ben Abbes for welcoming me for several meetings and stays, for the interesting discussions we had about the various topics of electricity generation scheduling and for the material they provided me.

During these four years I was fortunate to be financed by the Belgian Fund for Research in Industry and Agriculture (FRIA) and to be hosted by the University of Liï¿½ge. I also thank EDF for their financial support during my stays in Paris.

I would like to thank my colleagues and friends of the System and Modeling research unit and more broadly of the Montefiore Institute for their support and for the interesting discussions we had together during the last four years: especially Florence Belmudes, Arnaud Declercq, Boris Defourny, Renaud Detry, Raphael Fonteneau, Marie-Berthe Lecomte, Thibaut Libert, Laurent Poirrier, Diane Zander and all the others that I did not mention.

Finally I thank all my family, especially my father who has given me the taste of science and engineering and persuaded me to accomplish a Ph.D., my mother for the emotional support and of course my everyday and invaluable support, Laurie, to whom I dedicate this work.

Foreword

This thesis is the result of a collaboration with Électricité de France (EDF) for studying the application of machine learning to the electricity generation scheduling problem. We have voluntarily focused on one particular aspect of the problem that EDF is confronted to, namely the intra-day generation rescheduling problem, although we believe that machine learning may be of more general interest. In general the family of problems we are interested in are treated using independently the tools of statistics and mathematical programming. Our goal is to use machine learning tools and concepts to bring together these domains by somewhat merging the analysis of uncertainties impacting the problem and the process of finding near optimal solutions to the problem. Another attempt to bridge these domains from another perspective comes from the field of stochastic programming. The conceptual differences with our approach will be explained in this thesis.

This collaboration extends a long-standing relationship between the University of Liï¿½ge and EDF. My advisor Prof. Louis Wehenkel has devoted his earlier work to the application of Machine learning to dynamic security assessment of electrical power systems. The present thesis is also at the confluence of the topics treated by other researchers of the Systems and Modeling research unit of the University of Liï¿½ge, especially Pierre Geurts who has developed and analyzed the Extra-Trees supervised learning algorithm, Damien Ernst who is working on optimal control problems from the reinforcement learning point of view and Quentin Louveaux whose research topic is mixed-integer programming, i.e. the domain of mathematical programming that is currently used to solve the family of problems we are focusing on in this thesis.

Chapter 1

General introduction

Our work is driven by a class of practical problems of sequential decision making in the context of electric power generation under uncertainties. As we will see, these problems may be formulated either in a probabilistic setting, in which case they typically lead to an objective of minimizing an expected cost over a *distribution* of scenarios, or in a robust framework, where they lead to an objective of minimizing the maximal cost and/or ensuring feasibility over a *set* of scenarios.

In practice, these problems are natively very-high-dimensional and non-convex; while their deterministic counterpart may often be satisfactorily formulated as a large-scale mixed integer linear programming problem (MILP) and solved with existing state-of-the-art solvers, no satisfactory direct optimization strategy for solving the uncertain (robust or stochastic) counterparts of such problems is yet available from the shelf.

Our research goal was to investigate the possible uses of combinations of simulation, optimization, and machine learning, in order to help providing better solutions for such problems. Specifically, we have considered the use of tree-based supervised learning in order to offline determine decision strategies for online use in intra-day rescheduling of electric power generation, to help operators to react to deviations from planned conditions.

Our empirical investigations, reported in Part I of this thesis, show on a realistic test system that it is indeed possible to exploit supervised learning in this specific context in a practically useful way, by

- using Monte-Carlo simulations, in order to generate sets of scenarios representing the possible deviations of the next day's operating conditions with respect to the nominal conditions,
- computing for each scenario optimal intra-day adjustments of the nominal generation plan, by using state-of-the-art optimization solvers,
- applying machine learning algorithms based on ensembles of regression trees, in order to extract intra-day recourse strategies from the simulated datasets, and combining them with simple post-processing algorithms in order to restore feasibility.

While our work was mainly driven by this specific problem, we believe that our contributions are of general interest and could be applied in other contexts of planning under uncertainty and/or solving complex optimization problems. We therefore start in the first section of this chapter by describing in an abstract way the types of problems addressed, the proposed solution strategy, its open questions and its overall motivations. Then we will outline the main mathematical formulations of sequential decision making problems under uncertainty and introduce in an intuitive way some important ideas from the machine learning domain, in order to settle notations and provide the required background for understanding the rest of this thesis. We proceed by describing some practical examples of problems that fall in the general class of problems addressed by our work and conclude this chapter by explaining the organization of the rest of the manuscript and by stating our contributions.

1.1 Abstract view of the proposed approach

1.1.1 Families of similar optimization problems

We consider families of optimization problems, parameterized by a vector $p \in P \subset \mathbb{R}^{n_p}$ of parameters, with a decision space $U(p) \subset \mathbb{R}^{n_u}$, a performance criterion $f(u,p) : \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}$, and some inequality constraints $g(u,p) \geq 0$ (g is also a vector, say of dimension n_g). In a given family, the optimization problems have to be repeatedly solved, by repeatedly computing for different values of $p \in P$ an optimal decision

$$u^*(p) \in \arg_{\{u \in U(p) : g(u,p) \geq 0\}} \min f(u,p).$$

We assume that for a given family of optimization problems its description is completely known (meaning that functions f, g, and spaces P and $U(p)$ are known) and that we dispose of an optimization algorithm (denoting $\{P, U(p), f, g\}$ by z) $A_o^z(p)$ able to compute for each problem instance p a "point-wise" solution $u^*(p)$ or at least a "sufficiently accurate" approximation $u^*(p) + \epsilon$ (approximation that we will denote by $\tilde{u}(p)$ in the sequel).

With respect to the standard way of looking at optimization problems, the distinguishing feature of our setting is that instead of considering a single instance of an optimization problem, we consider solving a family of similar problems parameterized by a vector p residing in an a priori specified subset of parameter values P. By doing so, we hope to be able to exploit similarities of solutions of different problem instances in a same family so as to improve overall solution quality and/or speed of computation.

Example 1.
In day-ahead electric power system generation planning, every day, an optimization problem is solved for deciding on the operation schedules of the different (say n_{gen}) plants during the next day (say every 30 minutes); together these form the decision variable u (in practice $n_u = 48 \times n_{gen}$, i.e. in the order of a few thousand components for a large company such as EDF), while p in this case would represent a vector describing the forecasted system capacity requirements and generator availabilities specified for each time step (say $n_p = (n_{gen} + 3) \times 48$, or even more). The performance

criterion f would here be the cost of operation integrated over the next day, and the hard constraints g would reflect the technical constraints such as minimum down/up-time, ramping constraints, emission constraints, demand covering constraints, etc.

Example 2.
In intra-day operation of a power generation company, a set of such problem families need to be solved, one for each recourse opportunity, allowing to adapt the schedule to the information collected during operation (p would then denote deviations from forecasts re-estimated at the time of recourse) while u would become the vector of generation schedule adjustments over the remaining time horizon. Problem complexity, performance criteria, and hard constraints would essentially be similar to those used in day-ahead planning.

Example 3.
In medium-term (seasonal, or yearly) planning of electric power generation, a set of such problems would have to be solved every week, month, or seasonally, to determine optimal strategies for purchasing fuel, shutting down plants for maintenance, exploiting hydroelectric reservoirs, establishing forward contracts etc, over the next horizon. In this context, the optimization horizon would typically span over several weeks or months, with a predominantly economic criterion. Problem instances p would need to take into account the stochasticity of the environment, and thus each problem instance p would be described for example by a "tree" of possible future realizations of power demand and plant availabilities, water in-flows, temperatures, prices, etc., over the optimization horizon, while decision variables would correspond to the planned daily operation of the generators over the optimization horizon.

Each one of these three example problems corresponds to a different setting, but each one of them needs to be solved repeatedly in practice, for different values of the parameter, say every day, every 30 minutes, or every week or month. Here, the need for re-solving different problem instances is induced by the fact that, as time proceeds, additional information becomes available that can be fruitfully exploited in order to revise decisions that could not be anticipated for correctly. As a matter of fact, the complexity of these problems is such that with bounded computational resources it is not possible to determine once and for all a 'no-regret' decision strategy. Thus, computational resources are used in a time-receding fashion in a race between optimization and physical information flow. Notice that there exist many other classes of problems where repeated solutions are necessary, such as for instance problems where one wants to assess the influence of parameters on decisions by solving different problem instances for different values of parameters. In this case, one disposes as well of a specific "computing budget".

1.1.2 Generalizing solutions by supervised learning

Assuming that we have already solved a number N of problems using A_o^z (for fixed z, but for different values of p), which has yielded a dataset $D = \{(p_i, \tilde{u}_i)\}_{i=1}^N$ of problem-solution pairs, it is possible to exploit supervised learning (SL) in order to determine an approximation of A_o^z.

Generically, supervised learning (in batch mode) operates in the following way: given an input space $\mathcal{X}$, an output space $\mathcal{Y}$, a sample (dataset)

$\{(x_i, y_i)\}_{i=1}^N$, a hypothesis space of input-output functions $\mathcal{H} \subset \mathcal{Y}^{\mathcal{X}}$, and a loss-function $\ell \in R_+^{(\mathcal{Y}\times\mathcal{Y})}$, it determines an input-output function (a hypothesis)

$$h_D \in \arg\min_{h\in\mathcal{H}} \sum_{i=1}^{N} \ell(y_i, h(x_i)),$$

or at least a good approximation of it.

We will denote by $A_l^{\mathcal{H}} \in \mathcal{H}^{(\mathcal{X}\times\mathcal{Y})^*}$ such a supervised learning algorithm mapping any dataset D of any size N over $(\mathcal{X} \times \mathcal{Y})$ on an element of $\mathcal{H}$.

Here we propose to apply supervised learning to datasets provided by optimization algorithms repeatedly applied to different instances of a same family of optimization problems, by substituting x_i with p_i and y_i with $\tilde{u}_i$ and selecting an appropriate hypothesis space and search strategy.

Under suitable conditions, namely, if the dataset D is representative enough of the set of problem instances, if the optimization algorithm $A_o^z(p)$ is accurate enough, and if the supervised learning algorithm $A_l^{\mathcal{H}}$ is well adapted to the target strategy $u^*(\cdot)$ and dataset size, this approach may lead to good quality approximations $h_D(\cdot)$ of $u^*(\cdot)$.

1.1.3 Combining simulation, optimization, and learning

We consider situations where the generation of the dataset D is part of the overall solution. The design of a solution strategy is thus based on three main choices, namely

1. Simulation: how to generate problem instances $\{p_1, \ldots, p_N\}$.
2. Optimization: how to compute the solutions $\{\tilde{u}_1, \ldots, \tilde{u}_N\}$ to yield D.
3. Supervised learning: choosing $\mathcal{H}$ and its search strategy to get h_D.

The different design choices for these three steps should ideally be adapted to the practical problem of concern, so as to yield the best possible solution given the available computing budget.

In order to verify the quality of the computed approximations it is also necessary to develop a suitable validation strategy, so as to be able to compare in a sound way alternative solutions produced by alternative approaches. We will see in the subsequent chapters that the question of validation, already well studied in the context of supervised learning, needs further consideration in the context of our "learning to optimize" framework.

1.1.4 Overall motivations of the research theme

The interests of combining a supervised learning based approach with optimization are manifold. While, we will review them in the concluding chapter of this thesis, we state them now in order to clarify our motivations.

Speed up the search for an optimal response. First, from a computational point of view, one may hope that by using supervised learning based approximations h_D to the available direct optimization-based solution A_o^z, it may be possible to significantly reduce the response time needed to obtain a decision once a new value of p is presented. This may allow leveraging algorithmic solutions available in offline environments to real-time contexts. Thus the supervised learning based approach may be a successful strategy (among others) to significantly speed up rational decision making. In the context of intra-day generation scheduling this is a main goal, and we will assess computational speed-up in this context in Part I. Note that in this thesis we apply the supervised learning based approach to a MILP context where it is very hard to obtain information about the sensitivity of the solution to problem parameters. However this may also be of interest in a linear or convex programming context where we know how the solution and the objective vary with small variations of the parameters, but have only qualitative information about the global behavior of the solution.

Measure the influence of parameters. Second, since supervised learning looks at the global relationship between p and u^*, it may allow one to assess which of the components of p indeed influence the solution u^* in a significant way; indeed the supervised learning methodology includes methods that allow one to assess the importance of input variables and further to select a small subset of them that are sufficient to infer the output in a reliable way. From a practical point of view, this allows to reduce the complexity of the solution strategy and could help human experts to focus on the most important aspects of their problem. Both of these features may help engineers to better formulate their optimization problems and operators to become more confident in their solution strategies and more effectively justify them.

Improve optimization based solutions thanks to the properties of SL algorithms. Since supervised learning is already a very rich and mature, while still extremely lively field of investigation both from a theoretical and from a practical point of view, all its ongoing progresses may be exploited in the future. In particular, supervised learning theory provides guaranties about noise filtering, meaning that one can show that under suitable conditions the algorithms are able to recover near-optimal solutions even if the samples of the dataset are corrupted by noise. In practice this may help to find better solutions to optimization problems for which no tractable and sufficiently accurate direct solution algorithm is yet available.

Incorporate prior knowledge in the learning problem. From a more "machine learning oriented" point of view, recent developments (including those that will be developed in Part II of this thesis) now allow to exploit explicitly constraints that need to be satisfied by the decision strategy, so as to both improve accuracy of the inferred hypotheses and to make them more readily useful in practice. These constraints may be formulated jointly over input-output pairs or solely

among the different components of the output vector (as, e.g., in structured prediction methods to be shortly discussed in the next subsections).

1.1.5 Methodological approach

In order to carry out our research, we used a problem-driven approach. We started our work with the analysis of various electric power generation problems, in order to select a specific practical problem of interest for the industry; this work was done in collaboration with EDF and led to the choice of the intra-day rescheduling problem. Subsequently, we studied the existing approaches (in optimization and in supervised learning) so as to formulate a viable framework for combining these latter in order to address the selected problem.

Eventually, we decided to exploit existing open-loop optimization of a deterministic formulation of the problem as the basis for generating input-output samples of scenario-decision pairs and analyzed how (and which) available supervised learning algorithms could cope with such datasets in order to infer decision strategies for a large set of electric power generating units based on the given information state. Specifically, we considered a MILP formulation of the deterministic planning problem and based most of our investigations on supervised learning methods using ensembles of decision/regression trees (with some side experiments using linear, support-vector based, approaches).

We have then set-up a test-system with the help of engineers of EDF, deemed representative of their practical problems, and we have established a protocol for simulating datasets, for inferring from them decision strategies by supervised learning, and for evaluating these latter with respect to alternative ones. In this context, a large part of our work was devoted to developing software and using it for generating representative datasets and exploiting them to assess our supervised learning based approach on variants of the test system and for various classes of uncertainties.

Let us notice that the dimensions of the input and output spaces of the induced decision problems are of large scale (several thousands of dimensions for either space). Also, the golden standard optimization frameworks for our problem (multistage stochastic/robust programming) could not be used in order to generate datasets and/or to benchmark induced decision strategies, because no satisfactory algorithmic solution was available for them. This means that we had to address an unusually complex set of supervised learning problems, while being unable to generate representative samples of problem instances with an exact solution of them. To overcome this difficulty, a significant part of our research effort has been devoted to the generation of good quality input-output samples, formulating manageable supervised learning problems, and designing a sound and feasible evaluation methodology.

In order to introduce the reader to the kind of problems and methods investigated in this thesis, we next provide some background material about problem formulations and solution strategies respectively about direct optimization (Section 1.2) and supervised learning (Section 1.3).

1.2 Background in dynamic optimization under uncertainty

In this section we adopt an optimal control formalism to solve a sequential decision making problem under uncertainties. We consider a dynamic system that must be controlled over the course of time in order to satisfy some constraints and optimize some performance criterion (typically, minimization of economic cost of operation). The system lives in a stochastic environment, meaning that in addition to the control strategy, its dynamic trajectory is also influenced by the realizations of some random disturbance processes; the environment is exogenous, which means that these disturbances are not themselves influenced by control decisions. But these disturbances may render the usage of a fixed open-loop control sequence computed beforehand suboptimal, or even lead to catastrophic situations. It is therefore needed to define a control strategy allowing to adapt the controls at successive recourse stages, by taking into account the information that can be collected at each stage about the realization of the disturbance process issued by the environment.

The type of system we consider is too complex to be controlled using a classical PID regulator and would typically rather require human intervention to define and implement the control decisions. For example we may consider a large-scale system governed by nonlinear dynamics or containing complex time-varying operation modes. We assume that we know a good model of the system and of the performance criterion and that we dispose of information characterizing the environment behavior, either in the form of a set of possible disturbance scenarios, or in the form of a statistical model of the disturbance processes.

Below, we formalize this type of problem mathematically using some optimization formulations of increasing richness and complexity.

1.2.1 Adopting a deterministic view of the environment

Let us first assume that the environment is modeled by the decision maker in a deterministic way, i.e. without taking into account uncertainties. Thus the set of possible disturbance process realizations reduces for the decision maker to a single scenario that he assumes known beforehand.

1.2.1.1 Computation of an optimal open-loop control sequence

Computing an open-loop sequence of decisions for controlling optimally a system whose model is known (including that of the environment) – according to predefined criteria and for a horizon of T time steps – can be formulated, for example, as the optimization problem of Formulation 1.

We consider a discrete time decision process and denote by t the index of a time period[1]. The state variables at time t are gathered in the vector x_t, which is allowed to reside within a set $\mathcal{X}_t \subset \mathbb{R}^K \times \mathbb{Z}^L$; the set $\mathcal{X}_0$ may reduce to a singleton. We assume that the state vector is designed to contain enough information on the past of the system so as to avoid constraints implying the

[1] t is nevertheless referred to as "time" in this document.

Formulation 1: Deterministic open-loop control.

$$\min_{x,u} \quad \sum_{t=1}^{T} \|C_t x_t - Y_t\| + \alpha \sum_{t=0}^{T-1} R_t(x_t, u_t) \tag{1.1}$$

$$\text{s.t.} \quad x_{t+1} = F_t(x_t, u_t),\ \forall t \in \{0, ..., T-1\}, \tag{1.2}$$

$$x_t \in \mathcal{X}_t,\ \forall t \in \{0, ..., T\}, \tag{1.3}$$

$$u_t \in \mathcal{U}_t(x_{t-1}),\ \forall t \in \{0, ..., T-1\}, \tag{1.4}$$

state variable at more than two consecutive time steps (Markov property). The control at time t

$$u_t \in \mathcal{U}_t(x_t) \subset \mathbb{R}^I \times \mathbb{Z}^J$$

is typically composed of continuous and discrete control variables. It acts on the value of the state at time t through the function

$$F_t : \mathcal{X}_t \times \mathcal{U}_t \to \mathbb{R}^K \times \mathbb{Z}^L,$$

which represents the dynamics of the system. Globally, $\mathcal{X} = \mathcal{X}_0 \times \ldots \times \mathcal{X}_T$ represents the space of all possible evolutions of the state from $t = 0$ to $t = T$. As a shortcut we use $x \in \mathcal{X}$ to denote $x_{[0:T]} = (x_0, \ldots, x_T)$. Similarly, $\mathcal{U}(x) = \mathcal{U}_0(x_0) \times \ldots \times \mathcal{U}_{T-1}(x_{T-1})$ represents the space of all possible evolutions of the controls from $t = 0$ to $t = T - 1$, and $u \in \mathcal{U}(x)$ is a shortcut for $u_{[0:T-1]} = (u_0, \ldots, u_{T-1})$. The real matrix C_t maps the state vector into the quantities of outputs it corresponds to, and $Y_t \in \mathbb{R}^p$ is a vector of desired outputs at time t. The goal is thus to minimize a trade-off – tuned by the meta-parameter α – between a term which penalizes the discrepancy between the produced and desired outputs – according to a custom norm – and a term representing the cost of using u as sequence of controls. The second term implies the functions

$$R_t : \mathcal{X}_t \times \mathcal{U}_t \to \mathbb{R}$$

and depends on the state, since the same action taken at a certain time step while in different system states may lead to different costs.

1.2.1.2 Receding horizon model predictive control

Formulation 1 assumes that the exogenous environment behavior can be perfectly predicted, and the formulation thus incorporates these predictions in the definitions of the functions F_t, R_t, parameters C_t and Y_t and constraint sets $\mathcal{X}_t$ and $\mathcal{U}_t$. When the environment is not perfectly deterministic, such an open-loop strategy must be based on a best guess of the environment. Depending on the precise formulation of system dynamics, performance criterion, and environment, it may be difficult to define what would be the best guess for the problem; moreover, any single best guess formulation may lead to control

sequences that under some possible scenarios would drive the system along undesired or highly suboptimal trajectories.

Receding horizon model predictive control (Maciejowski, 2002) aims at circumventing these difficulties by allowing the periodical revision of the open-loop control strategy, by recomputing at each time step (or at a subset of them) the solution of the optimization problem over the remaining steps based on information gathered about the current system state (and possibly, by revising the predicted best guess of the environment behavior).

However driving the system by using such a deterministic receding horizon procedure may still lead to states which are costly to adjust, or even possessing no feasible control solution. It is thus maybe preferable or even necessary to link explicitly the information about the uncertain environment, system dynamics and control objective in order to obtain better ways to formulate the optimal control problem and compute its solutions.

1.2.2 Taking into account environment uncertainty

In the previous section, decisions are computed by either assuming that the environment is deterministic, or that near-optimal decisions can be computed (and later-on revised in a receding way) by assuming that the environment behavior can be compactly represented by a single "nominal" scenario over the remaining time steps, and consequently without taking into account the way subsequent decision steps could revise the control decisions.

When this strategy is not appropriate, one can explicitly model the uncertainty about the environment behavior and/or the fact that decisions may be revised based on the information that will be gathered later-on about the environment behavior.

In the present section we discuss formulations which consider explicitly environment uncertainties (we will illustrate this in a robust open-loop formulation) and the possibility to revise decisions (we will illustrate this in a multistage stochastic programming formulation).

For $t = 0, 1, \ldots, T$, we let ξ_t be a variable valued in a subset Ξ_t of a Euclidean space: ξ_t denotes the uncertain disturbances representing the environment at time t which we suppose to be observable by the decision maker at time $t+1$. Hence, $\xi_{[0:t-1]} = (\xi_0, \ldots, \xi_{t-1})$ represents the information collected by the decision maker about the environment behavior at time t, when he is supposed to choose the control input u_t. Globally, $\Xi = \Xi_0 \times \ldots \times \Xi_T$ represents the space of all possible evolutions of the environment ξ_t from $t = 0$ to $t = T$. As a shortcut we use $\xi \in \Xi$ to denote $\xi_{[0:T]}$.

In a robust approach, we will assume that the sole knowledge exploited by the decision maker is the set Ξ of possible disturbances, while in a stochastic approach, we will assume that a complete probabilistic model is also available in the form of a joint probability distribution over the random variables $\xi_{[0:T]}$. In a single stage approach, we will assume that the decision maker does not explicitly take into account the subsequent decision making steps, i.e. he commits decisions at early steps while neglecting his ability to revise decisions based on later observations of the environment. In a multistage approach, we

will explicitly model the recourse opportunities at later stages in order to select first-stage decisions.

To highlight these ideas, we first consider the single stage decision making approach under uncertainty in the robust framework, then we consider the multistage formulation in the stochastic setting, and finally we wrap up by briefly discussing the other possible combinations of these two approaches.

1.2.2.1 Robust single stage decision making under uncertainty

In order to take into account uncertainties in the decision making procedure, let us first consider how to compute an open-loop control sequence while ensuring that under the worst environmental conditions, the system is still able to comply with all constraints and would have an optimal performance. More precisely, we would like to compute a sequence of decisions u such that all the scenarios $\xi \in \Xi$ are manageable and that cost is minimized for the worst of them. We propose the following formulation to start our discussion.

Formulation 2: Robust single stage decision making problem.

$$\min_{x(\xi),u} \quad \max_{\xi\in\Xi} \left\{ \sum_{t=1}^{T} \|C_t x_t(\xi) - Y_t(\xi)\| + \alpha \sum_{t=0}^{T-1} R_t\left(x_t(\xi), u_t, \xi\right) \right\} \tag{1.5}$$

$$\text{s.t.} \quad \forall \xi \in \Xi, \forall t \in \{0, ..., T-1\},$$

$$x_{t+1}(\xi) = F_t(x_t(\xi), u_t, \xi), \tag{1.6}$$

$$\forall \xi \in \Xi,$$

$$x_t(\xi) \in \mathcal{X}_t, \ \forall t \in \{0, ..., T\}, \tag{1.7}$$

$$u_t \in \mathcal{U}_t(x_t(\xi)), \ \forall t \in \{0, ..., T-1\}. \tag{1.8}$$

With respect to Formulation 1, we still search for an open-loop control sequence u since we cannot revise our decisions, but the trajectory of the state $x(\xi) = (x_0(\xi), x_1(\xi), \ldots, x_T(\xi)))$ is particularized to each possible realization of the disturbance process, since F_t are now also function of ξ (1.6). The state at each time t must reside in an acceptable set $\mathcal{X}_t$ whatever the realization of the disturbance process (1.7), and the control sequence must be compatible with the value of the state (1.8). In the objective function (1.5), the targeted outputs Y_t and the penalization functions R_t are also function of ξ. Taking the maximum cost over all possible realizations of ξ provides a very conservative solution minimizing the effect of the worst possible realization. But in some applications it is necessary to ensure a minimal risk even if the resulting average cost is high, so that this formulation may be appropriate. Nevertheless, it is possible to derive variations of this formulation where the cost function depends only on a fixed "best guess" scenario, so that the solution of the problem would be optimal with respect to this best guess while being compliant with the constraints induced by all other possible scenarios. Notice also that the

formulation implies a constraint set for each possible scenario, which could lead in practice to an infinite number of constraints.

Rather than adopting a worst case or a nominal case based objective function, one could replace the "max" operator in the objective function by an expectation over a distribution of scenarios. This formulation would lead to a single stage stochastic programming formulation aiming to compute an open loop decision sequence which would comply in terms of constraints with all possible scenarios and lead to a minimal expected cost given the probability distribution of the environment behaviors. Taking the expectation needs additional information about the likelihood of scenarios, but typically leads to a less conservative solution whose average value is better than the one obtained with the max operator. One can also add other penalization terms to account for the variability of the solution (Value at Risk, Conditional Value at Risk, ..., see Shapiro et al. (2009, Chapter 6) for a comprehensive text on the subject in the case of convex programming and Märkert and Schultz (2005) in the case of Mixed Integer Programming).

Earlier remarks about the receding horizon approach discussed in Section 1.2.1.2 may be carried over to the single stage under uncertainty setting of the present section. A main difference is that in the present approach first stage decisions are computed in order to be compliant with constraints that may appear in later stages given the environment evolution, hence the term "robust planning".

1.2.2.2 Stochastic decision making with recourse

While in the above formulation uncertainties were explicitly considered in the model in order to provide robustness to the sequence u, the Formulation 2 is actually over-constrained, because it computes decisions as if they needed to be chosen once and for all in an open loop fashion and could not be modified later-on after having observed the partial realization of ξ.

As explained previously, in practice information may be gathered through time about the environment behavior, and this information will normally be used in one or another way by the decision maker to revise his decisions. The ability to revise decisions at later stages may have a strong impact on what should be considered as the optimal decisions at earlier stages. In order to incorporate this fact into the optimization problem, the formulation of a multistage approach is required, which has the essential feature that first stage decisions can be chosen without assuming that later decisions have to be chosen fully in advance, but instead can take into account the fact that the latter decisions can be adjusted later on in a way contingent on information collected at the later stages.

Multistage decision making problem formulations consist in defining one or more stages of recourses over the time interval; at each stage the control decisions corresponding to the current and future time steps can be revised and re-optimized given the information collected about the environment behavior. This leads to the necessity of optimizing over control strategies for later time periods, i.e. functions mapping information states (measurements about the

environment and the controlled system) towards decisions. This problem is notably more difficult than those of finite-dimensional optimization considered in the previous sections.

Below, we illustrate some formulations of such multistage problems within the stochastic programming framework. They could as well be adapted to the robust setting.

A single opportunity of recourse. Let us assume that we want to take into account the fact that there is a single opportunity before the final stage T to modify the control decisions computed at step 0, e.g. let us suppose that at a time step t_r, it is possible to (re)adjust the decisions for the subsequent time-steps until T given the information at hand at time t_r. The question then arises naturally as to how to compute the decision sequence at time 0, so that the combined process including the modifications computed and applied at t_r would lead to the best overall performances.

Thus, in order to take the decisions at the initial time 0, rather than assuming no-adjustment of control decisions, we should take into account the degrees of freedom that can be exploited at the later decision steps, by modeling the information flow about environment behavior and by taking into account the computational procedures that will be used at the later stages to adjust decisions, in addition to the dynamics of the controlled process (cf. Figure 1.1). Indeed it is preferable to compute the open loop sequences and the adjustment sequences jointly in order to reach a better overall optimum since both are interdependent. In the latter case, the open loop sequence will be referred to as the first stage and the adjustment sequence as the second stage and we thus face a two-stage problem. Note that a stage does not in general refer to an absolute time step but to a moment t_r at which the process ξ is observed and control decisions for subsequent time steps may be revised. In Formulation 3, the control actions u_t are thus now dependent of the realization of the random variables ξ_t from time 0 to time $t_r - 1$. However the first stage decisions must be unique, whatever the realization of ξ up to time $t_r - 1$, since we must implement it before having observed this realization. We formalize this setting by letting u_t become dependent on $\xi_{[0:t_r-1]}$ and by imposing that for $t < t_r$ they are independent on ξ (1.13). Assuming that a probabilistic model $\mathbb{P}_\xi$ of ξ is available, we formulate the problem as an expectation minimization problem.

To simplify the notations, we use the convention $\xi_{[t]} = \xi_{[0:t]} = (\xi_0, \ldots, \xi_t)$. The dependence of the sequence of controls on the uncertainty is denoted by

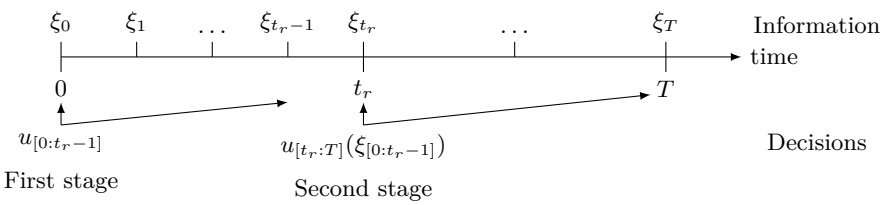


Figure 1.1: Information and decision flows.

Formulation 3: Two-stage decision making problem.

$$\min_{x(\xi),u(\xi_{[t_r-1]})} \quad \mathbb{E}_{\mathbb{P}_\xi}\left\{\sum_{t=1}^{T}\|C_t x_t(\xi)-Y_t(\xi)\| + \alpha\sum_{t=0}^{T-1} R_t\left((x_t(\xi),u_t(\xi_{[t_r-1]}),\xi\right)\right\} \tag{1.9}$$

$$\text{s.t.} \quad \forall \xi\in\Xi, \forall t\in\{0,...,T-1\},$$

$$x_{t+1}(\xi)=F_t(x_t(\xi),u_t(\xi_{[t_r-1]}),\xi), \tag{1.10}$$

$$\forall \xi\in\Xi,$$

$$x_t(\xi)\in\mathcal{X}_t,\ \forall t\in\{0,...,T\}, \tag{1.11}$$

$$u_t(\xi_{[t_r-1]})\in\mathcal{U}_t(x_t(\xi)),\ \forall t\in\{0,...,T-1\}, \tag{1.12}$$

$$u_t(\xi_{[t_r-1]})=u_t,\ \forall t\in\{0,...,t_r-1\}. \tag{1.13}$$

$u(\xi_{[t_r-1]})=\big(u_0(\xi_{[t_r-1]},\dots,u_T(\xi_{[t_r-1]})\big)$. Notice that if $(\xi_0,...,\xi_{t_r-1})$ has a discrete probability distribution the solution of the above problem thus provides an optimal adjustment sequence $u_t(\xi_{[0:t_r-1]})\ \forall t\in\{t_r,...,T\}$ for each value of $(\xi_0,...,\xi_{t_r-1})$, but otherwise the second stage decision is a policy, i.e. a function mapping an infinite number of realizations $(\xi_0,...,\xi_{t_r-1})$ towards second stage decisions. Even if the first stage decisions can be computed in some cases of this latter circumstance, one will in general need to recall for a computational procedure to calculate second stage decisions for each possible realization of the process over $\{0,\dots,t_r-1\}$.

As the deterministic and robust formulations, Formulation 3 may be used in a receding horizon manner by applying the first stage and discarding the second stage and repeating the optimization procedure periodically, the first stage being regarded as hedged against the uncertainty.

Several opportunities of recourse. More generally one can also consider several opportunities of recourse to adjust the decisions and thus consider a multistage problem. In that setting a recourse policy π_t for a recourse instant t is a mapping from the space Ξ to the space of continuous and discrete actions:

$$\begin{aligned}\pi_t:\quad & \Xi\to\mathcal{U}^{T-t}\\ & \xi\mapsto u_{[t:T]}.\end{aligned}$$

Among all the possible recourse policies, we are interested in the class of policies Π_t which use only the information collected until time t to output a decision (non-anticipativity). In this context, Π_0 is the class of constant functions. Note that we cannot make the assumption that the recourse policy is stationary, which means that we cannot apply the same policy at each recourse instant, for two complementary reasons: we are working over a finite horizon, and we target problems with a strongly time-dependent environment.

1.2.3 State-of-the-art in solution methods

We made no particular assumptions on the nature of the components of Formulation 1. Of course their nature determines the type of optimization problem that needs to be solved. In the application of Part I, the problem is usually formulated as a MILP (Appendix C.1 exposes the most widespread algorithms to solve such problems). But the formulation presented may as well encompass general nonlinear or convex problems.

For Formulation 2 and Formulation 3 one must take into account the stochastic process ξ. In such stochastic programs, often one cannot use as such the usual tools of mathematical programming except when ξ has a discrete probability distribution, in which case one can enumerate the possibilities under the max operator in Formulation 2 or replace the expectation operator by an average in Formulation 3. In the latter case these formulations usually have the same nature than Formulation 1 but result in much larger instances, for which specialized decomposition techniques are necessary. A common approach to tackle the case of a continuous distribution for ξ is to sample it and solve the problem as if the distribution were discrete. Each sample is called "scenario". Figure 1.2 depicts four scenarios $\xi^{(k)}$, $k \in \{1, 2, 3, 4\}$, represented by piecewise linear curves.

Scenario based optimization. A deterministic procedure would a priori aggregate the scenarios (e.g. by taking their mean if this is relevant) to produce a single scenario problem (cf. Formulation 1) and thus a single solution, whose value could however be poor regarding the uncertainty. Another procedure consists in optimizing independently the scenarios as such, thus discarding the non-anticipativity constraints, and would provide a distinct solution for each scenario (Figure 1.2). But as the realization is not known beforehand, the user must make a choice, possibly by comparing the solutions, and thus takes a risk. This risk can be mitigated by the possibility to switch to the solution of another scenario when parts of the realization of the uncertain process become

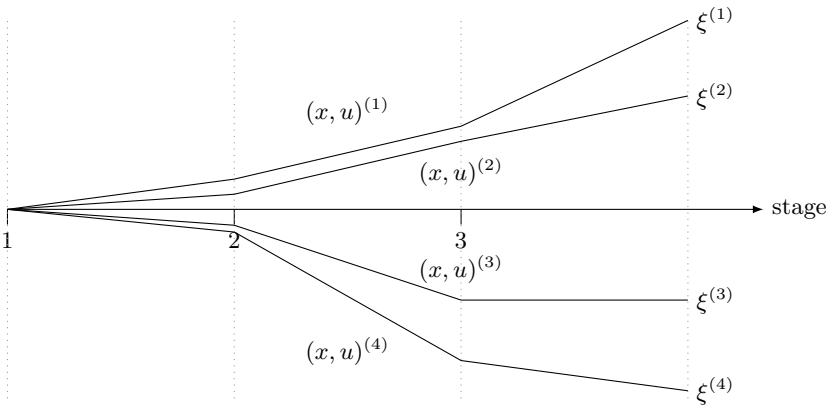


Figure 1.2: Four scenarios labeled with their individual optimal solutions.

available, i.e. in a receding horizon way as explained above. However this procedure has some drawbacks: switching from one solution to another may not be feasible or may require a transition period; this procedure overfits the information available when making the choice; and it does not provide an estimation of the overall risk related to this choice.

Some algorithms, e.g. progressive hedging (cf. Rockafellar and Wets (1991)) or Dual Decomposition in Stochastic Integer Programming (DDSIP, cf. Carøe and Schultz (1999)) compute a solution of the original problem based on the solutions of these single scenario sub-problems iteratively modified to impose the non-anticipativity constraints. Another method consists in reducing the size of the optimization problem by merging some scenarios and creating a scenario tree whose solution is a good approximation to the solution of the original problem. The tree structure imposes implicitly the non-anticipativity constraints. For example in Figure 1.3 the fan of scenarios of Figure 1.2 is replaced by a tree: sets of sufficiently similar sub-scenarios are incrementally merged (cf. Dupacova et al. (2000)) to form the branches of the tree and only one sequence of decisions is attached to a merged branch.

Thus when discretizing a continuous distributions one ends up with a set of action sequences for each scenario or for each branch of the scenario tree and not with a recourse policy. As we cannot apply them directly to novel unseen scenarios, it does not allow one to assess the quality of a policy computed using an independent test set of scenarios for the same problem or to exploit the policy in practice.

Quality of the recourse policy. In practice one can assess the quality of recourse policies from two viewpoints.

First, on the basis of their optimality, in the sense that it should lead to the smallest possible adjustment cost regarding the possible future realizations of the stochastic process. One should ideally take into account the effect of the

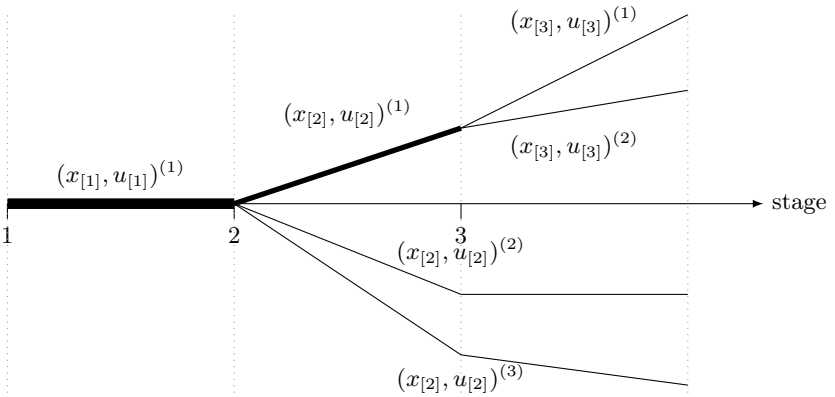


Figure 1.3: Scenario tree and associated optimal solutions for the different stages. Thickness of segments is representative of the number of merged subscenarios with respect to Figure 1.2

discretization of the disturbance process ξ in this respect.

Secondly, and maybe more importantly, based on the feasibility of the decisions that they provide. A policy should output feasible schedules for any realization of ξ, thus not lead to infeasibility at the subsequent recourse instants. This is the case of Formulation 2 if we do not discretize the disturbance process ξ. But as soon as we discretize the disturbance process, any of the formulations exposed so far applied in a receding horizon fashion may end up in a state where no sequence of actions may drive the system through a feasible sequence of states.

1.2.4 Summary

In the preceding subsections we exposed the problem of optimizing sequential decision making procedures under uncertainty. We have exposed both robust and probabilistic approaches, either in the context of single stage or in the context of multistage formulations. These formulations often lead in practice to intractable optimization problems, which need to be addressed by heuristic and suboptimal approaches. The avenues for solving these problems rely on a combination of problem statement simplification and systematic approaches for exploiting approximate solutions of problem instances. The rest of this thesis will elaborate on combinations of these two lines of research.

1.3 Background in machine learning

Machine learning aims at developing algorithms to make a computer able to learn decision rules from empirical data. It is divided in several sub-fields characterized by the type of data and by the properties and the abilities of the decision rules one wants to extract from it. We first present a typical categorization of these sub-fields and in the following subsections we detail the sub-fields that we are directly dealing with or which are related to the concepts developed in the remainder of this thesis. In this section we mainly aim at introducing the basic concepts to the reader who is not familiar with machine learning. For a more complete treatment and a more advanced theoretical analysis, we refer the reader to the large body of literature (an excellent treatment of supervised and unsupervised learning is given in Hastie et al. (2009)).

We assume that the data is organized as a collection of objects. Let Ω be the set of all possible objects, and let $(\Omega, \varepsilon, \mathbb{P})$ be a probability space, where ε denotes a sigma-algebra of measurable subsets of Ω and $\mathbb{P}$ is a probability measure defined over these *events*. Then let us consider a collection of random variables[2] each one mapping the space $(\Omega, \varepsilon, \mathbb{P})$ to some observation space. A dataset D is defined as a collection of joint observations of the values of these random variables, over a collection of objects drawn according to some sampling mechanism[3]. The generic objective of machine learning is to exploit such

[2]We use the expression "random variable" even when the value is a vector or any other complex data-structure.

[3]Normally one assumes that objects are drawn independently and identically distributed according to the measure $\mathbb{P}$ to form the dataset D, but in some circumstances other sampling

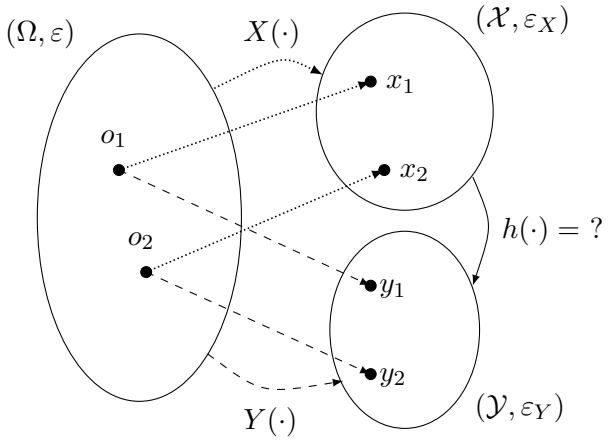


Figure 1.4: The supervised learning setting.

datasets in order to "learn" about the relations among the considered random variables and build predictive models allowing one to make predictions about their future values. The field of machine learning is concerned with the design of such algorithms with different objectives in mind, such as predictive accuracy (the ability to make correct predictions), computational scalability (the ability to exploit high-dimensional observation spaces and large datasets), and interpretability (the ability to produce results that may be easily confronted with human domain knowledge or assumptions).

In *supervised learning* (SL), one distinguishes between two types of variables X and Y mapping $(\Omega, \varepsilon, \mathbb{P})$ to respectively the measurable spaces $(\mathcal{X}, \varepsilon_X)$ and $(\mathcal{Y}, \varepsilon_Y)$ (see Figure 1.4). In a classical setting, $\mathcal{X}$ is called the input-feature space and has usually dimension greater than one, while $\mathcal{Y}$ is the output space and is often of dimension one. The observation of X is assumed to be easy or cheap, while Y is difficult or costly to observe. When both X and Y are observed, we say that the object is labeled, and we say that it is unlabeled when only X is observed. In supervised learning, the dataset D is composed of labeled objects, and one is essentially interested in building a predictive model able to estimate the output label of any object, given the observed values of its input-features. In other words, one is interested in estimating the conditional probability distribution of Y given X, $\mathbb{P}_{Y|X}$, from which one could derive a mechanism allowing to predict the value of Y given X that is optimal in expectation with respect to a given loss function $\ell(\cdot, \cdot) \in \mathbb{R}_+^{\mathcal{Y} \times \mathcal{Y}}$. Specifically, one searches for a function $h(\cdot) \in \mathcal{Y}^{\mathcal{X}}$, such that $h(x)$ is as close as possible to $\arg\min_{y'} E_{\mathbb{P}_{Y|x}}\{\ell(Y, y')\}$ for every possible value x of X. Notice that the conditional distribution $\mathbb{P}_{Y|X}$ can be written in terms of the joint probability distribution of X and Y as

$$\mathbb{P}_{Y|x}(y) = \frac{\mathbb{P}_{X,Y}(x, y)}{\mathbb{P}_X(x)},$$

mechanisms may as well be of interest.

so that the optimal predictor could as well be derived from $\mathbb{P}_{X,Y}$. Supervised learning approaches which operate by modeling the joint distribution are called "generative" approaches, while those which rely on modeling the conditional distribution are called "discriminative" approaches. The next subsection elaborates further on these approaches and provides some examples.

On the other hand, *unsupervised learning* (UL) aims at modeling the relations among all observed variables without distinguishing among inputs and outputs. In other words, based on the observation of a dataset over a collection of random variables, the goal of UL is to identify some relations among these variables. For example, a common task is to identify clusters of objects which exhibit the same property (Kaufman and Rousseeuw (2005)), which may be viewed as "modes" of the joint distribution. Another common task in unsupervised learning is to identify probabilistic independencies among subsets of variables, and to build models of their joint distribution exploiting these independencies.

Transductive learning (TL) is slightly different of SL. Only one part of the data is labeled and we want to estimate the output of the unlabeled objects using all the information of the data set. We do not try to generalize to all possible other objects, just to give an accurate prediction for the specific set of unlabeled objects given a priori. Semi supervised learning (SSL) is a combination of the SL and UL paradigms. Again only one part of the data is labeled (typically a small fraction), but here we want to use all the data set to generalize to other objects. Thus TL and SSL can be seen as combinations of SL and UL.

A sub-field which is directly related to sequential decision making is *reinforcement learning* (RL). The goal of RL is to design autonomous agents able to interact in an optimal fashion with an a priori unknown environment. The agent can take some actions influencing the state of the environment, and collect some information about the state of the environment and its performance by using sensors. By repeating this several times, it may then learn a control strategy from the collected information in order to improve its performance over time, and eventually reach near-optimal behaviors. At the end of this section, we will provide an example of a reinforcement learning algorithm, and make the connection with the optimization based optimal control approaches of the previous section.

1.3.1 Supervised learning

In the SL paradigm, the data is organized as a set of N objects described by their input features and their output label,

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}.$$

The features x_i are easily obtained by observation, while the output value y_i is provided by an expert, or is the result of a difficult observation and/or computation. Often instead of first computing an estimate of $\mathbb{P}_{Y|X}$ or $\mathbb{P}_{X,Y}$ and then deriving predictions from it, SL algorithms directly search for a mapping

$$h : \mathcal{X} \to \mathcal{Y}$$

between the feature space and the output space that generalizes well to elements for which the output value has not been observed. To this end, one defines a loss function

$$\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$$

and searches for a mapping h which minimizes the *expected loss*

$$\mathbb{E}_{\mathbb{P}_{X,Y}} \{\ell(h(x), y)\}$$

over the joint $\mathcal{X} \times \mathcal{Y}$ space. As one usually does not know the joint distribution of the objects, one can instead search for a function that minimizes an estimate

$$\frac{1}{N} \sum_{i=1}^{N} \ell(h(x_i), y_i)$$

of the expected loss, also called the *empirical loss*.

A SL problem can be categorized either as a classification problem if the output is qualitative and belongs to a finite set of labels, or as a regression problem if the output is a real value.

Bayes model of a prediction problem. For a moment, suppose that Ω is finite, that the distribution $\mathbb{P}$ is known, and that it is possible to evaluate the features $X(o)$ and the output $Y(o)$ $\forall o \in \Omega$. Then both $\mathcal{X}$ and $\mathcal{Y}$ are also finite sets, and the most probable value $B(x)$ of Y knowing that $X = x$ could be computed using the method of Table 1.1; $B(x)$ is called the Bayes classifier.

More generally, for a given problem defined by a loss function and a joint distribution of input and outputs, a function that actually minimizes the expected loss is called a *Bayes model* for that problem. For example in regression problems, and when we use the so-called square-loss (i.e. $\ell_{sq}(y, y') = (y - y')^2$), the Bayes model is obtained by computing the conditional expectation of the output given the input, i.e. $B(x) = E_{\mathbb{P}_{Y|x}}\{Y\}$. In these words, the Bayes classifier is the Bayes model obtained for the so-called 0/1 loss-function, $\ell_{0/1}(y, y') = \delta_{y,y'}$; it has the property of minimizing the probability of predicting a wrong label.

The expected loss of the Bayes model is called the *residual loss*, since by definition no hypothesis may yield a smaller expected loss. A learning problem is called "deterministic", if its residual loss is equal to zero, meaning that it is possible to find a hypothesis that leads to an average loss of zero.

Constraining the hypothesis space of supervised learning algorithms. In the context of supervised learning, neither can we assume in general that Ω is finite, nor do we have access to the probability measure $\mathbb{P}$ nor to the functions X and Y. The only knowledge about the problem that we have to help us guessing the Bayes model is a sample of input-output observations that is assumed to be drawn independently from $\mathbb{P}$. In this case, searching for a model in the space of all possible input-output functions (which is in almost all situations of practical interest of infinite dimension, or of a practically intractable finite dimension) while disposing only of a finite number of information items (the dataset) is an ill-posed problem.

Table 1.1: Computation of the Bayes Classifier.

1. Compute $(\forall x \in \mathcal{X}, \forall y \in \mathcal{Y})$

$$\begin{array}{rcl} \mathbb{P}_{X,Y}(x,y) & = & \sum_{o\in\Omega} \delta_{X(o),x}\delta_{Y(o),y}\mathbb{P}(o), \\ \mathbb{P}_X(x) & = & \sum_{o\in\Omega} \delta_{X(o),x}\mathbb{P}(o), \\ & = & \sum_{y\in\mathcal{Y}} \mathbb{P}_{X,Y}(x,y), \end{array}$$

2. deduce $(\forall x \in \mathcal{X}, \forall y \in \mathcal{Y})$

$$\mathbb{P}_{Y|x}(y) = \frac{\mathbb{P}_{X,Y}(x,y)}{\mathbb{P}_X(x)},$$

3. build the model $B(x)$, i.e. $\forall x \in \mathcal{X}$ compute

$$B(x) = \arg\max_y \mathbb{P}_{Y|x}(y). \tag{1.14}$$

Where

$$\delta_{a,b} = \begin{cases} 1 & \text{if } a = b, \\ 0 & \text{otherwise.} \end{cases}$$

One common approach in supervised learning therefore consists in restricting the search space of models to a finite dimensional space of input-output functions $\mathcal{H}$, called the hypothesis space of the supervised learning algorithm. Typically, this is carried out by defining a parameter vector of finite dimension Θ. Denoting by h_θ an element of $\mathcal{H}$, with $\theta \in \mathbb{R}^n$, supervised learning can then be reduced to solving the optimization problem

$$\theta^*(D) \in \arg\min_{\theta\in\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(h_\theta(x_i), y_i).$$

Choosing a hypothesis space thus consists in restricting the search space for h to a class $\mathcal{H}(\Theta)$ of functions parameterized by a finite set of parameters. Each SL approach (linear models, decision trees, artificial neural networks, support vector machines, ...) thus uses a particular hypothesis space. Choosing one of them amounts to assuming that the relation between X and Y has a particular structure. The Bayes model can represent any mapping B from the feature space to the output space and the set of possible Bayes models thus defines the largest possible hypothesis space. A supervised learning algorithm applied to a specific problem may lead to excellent results if its hypothesis space is both of small dimension (compared to the size N of the dataset) and contains models which are close enough to the Bayes model. For a given problem, the *representation bias* of a hypothesis space is defined by the degree

of sub-optimality of the best hypothesis in this space with respect to the Bayes model.

Intuitively, if the hypothesis space is of small dimension compared to the sample size, the principle of empirical loss minimization should lead to the identification with high probability of the best hypothesis in this space. On the other hand, the smaller the hypothesis space, the more likely it is that the Bayes model of a given problem cannot be well approximated within $\mathcal{H}$ (i.e. that the representation bias is large). Thus, a fundamental problem in supervised learning is to choose the right hypothesis space for a given problem. Often, smaller hypothesis spaces also lead to more efficient computational solutions to the problem of minimizing the empirical loss. The restriction of the hypothesis space is exemplified by the class of learning algorithms using linear models in Example 4.

Example 4.
In the regression problem illustrated on Figure 1.5 the true model is a parabola (continuous black curve) to which we have added a Gaussian noise of zero mean. We have repeated three times the procedure of sampling 5 points representing the realizations of the random variable Y for 5 values of X (represented by different colors) to obtain 3 datasets. The colored lines correspond to linear least ssquares fits of the points in each dataset. We see that the choice of a linear model is probably too restrictive. However we observe as well that by repeating the experiment several times the learned model does not change a lot.

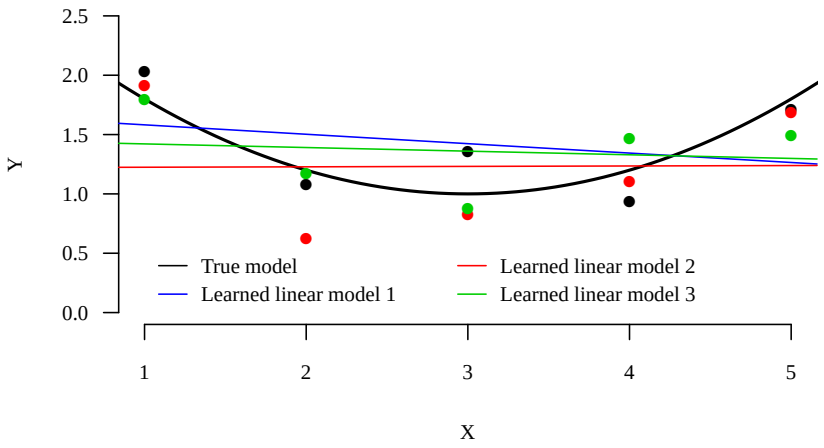


Figure 1.5: Restricting the hypothesis space to the class of linear models.

Rationality of the empirical risk minimization principle. Once a hypothesis space is chosen, designing an algorithm that produces an element of this space of minimal empirical loss seems to be a reasonable approach to supervised learning. Indeed, for a given hypothesis, its empirical loss is an unbiased estimate of its theoretical loss. Nevertheless, if the hypothesis space $\mathcal{H}$ is very large (e.g. if it contains all possible input-output functions), then, for any dataset the

empirical loss of the best function of $\mathcal{H}$ (one of those minimizing the empirical loss) may perform very well on the given dataset but be very far away from the Bayes model as illustrated in the Example 5.

Example 5.
Continuation of Example 4. Figure 1.6 illustrates the application of the empirical risk minimization principle in the hypothesis space of fourth degree polynomials. As there are 5 points in the dataset there always exists a polynomial of degree 4 interpolating the 5 points. The hypothesis space is thus too large and thus causes overfitting, i.e. the selected hypotheses focus too much on their dataset and it results in a high variability of the selected hypothesis when the dataset changes.

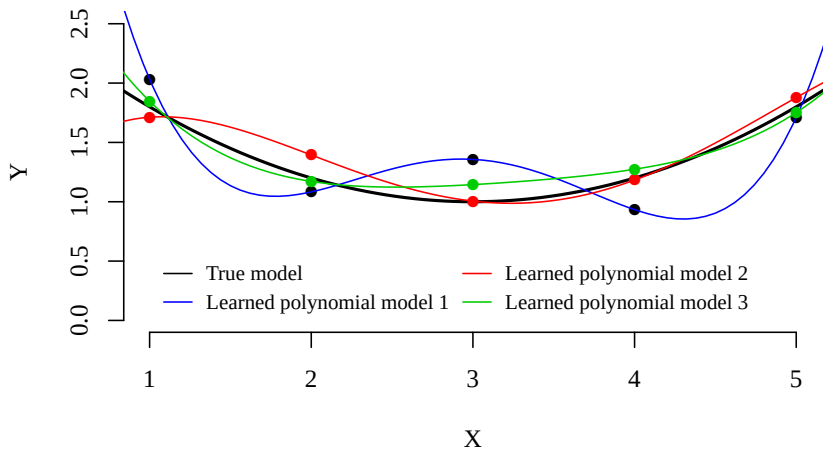


Figure 1.6: Choosing a too large hypothesis space: selecting the class of polynomial models of degree 4.

On the other hand, it is well known that for some classes of hypothesis spaces (e.g. for linear models), the principle of empirical loss minimization indeed leads to a procedure that will identify, with probability tending to 1 when the dataset size grows to infinity, the best model of this space in terms of the theoretical loss. Statistical learning theory (Vapnik (1998)) has delineated the general conditions on the hypothesis space that are both necessary and sufficient for the empirical loss minimization principle to be consistent. Its main result shows that $\mathcal{H}$ needs to be of finite "VC-dimension", in order to ensure that for any learning problem the empirical loss minimization leads to hypotheses which in the large sample regime will converge (in terms of theoretical loss) to the best possible hypothesis in this space. The theory also provides probabilistic bounds on the theoretical loss as a function of the empirical loss, although these bounds are typically very conservative. Note for example that the VC-dimension of the space of hyperplanes over n real-valued input-features ($\mathcal{X} = \mathbb{R}^n$, $\mathcal{Y} = \{0, 1\}$) is equal to $n + 1$, hence finite, while the VC-dimension of all possible functions in $\{0, 1\}^{\mathbb{R}^n}$ is infinite.

When we select a hypothesis space that is believed too large for the problem at hand, *regularization* may be used to automatically direct the search

towards the simplest model that explains sufficiently well the training data. Regularization thus establishes a trade-off between these concurrent goals. In the SL algorithms which are naturally formulated as constrained optimization problems, such as LASSO (Hastie et al. (2009, Chapter 3)) or Support Vector Regression (SVR) (cf. Appendix A.3), a term penalizing the complexity of the model is explicitly included in the objective function. This is illustrated in Example 6.

Example 6.
Continuation of Example 5. The blue curve of Figure 1.7 is the result of the search in a hypothesis space theoretically equivalent to the space of polynomial models of degree 4 but with a regularization term penalizing too complex models. While not totally perfect, we see that it is likely to perform better in generalization than the linear and fourth degree polynomial models fitted to the same dataset.

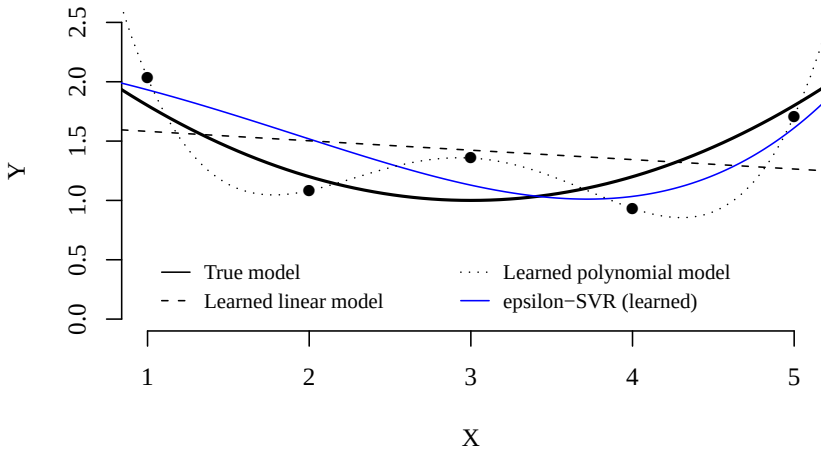


Figure 1.7: ε-SVR with polynomial kernel of degree 4 (cf. Appendix A.3), compared to the linear and polynomial models of Figures 1.5 and 1.6.

Constraining the algorithmic complexity of supervised learning algorithms. Many successful SL methods do not fit in the above framework, i.e. we do not know the VC-dimension of the hypothesis space they search. Instead of constraining a priori the hypothesis space to a space of finite VC-dimension, an alternative approach to the design of supervised learning algorithms consists in designing the algorithm mapping a dataset to a function h in such a way that it produces its result in an efficient way, i.e. so that the computing time of the algorithm grows relatively slowly with the size of the dataset. In this view, the challenge is to design fast algorithms producing hypotheses with a low empirical loss. When constraining the algorithmic complexity of a SL algorithm, one also constrains, although in an implicit fashion, the subset of reachable hypotheses. In other words, this approach consists in deciding a priori that some "details" about the dataset will not be exploited in order to choose a hypothesis, which is equivalent to reducing the hypothesis space.

Example 7.
Continuation of Example 6. The restriction of the search complexity is exemplified by the class of learning algorithms using tree-based models (cf. Appendix A.1). In tree-based methods, regularization is achieved either by early stopping the development of nodes or a posteriori by removing parts of the tree. By construction, one thus wants to limit the representation power of the learned hypothesis. In this example we illustrate the fact of limiting the depth of the trees during their top-down induction by acting on a parameter n_{min} which defines the number of objects contained in a node under which the node is not split. In Figure 1.8a a single tree with $n_{min} = 2$ interpolates all the points of the dataset since the tree is fully developed, while when $n_{min} = 3$ the tree is already less focused on the points of the dataset. These methods proved to have a high variance because the position of the splits, i.e. of the vertical segments in Figure 1.8a, is very sensitive to the content of the dataset. Thus several alternatives have been proposed to enhance their properties, relying on the aggregation of an ensemble of trees randomized in some fashion (bagging, random forests, Extra-Trees). In the Extra-Trees (cf. Appendix A.2), the simplification of the search complexity is operated by randomizing the composition, the size and the tested values of the subset of input features considered at each node during the tree induction. We illustrate the construction of an ensemble of 50 extremely randomized trees on Figure 1.8b. Again when $n_{min} = 2$ the model interpolates all the points of the dataset since each tree is fully developed, while when $n_{min} = 3$ the learned hypothesis comes closer to the true value, similarly to what was observed in Example 6 for the regularized model.

Such algorithms, which cannot be directly studied by the theory of the preceding paragraph, may be characterized as well theoretically by the notion of algorithmic stability (Poggio et al., 2004): a stable algorithm is an algorithm that produces hypotheses that are not very sensitive to small perturbations of the dataset, for example the removal of a few observations.

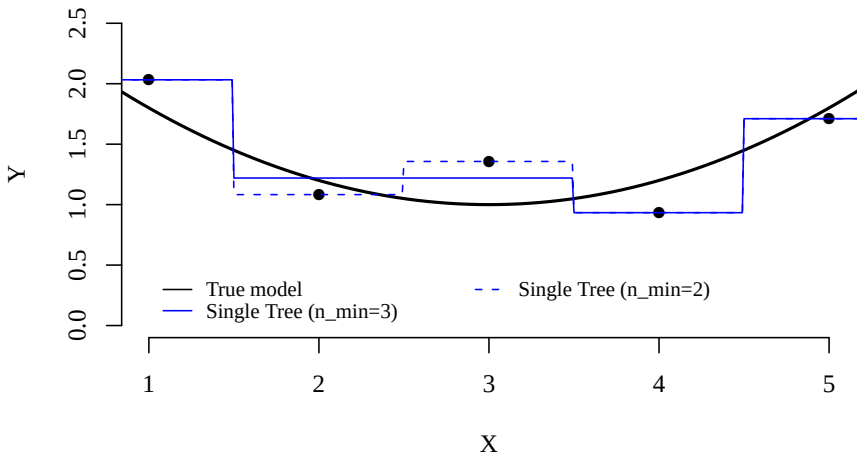
Bias and variance of learning algorithms. In addition to the residual error which is determined only by the supervised learning problem statement, two other sources of error linked to the choice of the learning algorithm arise in the SL context; they are respectively called bias and variance. We will explain these notions in the context of regression problems, although they are also relevant in the context of classification problems and more general output spaces.

On the first hand, any hypothesis space is characterized by a certain representation bias. For example if one restricts $\mathcal{H}$ to the class of linear models, there is no hope in approximating perfectly a non linear Bayes model[4] between X and Y. Also, the learning algorithm itself may under-exploit in a systematic way the information contained in the dataset, leading to a second source of bias. Both sources of error are normally difficult to distinguish and are therefore lumped into the term "bias". For a given learning problem (characterized by $\mathbb{P}_{X,Y}$ and ℓ which we suppose to be the square loss), and for a given dataset size N, the (squared) bias of a learning algorithm at a point x_0 is defined by

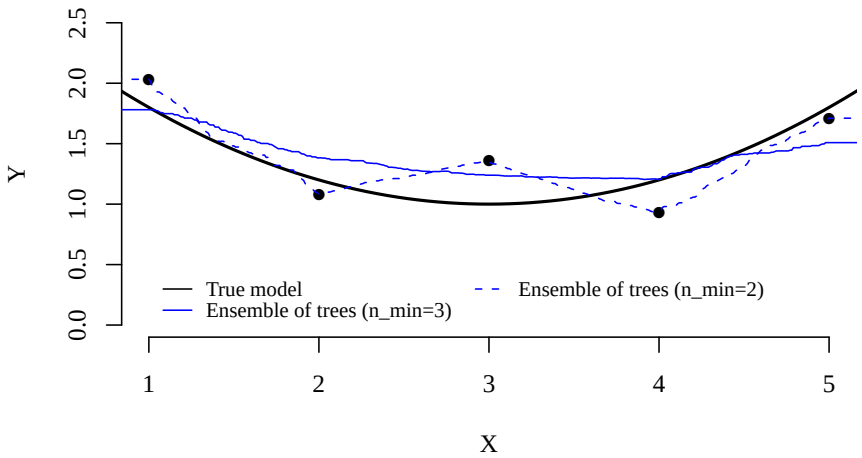
$$\left(B(x_0) - \mathbb{E}_D\{h_D(x_0)\}\right)^2.$$

Bias measures how far the average model produced by the algorithm (averaging over an ensemble of datasets) is from the Bayes model. Typically, using larger

[4]Without applying a transformation to the input features.



(a) One single regression tree for two values of the parameter n_{min}.



(b) One ensemble of randomized regression trees for two values of the parameter n_{min}.

Figure 1.8: Illustration of tree based methods.

hypothesis spaces leads to reducing the bias, while using more "rough" learning algorithms increases the bias.

On the other hand, the variance error originates from the sensitivity of the optimal parameters θ (or equivalently the hypothesis h computed by the algorithm) on the particular dataset of size N used as input. Namely, using the same algorithm with two different datasets of a given problem will normally produce two different models, both different from the average model, even if the average model is close to the Bayes model. This source of variation leads to additional suboptimalities, e.g. measured in regression at a point x_0 by

$$\mathbb{E}_D \left\{ (h_D(x_0) - \mathbb{E}_D\{h_D(x_0)\})^2 \right\}.$$

Variance measures how strongly the hypotheses computed by a learning algo-

rithm depend, for a given problem, only on the random nature of the dataset.

In the context of regression problems, with the square-error loss function, the three terms, namely residual loss ($\mathbb{E}_{Y|x_0}\left\{(Y - B(x_0))^2\right\}$), (squared) bias and variance, simply add up to produce the expected loss of the learning algorithm, i.e. the expectation over datasets of a given size of the expected loss of the hypotheses produced by this algorithm over the joint input-output distribution. These ideas also carry over to other settings, e.g. classification problems and problems with more complex output spaces. Thus one main direction of research in supervised learning is to characterize bias and variance of algorithms and designing algorithms with low bias and/or variance (see e.g. Geurts (2002)).

Model selection, model evaluation and cross-validation. In the context of SL one is interested in the ability of h to approximate the outputs of the data used to train it, but also in the generalization capacity of h to predict the outcome for independent data. For the inference on new objects, i.e. the generalization, all the available dataset is used by the learning algorithm to optimize the parameters of h, which is then used to make the prediction. However evaluating its error on the complete data set generally yields an optimistic estimate of the generalization error. Thus a usual way for evaluating the expected loss of h is to divide the data set in two disjoint sets. On the one hand a training set allows to optimize the parameter θ of the model. On the other hand a test set is used to evaluate the empirical error of the learned mapping. To obtain a more accurate estimate when the data set is too small to estimate the empirical error, one often resorts to $n-$fold cross-validation (CV) instead of the simple above procedure (cf. Table 1.2).

Table 1.2: $n-$fold cross-validation procedure.

Split the data set in n disjoint folds $S_1, \ldots, S_n$.
Then for $i \in \{1, ..., n\}$:

1. choose S_i as test-set and gather the $n-1$ other subsets of S in the training set,
2. learn h using the training set,
3. Evaluate h on the test set,
4. compute the empirical error of the learned method and optionally other byproducts of the method.

Afterward one can compute statistics (e.g. mean error) over the n iterations.

Batch and online learning. Batch learning means that the data set is fixed and processed entirely by the learning algorithm. On the other hand online learning algorithms have been developed either when the data collection process and the learning are imbricated, or when the memory or the processing capacity

of the computer are restricted so that the learning algorithm processes the data gradually. In this thesis we consider batch mode learning because there is apparently no need to learn in real time and there is no limiting effect of the available computation power and memory. However in our application of Part I the inference of decisions, or predictions, from the trained model should be relatively fast, since we aim to use it in quasi real time.

Structured prediction. Traditionally most of the work in the SL community dealt with univariate outputs, both in classification and in regression. Sometimes the related methods can be readily extended to multivariate outputs, e.g. in regression when passing from scalar to vector outputs, but may not model correctly the relations between outputs. Nowadays SL research is more and more oriented towards learning from and predicting complex structured data (labeled sequences, time series, images, XML trees, ...), i.e. when inputs and/or outputs exhibit a (known) structure which makes them interdependent.

For example, the book of Bakir et al. (2007) contains some variants of kernel-based methods adapted to structured data. The main idea is to embed the input-output pairs (x, y) in a feature space thanks to a function $\phi(x, y)$ and to formulate the prediction problem for a new input x^* as

$$h(x^*) = \arg\max_{\hat{y} \in \mathcal{Y}} w^\top \phi(x^*, \hat{y}), \tag{1.15}$$

where $w^\top \phi(x^*, \hat{y})$ measures the compatibility between x^* and $\hat{y}$, and w must be learned. However, under certain conditions, similarly to the SVR algorithm exposed in Appendix A.3, it is possible to use only the kernel

$$k((x, y), (x', y')) \doteq \phi(x, y)^\top \phi(x', y')$$

to compare two input-output pairs for learning and for prediction, without explicitly computing the dot product in the feature space induced by ϕ. To work well this technique thus requires the definition of a kernel appropriate to the particular structured prediction problem. The learning procedure is applicable to any problem without modification once the kernel is defined. The inference procedure, or *pre-image* problem (1.15), may be hard to solve in practice for general kernels, and some special properties are thus desired to render this problem tractable.

Geurts et al. (2006b) extend tree-based ensemble methods to the prediction of structured outputs using a kernelization of the algorithm that allows one to grow trees as soon as a kernel can be defined on the output space.

In Maes (2009) the structured prediction problem is cast on the RL framework (cf. Section 1.3.2 below). One of the particularities of this work is to construct incrementally the output prediction, as opposed to the previous methods.

Even though the emphasis is not exactly in line with these developments in this thesis, it will become clear that these works are very relevant for our application of Part I. Indeed, our work of Part II can be understood as a way to take into account the structure of the data by the regularization of a learned tree-based model, thanks to prior knowledge expressed as constraints between outputs or between input-output pairs.

1.3.2 Reinforcement learning

Usually, the aim of RL is to provide an agent with the ability to achieve a goal in an unknown environment. To achieve this goal the agent takes some actions to interact with the system and collect some data. RL aims at extracting from this interaction data a control policy for the agent. RL techniques are also used in the optimal control framework, when the system to control is a priori unknown.

Markov Decision Process. Markov Decision Processes (MDPs) (Puterman (1994)) provide a mathematical framework for RL. The agent's situation is represented by a state x_t. At each time t, the agent observes the state x_t and then selects an action u_t to make its state move from x_t to x_{t+1}. State and actions belong respectively to a state space $\mathcal{X}$ and an action space $\mathcal{U}$ (which could be either discrete, continuous or mixed). The agent's environment defines a function F which models such transitions and is assumed to be Markovian, meaning that the state x_{t+1} is fully conditioned by the state and the action at time t, irrespectively of previous states and actions:

$$x_{t+1} = F(x_t, u_t, \xi_t),$$

where ξ_t denotes a disturbance process that is dependent on past quantities only through x_t and u_t.

At each time step the agent receives a scalar reward which quantifies the quality of the action u_t taken while in state x_t. The function R associates this reward to a couple (x_t, u_t) and may as well be function of the disturbance ξ_t:

$$r_t = R(x_t, u_t, \xi_t).$$

We can distinguish two concepts at this point. The fact that little information about the system is available to the agent is obviously a source of uncertainty when exploring the system. However this does not reflect the stochasticity of the system, but rather the lack of knowledge of the agent about its environment. We must thus distinguish this from the uncertainty arising from the fact that F and R can be function of a random process ξ_t.

Formulation of the RL problem. Within the MDP framework, a stationary policy π is a mapping that assigns an action u to any state x. RL algorithms typically search for a policy which is able to achieve a goal in a minimum time, by collecting some instantaneous rewards. To do so they search for a policy that maximizes the expected infinite sum of discounted reward.

The state-value function of a stationary policy π starting from an initial state x can be defined as the expected value of the infinite sum of the discounted rewards that will be accumulated when following that policy,

$$V^{\pi}(x) = \mathbb{E}_{\mathbb{P}_{\xi}} \left\{ \lim_{T \to \infty} \sum_{t=0}^{T-1} \gamma^t R(x_t, \pi(x_t), \xi_t) \mid x_0 = x \right\},$$

where the discount factor $\gamma \in [0, 1[$ allows to tune the dominance of short-term rewards ($\gamma \to 0$) over long-term rewards ($\gamma \to 1$). The optimal state-value function is defined as

$$V^\star(x) = \max_\pi V^\pi(x),$$

from which one can extract an optimal control policy as follows:

$$\pi^\star(x) = \text{argmax}_u \mathbb{E}_{\mathbb{P}_\xi} \left\{ R(x, u, \xi) + \gamma V^\star(F(x, u, \xi)) \right\}. \tag{1.16}$$

Alternatively, the state-action value function of a policy π defined as

$$Q^\pi(x, u) = \mathbb{E}_{\mathbb{P}_\xi} \left\{ R(x, u, \xi) + \gamma V^\pi(F(x, u, \xi) \right\}, \tag{1.17}$$

and the optimal state-action value function as

$$Q^\star(x, u) = \max_\pi Q^\pi(x, u),$$

can also be used. In terms of the latter function we have

$$\pi^\star(x) = \text{argmax}_u Q^\star(x, u). \tag{1.18}$$

According to Busoniu et al. (2010), RL algorithms can be grouped in three categories. *Value iteration* algorithms consist in modeling $V^\star(x)$ or $Q^\star(x, u)$ first and exploiting (1.16) or (1.18) respectively. The latter option seems easier since the knowledge of R and F are not necessary. Note that often it is necessary to approximate the state-action value function using SL algorithms (Ernst et al., 2005). *Policy iteration* algorithms evaluate V^π or Q^π for a set of policies and searches for an improved policy based on these values. *Policy search* algorithms explore directly the policy space without modeling explicitly a value function.

Note that in an online setting there is a trade-off between the exploration which allows to collect potentially new information but can lead to bad performance of the agent, and the exploitation of the policy inferred from already sampled data.

Example of solution method. We adopt an offline setting and suppose that we dispose of a set $\mathcal{F}$ of 4-tuples (x_t, u_t, r_t, x_{t+1}) reflecting the interaction of the agent with the environment. To illustrate how such data can be used to derive a stationary policy for the agent, we reproduce in Table 1.3 the fitted Q iteration algorithm proposed in Ernst et al. (2005). This algorithm transposes the RL problem into a sequence of SL problems. The first SL problem yields the approximation $\hat{Q}_1$ of the reward function R. Then the solution of the N^{th} SL problem, denoted by $\hat{Q}_N$, is learned from a modified learning set, updated according to an equation derived from (1.17) and the solution of the previous SL problem, $\hat{Q}_{N-1}$. $Q^\star$ is approximated by the solution of the last SL problem, when some stopping criterion is satisfied.

Despite they are computationally limited when one attempts to solve problems in large state and action spaces, especially when they are continuous, RL algorithms have the advantage of making no (or few) assumption(s) on the

Table 1.3: The fitted Q iteration algorithm.

Inputs: a set of 4-tuples $\mathcal{F}$ and a regression algorithm $\mathcal{A}_l$.
Initialization:
Set N to 0.
Let $\hat{Q}_N$ be equal to zero everywhere on $\mathcal{X} \times \mathcal{U}$.
Iterations:
Repeat until stopping conditions are reached

1. $N \leftarrow N + 1$.
2. Build the dataset $D = \{(i^l, o^l), l = 1, \ldots, |\mathcal{F}|\}$ based on the function $\hat{Q}_{N-1}$ and on the full set of 4-tuples $\mathcal{F}$:

$$i^l = (x_t^l, u_t^l), \tag{1.19}$$
$$o^l = r_t^l + \gamma \max_{u \in \mathcal{U}} \hat{Q}_{N-1}(x_{t+1}^l, u). \tag{1.20}$$

3. Use the regression algorithm $\mathcal{A}_l$ to induce from D the function $\hat{Q}_N(x, u)$.

underlying model of the system (e.g. linear vs. nonlinear). Also, since the RL algorithms usually make no strong assumption on the model of the system, they are more likely to adapt the control policy if the system varies with time.

A connection, in the deterministic framework, between RL and the optimal control formulations presented in Section 1.2 can be found in Ernst et al. (2009) together with a detailed case study. It also briefly discusses the stochastic case.

1.4 Illustrative example: optimally driving a car

We provide a simple example not deemed to be representative of a real situation but which allows us to relate the different concepts and solution techniques discussed so far.

1.4.1 Problem setting

Imagine that a person drives a car from his home to his workplace, along a path composed of 3 segments, each characterized by a maximum speed V_i, $i = 1, 2, 3$ (cf. Figure 1.9). The driver can decide of the desired speed u_i, $i = 0, 1, 2$, of the car on each segment $i+1$ before entering that segment. The car is equipped with a speed regulator, hence the driver does not have to care about maintaining a desired speed u_i. Some exogenous uncertain events ξ_i, $i = 0, 1, 2$, (e.g. other drivers, traffic jams, ...) perturb his desired speed u_i on the road segment $i + 1$. After a deceleration, the speed regulator brings the speed back to u_i. We denote by $v_i(\xi)$, $i = 0, 1, 2, 3$, (for velocity) the actual average speed of the car on segment i. The average speed achieved on one segment may influence the average speed on the next segment. A first objective of the driver is to arrive

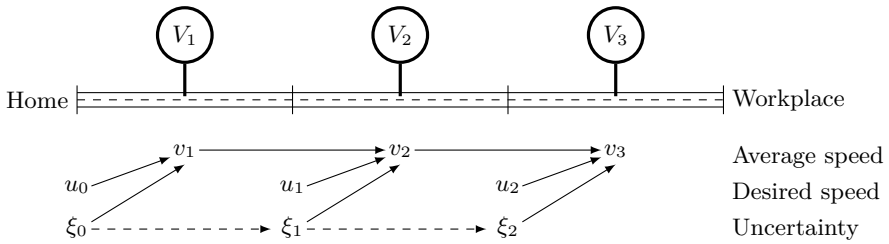


Figure 1.9: The three segments of the path together with their speed limitations, desired speeds, uncertainties and actual average speeds. Continuous arrows illustrate the direct influences between the variables of the problem. Dashed arrows illustrate that the uncertainties on the different segments may be correlated.

at his work as fast as possible, i.e. to minimize

$$\mathbb{E}_{\mathbb{P}_\xi}\left\{\sum_{i=1}^{3}(V_i - v_i(\xi))\right\}. \tag{1.21}$$

But depending on ξ_i, $v_{i+1}(\xi)$ may be different of u_i. The driver considers that he receives a penalty when his desired speed u_i is very different from its actual speed $v_{i+1}(\xi)$ on segment $i+1$. This penalty, measured by $R : \mathbb{R}^2 \to \mathbb{R}_+$, penalizes the excessive fuel consumption, usage of the brakes, etc. We do not detail this function, but require that $R(u, v(\xi)) = 0$ if $u = v(\xi)$. A second objective of the driver is thus to minimize

$$\mathbb{E}_{\mathbb{P}_\xi}\left\{\sum_{i=0}^{2} R(u_i, v_{i+1}(\xi))\right\}. \tag{1.22}$$

If the driver defines the desired speed u_i to drive at the highest allowed speed V_{i+1} on segment $i+1$, he is ensured to minimize the driving time (1.21) since he always drives at the maximum allowed speed except when some unexpected events perturb his desired speed. However the big disadvantage of this strategy is that the sum of penalties (1.22) may be very high, because he does not account for uncertainty to adapt u_i, $i = 0, 1, 2$.

Using Formulation 4[5], the driver may compute the speed u_0 for segment 1 without having any information about ξ, and then establish some recourses strategies $u_1(\xi_{[0]})$ and $u_2(\xi_{[1]})$ for the subsequent segments provided the observation of ξ on the previous segments. To establish a trade-off between the concurrent objectives (1.21) and (1.22), as they may not have the same importance for the driver, the coefficient α is introduced in the objective function (1.23). The functions F_i model the environment of the driver.

1.4.2 Learning and optimizing the driver's strategy

Experience based supervised learning. A first procedure to compute automatically a driving strategy would be to record in a dataset the actions u_i

[5] $v(\xi) = (v_0, v_1(\xi), v_2(\xi), v_3(\xi))$ (we may assume that $v_0 = 0$) and $\xi = (\xi_0, \xi_1, \xi_2)$.

Formulation 4: Driver's problem

$$\min_{v(\xi),u} \quad \mathbb{E}_{\mathbb{P}_\xi}\left\{\sum_{i=1}^{3}(V_i - v_i(\xi)) + \alpha\sum_{i=0}^{2} R\left(u_i(\xi_{[i-1]}), v_{i+1}(\xi)\right)\right\} \tag{1.23}$$

$$\text{s.t.} \quad \forall \xi \in \Xi,$$

$$v_{i+1}(\xi) = F_i(v_i(\xi), u_i(\xi_{[i-1]}), \xi),\ \forall i \in \{0,1,2\}, \tag{1.24}$$

$$v_i(\xi) \in [0, V_i],\ \forall i \in \{0,1,2,3\}, \tag{1.25}$$

$$u_0(\xi_{[-1]}) = u_0, u_1(\xi_{[0]}), u_2(\xi_{[1]}) \in \mathbb{R}. \tag{1.26}$$

taken by the driver along the complete path and the realization of the stochastic processes that have perturbed his behavior, for a series of trips from his home to his workplace. Then we would estimate a mapping allowing to predict the actions to take as a function of the stochastic events perturbing the driver. With this architecture we can use SL to approximate the strategy of the driver. But if the actions taken by the driver are too suboptimal with respect to the objective (1.23) defined over the whole path we cannot hope to improve his strategy. We can of course remove of the dataset the actions which led to bad performances and thus avoid learning from these bad actions.

Reinforcement learning. If we assign a reward to the actions of the driver (e.g. inspired of the objective function of Formulation 4), we could determine, given a collection of tuples of the form (current state, action, next state, reward) – where the next state is function of current state, the action and the stochastic process – a strategy which does not only take into account good actions, but is also aware of the actions yielding bad rewards. With RL we could thus optimize the strategy of the driver (e.g. by using algorithm of Table 1.3) from its sole experience. We could also generate tuples from simulation of the system, if the latter were known and that a statistical model of the uncertainty were available.

Optimal simulations based supervised learning. On the other hand if one has a model of the system i.e. of F_i and R, and also a statistical model to generate some disturbances ξ, thus a mean to simulate offline the quasi-optimal actions to respond to a sequence of simulated disturbances, one can hope to learn by SL a better strategy than the one obtained with experience based SL. This is the type of approach we investigate in Part I. Then comes the question of the information available at the time the driver takes the actions. In the experience based SL setting, the driver uses the information he currently has to take the decisions and thus acts non-anticipatively. In an offline computation setting one must take care of that in order to obtain decisions which do not depend explicitly of the future of the stochastic processes, but only of the information gathered until the moment at which the decisions are taken. This

is a fundamental concern of stochastic optimization paradigms (Section 1.2.2) which account for this gradual information flow.

Receding horizon optimization. If one is able to determine sufficiently fast an optimal sequence of actions accounting for the uncertainty, one can imagine applying it before each road segment when an action is needed, implement the first action proposed by the optimization algorithm and discard the rest of the sequence, observe the realization of the stochastic process until the next action is required, reapply the optimization algorithm, implement its first action, and so on. This would be a receding horizon strategy.

1.5 Thesis content and contributions

Sections 1.5.1 and 1.5.2 summarize the content and present the organization of the two parts of this document. They also state the contributions of this thesis, the published material as well as the developed software.

1.5.1 Part I – Short-term electricity generation scheduling and recourse policies computation

Content and organization. In Part I we develop the idea of computing a recourse policy by learning from simulations of the optimal actions to take to control a system under perturbed conditions. In particular, in the electricity generation scheduling context, Part I deals with the design of intra-day recourse strategies which may be used by operators to decide in real-time the modifications to bring to planned generation schedules of a set of units in order to respond to deviations from the forecasted operating scenario, i.e. when they collect information about the outcome of some exogenous random variables which influence the generation schedules. Chapter 2 introduces the short and very short-term generation scheduling problems in details, and presents the related work mainly founded upon the stochastic programming paradigm. Our aim is to design strategies that are interpretable by human operators, that comply with real-time constraints and that cover the major disturbances that may appear during the next day. To this end we propose in Chapter 3 a framework using supervised learning to infer such recourse strategies from simulations of the system under a sample of scenarios representing possible deviations from the forecast. This chapter describes the different steps and alternatives of the proposed supervised learning approach and contains the main methodological contributions of Part I. They are supported by experimental validation results on a realistic test system of medium size in Chapter 4. Chapter 5 concludes this first part and discusses directions for further work. Some background information about the supervised learning algorithms used, the formulation of the generation scheduling problem and the optimization-based solution techniques can be found in the appendices.

Contributions. The main contributions of the first part are

- the formulation of the problem of recourse policy computation from a set of optimal open-loop adjustment decisions to some disturbance scenarios as a supervised learning problem:
  - this approach allows to perform (nearly) all the time consuming computations offline,
  - to apply online, thanks to the generalization properties of supervised learning models, the resulting policy to any new scenario representing the actual conditions of the system,
  - and thus also provides an elegant and fast way to evaluate offline the resulting policy on an independent set of scenarios;
- the discussion of the connection and the complementarity between the approach we propose and the stochastic-programming based approaches.
- the validation of our method on a realistic problem encompassing a hydrothermal electricity generation system.
- the definition of several SL formulations (of which two are evaluated) from which we can select the degree of information provided by the recourse policy and the type of post-processing that one wants to apply.

Publications. One part of the work presented in Part I has been published in the proceedings of the IEEE PowerTech Conference (Cornélusse et al., 2009b). It covers approximately and with fewer details the material of Chapter 3 and of Section 4.4.

Another work (Cornélusse et al., 2007) indirectly related to this part was published during my first Ph.D. year and discusses the application of SL to the verification of the usage of primary control devices (cf. Section 2.2.1).

Implementation. A significant effort has been devoted to the development of several pieces of software to obtain the results of Chapter 4.

1. A C++ software interfacing the CPLEX MILP solver (about 3500 source lines of code[6]) takes a description of the generation system as input and computes a generation schedule, solution of the problem exposed in Appendix B. XML document types have been defined to represent scenarios and generation schedules. A set of options allows to use this software to generate the optimal adjustments to some generation schedules for perturbed scenarios, thus to generate the output part of the training set, and to post-process the decisions provided by the SL based recourse policies for offline validation or online exploitation.
2. A Python/Gtk User Interface (about 2000 source lines of code) for the simulation of scenarios, for the collection of data, for the formatting of data in order to apply SL, and for the synthesis of the results.

[6]We report on physical lines, cf. `http://en.wikipedia.org/wiki/Source_lines_of_code`.

3. Some Matlab routines (about 500 source lines of code) allowing the application of SL algorithms and the application of the recourse strategies.

4. A C++/Qt User Interface (about 1000 source lines of code) to ease the visualization of the generation schedules.

1.5.2 Part II – Prior knowledge in supervised learning algorithms

Content and organization. As explained in Section 1.1, we propose in Part I a supervised learning methodology to approximate the set of solutions of a family of optimization problems. In many cases the outputs of the resulting learning problem are restricted by complex relations arising from the constraints of the optimization problems. We thus believe that it is worth to analyze how this knowledge about the structure of the output space $\mathcal{Y}$, or more generally of the joint input-output space $\mathcal{X} \times \mathcal{Y}$, is or could be exploited by the learning algorithm.

- When we apply a SL algorithm to search in a hypothesis space based on data that satisfies a known set of constraints, can we guarantee that the selected hypothesis will make predictions that satisfy the constraints?

- Can we benefit of our knowledge of the constraints to eliminate some hypotheses while learning and thus enhance the generalization error of the selected hypothesis?

These questions initiated the work presented in Part II.

However in Part II we do not discuss these questions with respect to the application of Part I, but rather with respect to problems arising in the machine learning community that share some structural similarities. Chapter 7 motivates the interest of this work when learning from censored data and in the framework of semi-supervised learning. It then presents the optimization formulation developed in order to modify an ensemble of regression trees so as to incorporate prior knowledge. In Chapter 8, learning from censored data and manifold regularization for incorporating unlabeled data are cast in the general formulation, and the way to incorporate other types of information is also discussed. Chapter 9 concludes the second part.

Contributions.

- We have developed an original method for regularizing ensembles of regression trees using prior knowledge.

- We have applied this method to some problems which cannot be treated directly by standard tree-based methods.

Publication. Part II is mainly based on the work presented in Cornélusse et al. (2009a), but contains some additional illustrations and examples.

Implementation. I have used the Matlab interface to the Extra-Trees learning algorithm developed by Pierre Geurts as a starting point for the implementation of the method. I have used SeDuMi (Sturm (1998)) for the resolution of the optimization problem through the Yalmip interface (Löfberg (2004)).

Part I

Short-term electricity generation scheduling and recourse policies computation

Chapter 2

Day-ahead and intra-day electricity generation scheduling

In this chapter we introduce the problems that are faced everyday by an electric power generation company[1] in order to operate their generation system optimally. We start by a discussion of the typical partition of the global problem in Section 2.1. Then we provide some details about the two problems of this partition that we study in this thesis, namely day-ahead (Section 2.2) and intra-day (Section 2.3) scheduling, show how these problems fit in the framework exposed in Section 1.2, and outline the solutions proposed in the literature in this framework. We end this chapter by highlighting in Section 2.4 the interests of the approach exposed in Section 1.1, which makes use of the machine learning tools and concepts discussed in Section 1.3.

2.1 Partition of the generation scheduling problem

To allow the development of the electricity market in Europe, the activities of generation, supply, transmission and distribution of electricity, which were usually grouped inside electrical utilities, have been reorganized and attributed to different companies. Generation companies are in charge of physically generating the electricity from other sources of energy. Electricity retailers purchase electricity directly to generation companies or through the electricity markets and sell it to some customers. A transmission system operator (TSO) is in charge of the security and the safety of the high voltage transmission infrastructure over a geographical area (e.g. a country). TSOs are responsible for the transmission of electrical power from nodes of the network where power is injected to nodes where power is consumed. Distribution operators are companies that operate the low-voltage part of the network to distribute electricity to the end users. Some (small-scale) distributed generation may also be connected to this network (e.g. solar cells). Generation companies and electricity retailers often play the role of "balance responsible entities", i.e. they must ensure that the balance between their injected and consumed electricity (e.g. on a half-hourly basis) in a given "perimeter" is satisfied, otherwise they pay

[1] The shorthand "generation company" will be used from now on.

some penalties to the TSO in charge of this perimeter. TSOs are themselves responsible for the balance in their geographical area and must provide the physical means to adjust the generation to the consumption level dynamically.

In this thesis we take the point of view of a generation company which wants to generate electricity in an optimal way, i.e. to satisfy some demand requirements at minimum cost, and do not consider the network related problems, although some network related constraints appear indirectly in the studied problem, as will be explained further in this text.

Globally optimizing the electricity generation is a very complex problem. It implies decision making ranging from the creation of new power plants to the decision of which generators should be adjusted to maintain a perfect balance between generation and demand in a quasi real-time context. As a consequence, a hierarchy of easier problems spanning different time horizons has emerged. Problems over a given time horizon are decoupled from the problems over shorter horizons by aggregating some decision variables and simplifying the constraints they are subject to, and from those over longer time horizons by considering the decision variables related to these problems as fixed parameters.

Long-term problems span several years and relate to investments in new power plants and new generation techniques. Should the company opt for fossil or nuclear fuel? Does-it have the possibility to build new dams and hydroelectric capacities? Should it invest in renewables? These questions usually involve social and political considerations in addition to the strategic decisions of the generation company.

Typical generation systems contain a mix of generation units: classical thermal plants, nuclear plants, hydroelectric generators, wind turbines, and other renewable energy based plants. *Medium-term decisions* (one year) consist in determining the way these units will be used altogether to fulfill the load. One has to decide when to stop plants for maintenance, when to buy some fuel, when to refuel the nuclear reactors, how to use the water reserves, ... (see for example Wood and Wollenberg (1996, Chapter 8) for a detailed description of the problem and Aïd et al. (2004) for an application of RL in this context).

Today in the context of bulk power generation, the wind turbines are often considered as non controllable and must-run units; their generation can thus be seen as a stochastic negative load. On the other hand, the operation of thermal plants and hydroelectric generators has to be planned in advance every day. The process of deciding the schedule of the units one day for the next day (or possibly for several days) is called *day-ahead generation scheduling*, and is described in more details in Section 2.2.

Finally, during the day, when the schedule computed one day ahead is applied, the generation company has the opportunity to adjust the dispatching of its units to correct deviations of uncertain processes (demand, availability of units, ...) from their forecast. To this end the generation company elaborates *intra-day recourse strategies* that are used by operators in order to decide of economically efficient and feasible modifications to make to the generation schedule. To be useful, such recourse strategies must comply with real-time constraints and be interpretable by human operators. At the same time, they should cover all the major likely or unlikely disturbances that may appear dur-

ing the day, such as the loss of any generation unit and/or significant deviations from forecasted demand conditions. In current practice, a day-ahead schedule (unit commitment) of all power plants is typically computed one day ahead for the next 24 hours, by using a detailed model of the generation system and an appropriate optimization algorithm, while the intra-day recourse strategies are pre-determined by experts during offline studies. This problem is further discussed in Section 2.3. Within this context, the purpose of our work is to develop a systematic and essentially automatic approach to (re)design intra-day recourse strategies compatible with real-time constraints and interpretable by human operators, so as to allow them to respond to an a priori defined set of disturbances in a near-optimal way.

The above mentioned problems can all be described as sequential decision making problems under uncertainty. They are sequential because some decisions, or actions, must be taken repeatedly, with different time constants for each problem, and uncertain because all of these problems are subject (with varying degrees) to many uncertain events such as the uncertainty on the electrical load, the outage of some generation units, the uncertainty on natural water flows, the prices of fuels, ..., which are themselves function of the temperature, the precipitations, and so on.

2.2 Day-ahead electric power generation scheduling

We consider hydro-thermal generation systems, meaning that the generation company owns a number of thermal generation units and also exploits the water retained in reservoirs along some valleys. The generation company has to compute a schedule of its generation units for the next day such that the total generated power balances a demand curve[2]. Many causes influence the profile of the demand curve. Among them the temperature is probably the most prominent one. But in addition the generation capacity related to the renewable sources of energy is highly stochastic on a daily basis, and the power generated by such means must often be consumed directly. In a sense, the power generated from renewable sources can be deduced when estimating the demand forecast. Although some large-scale energy storage such as water reserves or batteries could provide the flexibility to cover deviations of the demand curve, they are in general either not available in sufficient amounts or of limited use because of operating and environmental constraints. Finally, some generation plants are scheduled for maintenance or refueling, and there may be fortuitous plant outages. In typical operating conditions all these factors imply the necessity of an automatized system for deciding when generation units must be started or stopped and which generation levels must be modified. The generation company has multiple objectives when computing the generation schedule. Maybe the most obvious one is to maximize the profit. We study this aspect from a slightly different point of view, namely the minimization of the power generation costs. Another quantity needs to be minimized, originating from a second objective which is to provide a generation schedule of good quality which is safe and

[2]The word "demand" reflects here the expected quantity of power that has to be served, while the word "load" refers to the realization of the process underlying the consumed power.

robust from the electrical network point of view. A third objective is to compute a generation schedule which does not violate the technical limitations of the individual generation units or their environment. A fourth objective is the robustness with respect to uncertainty on the parameters of the model used to compute the generation schedule.

Thus two sets of constraints appear in the context of short-term generation scheduling: on the one hand the cumulative generation pattern must satisfy some balancing constraints which couple all the generation units, on the other hand the generation of each thermal unit (Section 2.2.2) and each hydroelectric valley (Section 2.2.3) is restricted by individual operation constraints. To compute a generation schedule, the most widespread practice among generation companies is currently to partition the next day in small time periods (e.g. periods of 30 minutes) and to compute the schedule (Section 2.2.5) which best fits a demand curve, according to a criterion to be defined (Section 2.2.4), assuming a reference scenario for all the parameters of the system that can be subject to uncertainty. Typically the reference scenario contains the availability of the generation units for the next day as well as their generation restrictions and other characteristics, an estimate of the demand curve and an estimate of the other uncertain events having a significant influence on the generation schedule, e.g. the flows of water in valleys caused by rain, snow melting, ...

A schedule computed using this procedure is automatically adjusted in real-time the next day in order to compensate the differences between the real-time conditions and the forecasts (Section 2.2.1). Above the automatic devices acting in real-time, some periodical reorganizations of the generation schedule implying a decision making process are nevertheless required in order to restore a secure an economically efficient operation (Section 2.3) of the generation system.

2.2.1 Automatic balancing of the load

To maintain tightly the balance between generation and load in real time operation some generation reserves are typically required by the TSO. Using these reserves, some control systems – called ancillary services – guarantee a reliable operation of the synchronous electrical network to which the generation system is connected. In the context of this thesis, a hierarchy of three types of service exist, namely primary, secondary and tertiary control. The expression "generation reserve" must be understood as the generation capacity which can be activated or deactivated quickly, depending on the type of control, whatever the type of generation unit considered. For example, the quantity of reserve provided by a thermal generation unit is function of its extreme generating levels and of its ramp rates if it is started, and zero otherwise (except for some special units which can be started up or shut down very fast). The total quantity of reserve that must be provided by the generation company for each service is determined by the TSO in order to be able to restore the nominal situation after a perturbation. The required quantity can vary with the time of the day and thus typically one reserve curve is predefined for each ancillary service. The first two services are implemented on the generation units by automatic regu-

lators driven by measurements of some electrical network characteristics. Both types of control are complementary and aim jointly at balancing the generation and the load, and at restoring the frequency of the network at its nominal value after a perturbation. Their differences are pointed out below.

The reserve dedicated to primary control represents the total variation of generated power that can be accomplished within a few seconds after the occurrence of a perturbation of the frequency caused by an imbalance between generation and load. Primary control is a decentralized control achieved through the speed regulators of generators. This control is proportional to the difference between the measured frequency and the nominal frequency of the network. It is thus a common reaction among all the generators of the same synchronous area (the continental Europe network for example). Because of its proportional nature it leaves a permanent frequency error and since it is a reaction common to several generation companies it results in errors between planned and actual cross-border power exchanges.

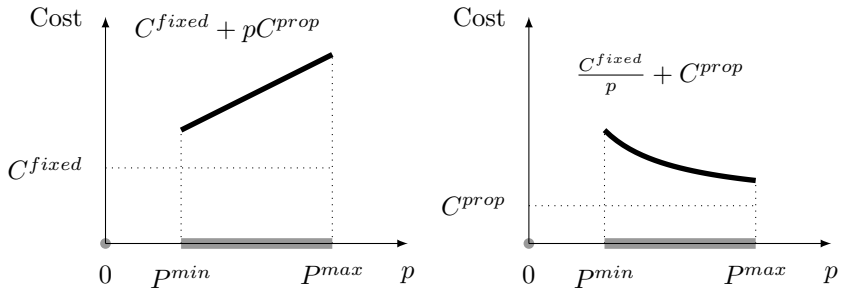
Secondary control is on the other hand a centralized and proportional-integral control based on the measurement of the frequency error, but also on the measurement of cross-border exchanges imbalances. It restores primary control reserves and acts in order to cancel the frequency error and to transfer the compensation of the imbalance inside the control area of the generation company where the perturbation originated. The total secondary reserve inside a control area should be sufficient to correct any perturbation in this area.

The third type of service, tertiary control, is triggered by human operators and aims at re-dispatching the action of secondary control in an economical way and at reestablishing the secondary reserve level to face another perturbation. Its associated reserve can be understood as a robustness margin to recover from usual random disturbances. As pointed out in Carpentier et al. (1996), this way of tackling the problem may lead to solutions distant from the optimal operation of the generation system, because it takes into account the stochastic nature of the problem very roughly. This is the reason why only the first two are considered in this thesis and that the tertiary reserve requirement has not been modeled in our work. Indeed it is the purpose of this thesis to design adjustment strategies for optimally rescheduling the generation means after a perturbation has been observed.

The actual quantity of reserve that is used in operation is thus function of some measurements, but also of the characteristics of the regulators for primary and secondary control, and of a decision making process for the tertiary control. Additional information about the purposes of the ancillary services can for example be found in UCTE (2004, 2009) or RTE (2004).

2.2.2 Thermal generation units

Thermal generation units of different types (nuclear, coal, fuel oil, gas) are characterized by different generation costs and dynamics. When a thermal unit is started it must operate above a minimum positive level P^{min} and below a technical maximum level P^{max}, i.e. in the gray interval of the p axis in Figure 2.1a. As illustrated on this figure, the operation cost is the sum of a fixed



(a) Cost per hour. (b) Cost per Mega Watt hour.

Figure 2.1: Illustration of the thermal generation cost as a function of the generation level.

cost and of a cost proportional to the generation level. Figure 2.1b represents the cost per unit of energy generated and stresses the fact that due to the fixed cost it is more profitable to operate near the maximum level. Costs are also incurred at each start-up and are function of the time the generation unit has been stopped before being started again. The dynamics of a thermal unit are restricted by a set of constraints. In the model we use in this thesis (described in appendix B), a generation unit must stay in the same On/Off state for a minimum duration, depending on that state, and maximum ramp rates limit the variation of the generation level between consecutive time steps. However many other practical constraints may restrict their dynamics, such as a limit on the number of times the generation level of a unit may be modified each day, or particular generation curve that the unit must follow in case of start-ups or of shutdowns. Knowing all these constraints, the generation company has to decide when to start and when to stop thermal units and to fix their generation level when they are committed.

2.2.3 Hydroelectric valleys

A hydroelectric valley contains a set of reservoirs interconnected by hydroelectric plants. Hydroelectric plants may themselves contain generation and/or pumping capacity. In the sequel hydroelectric generation plants and pumping plants are considered as distinct entities, even though they may be gathered in a single plant in practice. Figure 2.2 illustrates a valley composed of two reservoirs, two generation plants and one pumping plant. The arrows indicate the allowed direction of water flows. The generation and pumping plants are connected to the electrical network. Electricity can thus be generated and transmitted to the network by turbining water from reservoir 1 and sending it to reservoir 2, and/or by turbining water from reservoir 2. Alternatively one can choose to send water back from the second to the first reservoir using the pump, and thus consuming electricity. The volume of reservoirs must stay in a predefined interval at each time step (the interval may shrink to a single value in order to restore some levels at some particular instants). The volume of each

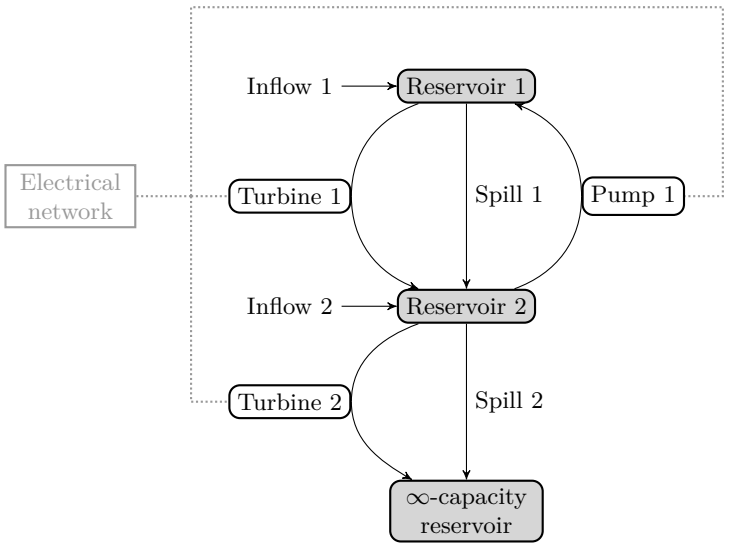


Figure 2.2: Representation of a hydroelectric valley.

reservoir evolves as a function of the water flows passing by the surrounding power-plants, the natural flows and the water spilled out when the reservoir is full. A hydroelectric generation plant is composed of turbines that must often, in practice, be started in a given sequence. In this work, the electric power generated by a hydroelectric plant is modeled as a piecewise linear function of the water passing through it. Pumping plants have rated discrete levels of operation. The variation of the head height has a significant impact on the power generated but causes nonlinearities and is not considered in our work.

The cost associated with the use of the water contained in a valley is computed on the basis of the value assigned to its upper reservoirs. This value – evaluated in the medium-term problem (cf. Section 2.1) – is considered given and constant in the short-term context, and reflects the opportunity cost of using the water, itself function of the abundance or scarcity of the water (seasonality, ...), instead of other sources of energy.

For the hydroelectric valleys one has to decide when to use water to generate electricity, when to store some water by pumping and when to spill water out of dams.

2.2.4 Targeted performance

The objective of the generation company is to minimize the generation costs, including fuel costs, thermal units start-up costs and opportunity costs for the utilization of water in hydroelectric power plants, in order to satisfy the balancing of generation and load as well as the ancillary services requirements. To some extent, computing a generation schedule which satisfies the demand requirement tightly may be hard to achieve numerically, and even unnecessary since some means are available to balance the actual load, which is generally not

equal to the demand forecasted one day ahead. To soften this requirement, some slack variables are introduced to represent the discrepancy between generation and demand and are penalized in the objective function through a piecewise linear and convex function. If the penalization function approximates, for example, a quadratic function, this tends to penalize more, proportionally, large discrepancies than small discrepancies. Similarly some slack variables are introduced to represent the discrepancy between required and provided ancillary services reserves. The penalization function equals zero when the provided reserve exceeds the requirement. Thus only the deficit of reserve is penalized. Again the penalization is convex, and piecewise linear if one wants to preserve a problem whose relaxation is a linear programming problem. Note that although excess of reserve is not penalized explicitly, it moves the solution away from the economical optimum.

2.2.5 Deterministic optimization model

The detailed optimization model that we use in our experiments of Chapter 4 is provided as an appendix (cf. Appendix B) to make the text more easily readable. We present a condensed version in Formulation 5 to provide a basis for the sequel.

Formulation 5: Deterministic generation scheduling.

$$\min_{x,x_{slack},u} \quad \sum_{t=1}^{T} L(x_{slack,t}) + \sum_{t=0}^{T-1} \sum_{i \in I} R_i(x_{i,t}, u_{i,t}) \tag{2.1}$$

$$\text{s.t.} \quad \forall t \in \{1, ..., T\},$$

$$x_{slack,t} = Y_{req,t} - \sum_{i \in I} C_i x_{i,t}, \tag{2.2}$$

$$x_{slack,t} \in \mathbb{R}^3, \tag{2.3}$$

$$\forall i \in I,$$

$$x_{i,t+1} = F_i(x_{i,t}, u_{i,t}), \quad \forall t \in \{0, ..., T-1\}, \tag{2.4}$$

$$x_{i,t} \in \mathcal{X}_{i,t}, \quad \forall t \in \{0, ..., T\}, \tag{2.5}$$

$$u_{i,t} \in \mathcal{U}_i(x_i), \quad \forall t \in \{0, ..., T-1\}. \tag{2.6}$$

The generation scheduling model presented in Appendix B contains continuous variables (the generation levels, the power reserves, the water flows, ...) and binary variables (the On/Off status of thermal units, the auxiliary variables used for expressing the piecewise linear relationships of hydroelectric turbines, ...). However the constraints and objective function are all linear expressions. The continuous relaxation of the problem obtained by replacing the binary variables by continuous variables ranging in $[0, 1]$ is thus a linear

program (LP), and the scheduling problem a MILP[3].

This formulation is consistent with the optimal control Formulation 1 except that the state, the control actions, the functions F and R and the matrix C are indexed by generation unit, and that the tracking error is represented by an auxiliary vector $x_{slack,t}$ at each time step. Here $x_{i,t}$ is a vector representing the state of the unit i at time t, belonging to the set of (thermal and hydroelectric) generation units I. For example the state vector of a thermal generation unit contains its generation level, its generation reserves and its status at time t, plus the information necessary to express the operation constraints (2.5) to (2.6), and the operation costs. This latter information actually consists of the generation and status of unit i for times $t' \in \{t - D_i \dots t - 1\}$ where the parameter $D_i \geq 1$ depends on the characteristics of the unit. The vector $u_{i,t}$ of control actions modifies the state $x_{i,t}$ through the function F_i. The continuous variables of $u_{i,t}$ are essentially the modification of the generation and margins levels while the discrete variables are the decisions to start or stop a thermal generation unit or turbines and pumps in a hydroelectric plant. The function $R_i(\cdot) : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ expresses the operation costs, e.g. the fixed, proportional and start-up costs of thermal generation units.

In the coupling constraints (2.2), the slack vector $x_{slack,t} \in \mathbb{R}^3$ gathers the imbalances between the required levels of generation and ancillary services $Y_{req,t} \in \mathbb{R}^3$ at time t and the actual levels provided by all the units. The contribution of a unit i to these requirements is obtained by multiplying $x_{i,t}$ by the matrix C_i. Non-zero values of the slack variables are penalized in the objective through the loss function $L(\cdot) : \mathbb{R}^3 \to \mathbb{R}_+$.

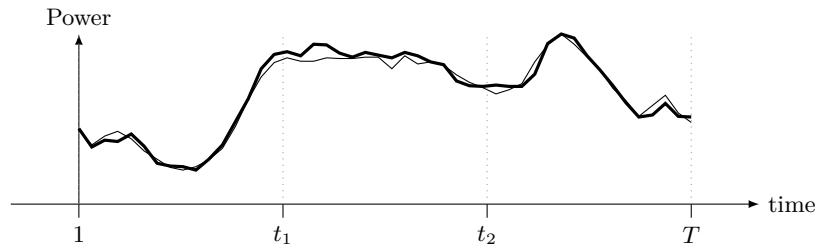
Instances of Formulation 5 are routinely solved by generation companies to derive every day the planned operation of their assets for the next 24 hours. Appendix C.1 describes the most common approaches to solve such instances. Finally, note that often there exists additional constraints which cannot be included in the optimization formulation and are imposed during a post-processing stage by altering the solution of Formulation 5.

2.3 Intra-day adjustment of the generation schedule

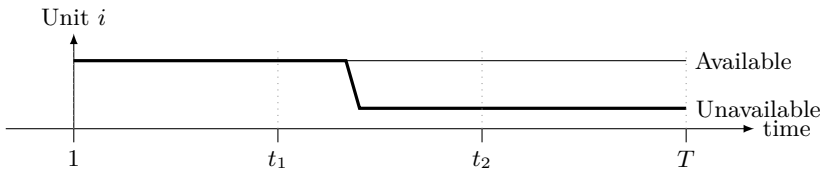
2.3.1 Real time exploitation

In the system we consider, a reference schedule for the complete optimization period must be communicated to the TSO one day ahead. The TSO can then perform network security analyses and optionally modify the generation schedule before its implementation. Then during the application of the schedule computed one day ahead, some events can motivate a generation company to re-dispatch its generation units. In practice there exists several predefined recourse instants (e.g. t_1 and t_2 in Figure 2.3) at which it is possible to observe the realization of the uncertain process (Figure 2.3a and 2.3b) impacting the generation system and to adjust the scheduled operations (Figure 2.3c). It is thus not possible to adjust the schedule continuously, or at least at arbitrary

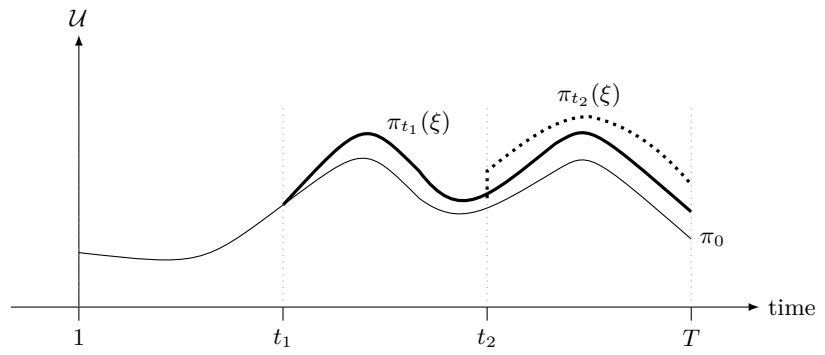
[3]A continuous relaxation of a MILP will be referred to as a *LP relaxation* in the sequel.



(a) Demand/load curve.



(b) Availability of generation units.



(c) Generation schedule.

Figure 2.3: Illustration of the intra-day generation rescheduling problem: (a) forecast (thin) and realization (thick) of the load curve, (b) forecast (thin) and realization (thick) of the availability of a generation unit and (c) illustration of the recourse policies $\pi_{t_1}(\xi)$ and $\pi_{t_2}(\xi)$, where the thin curve depicts the scheduled actions computed one day ahead, the thick curve represents the recourses taken at time t_1 thanks to $\pi_{t_1}(\xi)$ after having observed the load deviation on $[1 : t_1]$, and the dotted curves represents the recourses taken at time t_2 thanks to $\pi_{t_2}(\xi)$ after having observed the load deviation on $[1 : t_2]$ and the unit outage happening during $[t_1 : t_2]$.

instants, as primary and secondary controls do. An updated schedule can be provided at each recourse instant, and it must span the whole time interval until the end of the optimization period. The adjustments made to the schedule aim at maintaining the balance between generation and load at lowest cost, while also satisfying the operation constraints of the units and restoring the ancillary services reserves. This process is thus related to tertiary control, as explained in Section 2.2.1. The information gathered on the stochastic process ξ up to the recourse instant gives insight on the variation of the parameters influencing the generation schedule on the recourse period and thus on the adjustments to make to the reference schedule. Note that in practice there may be an additional constraint to the scheduling problem in the intra-day context that sets a limit on the number of generation units that can be modified (cf. Section 3.3.3).

The main uncertainty sources are the variation in forecasted temperature which cause load variations and possibly restrictions on the capacity of generation units, the outages of generation units, the variations of the natural water flows incoming in reservoirs, the fluctuations of the other renewables and the evolution of the fuel and of the spot market prices. In this thesis we have chosen to focus on load variations and outages of (or restrictions on) generation units because they have typically the most significant impact on the generation schedules. The other sources of uncertainty may sometimes be gathered in the load disturbances (e.g. solar and wind energy) or may influence other parameters of the models (e.g. water flows, fuel prices). Some may also require additional modeling effort (e.g. spot markets).

2.3.2 Deterministic receding horizon optimization

To adjust the reference schedule, a straightforward procedure consists in re-optimizing completely the schedule at each recourse instant, using one updated forecast of the demand and one updated forecast of the availability of the generation units, from the current state of the system at that instant, i.e. in a deterministic receding horizon manner.

Example. To give some insight on this procedure, Figure 2.4 illustrates the optimal adjustments of a subset of generation units owned by the generation company to a series of scenarios. The subset is composed of one hydroelectric plant and one pumping plant situated as Turbine 1 and Pump 1 in Figure 2.2, one 128 MW thermal generation unit and another one of 585 MW. First we optimized the generation schedule for a reference scenario, as would be done one day ahead in practice. Then we considered 11 perturbed scenarios containing each the outage of the same generation unit, but occurring at a different time step from time 0h00 to 10h00 with a 1 hour interval. The same load curve was used in all the simulations and the complete generation system is the same as in Cornélusse et al. (2009b). In each subfigure of Figure 2.4, the horizontal axis indexes the scenarios by the hour of occurrence of the outage. The vertical axis represents the optimization horizon, and the color indicates the magnitude of the adjustment. We assume that we can implement a recourse at each hour,

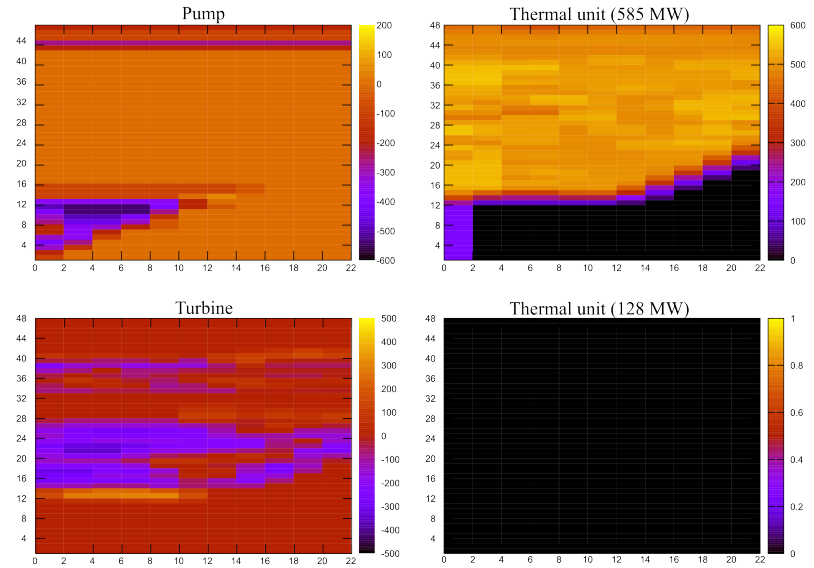
thus directly after the occurrence of the outage, and have enough time to re-optimize the generation schedule completely. Figure 2.4a shows the resulting adjustments if the lost unit is a 900MW unit, while Figure 2.4b shows the result of losing a 1300 MW unit.

In Figure 2.4a, the top left subfigure indicates that if the outage occurs early in the morning the pump stops sending water to the upper reservoir. As a consequence the turbine situated between the same reservoirs generates less power than planned in the reference schedule during the afternoon and during the evening peak. On the other hand, the 585 MW thermal unit is started as soon as possible and the unit of 128 MW remains stopped. If a 1300MW nuclear unit is lost instead of a 900MW one (Figure 2.4b), we observe nearly the same behavior, except that the turbine is almost stopped during the evening peak load. The 585 MW thermal unit operates at its maximal power, and the 128 MW thermal unit is started up. These kinds of pictures provide interesting information since they show the optimal adjustments resulting from a unit outage and also the influence of the occurrence time of the outage.

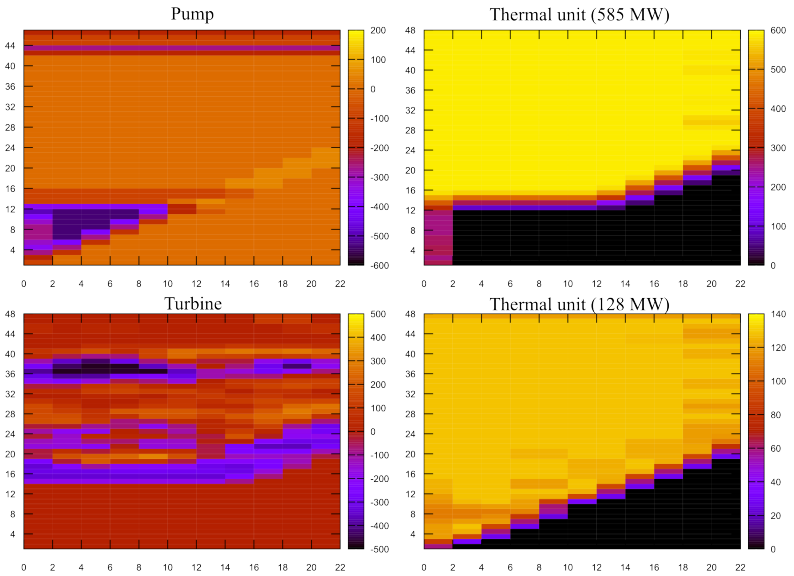
Discussion. The updated forecast on which the optimization is based is however never perfect, and as for the short-term generation scheduling problem, building the adjustments without taking into account the uncertainty on the forecast may lead to suboptimal decisions as uncertainty unveils. The overall problem, encompassing short to very short-term operation, should thus ideally be handled as a multistage stochastic problem.

As the number of recourse instants increases the available computation and implementation times decrease. It thus becomes harder to re-optimize completely the system from scratch at each recourse instant. Adjustments implementation related issues in general prevent a complete re-optimization of the generation schedule. Moreover it is maybe preferable for the generation company to have a small quantity of adjustments to perform. Thus some constraints may be incorporated in order to limit the number of adjustments (cf. Appendix B.2.2.2). Although such a requirement may seem to decrease the difficulty of the adjustment computation task since one has to search for simpler solutions, this is generally not the case for an optimization based solution, since it adds a combinatorial layer to the problem. To speed up the computation of the adjustments one thus typically uses some hand-made and empirical recourse rules.

As discussed in Section 1.2.2, since the reference schedule can be adjusted at several recourse instants to compensate stochastic variations of the model parameters, it would be comfortable to compute on the one hand a reference schedule accounting for the uncertainty on the model parameters and the recourse opportunities, and on the other hand some recourse policies that would be used to adjust the reference schedule at the recourse instants. Actually, the recourse policies should be computed jointly with the reference schedule since the recourse actions for a particular recourse instant depend of the reference schedule and of the recourse actions implemented at previous recourse instants, and should not lead to costly adjustments or infeasibility at subsequent time steps.



(a) Outage of a 900 MW generation unit.



(b) Outage of a 1300 MW generation unit.

Figure 2.4: Optimal adjustment of a subset of units a generation company after the outage of a thermal generation unit and influence of the occurrence time of the outage. The color range indicates the power variation with respect to the generation schedule.

2.3.3 Example of scenario-based stochastic programming formulation

We provide in Formulation 6 a two-stage stochastic programming formulation of the generation scheduling problem in line with Formulation 3, in order to discuss its interpretation and to ease the understanding of the related work surveyed in Section 2.3.4. Formulation 6 is an alternative to Formulation 5 assuming that a single opportunity of recourse arises at time t_r. Since optimizing

Formulation 6: Two-stage stochastic generation scheduling.

$$\min_{x^{(k)},x^{(k)}_{slack},u^{(k)}} \quad \sum_{k=1}^{K} w_k \left(\sum_{t=1}^{T} L(x^{(k)}_{slack,t}) + \sum_{t=0}^{T-1} \sum_{i \in I} R_i(x^{(k)}_{i,t}, u^{(k)}_{i,t}) \right) \tag{2.7}$$

$$\text{s.t.} \quad \forall k \in \{1, ..., K\},\ \forall t \in \{1, ..., T\},$$

$$x^{(k)}_{slack,t} = \sum_{i \in I} C_i x^{(k)}_{i,t} - Y^{(k)}_{req,t}, \tag{2.8}$$

$$x^{(k)}_{slack,t} \in \mathbb{R}^3, \tag{2.9}$$

$$\forall i \in I,\ \forall k \in \{1, ..., K\},$$

$$x^{(k)}_{i,t+1} = F^{(k)}_{i,t}(x^{(k)}_{i,t}, u^{(k)}_{i,t}), \quad \forall t \in \{0, ..., T-1\}, \tag{2.10}$$

$$x^{(k)}_{i,t} \in \mathcal{X}_i, \quad \forall t \in \{0, ..., T\}, \tag{2.11}$$

$$u^{(k)}_{i,t} \in \mathcal{U}_i(x^{(k)}_{i,t}), \quad \forall t \in \{0, ..., T-1\}, \tag{2.12}$$

$$\forall i \in I,\ \forall t \in \{0, ..., t_r - 1\},$$

$$u^{(1)}_{i,t} = u^{(2)}_{i,t} = \ldots = u^{(K)}_{i,t}. \tag{2.13}$$

the recourse policy implies optimizing over a function space, which is hardly achievable for practical problems, a commonly used approach consists in approximating the stochastic process ξ by a set of K scenarios $\xi^{(k)}$ corresponding to realizations of the stochastic process (cf. Section 1.2.3). A scenario represents the temporal evolution of the demand curve ($Y^{(k)}_{req,t}$) and of the limitation or the unavailability of some generation units ($F^{(k)}_t$). The objective (2.7) is to minimize the expected cost of adjusting the first stage to the different scenarios. The expectation is replaced by the average over all the scenarios, and a weight $w_k \in [0, 1]$ (with $\sum_{k=1}^{K} w_k = 1$) is assigned to each scenario to represent its probability.

The first and second stages contain commitment (integer) and generation level (real continuous) variables. Constraints (2.13) express that there cannot be any adjustment or specialization to a scenario until time period t_r (non-anticipativity). We impose the non-anticipativity of the control actions and not of all the optimization variables indexed by a time smaller than t_r, since the state may differ in the different scenarios. From time t_r on, the schedule may be adjusted to track a particular scenario. This formulation allows the

decomposition of the problem in sub-problems related e.g. to single scenarios using the DDSIP algorithm described in Carøe and Schultz (1999). However for computational reasons we cannot hope to include a large number of scenarios in the tree. This may not be harmful to compute a good first stage solution, but it renders impractical the application of the second stage, since in the real time context the realization of the uncertain process is likely to differ to a large extent from the scenarios considered in the above formulation.

We can arbitrarily define the reference schedule, i.e. the schedule that would be submitted one day ahead to the TSO, as the schedule obtained for scenario $\xi^{(1)}$. This reference could be used to state some constraints limiting the magnitude of the adjustments made to reschedule the first stage for other scenarios. However each such constraint would couple the reference schedule with one schedule fitted to another scenario $\xi^{(k)}, k \neq 1$ and, as non-anticipativity or constraints coupling the generation units, must be relaxed to obtain a tractable problem. We will see in Chapter 4 that such constraints applied to a single scenario rescheduling problem already lead to a large increase of the computation time.

2.3.4 Literature survey on the incorporation of uncertainty in the generation scheduling problem

This section reviews the different ways under which uncertainty has been introduced in short-term electric generation scheduling as well as the solution and evaluation methods that have been proposed.

A multistage stochastic programming formulation is proposed in Takriti et al. (1996). Decisions, i.e. status variables and generation levels, are attached to the nodes of a scenario tree. Scenarios contain perturbed load curves and a generator failure is handled as a load increase of an amount equal to the maximum generation level of the corresponding unit. Stages are delimited by the precise instants at which the recourse can be taken. The progressive hedging algorithm (Rockafellar and Wets, 1991) is used to decompose the problem in single scenario sub-problems, i.e. to relax the non-anticipativity constraints, and to iteratively modify these sub-problems to impose the non-anticipativity constraints progressively. Each scenario related sub-problem is solved using Lagrangian relaxation of the coupling constraints. Progressive hedging is proved to converge to a global optimum in the convex continuous case but has no guarantee to converge in the mixed-integer framework. However the authors report quick convergence and good quality of the solution. The authors apply this procedure to a weekly generation scheduling problem containing about 20 scenarios. To measure the value of the hedged schedule, the authors first apply the deterministic schedule obtained for each scenario to all the other scenarios and compute the average cost. Then they apply the hedged schedule to all the scenarios that composed the scenario tree and again measure the average cost. They observe significant savings thanks to the stochastic formulation in these experiments. Spinning (tertiary) reserve constraints are not implemented in the stochastic model, the authors stating that the extreme system conditions may be hedged by including special scenarios in the scenario tree.

In Renaud (1993), EDF presents its new tools for computing daily generation schedules. The Augmented Lagrangian technique is used to solve the deterministic generation scheduling problem. The author points out the idea to move toward a quasi real time actualization of the generation schedule in order to correct the inaccuracies in load forecasts or to face any other unforecasted event; he thus argues for a receding horizon procedure in the intra-day context. The author also pins down the main difficulties of this upgrade: the necessity to satisfy accurately all the operating constraints because a time consuming manual post-processing is by nature not affordable, the integration of network secure operation, etc.

Almost at the same time as the work presented in Takriti et al. (1996), Carpentier et al. (1996) report on applying the Augmented Lagrangian technique to the multistage stochastic generation scheduling problem. Instead of relaxing the non-anticipativity constraints, the coupling constraints are relaxed first. The problem then decomposes into stochastic sub-problems (one for each unit of the thermal system considered). To compare the stochastic solution with respect to the deterministic one containing spinning reserve constraints, *simulation* trees are used and an "Open Loop Feedback Optimization" scheme is considered. A simulation tree has the same characteristics as a scenario tree used for optimization, but is larger in order to assess the decision methods for a larger spectrum of uncertainties. To evaluate the stochastic approach, a stochastic problem is solved at each node of the simulation tree by using an optimization tree, and the root decision only is implemented. To evaluate the deterministic approach, a deterministic problem with spinning reserve is solved at each node of the simulation tree (for the remaining horizon at the time the current node corresponds to), and the first decision is implemented. They report a benefit of approximately 2 % for using the stochastic solution when considering both load uncertainties and one unit failure events (but this depends on the way the optimization trees are built).

A two-stage formulation is adopted in Schultz and Carøe (1998) in order to obtain a robust schedule, in opposition to the work of Takriti et al. (1996) and Carpentier et al. (1996) which implement a receding horizon procedure, where only the first stage decisions are implemented while the decisions related to the other stages are discarded. In Schultz and Carøe (1998) the first stage gathers the start-up decisions of the coal fired units for the whole time horizon, and the recourse stage consists of the generation levels of all the started units as well as the start-up decisions of quick gas fired units for each scenario. Here both stages thus cover the same time horizon. Hence the recourse costs are taken into account when computing the first stage. Schedules are hedged against start-ups only, but are implementable for the whole time horizon. As in Takriti et al. (1996), non-anticipativity constraints are relaxed, but a branch-and-bound algorithm consisting in progressively imposing the non-anticipativity constraints is used to obtain convergence in a finite number of steps. Bounds are obtained thanks to the value of the dual problem (cf. Carøe and Schultz (1999)). The reported results are mainly oriented towards computational experience and convergence evaluation, without a real evaluation of the hedged policy. Similar considerations are reported in Gollmer

et al. (2000). Nowak et al. (2005) formulate a two-stage stochastic program that incorporates day-ahead trading in the problem, and also uses the solution method proposed in Carøe and Schultz (1999).

The paper of Dentcheva and Römisch (1998) somewhat unifies the preceding discussion by presenting two kinds of models, which differ by the quality of the information available about the demand uncertainty: a dynamic model and a two-stage stochastic model. The dynamic model is dedicated to short term operation – it is in fact a multistage model similar to the models of Takriti et al. (1996) and Carpentier et al. (1996) – where the future consequences of the current scheduling decisions are taken into account as well as future demand uncertainty: demand is supposed to be completely known at the beginning of the horizon and the uncertainty grows with the number of time periods. The two-stage model is comparable to the model in Schultz and Carøe (1998), except that start-up of large units may also happen in the recourse stage. It is dedicated to a longer term operation than the dynamic model. The objective is thus to find an optimal schedule for the whole system and scheduling horizon such that the uncertain demand can be compensated by the system (in probability), the system constraints are satisfied and the generation plus expected recourse costs is optimal. The paper then describes in details the solution method for the two models, the Lagrangian relaxation of the coupling constraints as well as the sub-gradient algorithm used to solve the dual problem. No evaluation of the schedules is reported. Gröwe-Kuska et al. (2000) reports on the same methodology to solve the multistage model, but adds an algorithm to build representative but not too complex scenario trees in order to obtain a tractable problem. Only load uncertainty is considered. Nowak and Römisch (2000) is again devoted to the same topic, but puts the emphasis on the sub-problems solution methods and on the Lagrangian heuristics needed to recover a feasible primal solution. Nürnberg and Römisch (2002) incorporate stochastic fuel prices in the two-stage model described in Dentcheva and Römisch (1998). Gröwe-Kuska et al. (2002) add stochastic fuel prices and water inflows in the dynamic model described in Nowak and Römisch (2000). Heitsch and Römisch (2005) reports on the reduction of multivariate scenario trees, in particular load/water inflow trees.

In all the above stochastic formulations, the spinning reserve requirement is neglected because it is assumed that taking into account multiple scenarios when planning provides enough flexibility. Ruiz et al. (2008) nevertheless incorporate spinning reserve requirements in a two-stage formulation where the first stage contains the commitment decisions and the second stage the economic dispatch to meet specific scenarios load (thus close to the model of Schultz and Carøe (1998)). The rationale is that the (few) scenarios that may be included in a tractable stochastic formulation cannot capture the whole spectrum of uncertainty. Furthermore they claim that there should be a balance between the reserve requirements enforced and the scenarios included in the stochastic problem, and the reserve requirements may be scenario dependent. An interesting evaluation procedure is used to compare unit commitment decisions, i.e. the on/off part of the generation schedule. One deterministic and two stochastic unit commitments are computed – the stochastic ones differ

only in the reserve requirements associated to the scenarios – and only unit outages are considered. Then the economic dispatch problem is solved for a large number of simulation scenarios, distinct of the scenarios used to compute the unit commitments. Finally the different first stages are compared on the basis of their expected cost and energy slack on the simulation scenarios for several reserve requirement levels. The stochastic unit commitment where no reserve is required for scenarios containing an outage turns out to be a good compromise. It is interesting to note that a small number of scenarios are used and that they do not use any decomposition technique to solve the problem for the thermal based generation pool considered.

2.4 Motivation of a simulation based supervised learning approach

Real instances of the short-term scheduling problem are typically large-scale. For example, the real system of which the test system of Section 4.1 is a subset contains more than one hundred heterogeneous generation units and ten to twenty recourse stages. By nature, the electricity generation scheduling problem is very complex because of the presence of discrete variables in addition to continuous variables. Thus currently the formulations discussed in Section 2.3.4, which use scenario-based stochastic programming, are intractable from a computational point of view for the kind of system we consider in a day-ahead or intra-day context. Furthermore they do not provide explicit recourse policies. Currently, empirically designed good practice rules are thus used to propose adjustments for the very short term management of the generation pool (Renaud, 1993).

In this work we study an alternative solution to scenario-based stochastic programming. We compute the optimal schedules individually for a set of disturbance scenarios to build a dataset of scenario-adjustment pairs, and then apply supervised learning to this dataset to learn an explicit mapping representing the recourse strategy. The interests of such a formulation are manifold.

1. Instead of providing schedules specialized for particular scenarios, this approach provides a policy which generalizes these specialized schedules and outputs tuned decisions for scenarios which were not used to construct the policy.

2. This allows to validate the recourse policy before actually using it, thanks to some independent scenarios held out of the learning dataset.

3. We may use the regularity properties of SL algorithms to compute a recourse strategy that generalizes well the information contained in the scenario-schedule pairs of the dataset, and which is robust to small variation of the parameters, i.e. in our case of the uncertainty (cf. Examples 6 and 7). The predicted decisions are based not only on the solution of one scenario, but are also sensitive to the solution of scenarios in a neighborhood. Neighborhoods are automatically induced from data by

the learning algorithm, depending on the value of some meta-parameters which can be optimized during the validation (cf. item 2).

4. SL algorithms may be able to filter the schedules associated to scenarios for which it was not possible to compute a near-optimal solution, assuming that such scenarios do not represent a large proportion of the dataset, and thus to increase the performance of the optimization algorithm on these particular scenarios.

5. Heavy computations would be done offline and a relatively long time before the exploitation of the policy, using standard optimization techniques and custom scenario generation procedures. The time required to accomplish the whole procedure may be easily forecasted, thus the available computation time could be fully consumed.

6. Consequently the recourse strategy is applicable in quasi real-time if the inference process is kept simple and if its decisions can be made compliant with operation constraints thanks to a post-processing stage.

7. Some SL techniques algorithms provide side information, such as the degree of influence of some parameters on the decisions, the weight associated to the learning samples in the learned model, the degree of confidence on the decisions, or simply an interpretable explanation of the decision as a set of if-then rules.

8. There exists some flexibility on the formulation of the SL problem: first on the description of the input features, secondly on the choice of the output variables to predict. The latter option allows to tune the extent to which we believe the method is useful by selecting the set of variables that one wants to predict.

9. This procedure is to a large extent automatic (automatic learning and automatic validation) and can handle the dimensions of the real problem.

10. Finally, simulations could be replaced by or complemented with the schedules computed for scenarios which occurred in the past, allowing the exploitation of historical data.

These aspects will be exemplified in the next chapters. The proposed approach is exposed in greater details in Chapter 3, and validated in Chapter 4 on a representative benchmark problem.

Chapter 3

Supervised learning of recourse policies for intra-day generation rescheduling

In this chapter we expose our vision of the connections between the problems of day-ahead and intra-day generation (re)scheduling. We propose a way to combine simulation, optimization and SL in order to compute recourse strategies for the intra-day rescheduling problem.

3.1 Overview of the proposed approach

As already suggested in the previous chapters, in the short-term generation scheduling problem of a generation company, there are two main sub-problems that need improved solution methods.

1. Given uncertainties about the next day conditions, compute offline an "ideal" reference schedule for the next day given the information available the day before, so as to choose market positions and then announce the resulting schedules to the TSOs that are involved. This problem may actually lead to several iterations, if one or the other of the involved TSOs is unable to plan secure operation of his area with the schedules submitted by the different generation companies bidding in his area;

2. Define "recourse" strategies that will be used on the next day by the operators of the generation company in order to exploit incoming information about unexpected events gathered during operation, so as to reschedule the generation system with minimal cost, given the schedules and opportunities of recourse that the generation company has negotiated. For a particular recourse opportunity, this problem depends on the decisions committed at the previous steps, including both the day ahead "ideal" schedules negotiated with the TSOs he is interacting with and the decision policies settled and used for previous intra-day "recourse" opportunities.

These two problems are strongly intertwined, because what is optimal to announce the day before depends on how well one will be able to react the next

day and vice versa, and because the optimal reactions during the next day depend on what has been announced the day before.

In this chapter, we approach these two problems by proposing an approach that may be used in order to compute, the day ahead, strategies for rescheduling operation based on the incoming information flow, once the reference schedule has been decided. We will thus assume that the problem 1 of defining an ideal reference schedule is already solved, and exploit this reference schedule to simulate possible unexpected events and pre-compute ways to react to them. As concerns problem 2, given that in practice it is not possible to analyze in advance all possible unexpected events that could happen the next day, we rely on sampling "disturbance" scenarios in a finite number to represent this uncertainty. For each possible unexpected event (or sampled disturbance scenario), it is also not possible to exactly compute how to react to it in an optimal way, and we will only be able to rather roughly determine in advance what would be the optimal recourse decisions for coping with it, by formulating tractable approximate optimization problems to model these decision stages.

With these limitations in mind, we propose to apply a combination of Monte-Carlo simulation, optimization, and supervised learning methods in order to do the following (see Figure 3.1 for an overview of the different steps):

- the day head, and given a reference scenario/schedule, sample a finite relevant set of disturbance scenarios for the next day (step 1) and obtain near-optimal recourse decisions coping with each one of them (step 2);
- use supervised learning methods to (step 3):
  - extrapolate generic intra-day decision rules (recourse strategies) from the finite number of simulations carried out in steps 1 and 2, based on the reference scenario and reference schedule;
  - filter out in this way inconsistencies and sub-optimalities from this available finite sample of simulations;
- combine the learned decision rules with fast constraint satisfaction procedures (post-processing) in order to provide recourse strategies which are feasible and still fast to validate by Monte-Carlo simulation (step 4) and easy to implement in online exploitation;
- assess which parameters describing the uncertainty set are most important for formulating intra-day recourse strategies.

Anticipating on the subsequent discussions, let us mention the following stakes of the proposed approach:

- intra-day decisions are subject to constraints, which may not be ensured in general by the predictors computed by machine learning; hence a post-processing step is required to make the learned policies of practical use;
- to model correctly the context of application of a recourse strategy, it is necessary to model the reaction of the system during the time period preceding the application of the recourse decisions; it turns out that this

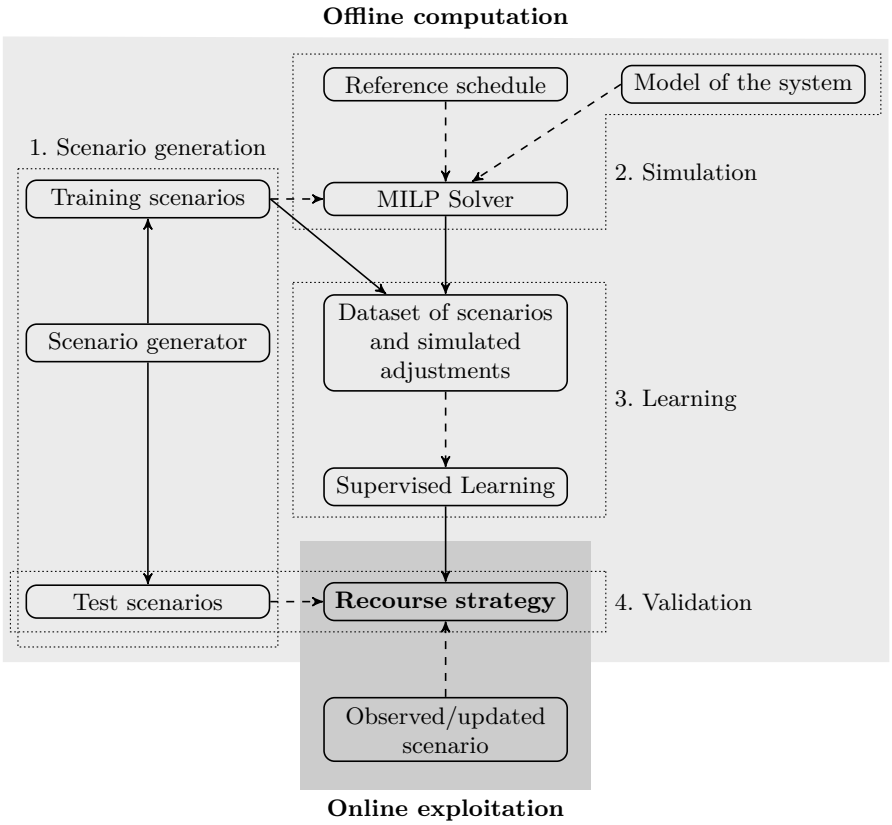


Figure 3.1: Schematic overview of the proposed supervised learning-based approach for building intra-daily recourse strategies. Continuous arrows illustrate the creation of the content of the destination node from the origin node. Dashed arrows illustrate the utilization of the content of the origin node by the destination node.

problem is rather difficult since it involves information about the whole network which is normally not available the day ahead; we will expose how we treat this problem in this thesis;

- to compute a recourse decision strategy for a given recourse instant, it would be necessary to already know how the subsequent recourse opportunities are going to be exploited, and vice versa, for computing these latter it is also useful to know how previous decisions will be taken.

Our approach consists in developing a method to design recourse strategies for a given recourse instant assuming that previous and posterior decision making schemes are already established. However, our recourse strategies are designed to be efficiently implementable given any simulated scenario; hence there is a possibility of iterating in order to adapt anterior and posterior decision stages

by taking into account results produced by our approach. These latter opportunities are not evaluated in this thesis but will be discussed in Chapter 5.

Finally, Figure 3.1 illustrates the steps which can be performed offline (light gray area), i.e. at least one day ahead, and the steps which are performed online (dark gray area), i.e. at each recourse stage.

3.2 Generation of perturbed scenarios

We need a way to generate a set of demand and availability scenarios representing the range of disturbances of the forecast that we want to cover with our recourse strategies. In practice, there are essentially two approaches to gather a set of such scenarios, namely their collection from actual operation of the system or the use of a Monte-Carlo simulation approach exploiting a probabilistic model of possible deviations from forecasts. Within our proposed methodology we can use either of these two approaches, or even a combination of them, since the input to the next step is merely a set of time series representing the deviation of load from the forecast and the moments at which a particular generation unit becomes unavailable.

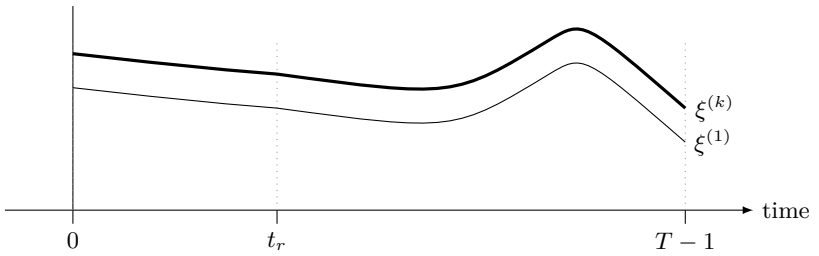
Consider a set $\mathcal{D}$ of demand patterns, and a set $\mathcal{O}$ of generation units which are supposed available along the complete optimization horizon in the reference schedule, but could become unavailable the next day. In our case study of Chapter 4 (cf. Section 4.2), we start from the reference scenario containing the day ahead demand forecast and the generation units availabilities used to compute the reference schedule. To obtain $\mathcal{D}$, we then generate different perturbed curves randomly around the reference demand curve by combining a probabilistic model of demand-forecast errors. To obtain $\mathcal{O}$, we consider all the thermal generation units that are started before the recourse opportunity.

Let Ξ be the set of scenarios $\xi^{(k)}$, $k = 1, \ldots, K$ made of one demand curve $d \in \mathcal{D}$ and of a unit outage of $\mathcal{O}$ imposed at a time in $\{0 \ldots t_r - 1\}$, where t_r is a fixed recourse instant belonging to $\{0 \ldots T - 1\}$.

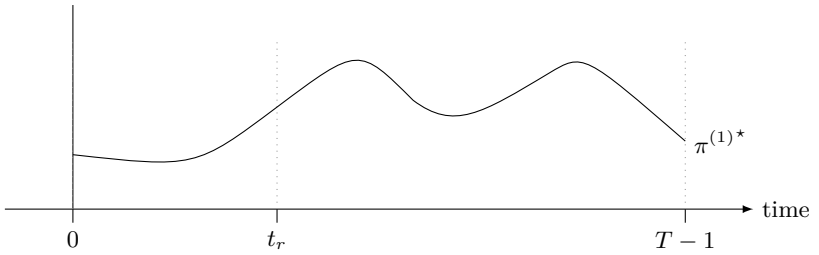
3.3 Computation of the adjustments to the perturbed scenarios

Let $\pi^{(1)\star} = (x^{\star}_{[0:T-1]}, u^{\star}_{[0:T-1]})$ be the optimal schedule associated to a scenario $\xi^{(1)}$ (cf. Figure 3.2) that we consider as the reference, i.e. the scenario used one day ahead for solving problem 1 (cf. Section 3.1). $\pi^{(1)\star}$ is thus the reference schedule. We want to compute an optimal recourse strategy $\pi^{*}_{t_r}(\xi_{[0:t_r-1]})$ for a single a priori fixed recourse time t_r. Thus, we want to know the modifications to make to the day-ahead schedule of all the generation units from time t_r to time $T-1$ once that the real behavior of the system between time 0 and time $t_r - 1$ is observed (e.g. $\xi^{(k)}_{[0:t_r-1]}$) and that, possibly, an updated load forecast is available[1].

[1]Note that there is a slight change of convention from here until the end of Part I, as well as in Appendix B. We consider that the first step of the generation schedule for the next



(a) Two scenarios of Ξ. $\xi^{(1)}$ is the scenario used to compute the reference schedule.



(b) The reference generation schedule $\pi^{(1)\star}$.

Figure 3.2: Illustration of the reference scenario and schedule, and of the perturbed scenario $\xi^{(k)}$.

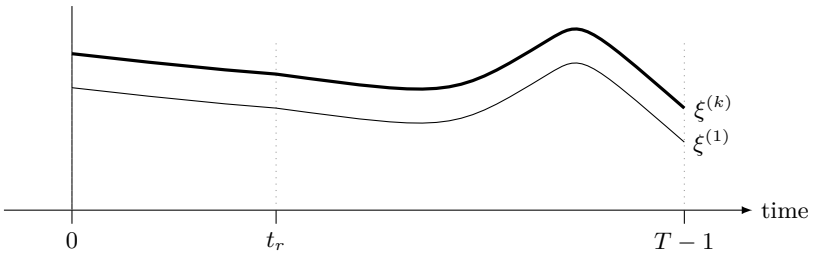
3.3.1 Simulation of the ancillary services

We suppose that unforecasted events occurring in the interval $\{0 \dots t_r - 1\}$ are compensated by the ancillary services which act in continuous time. Thus the automatic regulation processes associated to the ancillary services (cf. Section 2.2.1) modify the schedule planned between two recourse opportunities, and thus also between the beginning of the implementation of the day ahead schedule and the first opportunity of recourse. We denote by $\pi^{(1,k)\dagger}$ the generation schedule corresponding to the reference schedule automatically adjusted thanks to the ancillary services if scenario $\xi^{(k)}$ happen instead of $\xi^{(1)}$ (cf. Figure 3.3). The automatically adjusted schedule, $\pi^{(1,k)\dagger}$, must comply with the margins of ancillary services allocated in the reference schedule $\pi^{(1)\star}$.

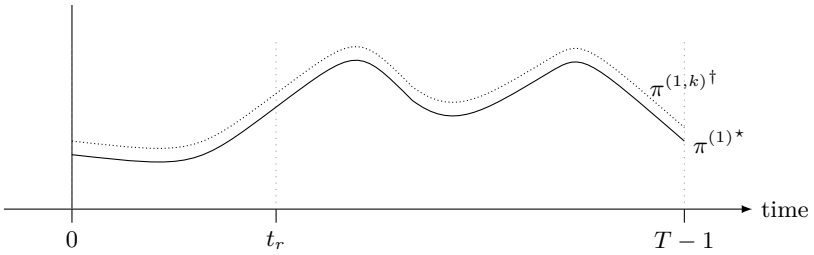
3.3.2 Re-optimization of the day-ahead schedule

The reference schedule $\pi^{(1)\star}$ optimally adjusted to satisfy the scenario $\xi^{(k)} \in \Xi$ on $\{t_r \dots T-1\}$ is denoted $\pi^{(1,k)\star}$. To compute $\pi^{(1,k)\star}$ under real system conditions, the reference schedule automatically adjusted thanks to the ancillary services, $\pi^{(1,k)\dagger}$, must be imposed on $\{0 \dots t_r - 1\}$. In Figure 3.4b, the dashed curve represents the schedule $\pi^{(k)\star}$, which is optimal with respect to scenario

day is time 0 and that the schedule ends at time $T-1$. It thus spans T time steps, and the recourse occurs at time t_r, based on the information gathered up to $t_r - 1$.



(a) Two scenarios of Ξ. $\xi^{(1)}$ is the scenario used to compute the reference schedule.

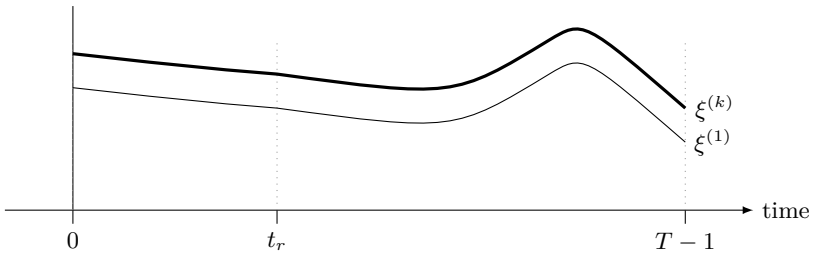


(b) Automatic adjustment of the reference generation schedule $\pi^{(1)\star}$.

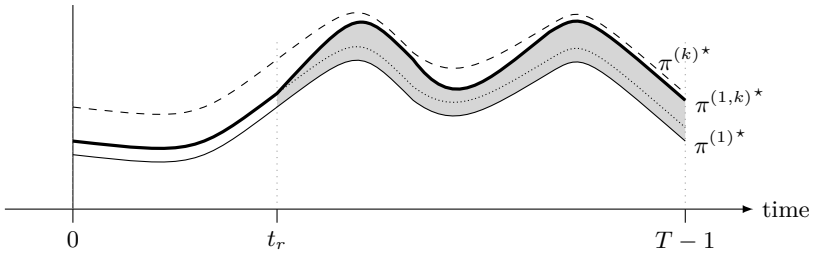
Figure 3.3: Effect of the ancillary services.

$\xi^{(k)}$ for the entire optimization period $\{0 \dots T-1\}$, while $\pi^{(1,k)\star}$ is thus optimal for scenario $\xi^{(k)}$ on $\{t_r \dots T-1\}$ conditionally on the assumption on its behavior on $\{0 \dots t_r - 1\}$. Because schedule $\pi^{(1,k)\star}$ is constrained to stay closer to $\pi^{(1)\star}$ on $\{0 \dots t_r - 1\}$, $\pi^{(1,k)\star}$ and $\pi^{(k)\star}$ are quite likely to differ on $\{t_r \dots T-1\}$.

The difference $\pi^{(1,k)\star} - \pi^{(1)\star}$ (Figure 3.4b, gray shaded area) actually represents the difference between the open loop schedule computed the day ahead from the forecasts and the optimally adjusted schedule if at time t_r perfect information became available about the realization of the scenario for the whole period $\{0 \dots T-1\}$. Notice that in real-time operation the information available at time t_r to take the recourse decision is not so strong; while it may perfectly describe the realization up to t_r it will in general only reduce, but not totally remove, the uncertainty about the future realization of the process during $\{t_r \dots T-1\}$. Therefore, the re-optimization of the perturbed scenarios provides a dataset of generation adjustments which are optimistically biased because they assume perfect information about the behavior of the system at time steps subsequent to t_r. This over-fitting of the sample of perturbed scenarios can however partially be countered by the application of supervised learning at the next step of the proposed approach which can enforce the projection of this information on a set of non-anticipative decision strategies which are only function of the information actually available at time t_r.



(a) Two scenarios of Ξ. $\xi^{(1)}$ is the scenario used to compute the reference schedule.



(b) Re-optimization of the automatically adjusted reference generation schedule $\pi^{(1,k)\dagger}$ on $\{t_r \dots T-1\}$.

Figure 3.4: Obtaining quasi-optimal adjustment to $\xi^{(k)}$ on $\{t_r \dots T-1\}$.

3.3.3 Limitation on the number of adjustments per recourse

As exposed in Section 2.3, the number of adjustments that can be implemented at each recourse opportunity may be limited for practical and regulatory reasons. This constitutes an additional requirement for the re-optimization phase. From the schedule computed one day ahead, we can only switch to a relatively close schedule when a recourse opportunity occurs: the number of thermal units and hydroelectric valleys on which we can act for all the recourse period is limited to an integer K smaller than the total number of units $(N_t + N_v)$ allowed to be adjusted. The problem is thus to select the best subset of units and then to re-optimize their generation schedule to cover the new demand forecast. Selecting K units to adjust beforehand, i.e. independently of the optimization of the generation levels, obviously leads to a suboptimal solution.

3.3.4 Remarks

To obtain quasi-optimal adjustments $\pi^{(1,k)\star}$ to the reference schedule $\pi^{(1)\star}$ for a scenario $\xi^{(k)}$ on the period $\{t_r \dots T-1\}$, we must account for the ancillary services that automatically adapt $\pi^{(1)\star}$ to scenario $\xi^{(k)}$ on the period $\{0 \dots t_r - 1\}$. Some additional constraints may apply to the re-optimization on the period $\{t_r \dots T-1\}$ to limit the number of adjusted units.

This re-optimization procedure can be repeated for each scenario of Ξ. Practically, it can be performed with the same optimization algorithm as the one

used to compute the reference schedule by applying a few modifications to Formulation 5 (cf. Appendix B.2). As exposed in Section 2.2.1, the ancillary services affect the power generated by the units of the system according to deviations of the synchronous network frequency and of the planned exchanges between areas, which are themselves influenced by events occurring in and outside the generation system under consideration (load variations, ...). In real-time exploitation the local modifications to the planned schedule caused by the regulators implementing the ancillary services can be measured and centralized to provide an appropriate estimation of the state of the generation system before taking some adjustment decisions. However, during an offline a priori study, the information available is not sufficient to evaluate the next day use of the ancillary services, since the information contained in scenarios is restricted to the area of the generation company. Without a model and scenarios encompassing the whole network it is thus not possible to reconstruct the actual realization of the ancillary services.

We expose in Appendix B.2 the procedure that we implemented to tackle this problem for the experiments of Chapter 4.

Appendix B.2 also exposes how the subset selection problem can be incorporated in the re-optimization procedure.

3.4 Supervised learning application

We want to exploit the datasets generated by the approaches explained in Sections 3.2 and 3.3 to obtain an approximation $\pi_{t_r}(\xi_{[0:t_r-1]})$ of an optimal recourse strategy $\pi^{\star}_{t_r}(\xi_{[0:t_r-1]})$ mapping the information about the realization of the uncertain processes available at time t_r to the optimal generation adjustments at subsequent time steps. We propose to formulate a supervised learning problem in order to compute an approximation of this mapping. We have a sample of K realizations of the uncertain process ξ, $\left\{\xi^{(1)}, \ldots, \xi^{(K)}\right\}$ and the corresponding set of $K-1$ optimal adjustments of the reference schedule $\pi^{(1)\star}$, $\left\{\pi^{(1,2)\star}, \ldots, \pi^{(1,K)\star}\right\}$ (cf. Figure 3.5). Schematically, we want to learn a mapping from the gray shaded area of the top graph of Figure 3.5 to the gray shaded area of the bottom graph of this figure. For example, a direct formulation of this SL problem consists in setting:

- as output space $\mathcal{Y}$ the space of multivariate time series representing the evolution of the generated power of all the generation units for $t \geq t_r$,
- as input space $\mathcal{X}$ the space representing the state of the generation system at time t_r and the information about the realization of the load and generation availability scenario collected until t_r (which are also time series).

The overall description of the methodology used to learn a recourse strategy is summarized in Table 3.1. One can in principle apply any available supervised learning algorithm to this problem.

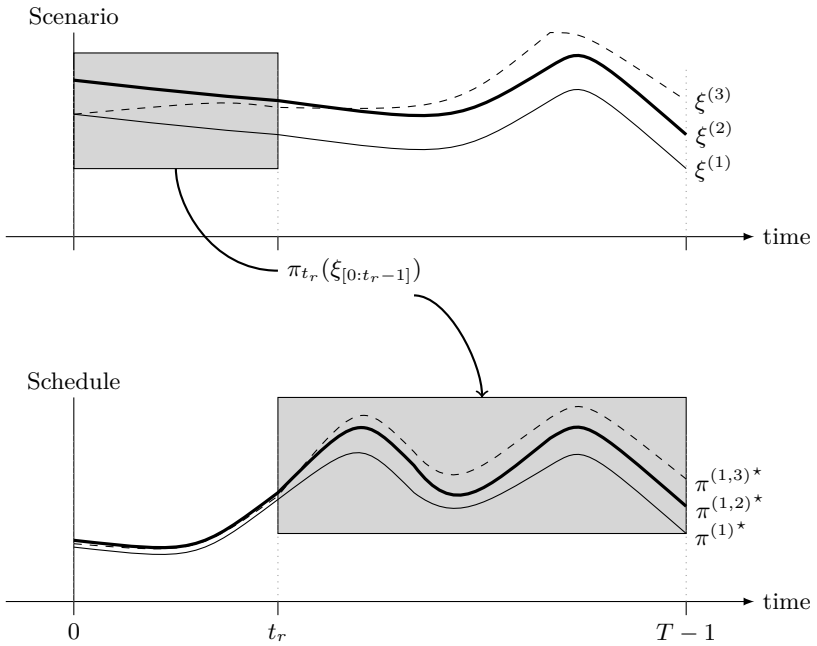


Figure 3.5: Illustration with $K = 3$ of the supervised learning application to compute an approximation of $\pi^{\star}_{t_r}(\xi_{[0:t_r-1]})$.

Table 3.1: Computation of a recourse strategy.

Input:
an optimal schedule $\pi^{(1)\star}$ associated to a reference demand scenario $d^{(1)} \in D$.
Output:
a recourse strategy $\pi_{t_r}(\xi_{[0:t_r-1]})$ for $\pi^{(1)\star}$ and for the recourse time t_r.

1. Let Ξ be the set of scenarios made of one demand of D and of a unit outage event of $\mathcal{O}$,
2. compute the automatic adjustment $\pi^{(1,k)\dagger}$ of $\pi^{(1)\star}$ from 0 to $t_r - 1$ to each scenario $\xi^{(k)} \in \Xi$,
3. compute a schedule $\pi^{(1,k)\star}$ adjusted from t_r to $T - 1$ to each scenario $\xi^{(k)} \in \Xi$ by imposing the schedule $\pi^{(1,k)\dagger}$ on $\{0 \dots t_r - 1\}$,
4. solve a supervised learning problem in order to derive an approximation $\pi_{t_r}(\xi_{[0:t_r-1]})$ of the optimal recourse strategy for time t_r, $\xi_{[0:t_r-1]}$ being the state information available at time t_r.

3.4.1 Choice of the input space $\mathcal{X}$

There exists some flexibility on the inputs of the SL formulation. We can define the input space so as to enforce the learned strategy to exploit exclusively the observation of the realization of ξ up to $t_r - 1$. The learned recourse strategy will tend to predict adjustments which are good on average for multiple possible futures given the observed realization on $\{0 \dots t_r - 1\}$. Alternatively the learned strategy may also exploit an updated forecast of the demand on $\{t_r \dots T - 1\}$. However, since the coupling constraints are only implicitly present in the learning sample and not explicitly imposed during the learning of the recourse strategy, there is no guarantee to obtain schedules with a cumulative generation curve close to the updated demand forecast. It is not possible to conclude a priori which formulation is the best, because it is related to the quality of the updated demand forecast and to the possible deviations of the load from the latter. This must thus be kept in mind when validating (cf. Section 3.5.2 and Section 5.6).

3.4.2 Choice of the output space $\mathcal{Y}$ and induced post-processing

One interesting aspect of a SL-based approach to compute the recourse strategy is that we can a priori choose which part(s) of the available information on the adjusted generation schedules we actually want to predict, and define the output space $\mathcal{Y}$ consequently. For example, instead of predicting the precise generation level of the units we can predict a cruder approximation of the generation schedule, e.g. only the on-off state of the thermal generation units and the number of turbines or pumps that are activated in each hydroelectric plant. The projection on a precise updated forecast may then be performed during a post-processing phase, as we do in Section 4.5.

Depending on the choice of $\mathcal{Y}$, a post-processing is required for one or more of the following reasons.

1. We must obtain detailed generation schedules for all the generation units.

2. We must ensure that the predictions are feasible with respect to the individual dynamic and operating constraints of the generation units: for example, the generation interval of a unit is non-convex (it is either 0 MW, or between P^{min} and P^{max}), a unit must stay on for a certain amount of time once it has been started, etc. There is no guarantee that a recourse policy computed with a SL algorithm generates adjustments which satisfy the operating constraints of the generation units, since these constraints are only implicitly present in the learning sample. However, the feasibility of the adjustments in this sense is a strict requirement. Thanks to a post-processing phase the predicted adjustments can be projected onto the feasibility domain delimited by the operating constraints of the thermal and hydroelectric generation units.

3. We must ensure the load-generation balance with respect to a particular load forecast.

Furthermore, depending on the structure of $\mathcal{Y}$ and on the choice of the learning algorithm, a decomposition or a reformulation of the problem may be necessary. In the next subsections we analyze these aspects for a few choices of $\mathcal{Y}$.

3.4.2.1 Predicting the subset of generation units that should be adjusted

One of the simplest choices for $\mathcal{Y}$ is, following the discussion of Section 3.3.3, to predict the optimal subset of K generation units that should be adjusted. Indeed from the simulations of Section 3.3 we can define an adjustment indicator Y_i for each unit i, that is equal to one if the corresponding unit has been adjusted with respect to the reference schedule, and equal to zero otherwise. As output space we have thus

$$\mathcal{Y} = \{0,1\}^{N_t+N_v}.$$

Even though the adjusted schedules from which we evaluate the indicators Y_i comply with the limitation on the number of adjustable units (K), the learned recourse strategy may produce predictions that are not compliant with the latter limitation and thus require a post-processing stage to enforce it.

Then, to obtain generation schedules for all the generation units of the system, one must have an updated forecast of the demand curve and solve a smaller instance of Formulation 5 containing at most K adjustable generation units (e.g. typically, containing $K = 30$ adjustable units in a system made of $N_t + N_v = 150$ units). This setting is experimented in Section 4.5.

3.4.2.2 Predicting start-up and shut-down decisions

Predicting start-up and shut-down decisions is equivalent to predicting the On/Off status of the generation units and the number of turbines or pumps activated in each hydroelectric plant, i.e. the value of the integer variables of the problem. These predictions cannot be used directly to obtain the detailed adjustments of the generation schedules because

1. first, one must ensure that the sequences of integer decisions predicted are feasible (since SL algorithms do not guarantee this per se),

2. then, one has to compute the power generation levels (continuous variables).

If these two problems can be solved sequentially, then one could imagine to solve problem 1 for each unit independently, and to solve problem 2 globally, i.e. for all the generation units together, to target an updated demand forecast.

These two types of post-processing may be solved sequentially for thermal generation units since for the integer variables one just needs to ensure that the delays between start-ups and shut-downs are sufficiently long, without having to consider the continuous variables. Once integer variables are fixed, continuous variables can be optimized.

However for hydroelectric units these two types of post-processing cannot be solved sequentially. Integer variables determine the range of water flows and

there is a bijective correspondence between the water flows and the power generation levels. Water flows determine the evolution of the volume of reservoirs. Since the allowed volume variation range is at some time steps very narrow (or even null), the activation or deactivation of some turbines or pumps may be necessary. Thus one must be able to act on both integer and continuous types of variables simultaneously.

In summary, predicting directly start-up or shut-down decisions seems valuable only for thermal generation units. In this setting, Formulation 5 could be solved with the integer variables of the thermal units set according to the post-processed predicted values while the hydroelectric sub-problems may contain both integer and continuous variables.

3.4.2.3 Predicting power generation levels

The evolution of the generation level of each generation unit over the recourse horizon is the most informative output that we can predict, since it also defines the On/Off status of the thermal units, the water flows and the ancillary services reserves. As mentioned in Cornélusse (2008), predicting the individual generation level of all the hydroelectric units of a valley is quite harder than for their thermal counterparts, because of the more chaotic structure of their schedule due to severe restrictions on the volume of reservoirs and to the couplings induced by the water flow networks (cf. Figure 2.2). We therefore propose to predict the total power that must be generated by a hydroelectric valley along the recourse horizon. Learning one regression model for all generation units corresponds to a multivariate real valued time series prediction problem.

In this setting, we can interpret the output of the learned strategy as a demand to satisfy for each thermal generation unit and for each valley. We thus obtain a unit by unit decomposition of the generation re-scheduling problem. However we have no guarantee that these demands are individually satisfiable when taking into account the operating constraints of the generation units. For example, using tree-based ensemble methods (Appendix A.2) to compute a regression model, the prediction for any input scenario is essentially a convex combination of some outputs present in the learning set; a SVR hypothesis (cf. Appendix A.3) predicts weighted sums of kernel evaluations based on the input features of different objects of the training dataset; etc. Instead of customizing the learning algorithm to obtain predictions that comply with the constraints of the generation scheduling problem, we decided to use classical learning algorithms and to post-process the prediction to ensure the feasibility of the adjustments (cf. Appendix B.3.1). Each post-processing sub-problem is nevertheless much smaller than the problem of computing the reference schedule, and we can thus hope that it fits in the time requirements of intra-day recourse management.

As we decompose the problem and do not enforce the coupling constraints a posteriori, if the adjusted generation schedule leads to a discrepancy between the total generation and the total load, it must be compensated by reserve energy purchase and is (strongly) penalized when evaluating the cost induced

by this strategy. The quality of the predicted adjustments in this respect must thus by experimentally analyzed (cf. Section 4.4).

3.4.3 Decomposition or reformulation of the learning problem

All the learning problems exposed in Section 3.4.2 exhibit a complex structure. We discuss below four formulations of the SL learning problems to deal with their structure when the latter contains both a spatial dimension (i.e. several units) and a time dimension.

Case 1. We solve the problem by taking as output space the space of multivariate time series (cf. Figure 3.6). Each output can be seen as a matrix where rows represent units and columns represent time steps of the recourse horizon.

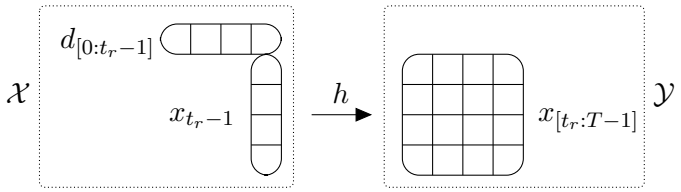


Figure 3.6: Illustration of decomposition of case 1. Learning a multivariate time series as a function of the state of the system and the observation of the demand until time $t_r - 1$.

Case 2. To decrease the complexity of the problem we decompose the learning problem unit by unit into univariate time series prediction problems (cf. Figure 3.7), i.e. by rows of the output space of Case 1. We thus have $N = N_t + N_v$ supervised learning problems and the recourse strategy $h = (h_1, \ldots, h_N)$ is the composition of N mappings. However we somewhat lose the coupling of the units even though the decisions on which each h_i is trained are globally optimal.

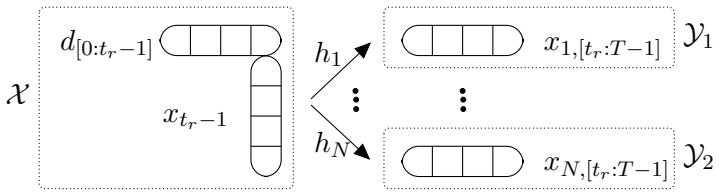


Figure 3.7: Illustration of the decomposition of case 2. Decomposition along generation units.

Case 3. Instead of decomposing unit by unit another possibility is to decompose Case 1 along the time dimension by explicitly appending the prediction time in the inputs of the learning problem, i.e. column by column

(cf. Figure 3.8). Learning from this formulation can take into account the correlation between outputs at consecutive time steps since, e.g. a tree-based model may contain splits on the time periods and thus explicitly partition the time domain. Indeed this is rather a reformulation than a decomposition since we keep a single learning problem. We somewhat break the dimensionality of the output space since in this setting the number of learning examples is multiplied by the number of time steps of the recourse horizon, $T - t_r$, and the dimension of the output space is consequently divided by $T - t_r$. We thus gain a $(T - t_r)^2$ factor on the ratio of the dimension of the output space over the number of learning examples with respect to Case 1.

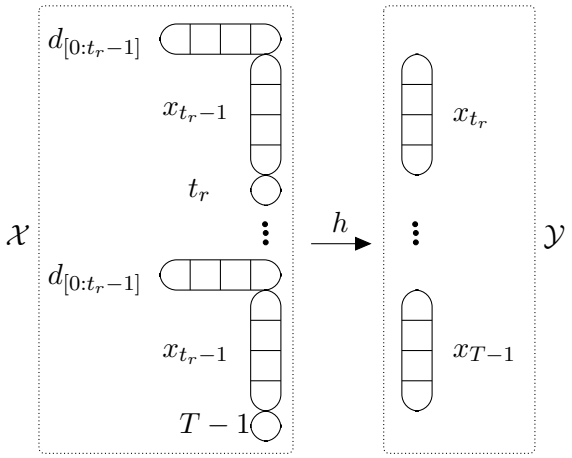


Figure 3.8: Illustration of the decomposition of case 3. Incorporation of the time variable in the input space.

Case 4. In order to apply some standard SL algorithms which operate with a scalar (i.e. one-dimensional) output space, we note that we can combine Case 2 and Case 3 to yield a scalar output learning problem for each generation unit.

An alternative to the above approaches is to identify a priori clusters of adjustment decisions for each generation unit and to turn the problem into a classification problem. If cluster centers are feasible, this alternative requires no post-processing stage. However, the a priori identification procedure may be cumbersome and the definition of the optimal number of clusters hard to define.

3.5 Exploitation and validation

3.5.1 Online exploitation

The most important requirement is to obtain recourse decisions that satisfy the operating constraints of the generation units. To ensure the satisfaction of

real-time feasibility constraints of a given unit, we impose them a posteriori, at the moment where the recourse action is applied. First we compute a recourse based on the information gathered in $\xi_{[0:t_r-1]}$ and the decision rules built by supervised learning. Then we modify these recourses and impose constraints unit by unit.

3.5.2 Offline validation of a recourse strategy

In order to validate the recourse strategies computed by supervised learning, we use an independent set of scenarios in the following way. First, each scenario is solved optimally, in the same fashion as we computed the recourse decisions for the learning sample. This yields for each validation scenario a generation schedule that minimizes the costs of operation under the hypothesis of perfect information. Then, for each scenario the recourses are computed by using the learned decision rules, by post-processing them, and by computing the overall induced operating costs, including the penalization of the possible violation of the coupling constraints. Finally, by comparing the resulting costs, we may measure the distance between the inferred strategies under different conditions (e.g. using different settings of the learning method, different sets of input features, or different sizes of learning samples) and assess them also with respect to the (admittedly unreachable) ideal strategy based on perfect information, or any other candidate strategy.

Chapter 4

Experiments

We begin with a description of the generation system (Section 4.1) used throughout this chapter and provide some details about the scenario generation procedure (Section 4.2). Then we analyze under different settings the computation of quasi-optimal adjustments to the reference schedule for the set of perturbed scenarios considered (Section 4.3), the learning of adjustment strategies, their post-processing to obtain some implementable decisions and the overall performance of the proposed approach (Sections 4.4 and 4.5). These settings differ by the information assumed available when making the recourse decisions and by the type of decisions one wants to infer.

4.1 Test system

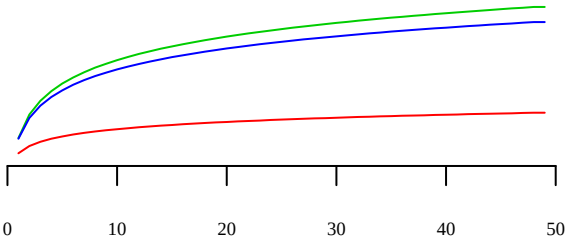
4.1.1 Composition of the generation system

The test system we use in our experiments is composed of 16 thermal units of different capacities, generation costs and technical characteristics (Figure 4.1), and of 6 hydroelectric valleys (Figure 4.2) (this is twice the size of the generation system considered in Cornélusse et al. (2009b)). It is composed of elements of the generation system of EDF, which contains about 111 nuclear and thermal units and 38 Valleys. The maximal peak load that we consider in our experiments is about 15 GW while the maximal peak load for France is about 90 GW (cf. Section 4.2.1). Roughly there is thus a 1 to 6 ratio between our test system and the true EDF system. On the other hand our test system is comparable to the generation system of Electrabel in Belgium in terms of generated megawatts, but contains proportionally more hydroelectric capacity. The values of the parameters are adapted from real data provided by EDF.

The reference load curve is shown in Figure 4.3. The ancillary services requirements are approximately constant all along the optimization period (130 MW for the primary reserve, 100 MW for the secondary reserve). The simulations follow EDF's industrial practice: to compute the schedule for the next day the optimization horizon is divided into periods of 30 minutes and contains two days, although the procedure is repeated every day, to obtain scheduling decisions which drive the system into an acceptable state for repeating the

Unit	Type	P^{min} [MW]	P^{max} [MW]	Fixed cost	Proportional cost
$ARRI5T_1$	Gas	77	128	very high	very high
$BLENOT_4$	Coal	60	220	medium	medium
$CORD5T_4$	Coal	210	510	high	medium
$PORC2T_3$	Fuel oil	175	585	high	high
$GRAV5T_2$	Nuclear	210	903	very low	very low
$SSEA2T_1$	Nuclear	180	910	very low	very low
$CATTET_1$	Nuclear	310	1303	very low	very low
$FLAMAT_2$	Nuclear	260	1330	very low	low

(a) Technical characteristics.



(b) Start-up costs against stopped time: $BLENOT_4$ red, $CORD5T_4$ green, $PORC2T_3$ blue. No start-up cost are incurred to other units.

Figure 4.1: Description of thermal generation units. Two units of each type are included in the model, with different fixed and proportional costs.

procedure the next day.

4.1.2 Example of generation schedule

After an automatic pre-processing[1] the optimization problem contains about 39,000 variables, one half of them being binary, and about 56,000 constraints. The optimization problems are solved using the branch-and-cut algorithm implemented in CPLEX (ILOG, 2007) (cf. Appendix C.1.2).

If we want to solve the problem to optimality, a solution is found after 752 seconds after having explored more than 11,000 nodes of the branch-and-bound tree. When solving the problem with a tolerated integrality gap of 1% and a maximum allowed computation time of 600 seconds, a solution is found after 51 seconds and only ten nodes explored thanks to a heuristic which quickly finds feasible solutions and tightens the optimality gap. In this case the estimated achieved integrality gap is about 0.8%. The actual gap between the provably optimal solution and the approximated solution is about 0.5%.

[1] A preliminary stage performed before starting the branch-and-cut to tighten the allowed range of variables and remove redundant constraints. This usually strengthens the LP relaxation of the problem and decreases its size.

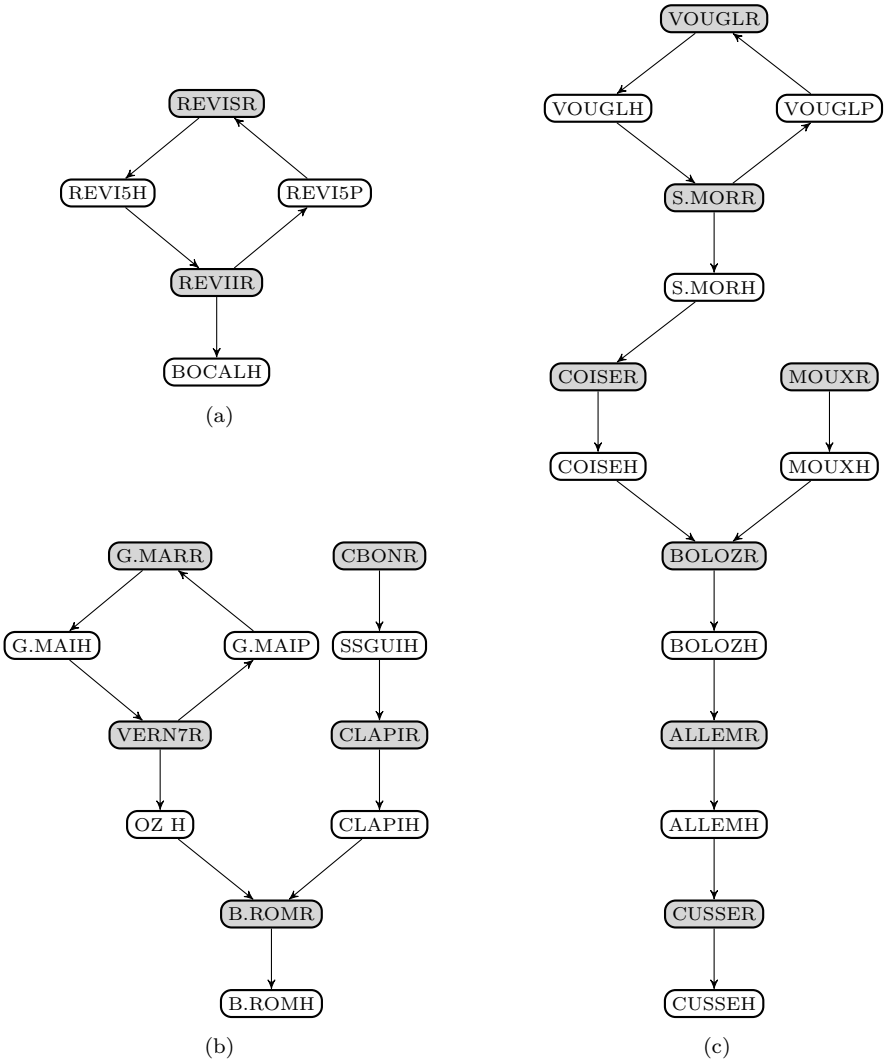


Figure 4.2: The three types of valleys used in the simulations. Two instances of each type are included in the model, with different water values assigned to the head reservoirs. Natural water flows and spills are not shown (cf. Figure 2.2).

The total power generated by hydroelectric units almost follows the variation of the load, while the total power generated by thermal units (Figure 4.3) is nearly piecewise constant along the optimization period and thus offsets the hydro generated power. The generation curves of some thermal units are depicted in Figure 4.4 (all nuclear power plants operate close to their maximal output and are thus not illustrated). The evolution of the volume of the reservoirs and of the power generated by the plants in one of the six valleys are depicted on Figure 4.5.

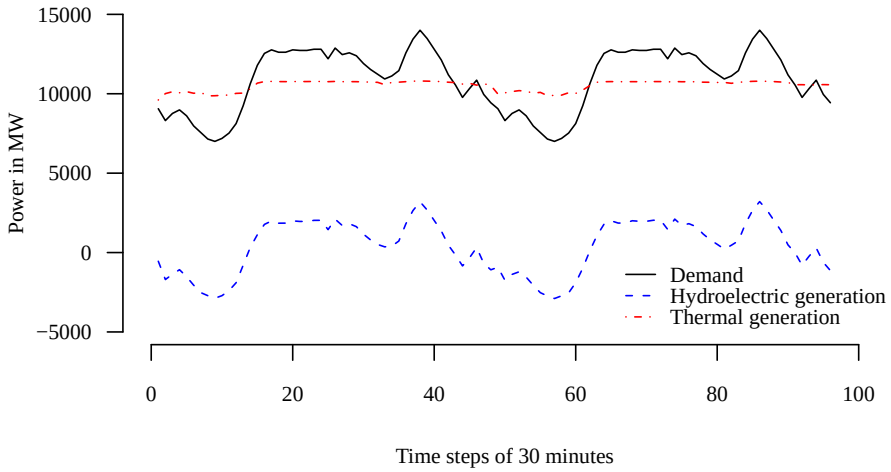


Figure 4.3: Two days demand forecast and corresponding cumulative thermal and hydroelectric generation in the reference schedule.

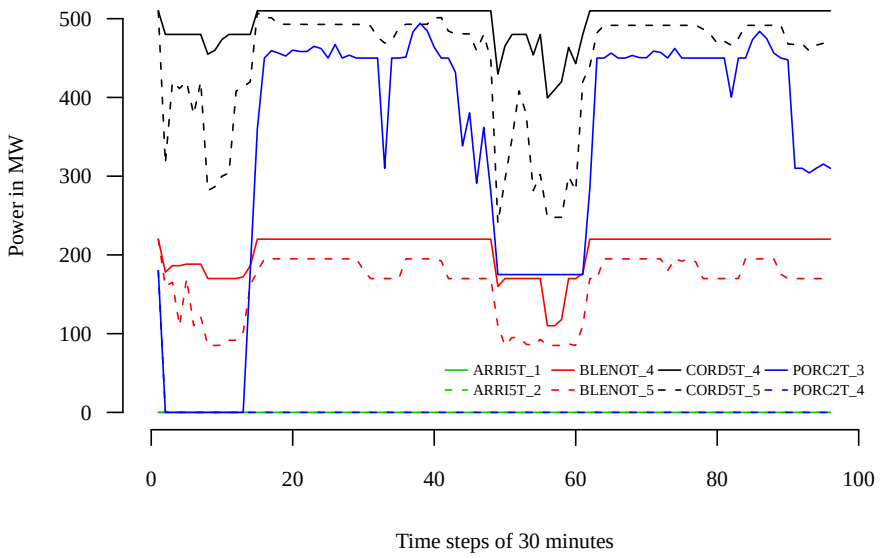
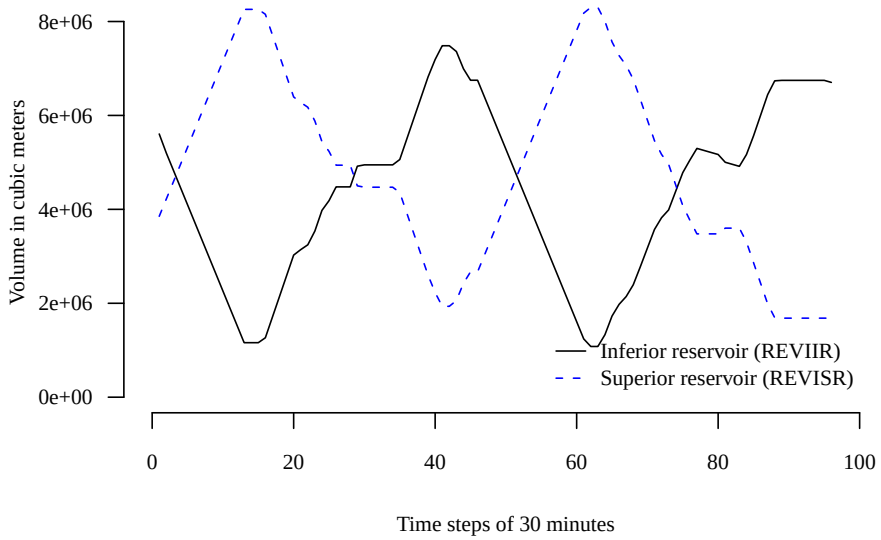
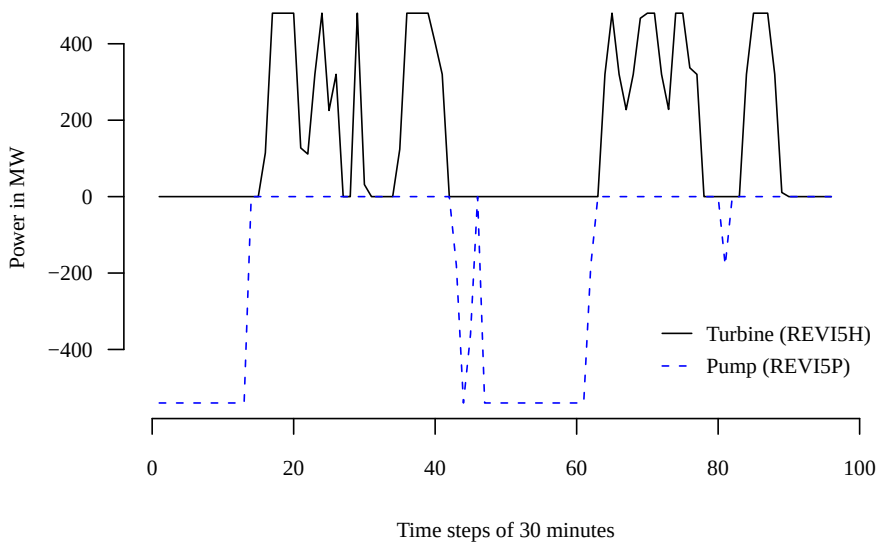


Figure 4.4: Evolution of the generation level of the classical thermal units (i.e. non-nuclear) in the reference schedule. Units of the same type are grouped by color. The two small gas units are not used, as well as one fuel oil unit ($PORC2T_4$). In the remaining groups, the less expensive units operate at their maximal level during high demand periods while the most expensive units provide reserves for ancillary services.



(a) Reservoirs.



(b) Plants.

Figure 4.5: Evolution of the volume of the reservoirs and of the generation levels of the plants in a valley of type (a) (cf. Figure 4.2) in the reference schedule. In this schedule the lower turbine (BOCALH) is never activated, and thus all along the optimization period the sum of the water stored in each reservoir is constant.

4.2 Generation of training and validation scenarios

In order to generate our dataset of scenarios for the next day, we have to make two choices:

1. Choose a reference scenario for the next 48 hours (96 time steps of 30 minutes).
2. Generate perturbed versions of the reference scenario representing possible deviations from the reference scenario, which should be covered by the recourse strategy.

In our setting a scenario encompasses a demand curve and a generation unit outage. In order to achieve the generation of perturbed demand curves in a realistic way, we have used historical demand forecast and realization curves from three years provided by EDF, properly scaled down to the size of our generation system, both in order to choose the reference scenario and to build a generative model of deviations from the forecasted demand curves. The exact procedure is explained in Section 4.2.1. We then describe and discuss the generation of joint demand-generation perturbation scenarios in Sections 4.2.2 and 4.2.3 respectively.

4.2.1 Generation of perturbed demand curves

In order to obtain a generative model of the demand forecasting errors, we proceeded in two steps exploiting the historical data: first we have pre-processed the historical forecasts in order to obtain a normalized and sorted set of demand patterns, then we have derived a model of the forecasting error by building an autoregressive model of the difference between the forecasted and realized demand. We eventually used the autoregressive model in order to generate a sample of fixed size (in our case 900) of demand deviations around a reference demand forecast over 96 time periods.

Normalizing and clustering the demand forecasts. The following procedure is inspired by Gröwe-Kuska and Römisch (2005). Our historical data is derived from a set of forecasted demand levels, $d_{i,t}$, where $i = 1, \dots, I = 1095$ represents a particular day, and $t = 1, \dots, T = 48$ represents a particular time step during a day. We first compute for each day the average consumption of that day, namely

$$\bar{d}_i = \frac{1}{T} \sum_{t=1}^{T} d_{i,t},$$

which we subtract from the original data by computing a new series of hourly deviations from the average daily values by

$$\tilde{d}_{i,t} = d_{i,t} - \bar{d}_i,$$

which we gather in an $\mathbb{R}^{I \times T}$ data matrix D, whose lines represent daily demand deviations from the average of that day.

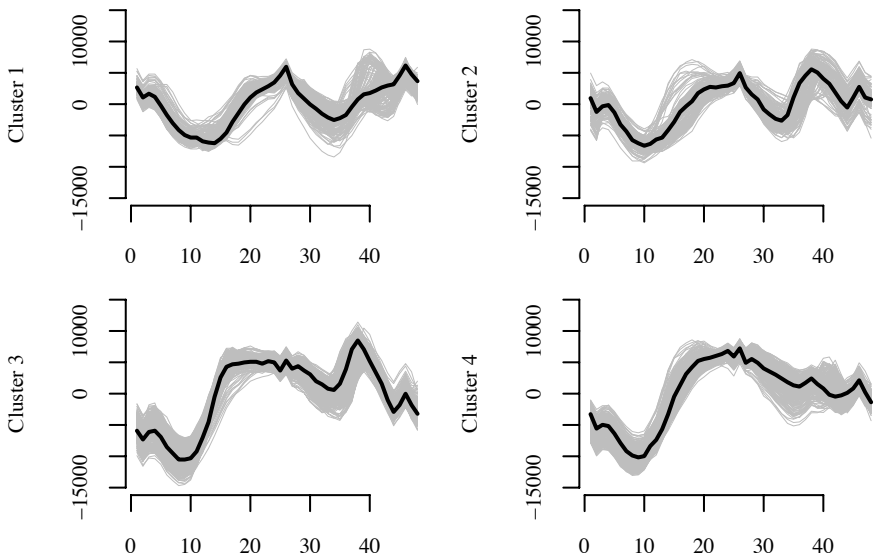


Figure 4.6: Clustering of the daily mean-corrected forecasts contained in D. In each cluster the thick black curve represents the medoid, and the grey curves are the remaining demands of the cluster.

The next step of the procedure consists in partitioning the lines of D into clusters of similar daily patterns. We used a variant of the $k-$medoids algorithm (Kaufman and Rousseeuw, 2005) to do this and chose a value of $k = 4$ by trial and error. As a matter of fact, the 4 clusters (Figure 4.6) of demand shapes correspond roughly to a partition into week-end (clusters 1 and 2) and week days (clusters 3 and 4) combined with a partition along the two daylight saving time periods (winter for clusters 1 and 4, summer for clusters 2 and 3). Each cluster $c_j, j = 1, \ldots, J = 4$ corresponds to a subset of days, and is characterized by its medoid denoted below by i_j^m and the average value of daily average consumptions in that cluster, namely

$$\bar{d}_{c_j} = \frac{1}{\#c_j} \sum_{i \in c_j} \bar{d}_i.$$

From this we get a very simple (and admittedly rough) forecasting method to obtain a reference demand:

- choose the cluster j corresponding to the next day
- predict its demand by $d_{i_j^m,t}$, $(t = 1, \ldots, T = 48)$
- if the prediction is carried out over the next two days, as in our simulations, it is completed by $d_{i_j^m+1,t}$, $(t = 1, \ldots, T = 48)$.

Computation of a demand forecasting error model. Assuming that the cluster is always chosen correctly, the forecasting errors of this procedure are composed of two parts, namely the deviation of the average daily load forecast from that of the realization and the deviation of the shape of the forecast from the shape of the realization. In the same way than for the demand forecast, we introduce the realized demand levels, or load, denoted by $l_{i,t}$, and define the daily average load as

$$\bar{l}_i = \frac{1}{T}\sum_{t=1}^{T} l_{i,t},$$

which we subtract from the original data by computing a new series of hourly deviations from the average daily values by

$$\tilde{l}_{i,t} = l_{i,t} - \bar{l}_i.$$

We can then define the prediction error for the daily average consumption $\bar{e}_i$ and the mean-corrected shape $\tilde{e}_{i,t}$ as

$$\bar{e}_i = \bar{d}_i - \bar{l}_i$$

and

$$\tilde{e}_{i,t} = \tilde{d}_{i,t} - \tilde{l}_{i,t}.$$

In order to build a generative model of these forecasting errors, we have proceeded as follows:

- In each cluster we assume a normal distribution of $\bar{e}_i$ of mean

  $$\mu_{\bar{e}_j} = \frac{1}{\#c_j}\sum_{i\in c_j} \bar{e}_i$$

  and standard deviation

  $$\sigma_{\bar{e}_j} = \sqrt{\frac{1}{\#c_j}\sum_{i\in c_j}(\bar{e}_i - \mu_{\bar{e}_j})^2}.$$

- To build a generative model of the intra-day deviations from the medoid, we build a second dataset E in the same way than D but containing the mean-corrected error shapes $\tilde{e}_{i,t}$, and then according to Le Goazigo and Collet (2006) fit an autoregressive (AR) model (cf. Brockwell and Davis (1996, Chapter 5)) of these forecasting errors. Assuming that X_t is the process of zero mean that we want to model and $\varepsilon_t \sim WN(0, \sigma_\epsilon^2)$, the AR model of order p writes

  $$X_t = \sum_{j=1}^{p} \phi_j X_{t-j} + \varepsilon_t.$$

  We have estimated the order p of the model and the values of the parameters $\phi_1, \ldots, \phi_p$ using the default Akaike Information Criterion (AIC) implemented in the R[2] function `ar`.

[2] `http://stat.ethz.ch/R-manual/R-devel/library/stats/html/ar.html`

Scaling the demand curve to our reduced generation system. The motivation for working with a reduced generation system is to reduce the size of the optimization problem instances that we need to solve repeatedly. However we are not constrained to divide the demand curve of the complete generation system by the factor characterizing the generation system reduction (i.e. about 6 as stated in Section 4.1). Thus we have proceeded as follows to establish the reference demand of the reduced generation system:

1. choose the peak demand d^{max} of the desired reference demand curve,
2. set its minimal level at $M\%$ of the peak demand,
3. compute the minimum $d^{min}_{c_j} = \min\{d_{i,t} : i \in c_j\}$ and the maximum $d^{max}_{c_j} = \max\{d_{i,t} : i \in c_j\}$ of the medoid of the cluster j.
4. Subtract $d^{min}_{c_j}$ from the medoid of the cluster j and divide it by $(d^{max}_{c_j} - d^{min}_{c_j})$ so that its range is in $[0; 1]$,
5. multiply the result by $(100 - M)\% \times d^{max}$ and add $M\% \times d^{max}$ to obtain the desired reference demand.

We have chosen $d^{max} = 14GW$ and $M = 50$, which yields proportionally to the real system a wider variation range since $\frac{d_{max}}{2} = 7GW$ while the range of the medoid is of $19GW$. With these values the range of the reference schedule and of the perturbed curves are sufficient to cause start-ups or shut-downs of thermal units. We have generated perturbations around the medoid before scaling and applied the same procedure than above (i.e. steps 4 and 5) to scale the perturbed curves.

4.2.2 Generation of the reference scenario, a sample of perturbations, and their adjusted generation schedule

Our reference scenario is directly obtained from the historical data, by choosing a typical sequence of two days, and assuming that all generation units are available. Specifically, we chose as reference scenario the historical data corresponding to the medoid of the third cluster, thus corresponding to a week day in the winter season, together with the day immediately following which is of the same type, in order to yield a 96 time steps scenario used in our optimization phase. The scenario is solved, yielding a reference generation schedule $\pi^{(1)\star}$ over the corresponding 96 time periods (NB: the reference scenario is actually the one used for illustration in the Section 4.1).

Next we choose a recourse stage t_r, a total number N of scenarios to generate and the proportion p_o of them for which we impose the outage of a generation unit outage prior to t_r. In our experiments reported in the rest of this chapter, we set $N = 900$, $t_r = 12$ (i.e. 6AM) and $p_o = 50\%$.

The dataset of scenario perturbations and schedule adjustments is then generated as follows. To obtain scenario $\xi^{(k)}$:

- use the generative model of the cluster from which the reference scenario was drawn in order to generate deviations for 96 time steps from the

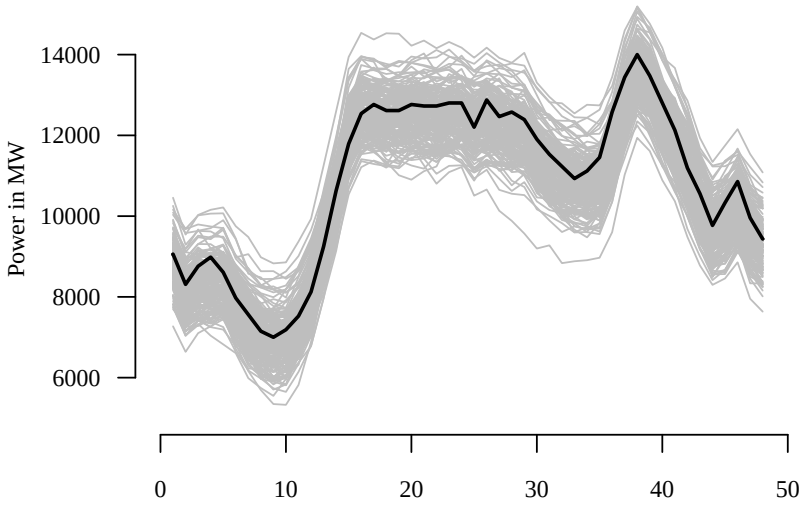


Figure 4.7: Reference (thick black) and perturbed (gray) demand curves. The horizontal axis indexes time steps of 30 minutes.

reference, by first drawing a deviation of the daily average using the normal distribution and then using the autoregressive model over 96 time steps (Figure 4.7),

- with probability p_o choose at random a generating unit among those in operation in the reference scenario at time $t = t_r - 1$ and make it unavailable for the rest of the horizon (until $t = 96$).
- adjust the generation schedule over the period $t_r, \ldots, T-1$ (cf. Section 4.3) to obtain $\pi^{(1,k)\star}$.

4.2.3 Discussion

Admittedly choosing as demand forecast for the next day the medoid of one of the cluster is a rough procedure. However this is just intended for a simulation purpose and in real-life one should use an up to date forecast if available. We have put more emphasis on the demand perturbation generation procedure which to us is a real need in our approach. We believe that this is done properly since the forecasting error is modeled quite accurately. An alternative would have been to use as perturbation signals the history of forecasting errors. Note that in Figure 4.7 the fact that perturbed curves are biased towards smaller demands reflects the reality of our dataset.

The choice of a probability of outage occurrence of 50% may seem exaggerated. However on the first hand we do not have accurate estimates of these probabilities, and on the other hand we do not believe that it is an inconvenient, since we are almost sure to generate scenarios encompassing the outage of each type of unit for each type of demand curve.

Furthermore, for each scenario we compute a sequence of quasi optimal decisions assuming the reference schedule given – as opposed to what stochastic programming would do, i.e. compute a first stage hedged against all possible scenarios – and the extraction of a recourse strategy is performed at the next step with the possibility to filter out outliers.

The choice of $t_r - 1$ as the outage occurrence time is motivated by the fact that we re-optimize only from time t_r on and thus, if the outage had occurred earlier, adjustments to the planning would have been made at earlier recourse stages (since in reality there are such opportunities say every hour).

Finally note that a few other settings have been investigated during the research but we have chosen to standardize the setting in this manuscript for the sake of comparison between experiments of Sections 4.4 and 4.5.

4.3 Computation of the adjustments to the reference schedule

In this section we provide some information about the computation of the quasi-optimal adjustments to the perturbed scenarios generated according to Section 4.2. In a first run we do not impose any limitation on the number of adjusted units and in a second run we set a limit of 6 units to be adjusted (cf. Appendix B.2.2). From this section to the end of the chapter the results related to the costs, the demand satisfaction and the illustration of the adjustments are presented for the recourse horizon cropped at the end of the first day, although we compute adjustments spanning also the second day. However for the results related to the computation time, the optimality gap and the variable importance we do not (or cannot) make this distinction.

4.3.1 No limitation on the number of adjustments

Figure 4.8 illustrates the timing of the quasi-optimal adjustments computation phase when a budget of 600 seconds is devoted to the rescheduling of each scenario and an optimality gap[3] of 0.5 % is tolerated. The time budget is exhausted for almost 400 scenarios and the optimality tolerance is not always satisfied. By plotting the computation time against the optimality gap one can indeed verify that the set of scenarios requiring the whole computation time budget corresponds to an optimality gap greater than 0.5 % while the complementary of this set corresponds to an optimality gap smaller or equal than 0.5 %.

Figure 4.9 represents the adjustment cost (thus integrated over $[t_r : T - 1]$) against the difference of energy between the reference and the targeted demand curves. In this plot the size of points is an affine function of the lost generation capacity consecutive to a unit outage. One can distinguish clusters of points corresponding to the same type of unit outage. In each cluster the points seem to be situated on a piecewise linear curve. Large outages together

[3]The optimality gap represents the relative difference between the best integer solution and the lower bound obtained from the continuous relaxation (cf. Appendix C.1.2).

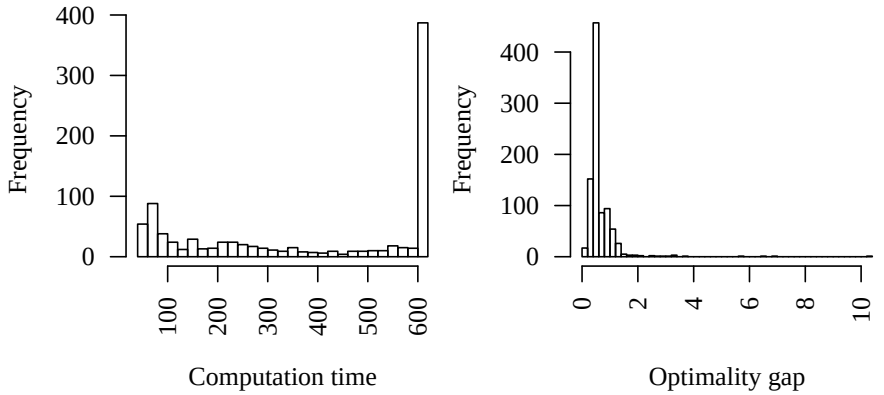


Figure 4.8: Left: histogram of the time required to compute the optimal adjustments (in seconds), with a time limit of 600 seconds. Right: histogram of the achieved Optimality gap (in %) with a tolerance of 0.5 %.

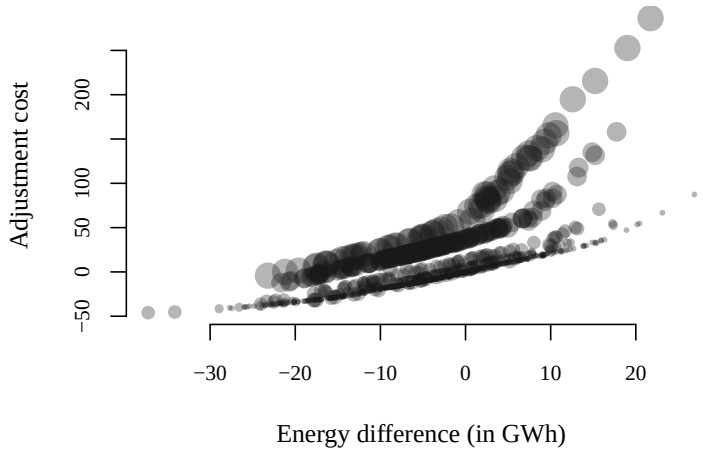


Figure 4.9: The adjustment cost against the difference of energy between the forecasted and realized scenarios (the point size is function of the unit outage).

with large increases of the demand with respect to the day-ahead forecast generally lead to high adjustment costs. Figure 4.10 gives further insight in the repartition of the cost of the adjusted schedules between the costs related to thermal generation, hydroelectric generation, and the penalization of the non-satisfaction of the demand and of the ancillary services requirements. The box[4] corresponding to the demand satisfaction indicates that high adjustment

[4] A box plot is a convenient way for graphically comparing several samples. The left and right extremities of one box represent respectively the lower and upper quartiles of one sample, while the median of the sample is depicted by the thick segment inside of the box. Whiskers extend (dashed line) from each extremity of the box to the farthest adjacent sample point within 1.5 times the interquartile range. Outliers (represented by circles) are sample points with values beyond 1.5 times the interquartile range from each side of the box.

costs of Figure 4.9 correspond to scenarios for which the optimization algorithm is not able to compute a schedule matching the targeted demand curve, thus yielding a high penalization of the discrepancy between the generation and the demand. However the median of the adjustment cost induced by the demand and ancillary requirements satisfaction is equal to 0 %, and most of the time the costs are distributed among the thermal generation costs and the hydroelectric opportunity costs.

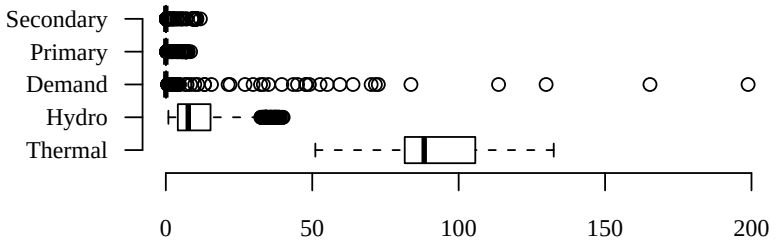


Figure 4.10: Box plot of the costs of the adjusted schedules as a percentage of the cost of the reference schedule.

Figure 4.11 and Figure 4.12 provide some insights on the variability of the adjustments of the thermal generation units and of the valleys respectively. Note that the frontiers of the envelopes do not necessarily represent feasible schedules since we have represented the pointwise extrema among all the adjusted schedules. Also the thermal unit outages were removed from the dataset before computing these envelopes to make the distinction between voluntary and involuntary shutdowns. The reference schedule is depicted by the thick curve. From Figure 4.11 one can observe that the eight smallest units in terms of generation capacity are the most frequently adjusted and are sometimes switched on or off. Indeed only two nuclear units are adjusted, but they are never shut down. Figure 4.12 shows that the adjustments to the hydroelectric generation fluctuates a lot around the reference schedule.

Over the full set of 900 scenarios, the median number of units (i.e. thermal units and valleys) for which the schedule is modified is about 12 and oscillates between a minimum of 10 and a maximum of 15 units.

4.3.2 Limitation on the number of adjustments

In a second experiment we add some constraints to the optimization problem, according to Section 3.3.3, to limit the maximum number of adjustments to 6. Although this might seem a very strict limitation since on average 12 units are adjusted when there is no limitation, this value creates an arbitrage between adjusting the hydroelectric units or the thermal generation units. By experiment, a higher value of K results in making adjustments to all the valleys and using the remaining adjustments on the thermal units.

For modeling the adjustment indicators we have tried the two formulations presented in Appendix B.2.2.2. Experimentally the second formulation

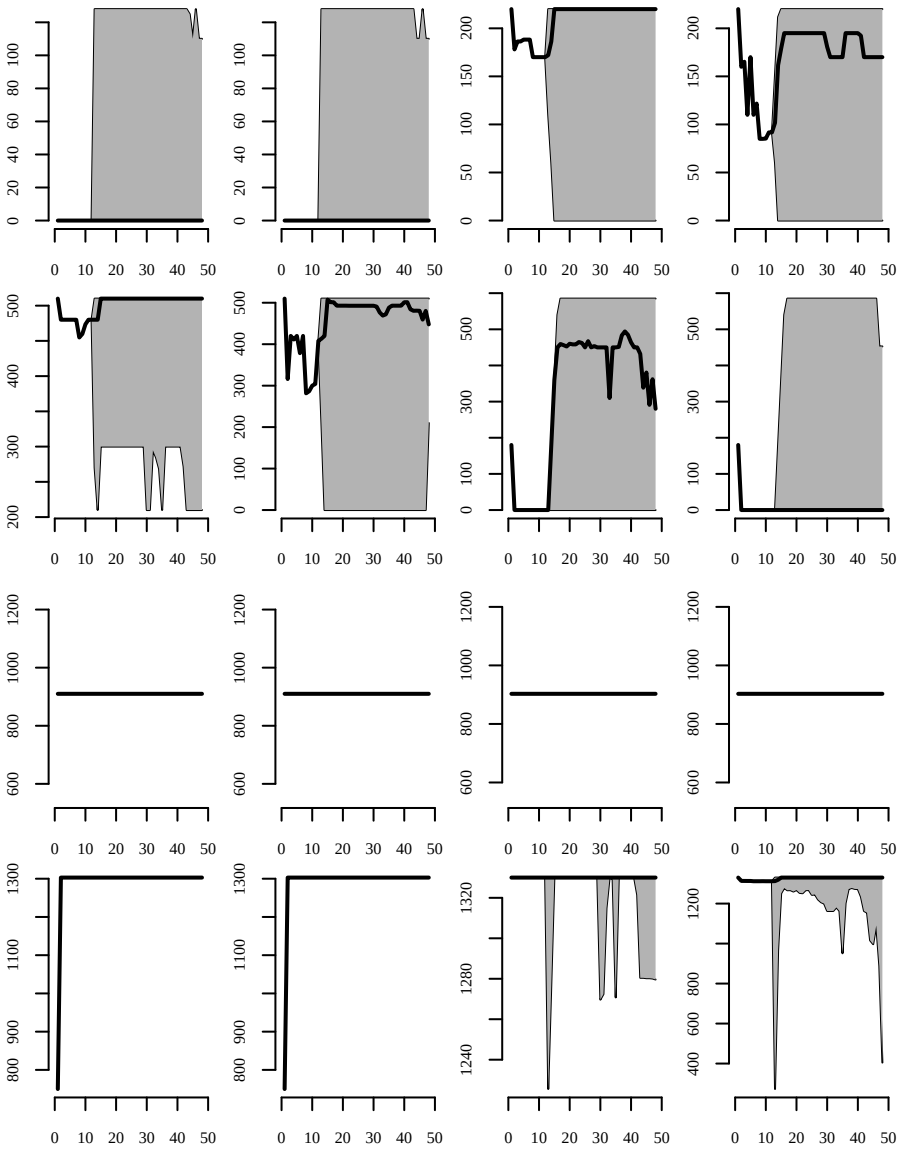


Figure 4.11: Illustration of the adjustments unit by unit. Each graph shows the envelope of the pointwise minimum and maximum power generated by the corresponding thermal unit, while the thick curve corresponds to the reference schedule. The units are ordered from left to right and form top to bottom according to the order of Table 4.1a. The Y-axis is indexed in MW.

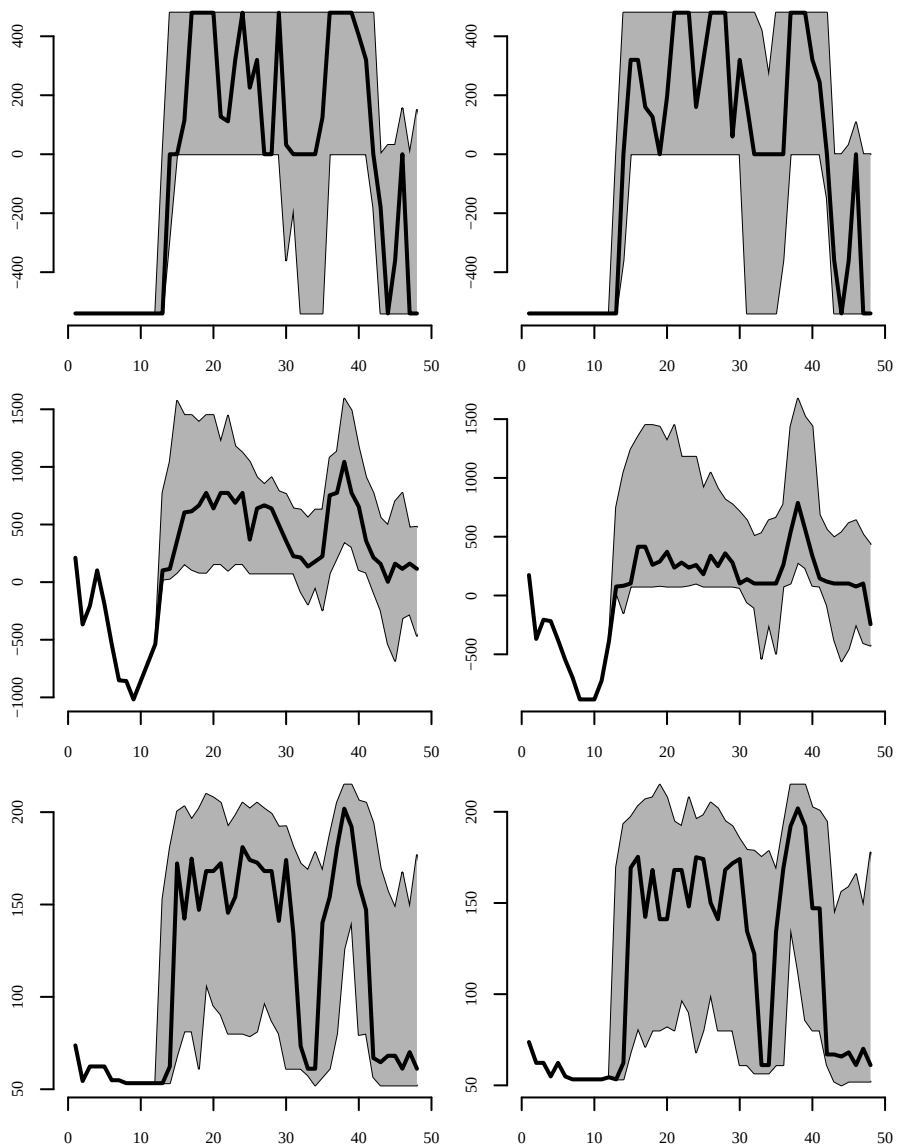


Figure 4.12: Illustration of the adjustments valley by valley. Each graph shows the envelope of the pointwise minimum and maximum power generated in the corresponding valley, while the thick curve corresponds to the reference schedule. From top to bottom, each row represents the valleys of type (a), (b) and (c) (cf. Figure 4.2). The Y-axis is indexed in MW.

has turned out to be more efficient and we used it exclusively. Nevertheless, we had to adapt the time limits and the tolerated gap in the following way. We did a first run with the same values as in Section 4.3.1. However the solver did not manage to find a feasible solution for almost 300 scenarios. We then restarted the optimization procedure for these scenarios with a tolerance of 1% and a time limit of 1 hour. Compared to Figure 4.8, Figure 4.13 shows a much bigger span for the optimality gap in addition to an increase of the computation time. Indeed for many scenarios for which a feasible solution is found after 600 seconds the gap is very high compared to the experiment of Section 4.3.1. For the scenarios requiring an extension of the computation time, the proportion of solutions below the tolerated gap is higher than for the feasible solutions found in 600 seconds or less. Yet, for about 30 scenarios the allowed time is consumed, and 30 scenarios are not solved. With respect to the simulations

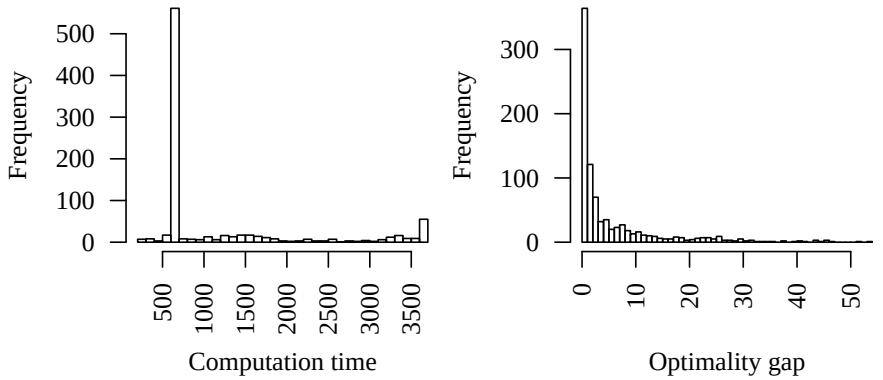


Figure 4.13: Limitation on the number of adjustments. Left: histogram of the time required to compute the optimal adjustments (in seconds). Right: histogram of the achieved Optimality gap (in %).

of Section 4.3.1, the decrease of performance of the branch-and-cut algorithm can be understood as follows. The linear relaxation of the constraints added to model the adjustment indicators is very weak and does not provide a lot of information about the actual optimal values of the adjustment indicators. This results in the necessity to develop a lot the branch-and-bound tree, and increases dramatically the resolution time.

With respect to Figure 4.9, Figure 4.14 shows the degradation of the adjustment cost due to the limitation on the number of adjustments. One can see on Figure 4.15 that this is due to generation-demand imbalances and not to thermal or hydroelectric costs.

4.4 Predicting power generation levels

Now that we have generated a set of scenarios and computed quasi-optimal adjustments of the reference schedule to these scenarios, we can apply supervised learning to derive a recourse policy. In this section, we want to predict for each

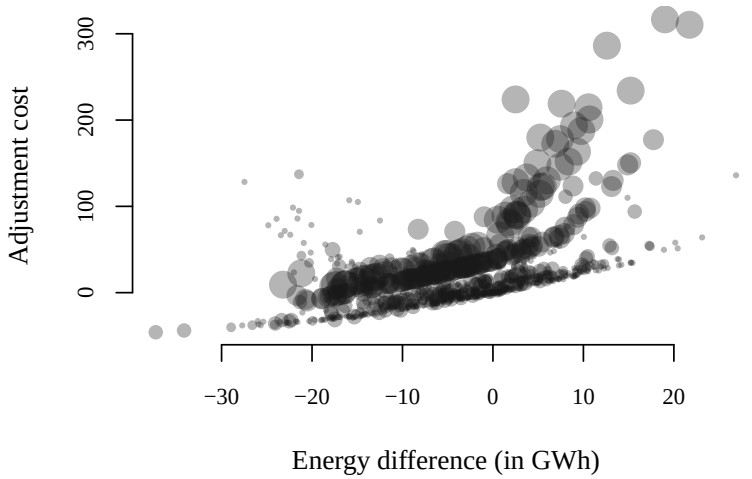


Figure 4.14: The adjustment cost against the difference of energy between the forecasted and realized scenarios when a limitation on the number of adjustments is imposed. The size of the points is function of the lost capacity consecutive to a unit outage.

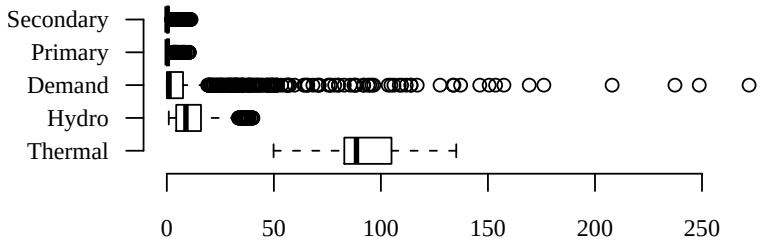


Figure 4.15: Repartition of the costs of the adjusted schedules as a percentage of the cost of the reference schedule when a limitation on the number of adjustments is imposed.

thermal unit and each valley which trajectory to follow inside the grey shaded area of Figure 4.11 and Figure 4.12 to adjust the reference schedule, given the observation of the realized scenario on $[0 : t_r - 1]$. The adjustments predicted according to the learned recourse policy are individually post-processed to ensure their feasibility (see Section 4.4.3). We analyze the fact of knowing an updated demand forecast when computing the adjustments with respect to the fact of ignoring this information. When an updated demand forecast is available it is included in the inputs of the formulation, but the post-processing phase ignores this information and thus does not aim at improving the satisfaction of the adjusted generation versus updated forecast balance. We compare our approach to the adjustments obtained when re-optimizing the day-ahead schedule in order to satisfy the updated forecast.

4.4.1 Learning the recourse policy

We opt for the formulation of Case 3 in Section 3.4.3 since it yields a single learning problem which takes into account explicitly the influence of the time variable in the adjustments sequences and preserves the coupling of the generation units. Table 4.1 describes an entry of the learning set for this formulation.

Table 4.1: An entry of the learning set for the recourse prediction task

Inputs	Outputs
• state of the system at time $t_r - 1$, • observed load deviation from the demand forecast on $[0 : t_r - 1]$, • unit(s) outage before time t_r, • prediction time t_p, • updated demand forecast on $[t_r : T - 1]$	vector of thermal units generation levels and power generated in each valley at time t_p.

The state of the system is represented by

- the generation level and the ON/OFF status of each thermal unit and the number of time steps since which the unit has the same status,
- the volume of the reservoirs,
- the generation level and the number of groups activated of each hydroelectric plant.

The observed load deviation as well as the updated demand forecast are represented as time series, with some additional features computed from these series such as the energy. The outage is described by the maximal, minimal and actual generation levels of the lost unit as well as its costs.

The problem formulated in this way contains 340 input variables and 22 output variables, while the number of objects is multiplied by the number of time steps over the period $[t_r : T - 1]$. Note that for Random-Access Memory saturation reasons (using the Matlab interface to the Extra-Trees), we have randomly selected a subset of 500 scenario-schedule pairs among the 900 that we have simulated for the rest of the experiments of this Section. The dataset thus contains $500 \times (T - t_r) = 500 \times 84 = 42000$ objects.

We adopt the Extra-Trees (cf. Appendix A.2) to model the recourse strategy, since our preliminary experiments (cf. Cornélusse (2008)) showed that the Extra-Trees clearly outperformed the ε-SVR (cf. Appendix A.3) on this problem, at least for the common kernels that we have tested.

The experiment is structured as follows. The training set is partitioned in 5 folds randomly. Then repeatedly one fold is isolated for testing and a recourse strategy is learned from the 4 remaining folds. The parameters of the ensemble

of trees are tuned using a grid search, and we have implemented an inner two-fold cross-validation procedure to this end. We build an ensemble of 100 trees. The best values of K and n_{min} are respectively 340 and 5. The overall time required to run the nested CV is about several hours on a single threaded machine, and it takes about one hour to run the outer CV with the best values of the parameters. Note that the Extra-Trees can easily be computed in parallel, thus the computation time may be roughly divided by the number of trees if sufficient computer capacity were available. In addition the inner CV loop used to optimize the parameters may be skipped since the selected parameters values are almost constant over repeated experiments in practice. Indeed one can observe the following phenomena: over all the experiments the best value of K is equal to the number of input features. The accuracy in terms of learning error and the overall performance (as will be described below) constantly increases when the number of trees in the ensemble increases. The optimal value of the parameter n_{min} which sets the number of learning objects under which a node is not split any further (cf. Appendix A.2) is about 5, meaning that the trees must almost be completely developed (otherwise the predicted adjustments would be excessively smoothed). The overfitting effect that results from such a small value for n_{min} is countered by the ensemble effect, hence the necessity of a large number of trees in the ensemble.

At this stage we can analyze the prediction error with respect to the adjustments of the data set used to train and validate the learned recourse policy. Figure 4.16 depicts the distribution of the prediction error unit by unit. Clearly, the thermal units' adjustments are better predicted than the hydroelectric adjustments where errors of several hundreds of MW occur frequently. However there may be many good solutions to the adjustment problem and thus the unit by unit prediction error analysis does not tell much about the optimality of our recourse policy in terms of coupling constraints satisfaction and overall adjustment cost. For example, with respect to the quasi optimal adjustment, a large positive error in one valley may be compensated by a large negative error in another valley. Furthermore the training examples are not totally optimal.

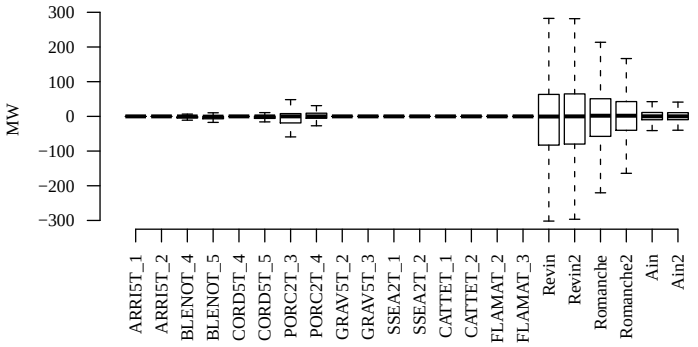


Figure 4.16: Distribution of the prediction error unit by unit. Outliers are not shown.

Thus we must also analyze the optimality in terms of the satisfaction of the coupling constraints. Contrary to the unit by unit analysis it makes sense to compare the solution of several policies. Figure 4.17 illustrates the policy which re-optimizes the reference schedule based on the updated demand forecast (Figure 4.17a), the reference schedule itself (Figure 4.17b) and the learned policy (Figure 4.17c) in this respect. In the left column we have illustrated the time evolution of the distribution of the difference between the updated demand and the adjusted generation on the recourse period, i.e.

$$\left\{ d'_t - \left(\sum_{i=0}^{N_t-1} \hat{p}_{i,t}^t + \sum_{j=0}^{N_v-1} \hat{p}_{j,t}^h \right) \right\}_{t=t_r}^{T-1},$$

where d'_t is the updated demand forecast at time t, N_t and N_v are respectively the number of thermal generation units and hydroelectric valleys, and $\hat{p}_i^t$ and $\hat{p}_j^h$ are respectively the predicted generation of thermal unit number i, $i = 0, \ldots, N_t - 1$, and valley number j, $j = 0, \ldots, N_v - 1$, at time t. In the right column we have illustrated the histogram of the energy slack, defined as

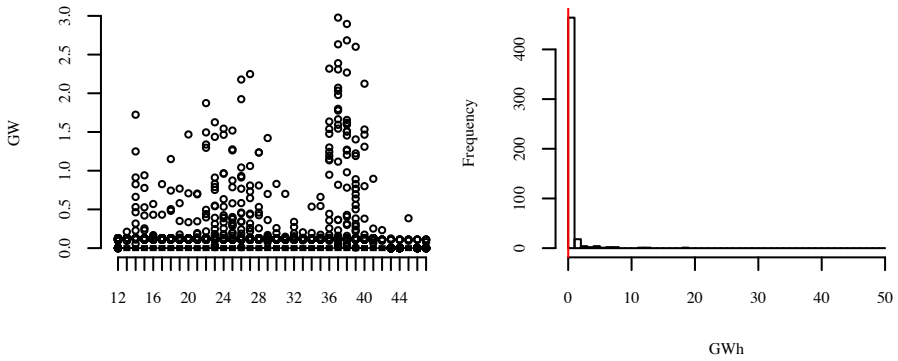
$$\sum_{t=t_r}^{T-1} \left| d'_t - \left(\sum_{i=0}^{N_t-1} \hat{p}_{i,t}^t + \sum_{j=0}^{N_v-1} \hat{p}_{j,t}^h \right) \right|.$$

In the first row one can observe, as in Figure 4.10, that the updated demand forecast is not always satisfied after the re-optimization of the reference schedule (circles are outliers). However the median energy slack is about 0.004 GWh. The second row illustrates the large discrepancy between generation and demand if the reference schedule were applied without adjustment. The discrepancy fluctuates over the recourse horizon with no apparent trend and the median energy slack equals 8.7 GWh. On the other hand the discrepancy of the learned recourse policy is larger from time steps 12 (6 AM) to 28 (2 PM) than on the rest of the recourse period, except at the moment of the peak load, and the median energy slack equals 2.43 GWh.

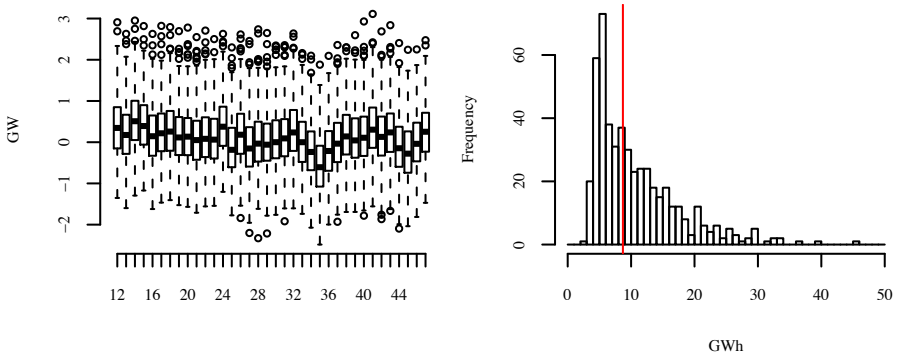
We show an illustration of the predicted adjustments for one particular scenario on Figures 4.18, 4.19, and 4.20 of the adjusted generation schedule (blue) which yields an energy slack of 1.3 GWh while the energy slack would be 11.9 GWh if the reference schedule were applied without modification to that scenario (red).

Figure 4.18 shows the aggregated generation curves obtained by summing the contributions of all the generation units for the three policies (upper part) as well as the residuals with respect to the updated demand forecast (lower part). In this scenario the updated demand is higher than the demand curve used to compute the reference schedule, and there is no unit outage. In accordance with Figure 4.17, the residuals of the approximated recourse are much smaller than those of the reference schedule.

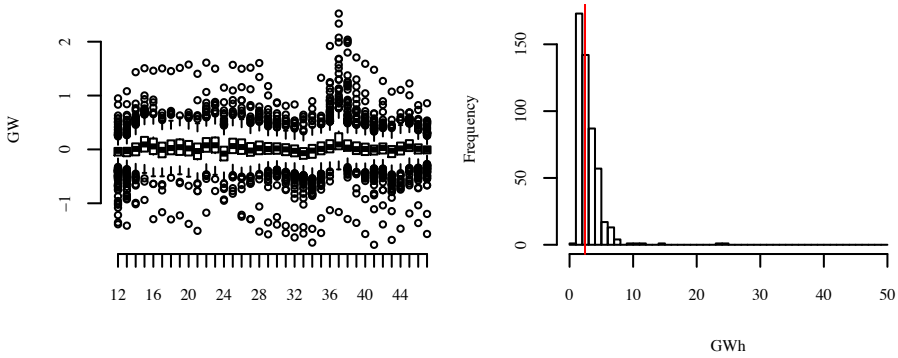
Figure 4.19 shows the adjustment made to the thermal units. Almost no modification is done to the nuclear units (the lower eight units). On the other hand there are some interesting phenomena to notice about the classical thermal units. Reading the figure from left to right and top to bottom, the first



(a) Quasi-optimal adjustment.



(b) Day-ahead schedule without adjustment.



(c) Predicted adjustment.

Figure 4.17: Analysis of the demand satisfaction error for three policies. Left column: time evolution of the distribution of the error as box plots. Right column: histogram of the energy slack (breaks are distant of 1 GWh). The red vertical line represents the median of the distribution.

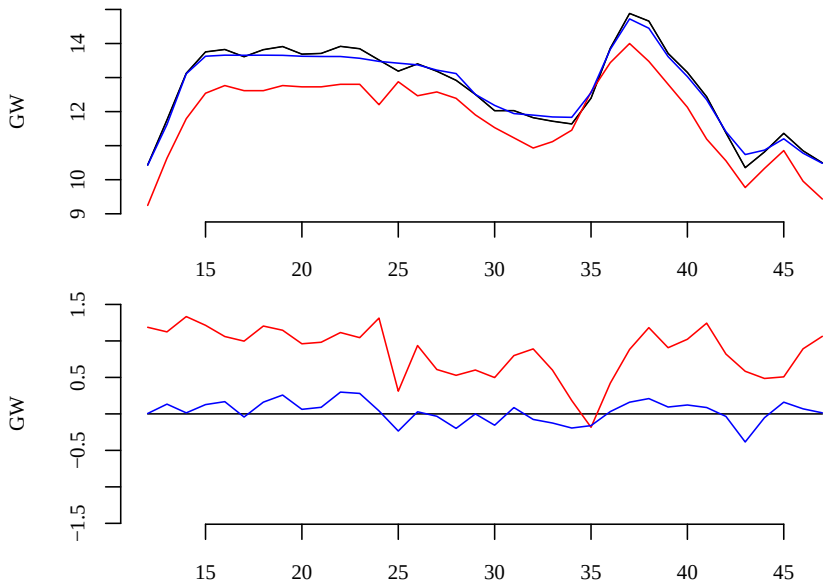


Figure 4.18: Top: total power generation of (black) the optimally adjusted schedule, (blue) the predicted recourse, (red) the reference schedule, over the recourse period. Bottom: residuals with respect to the updated forecast.

two units are stopped in the reference schedule and remain stopped in the quasi-optimal adjustment, but our approach predicts a non-zero generation level. However the generation levels evolve well under the technical minimums of these generation units, which are indicated by the dotted black horizontal lines, and are thus not feasible. This illustrates the necessity of a post-processing phase, as discussed in Section 3.4.2.3, which in this case should obviously decide to leave the two units shut off. The predictions are almost correct for units 3 to 7. Finally the learned adjustment strategy decides to start-up unit 8 almost at same time than the quasi-optimal adjustment, and evolves similarly, except at the end of the first day where the level stays inside the interval between the off state and the technical minimum. This again needs to be corrected in the post-processing phase.

Figure 4.20 shows the adjustment to the power that should be generated in each valley. These signals are more complex and the learned recourse policy appears to be less accurate than for the thermal units. The discrepancy between the total generation and the updated demand is due to the incapacity of the model to capture the fluctuation of the generation level on the time interval $[12 : 28]$ but rather outputs a smoothed generation curve (although it looks acceptable for the valleys of row 2). To decrease the smoothing effect one solution might be to select a smaller value for the parameter n_{min}, but this turned out to decrease the estimate of the overall generalization error in our experiments. It is also less evident to evaluate a priori whether the predicted curves can be satisfied tightly when considering the constraints on the volume of the reservoirs in each valley.

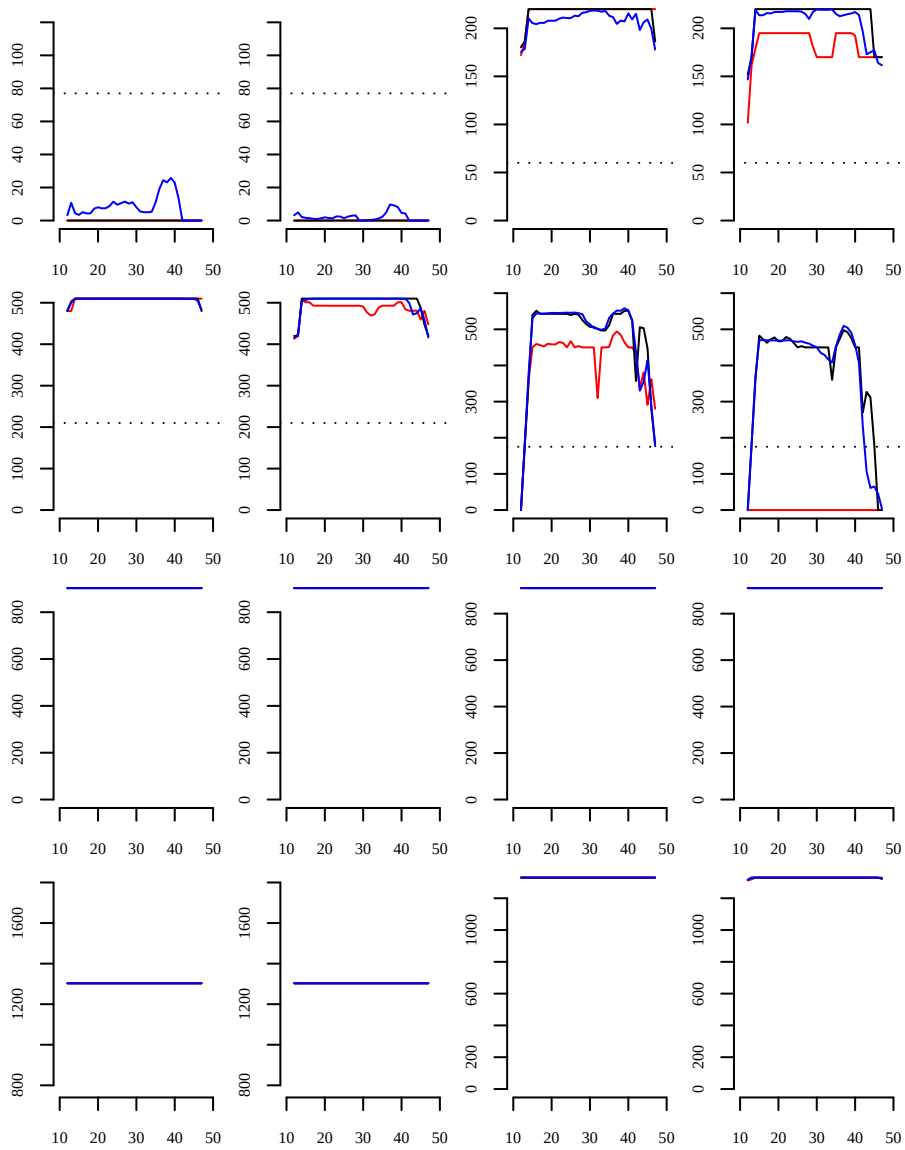


Figure 4.19: Illustration of the adjustments of the thermal units. Each graph shows the optimal adjustment (black), the reference schedule (red) and the predicted adjustment (blue) of the corresponding thermal unit. The units are ordered from left to right and from top to bottom according to the order of Table 4.1a. The Y-axis is indexed in MW.

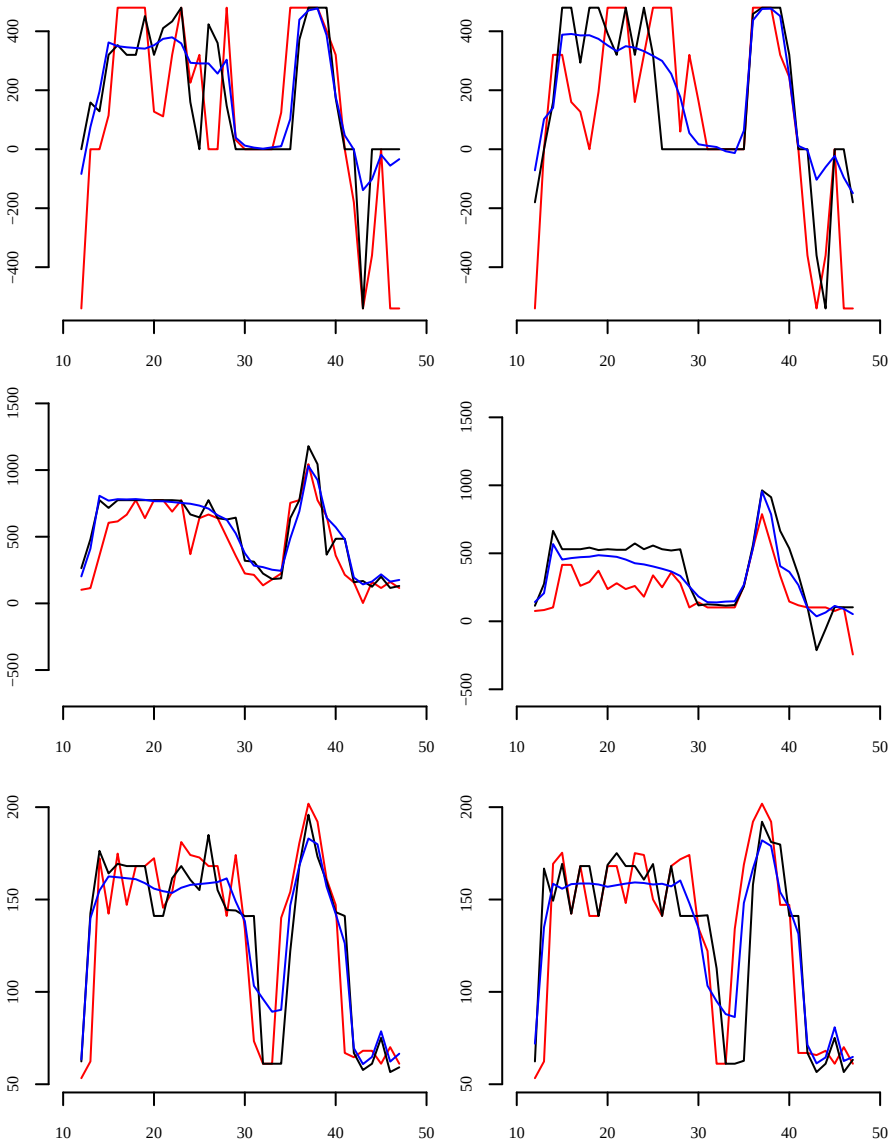


Figure 4.20: Illustration of the adjustments valley by valley. Each graph shows the optimal adjustment (black), the reference schedule (red) and the predicted adjustment (blue) in the corresponding valley. From top to bottom, each row represents the valleys of type (a), (b) and (c) (cf. Figure 4.2). The Y-axis is indexed in MW.

4.4.2 Importance of variables

To analyze the influence of input features on the recourse decisions, we show their importances as evaluated by the Extra-Trees algorithm on Figure 4.21. The pie chart on the left indicates that the variables related to the state of the system just before the recourse period are the most important in the decision model, followed by the variables related to the updated demand, the variables describing the outage, and the time variable spanning the recourse horizon. Indeed the state of the system just before the recourse period describes, in addition to the effect of the deviation of the demand on the period $[0 : t_r - 1]$, partially the effect the unit outage; this explains why it has such a high importance in the decision process. The right part of the figure indicates the repartition of the importance among the input features representing the evolution of the updated demand forecast along the scheduling horizon. The percentage of variance reduction fluctuates between 0% and 1%. The variables related to the recourse period are more important than the variables related to the pre-recourse period, except the realized demand at time 0 which is indicative of the bias added in the scenario generation process (cf. Section 4.2).

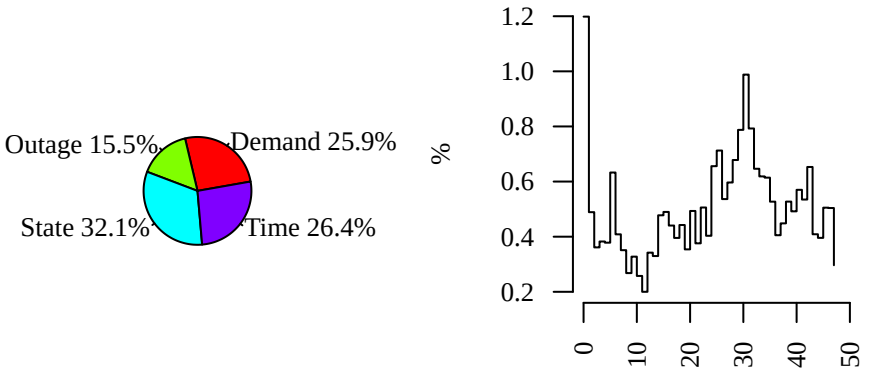


Figure 4.21: Left: Importance of the categories of input features in the decision model. Right: percentage of variance reduction of the features representing the updated load forecast.

4.4.3 Obtaining feasible adjustments

Necessity of a post-processing stage. We need to post-process the predicted adjustments to render them feasible since the learned model does not guarantee it per se. By feasible, we mean "satisfying the individual operating constraints of the units". Although the learning algorithm does not handle the coupling constraints explicitly, it is evaluated on this criterion, and we can penalize the energy slack and the lack of ancillary services reserves. Thus the non-satisfaction of the coupling constraints penalizes economically the recourse strategy, and may render it more or less valuable than another strategy, while

the non-satisfaction of the individual constraints renders the recourse strategy useless. The details of the post-processing procedure that we apply to predictions in order to obtain feasible adjustments are described in Appendix B.3.1.

Impact of the post-processing on individual adjustments. Figure 4.22 illustrates the distribution of the modifications imposed by the post-processing stage to the adjustments made to each generation unit. More precisely, it rep-

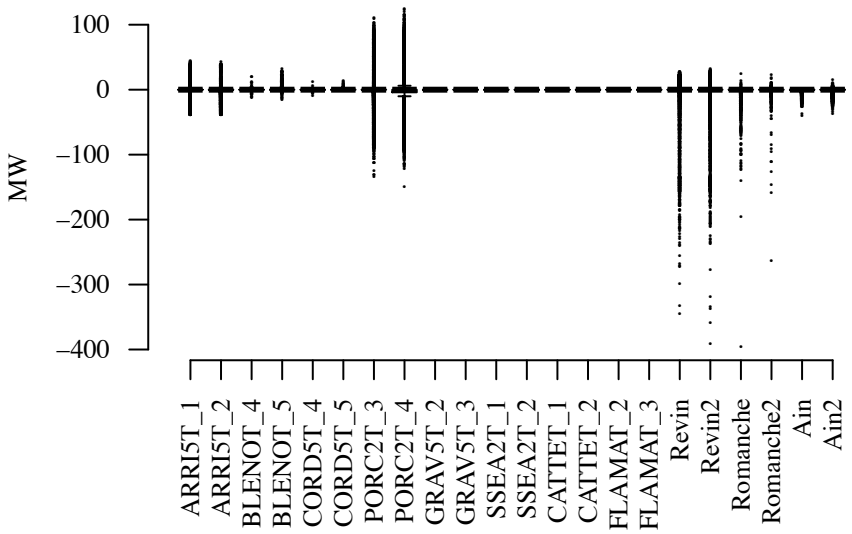


Figure 4.22: Box plots of the modifications to the predicted adjustments caused by the post-processing stage.

resents the distribution of the post-processed adjustment minus the predicted adjustment at any time step. Nearly all boxes shrink to a 0 variation range, and the large modifications are indeed outliers (there are 500 * 84 = 42000 points to represent for each unit). The distributions are symmetrical for the thermal units and the main corrections are indeed made to prevent the generation curve from containing points in $[0, P^{min}]$ except for start-ups or shut-downs. On the other hand the distributions are asymmetric for the valleys. This stresses the fact that the predicted demand is sometimes superior to what can actually be generated.

Satisfaction of the load-generation balance. Similarly to Figure 4.17, Figure 4.23 illustrates the distribution of the mismatch between generation and demand for the post-processed adjustments. One can observe that over all the scenarios the post-processing only slightly increases the energy slack. The median energy slack is about 2.59 GWh after post-processing (2.43 GWh before the post-processing).

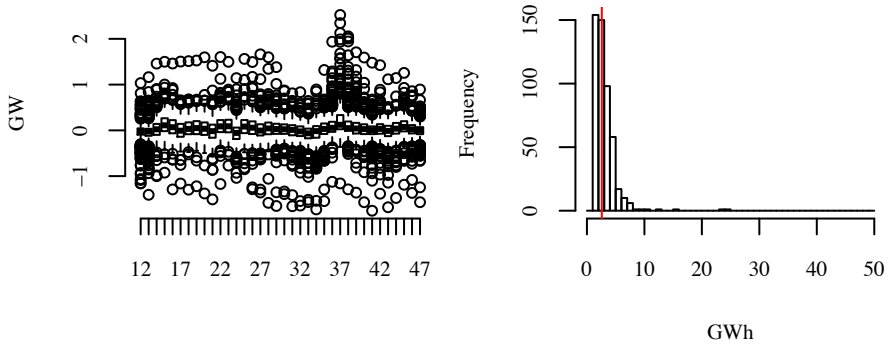


Figure 4.23: Analysis of the demand satisfaction error for the post-processed adjustments. This figure must be compared with the last row of Figure 4.17.

Duration. Figure 4.24 illustrates the time required for post-processing the predicted adjustments. The median post-processing time is about 22 seconds, and is, except for two scenarios, always much smaller than the time required to compute the optimal adjustments whose median time is about 600 seconds.

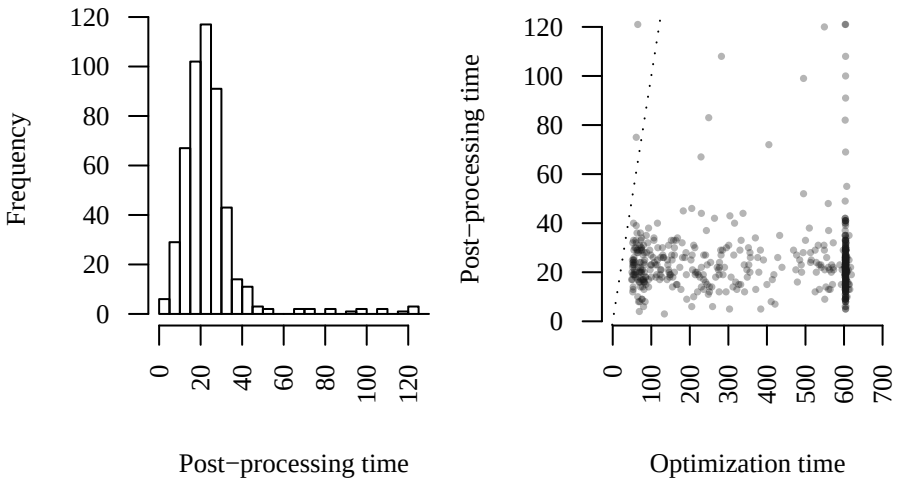


Figure 4.24: Left: Histogram of post-processing time. Right: Scatter plot of the post-processing time against the time required to compute the quasi-optimal adjustments. The dotted line has unit slope and intercepts the origin.

Illustration on one scenario. In each of the Figures 4.25, 4.26 and 4.27 the blue continuous curve is the post-processed version of the dotted blue curve. The dotted blue curve itself corresponds to the predicted adjustment in Figures 4.18, 4.19 and 4.20. Figure 4.25 shows the small degradation in terms of demand satisfaction.

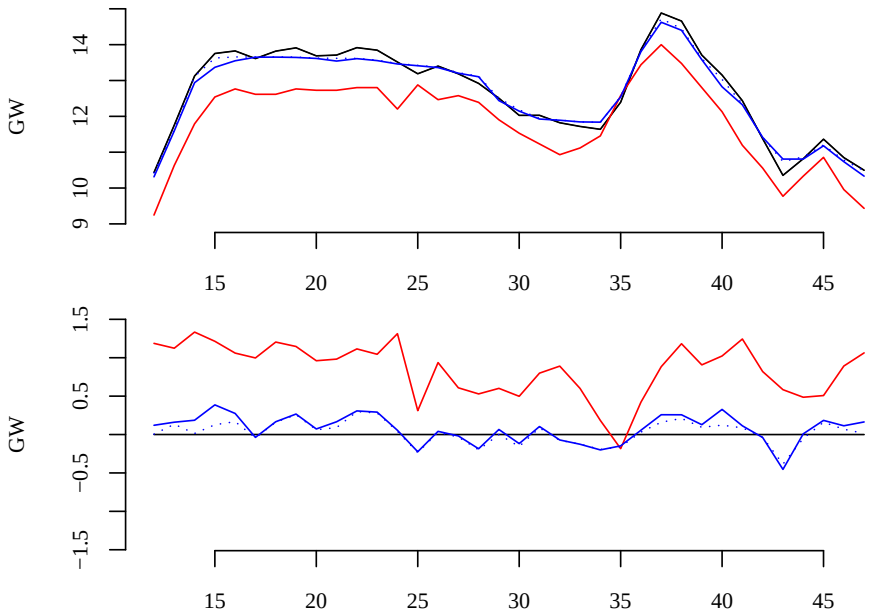


Figure 4.25: Top: impact of the post-processing on the satisfaction of the updated demand forecast. Black: optimally adjusted schedule. Dashed blue: predicted recourse. Blue: post-processed predicted recourse. Bottom: residuals with respect to the updated forecast.

Figure 4.26 illustrates the modification made to the predicted adjustments of the thermal units. Units 1 and 2 are shut off as in the quasi-optimal adjustment, and the shutdown of unit 8 is slightly modified. The other units are not impacted.

Figure 4.27 illustrates for each valley the achievable generation with respect to the predicted demand. A discrepancy between generation and demand is observed in the valleys of the first and third rows. In the third row one can observe that the generated power is too high from time steps 30 to 35, preventing the peak demand to be satisfied.

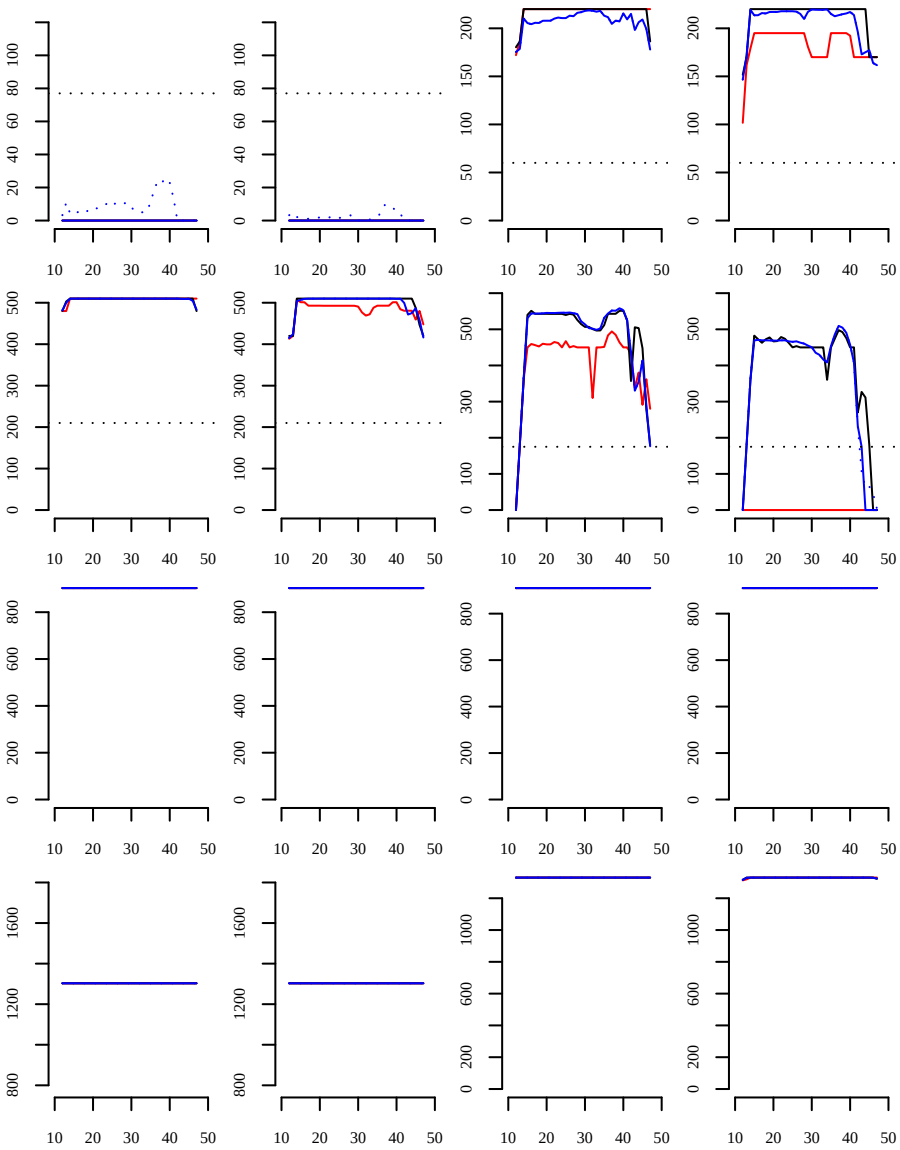


Figure 4.26: Illustration of the adjustments of the thermal units. Each graph shows the optimal adjustment (black), the predicted adjustment (blue) and the post-processed predicted adjustment (dashed blue) of the corresponding unit. The units are ordered from left to right and from top to bottom according to the order of Table 4.1a. The Y-axis is indexed in MW.

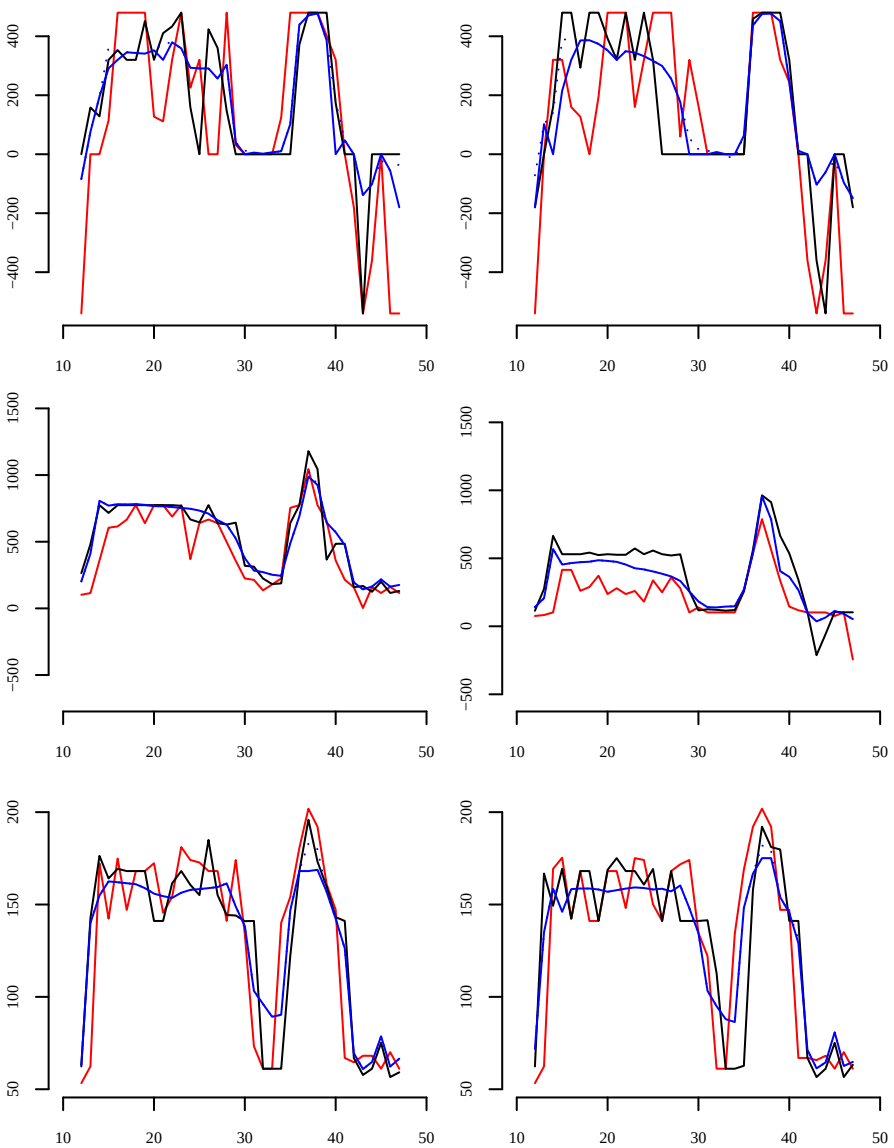


Figure 4.27: Illustration of the adjustments valley by valley. Each graph shows the optimal adjustment (black), the predicted adjustment (blue) and the post-processed predicted adjustment (dashed blue) of the corresponding valley. From top to bottom, each row represents the valleys of type (a), (b) and (c) (cf. Figure 4.2). The Y-axis is indexed in MW.

4.4.4 Overall adjustment costs

Finally we have analyzed the optimality in terms of generation costs of the schedules computed by the recourse strategies approximated with the Extra-Trees and after the post-processing stage. The previous analysis shows that the main source of additional cost is the non-satisfaction of the demand or of the ancillary reserve requirements.

Figure 4.28 shows the results obtained on the 500 scenarios. Each point refers to a scenario of deviations combining the loss of a generation unit before 6 AM and a deviation of the load curve from its forecast. Over the horizontal axis these scenarios are sorted according to the total cost associated to them if perfect (full) knowledge of the scenario is exploited to re-optimize the generation plan; over the vertical axis they are sorted according to the actual incurred cost depending on the adjustment strategy. The dotted vertical line which reflects the cost of the reference schedule (about 4.1×10^6) partitions the horizontal axis in the scenarios whose realized and updated demand are on average lower than the reference demand (left) and scenarios whose realized and updated demand are on average higher than the reference demand (right). For each scenario, three different adjustments have been evaluated, corresponding to three different points at the same horizontal coordinate:

- the first strategy consists in applying no recourse action at all (the corresponding points are depicted using red + symbols). In this case the

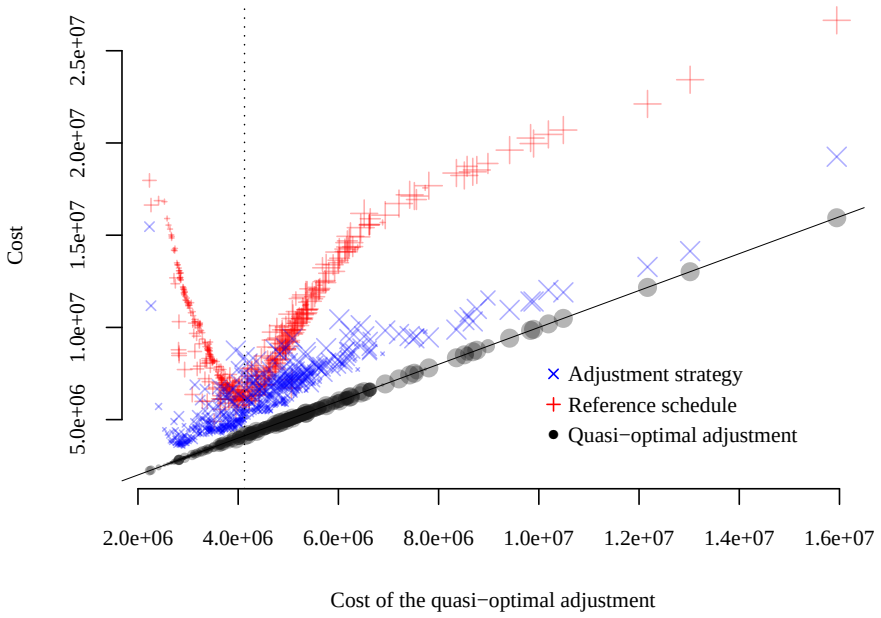


Figure 4.28: Scatter plots of the schedule cost during the recourse period vs. the cost of the planning quasi-optimally adjusted. Symbol size is an increasing affine function of the lost capacity consecutive to a unit outage.

loss of the generation unit and the deviation of the load are compensated by purchasing rather expensive reserves. Their price is modeled by a piecewise linear and convex function, hence the 'V' shape centered on the cost of the reference schedule and spanning from 2×10^6 to 6×10^6. The right hand slope decrease from 6×10^6 results from the fact that the corresponding cost of the optimal strategy increases accordingly since it is unable to satisfy the demand (cf. the outliers of the 'Demand' box in Figure 4.10). This strategy constitutes the worst case behavior;

- the second strategy (represented by black • symbols) corresponds to the perfect information case. In this case the points are located on the line $y = x$; their cost on the vertical axis represent a lower bound for all possible recourse strategies;

- the last strategy is the one built using the proposed procedure described in Table 3.1 (it is represented by blue $\times$ symbols).

We note that our approximated schedules yield costs which are often much lower than the day-ahead schedule applied without modification and quite close to those assuming perfect knowledge. Furthermore when considering the range spanned by the cost of the quasi-optimal adjustment, one can observe that the difference of cost with respect to the quasi optimal adjustment is almost constant. Thus although the dataset contains a few extreme scenarios, they do not seem to impact the overall performance. Figure 4.29 is another representation of the information contained in the Figure 4.28. It is a cumulative histogram of the additional cost induced by the strategies compared to the cost of the schedules deterministically optimized knowing a perfect forecast of the system conditions, i.e. the black points of the top subfigure. The red curve corresponds to the red $+$ symbols, while the blue curve corresponds to the blue $\times$ symbols.

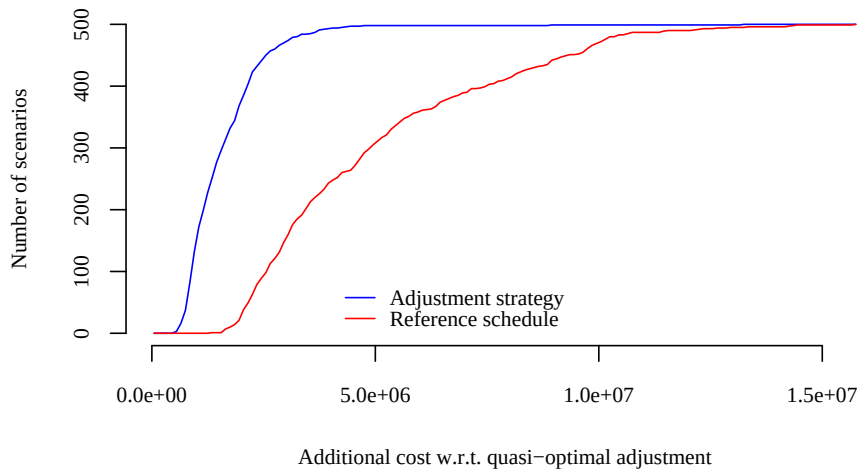


Figure 4.29: Cumulative histogram of the additional cost compared to the quasi-optimally adjusted schedules.

Reduction of the learning set size. We repeat the same experiment (learning, post-processing, evaluation) assuming less scenarios are available to learn the recourse strategy, in order to have some insight on the sensitivity of the overall performance of the proposed approach in this respect. We still have 500 scenarios, but when performing the five-fold cross-validation we randomly sub-sample the four folds devoted to learning to keep only 50% or 10% of the available data (LS). Using 50% of the dataset shifts the full learning set curve to the right of an amount of 10^5, while with 10% of the data set only the bottom part of the curve is shifted of 7×10^5 to the right, and the top part (the last 200 scenarios) is even more flattened.

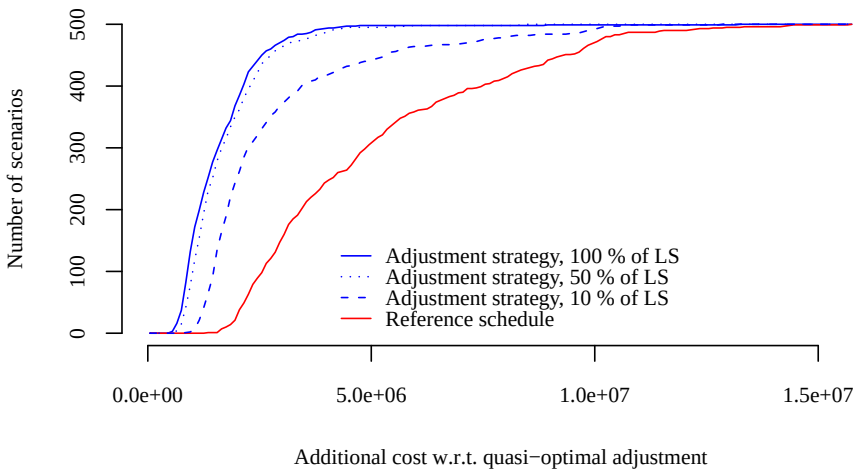


Figure 4.30: Cumulative histogram of the additional cost compared to the quasi-optimally adjusted schedules. Effect of the reduction of the learning set size.

No updated demand curve. As mentioned in Section 3.3, the re-optimization of the perturbed scenarios yields a dataset of generation adjustments which are optimistically biased because they assume perfect information about the behavior of the system at subsequent time steps. To decrease the over-fitting of the sample of perturbed scenarios, we propose to suppose that the updated demand forecast is unavailable when the recourse opportunity occurs, and thus to discard this information from the inputs of the supervised learning formulation. Doing so, we can enforce the projection of the over-fitted schedules on a set of non-anticipative decision strategies which are only function of the information available at time t_r. We thus end up this section by repeating the same experiment using all the available dataset for learning, but suppose that the updated demand forecast is unavailable when the recourse opportunity occurs.

This formulation yields the dotted blue curve of Figure 4.31, which is shifted to the right of about 9×10^5 with respect to the continuous curve. This gap reflects the value of knowing the updated forecast in the present setting, since we suppose that the realization will be exactly equal to the updated forecast. Contrary to the experiment above where we use only 10% of the dataset for

learning, the top part of the curve is closer to the continuous one. This can be explained by the fact that scenarios leading to a high additional adjustment cost with respect to the quasi-optimal recourse correspond to the scenarios containing a unit outage. Clearly when reducing the dataset to 10% of its content one loses a lot of information about the adjustment consecutive to a unit outage, while when reducing the information about the demand this information is obviously still available.

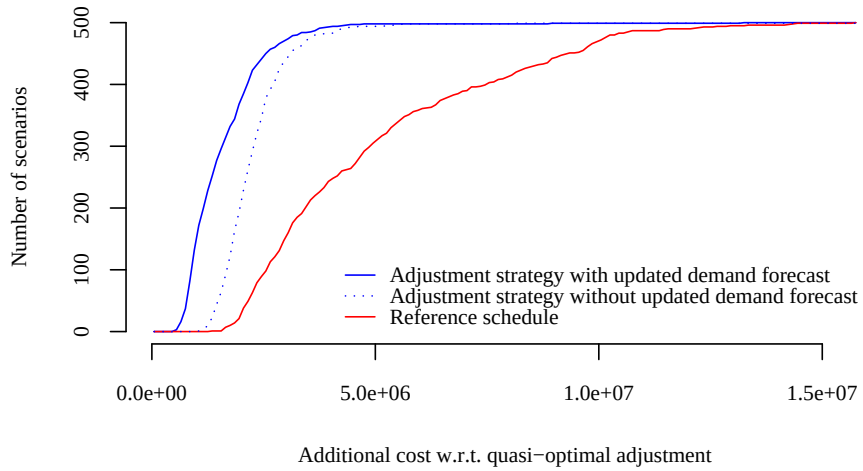


Figure 4.31: Cumulative histogram of the additional cost compared to the quasi-optimally adjusted schedules. Effect of the knowledge or of the ignorance of an updated demand forecast.

4.5 Predicting the subset of thermal generation units to adjust

In this section we analyze an alternative formulation of the adjustment strategy learning problem. In particular we analyze the formulation exposed in Section 3.4.2.1 that allows to instantiate a simpler version of the generation rescheduling problem faced in the intra-day context.

We use the simulations of Section 4.3.2 where a limit on the number of adjustments per recourse was imposed. Our aim is to decide which units must be adjusted using the same input information than in Section 4.4. Instead of a multivariate real valued time-series, the output of the learning problem is thus a vector of binary values containing at most K non-zeros. From the prediction of these values we instantiate and solve a downsized instance of Formulation 5 to obtain the adjusted scheduling decisions.

As we solve explicitly an instance of Formulation 5, an updated demand forecast is mandatory to specify the problem properly. The goal of this section is to analyze how the resulting schedule satisfies the updated demand and how it compares in terms of computation time with respect to the simulation phase.

As illustrated in Section 4.3.2, the proven optimality gap of the schedules that we will use for learning are far from the tolerated gap because the allowed computation time was exhausted for many scenarios. Thus, in opposition to the experiment of Section 4.4, the dataset now contains samples which cannot be qualified as "quasi-optimal". We will analyze how the learned adjustment strategy is impacted.

Figure 4.32 illustrates how the selected subset of K units fluctuates among the scenarios. A black dot indicates the adjustment of the corresponding unit. An explicit analysis shows that on 900 scenarios there are 213 different patterns. The most common pattern, which occurs for about 150 scenarios, consists in adjusting only the hydroelectric units, which are in a sense the most flexible. As there are more than 150 scenarios containing no unit outage, the demand fluctuation is however sufficient to cause the adjustment of the thermal units instead of the valleys. We can identify some interesting behaviors. For example, when $PORC2T_3$ fails, the unit $PORC2T_4$, which is identical but a bit more expensive, is started up to compensate. Note that units $ARRI5T_1$, $ARRI5T_2$ and $PORC2T_4$ are shut off in the reference schedule shown on Figure 4.4, thus the adjustment of any of these units involves a start-up.

4.5.1 Solving the learning problem

We again face a learning problem where the output has some known structural properties that are not explicitly taken into account by the learning algorithm. However this time they are much easier to characterize, since the only constraint is that there may be no more than K non-zero values in each output.

We opted for a regression setting where we consider the 900 scenarios-adjustment pairs of Section 4.3.2, the input-features are the same than in the experiment of Section 4.4 (cf. Table 4.1), and the outputs are the vectors of 22 adjustment indicators. The problem is thus much smaller than in Section 4.4 since we do not consider the time dimension. The learned adjustment strategy will predict vectors having components lying in the $[0, 1]$ interval. We retain the K highest values of the predicted vector as indicators of the units to adjust. We can thus use the same regression tree induction algorithm than in Section 4.4.

Indeed since the number of possible outputs is finite we could handle this problem as a classification problem with as many classes as there are combinations of $K = 6$ units among 22, i.e. 74613 classes. However only 213 patterns are represented in the dataset. But 213 classes is still a large number since we have only 900 objects in the dataset and that some classes are much more represented than others. A solution could be to use a clustering algorithm to identify a subset of the most representative output patterns, and then use these latter clusters as classes. But we do not know how to select the number of clusters a priori. To simplify, we may construct a classifier for each generation unit and decompose this problem into 22 binary classification sub-problems. Doing so we lose the coupling information and need to post-process the model in order to enforce the limitation to K adjustments. However there is no clear way how to perform this post-processing in a classification setting, except by counting the number of positive votes for each unit and retaining the adjustment of the K

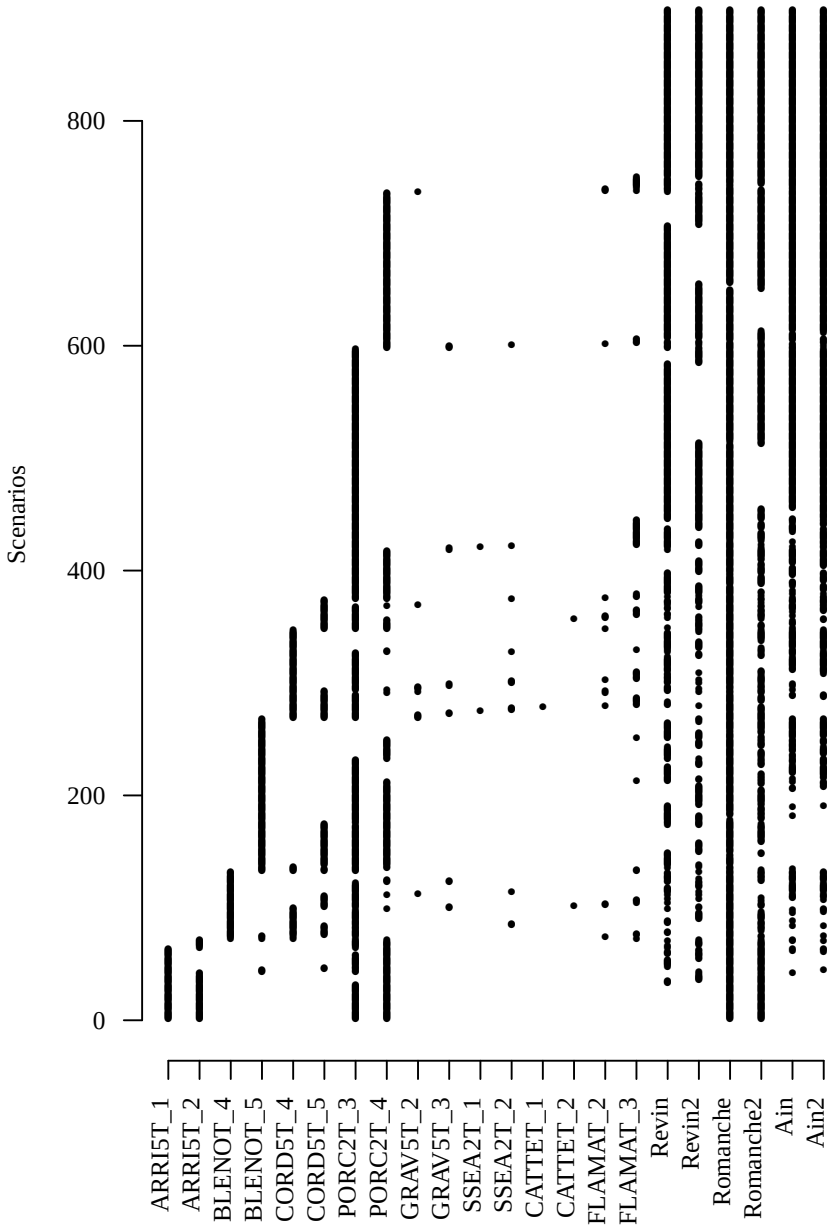


Figure 4.32: Illustration of the units which were adjusted in the simulations: one can verify that there are at most $K = 6$ black dot in each row. With this value for K, there is a trade-off between adjusting hydroelectric or thermal units. Nuclear units (from $GRAV5T_2$ to $FLAMAT_3$) are almost never adjusted. Among groups of units with the same operating characteristics, the cheapest ones are the most adjusted.

units having the highest number of positive votes. In this case it is equivalent to work in a regression setting, where the predictions lie in the $[0, 1]$ interval, and to adjust the units having the K highest values. Note that in the case of a vector of ones and zeros, minimizing the square loss is equivalent to minimizing the number of binary errors.

The experiment is structured as in Section 4.4, i.e. we use 5-fold cross-validation, and the parameters of the ensemble of trees are tuned by grid search performed using an inner two-fold cross-validation procedure. We build an ensemble of 100 trees. The best values of K and n_{min} are respectively 340 and 5, i.e. the same value as in the experiment of Section 4.4. We obtain the results illustrated on Figure 4.33. The intersection of the prediction with the adjustments of Figure 4.32 is shown on Figure 4.34, and the prediction error is illustrated on Figure 4.35. Figure 4.36 gives further insight on the repartition of the prediction error. The number of permutations (0 if the model makes no error, 6 if the model predicts incorrectly all the adjustments), is distributed according to the histogram of Figure 4.36a. The median number of permutations is equal to one and there are up to four permutations in a few cases. From Figure 4.36b, we can observe that the negative correlation coefficients correspond to the part of the correlation matrix relating valleys to thermal units. Permutations occur thus more often between thermal units and valleys than between thermal units (or between valleys) of similar characteristics.

4.5.2 Solving the simplified intra-day scheduling problem

Using the predictions of the adjustment indicators obtained in Section 4.5.1, we can now compute the precise adjustments by enforcing the selected subset of generation units which are allowed to be adjusted in order to satisfy the updated demand. The details of the post-processing step are provided in Appendix B.3.2. The computation time limit is set to 600 seconds and the tolerated optimality gap is set to 0.5 %.

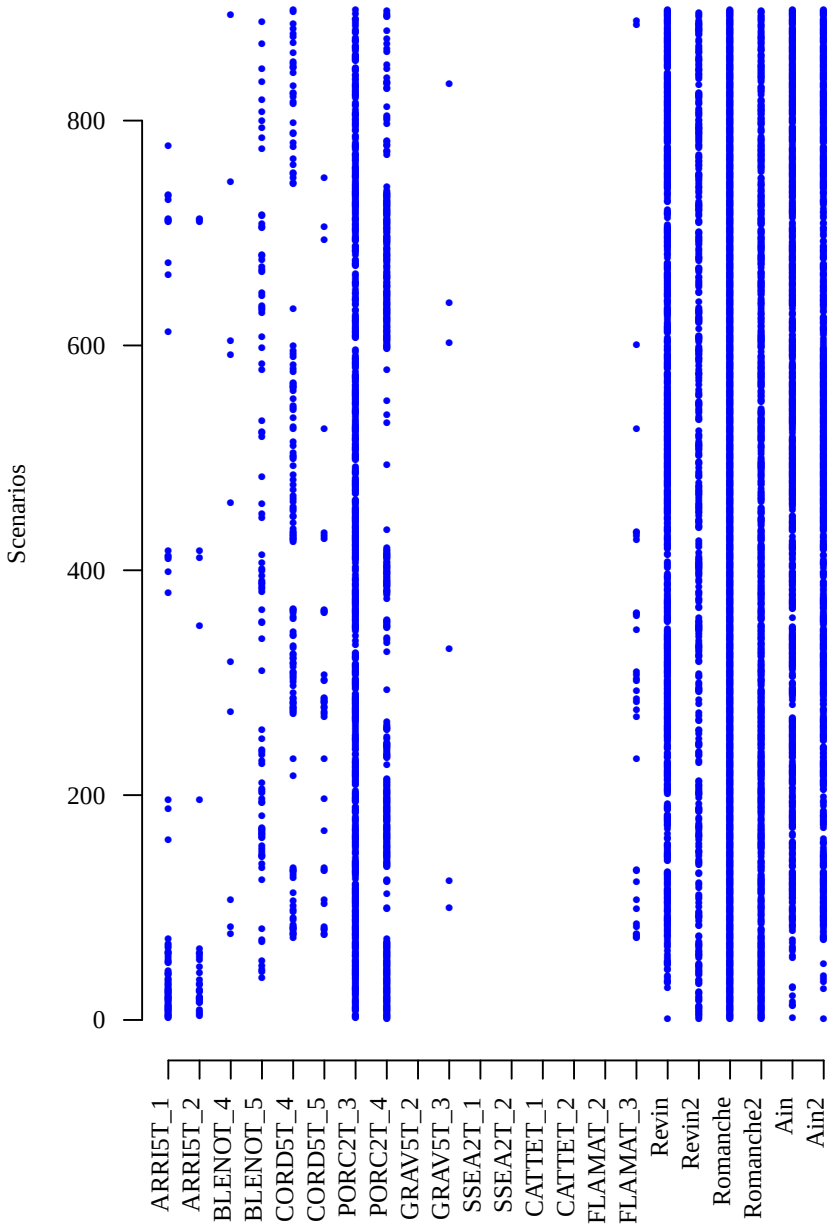


Figure 4.33: Prediction corresponding to Figure 4.32

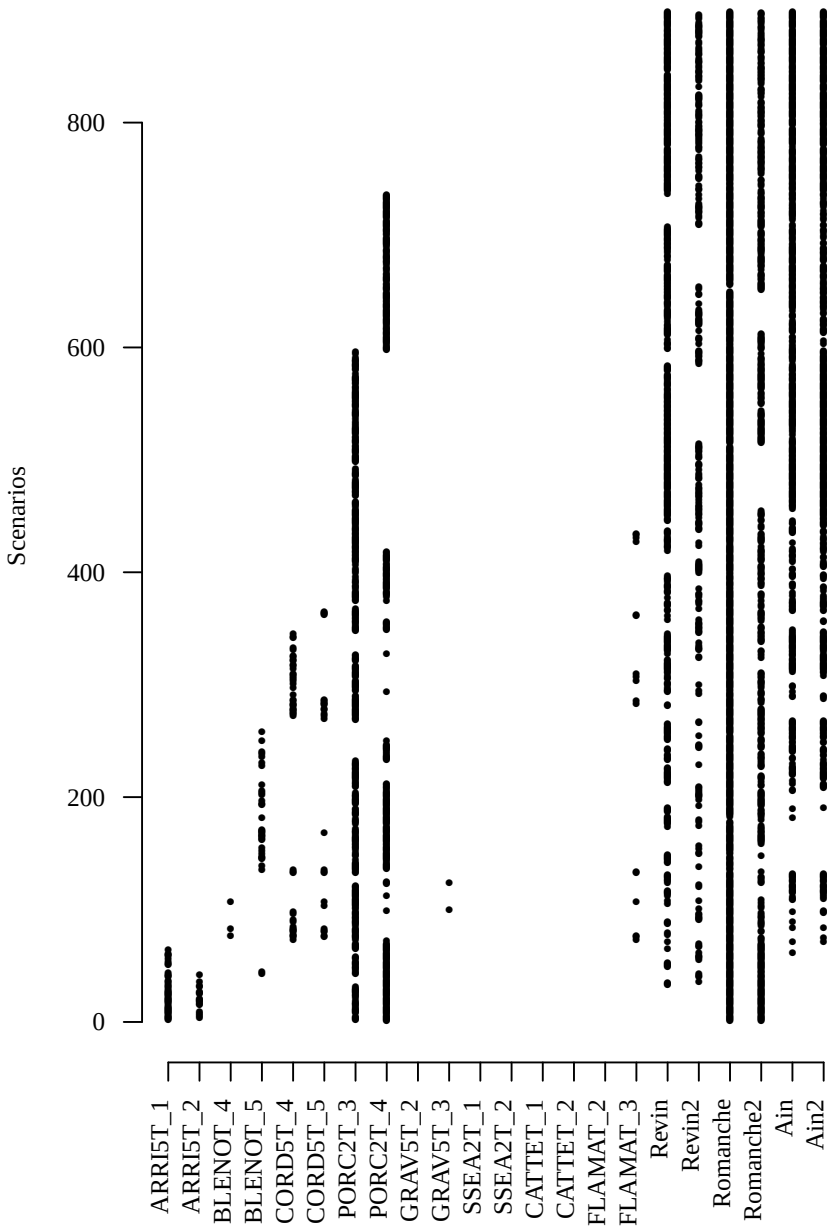


Figure 4.34: Intersection of the adjustments of Figure 4.32 and Figure 4.33.

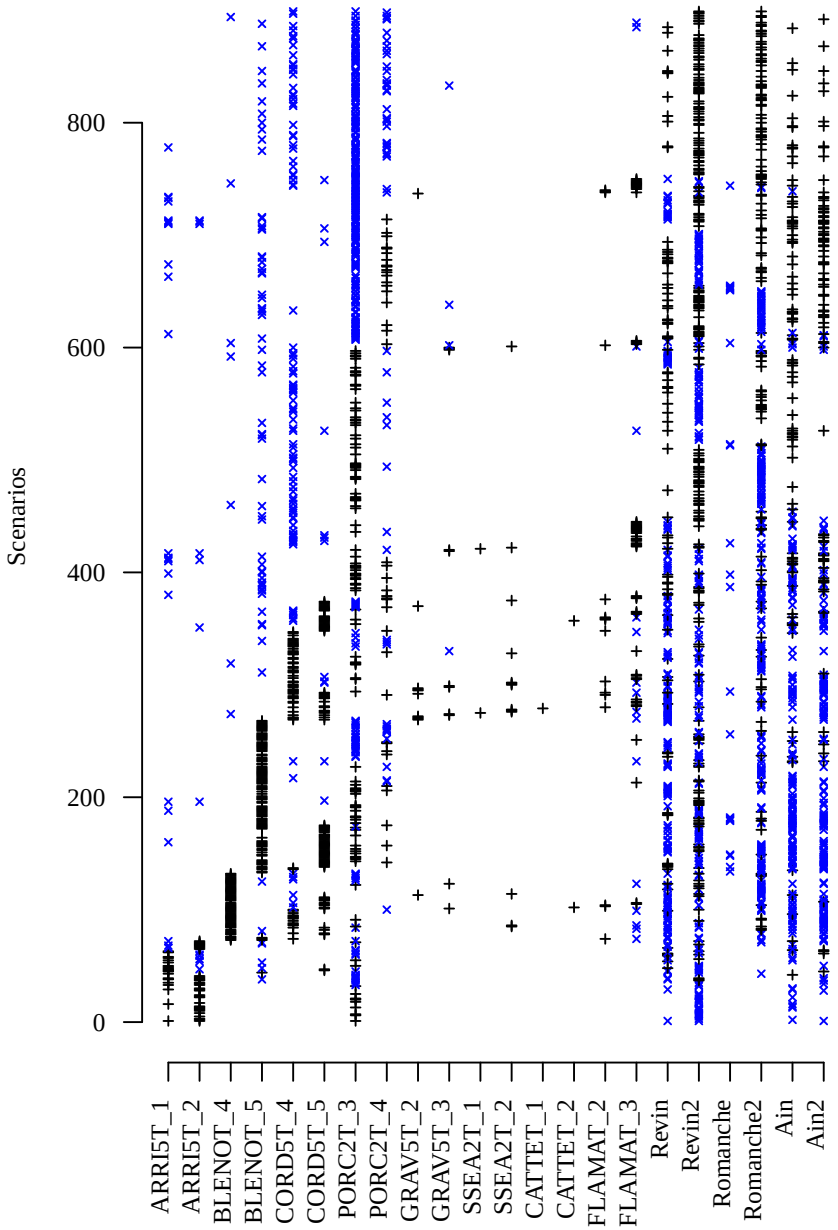
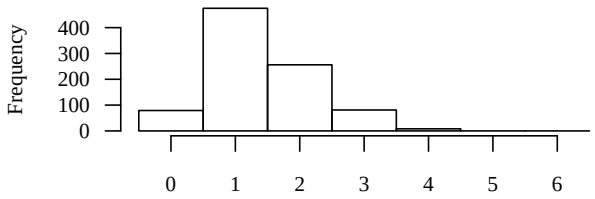
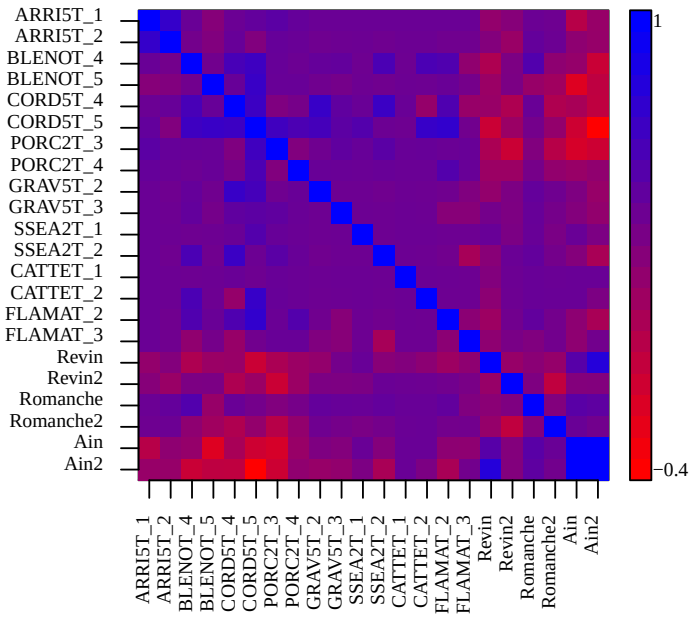


Figure 4.35: Difference of the adjustments of Figure 4.32 and Figure 4.33. Black + symbols correspond to adjustment predicted in Figure 4.32 and not in Figure 4.33, while blue × symbols correspond to the opposite.



(a) Histogram of the number of permutations.



(b) Correlation matrix of the errors between the generation units. Blue corresponds to positive correlation and red to negative correlation.

Figure 4.36: Analysis of the prediction error.

Optimization time and achieved optimality gap. In Figure 4.37 we compare the post-processing time to the time required for computing the optimal recourses and in Figure 4.38 the achieved optimality gap of these two alternatives. We can observe that both characteristics are dramatically decreased when using the adjustments predicted using our approach. The median re-optimization time drops form 600 seconds to 200 seconds and the tolerated gap of 0.5 % is achieved for more than 850 scenarios. However the gap does not say anything

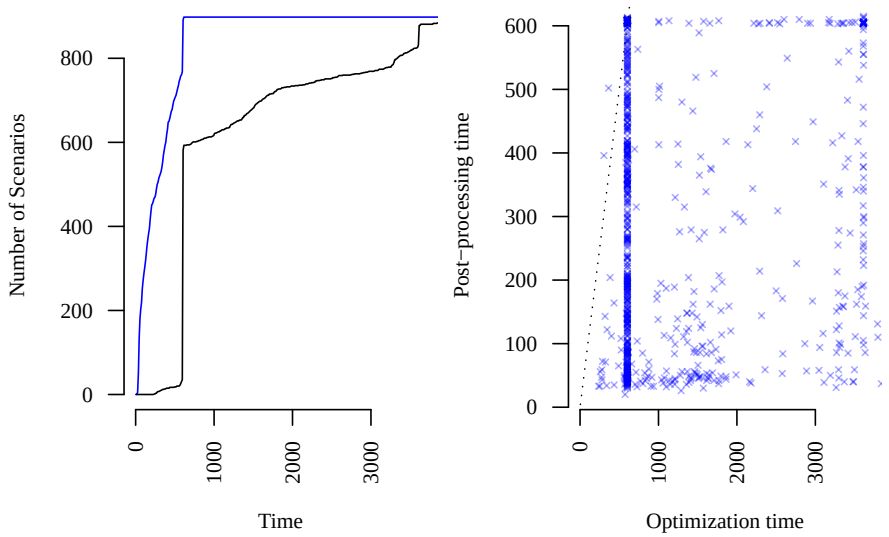


Figure 4.37: Processing time (in seconds) of our approach vs. time to reach the quasi-optimal solution, when using our approach to predict the units that need adjustments and re-optimizing these units only (blue) or computing the optimal schedule with the cardinality constraint (black).

about the cost of the adjusted schedule. This is analyzed in the next paragraph. Note that in Figure 4.37 the large increase of the black curve around 600 seconds is due to our arbitrary decision to constrain the allowed computation time to 600 seconds in the first run of the simulations. If this constraint were released, the black curve would most probably increase more smoothly, and the black curve of figure 4.38 would be closer to the blue curve.

Comparison of the costs. The repartition of the total costs between the different sources is distributed according to Figure 4.39. Compared to Figure 4.15 we see that the costs related to the non-satisfaction of the coupling constraints are much smaller. Thus the generation now satisfies the demand in most cases. Figure 4.40 shows the impact in term of total costs. On the top part of the figure we can observe that, with respect to the cost of the recourses optimized with the cardinality constraints (black dots), the cost of the post-processed schedules (blue $\times$ symbols) are sometimes almost equal, sometimes a bit more expensive, but more importantly also sometimes lower. This results from the

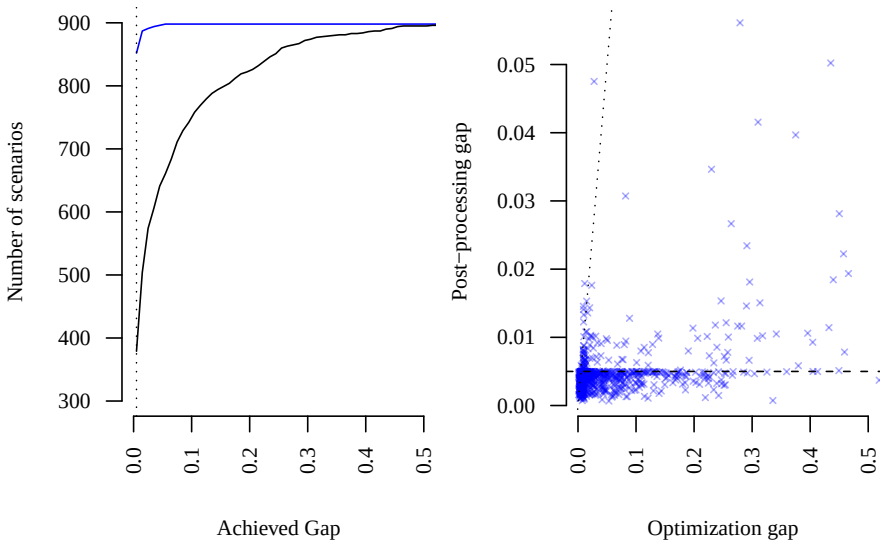


Figure 4.38: Gap of our approach vs. Gap of the optimal solution (in %), when using our approach to predict the units that need adjustments and re-optimizing these units only (blue) or computing the optimal schedule with the cardinality constraint (black).

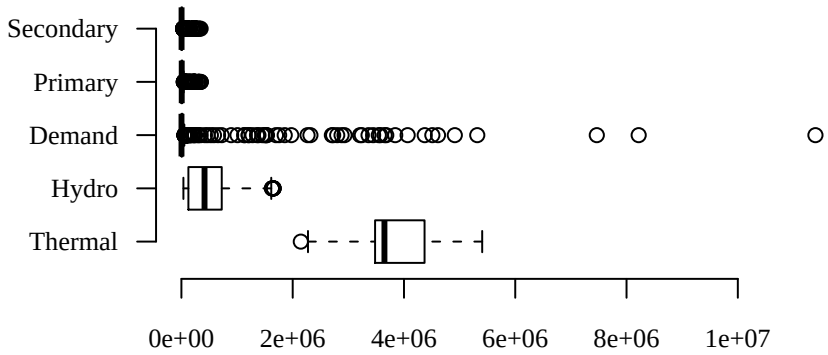


Figure 4.39: Repartition of the costs of the adjusted schedules when a limitation on the number of adjustments is imposed.

fact that the adjusted units are properly selected and that the optimization algorithm achieves a smaller integrality gap since the cardinality constraint is dropped. We can evaluate the proportions of scenarios falling in these three categories of relative cost thanks to the bottom part of Figure 4.40. Indeed there are about 750 schedules which are less or equally costly than the adjustments computed with the cardinality constraint, and only about 150 schedules which are slightly more expensive. Finally the green curve corresponds to the strategy consisting in adjusting exclusively the 6 hydroelectric valleys, since they are the most frequently adjusted units according to Figure 4.32. The left part of the latter curve is almost equivalent to the curve corresponding to the learned strategy. However there are many scenarios for which this strategy leads to a significant additional adjustment cost.

4.6 Summary

In this chapter we have applied the approach presented in Chapter 3 to learn a first recourse strategy to predict the power generation levels of all the thermal units and of all the valleys, and a second recourse strategy to predict some adjustment indicators. We described the generation system used for both experiments, the scenario generation procedure, and provided some insight on the results of two formulations of the rescheduling problem, differing by the presence or the absence of a constraint limiting the number of adjusted units. The inclusion of this constraint resulted in a high increase of the achieved optimally gap and of the computation time.

For the first recourse strategy, we analyzed the prediction error unit by unit then globally, the effect of the post-processing stage necessary to ensure the feasibility of the adjustments, and finally the impact in terms of adjustment costs. We also analyzed the effect of a variation of the number of scenarios available for learning, and the effect of the unavailability of an updated demand forecast. In terms of costs, the learned strategy is much closer to the strategy that computes quasi-optimal adjustments of the reference schedule to satisfy the updated forecast than from the strategy that applies the day-ahead schedule without modification and pays the penalties due to load-generation imbalances. In addition the learned strategy is far more efficient from a computation time point of view than the strategy that computes quasi-optimal adjustments of the reference schedule.

For the second recourse strategy, we used the adjustments rescheduled according to the formulation including the constraint on the limitation on the number of adjusted units. We analyzed the prediction error and compared the achieved optimality gap, computation time and costs to the quasi-optimal adjustments. We observed a significant improvement of these three aspects using the learned strategy.

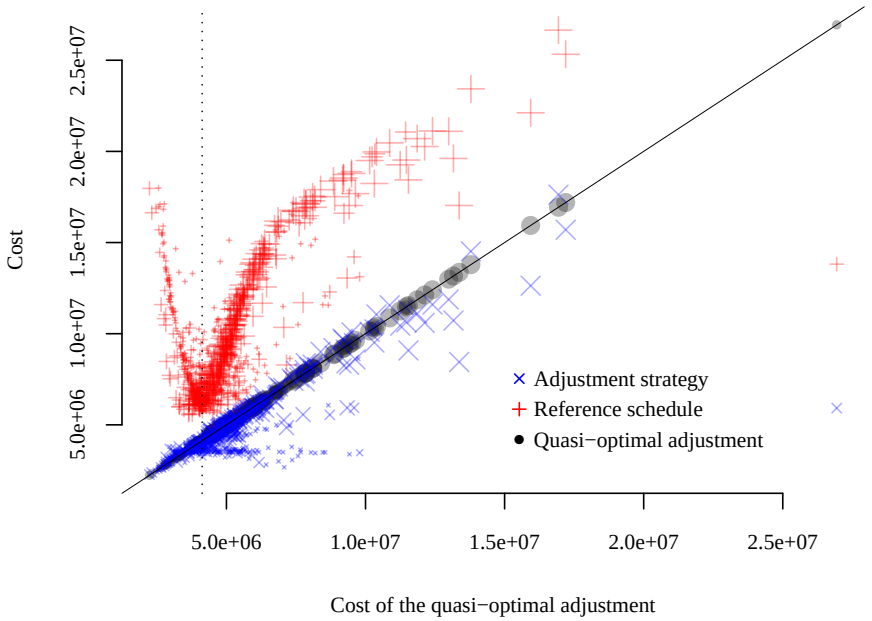


Figure 4.40: Scatter plots of the schedule cost during the recourse period vs. the cost of the planning quasi-optimally adjusted. The conventions for the color, the type and the size of the symbols are the same than in Figure 4.28.

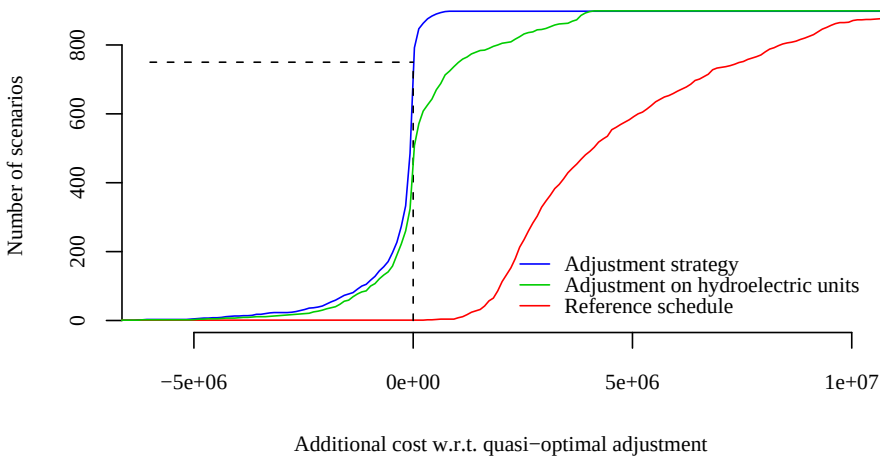


Figure 4.41: Cumulative histogram of the additional cost compared to the quasi-optimally adjusted schedules. The conventions for the color of the curves are the same than in Figure 4.29. The green curve corresponds to a strategy adjusting exclusively the hydroelectric part of the system.

Chapter 5

Conclusion of Part I

In this chapter we draw a first set of conclusions and discuss lines of further research related to Part I of this thesis. After a first section summarizing our work reported in chapters 2, 3, and 4, we discuss our findings along different topics, each time by making a critical analysis of our related results followed by a discussion of possible directions of further research. Notice that the general conclusions of our work are given in the last chapter of this manuscript, after having also presented our work of Part II.

5.1 Summary

The objective of Part I of this thesis was to propose, explain, motivate and validate an approach for supporting decision making under uncertainty in order to manage large electric power generation systems composed of a mixture of thermal and hydroelectric power plants of various types. Our approach uses

- Monte-Carlo simulations to generate scenarios of exogenous conditions under which the considered power generation system should be able to operate,
- state-of-the-art formulations of optimal control problems and their associated solution algorithms to pre-compute optimal open-loop generation schedules for each scenario,
- state-of-the-art machine learning methods to extract from the datasets so generated synthetic information which may be used to enhance the decision making process.

We have worked this out in the context of day-ahead and intra-day decision making, by proposing a method for pre-computing recourse strategies, from the situation anticipated during the previous day, that may be used by operators to manage demand variations or plant outages during the next day.

In chapter 2 we have precisely stated the problem that we want to tackle and discussed the limitations of problem solving strategies used in practice or proposed in the literature in this context, either based on "deterministic", "robust", or "stochastic" paradigms. From this analysis we have concluded that, although

built on solid theoretical foundations, classical optimization tools such as mixed integer linear programming solvers still suffer from their complexity when one wants to quickly infer solutions in real-time (e.g. for similar instances of a problem for which one has already solved one or a few instances) or when one wants to incorporate explicitly uncertainty on problem parameters for computing robust day-ahead decisions. On the other hand, scenario-tree based multistage stochastic programming problem instances of practical size are so complex that they are hardly solvable in the day-ahead time frame of generation scheduling, and even if this could be overcome, there would still remain the problem of computing in real-time the decisions of the second (and subsequent) stage(s). In addition, the scenario-tree based approach is so complex that it does not lend itself easily to a non-biased evaluation on independent scenarios (cf. Section 2.3.4).

In chapter 3 we have presented the main elements of the machine learning based approach that we propose to overcome these problems. Specifically, we have explained how to generate exogenous disturbance scenarios, how to compute scenario-wise adjusted generation schedules, how to apply supervised learning to obtain an approximate recourse strategy, how to restore feasibility of the decisions predicted by this recourse strategy in real-time, and how to evaluate a recourse strategy and its associated feasibility restoration mechanism on a set of independent scenarios. The proposed approach is intrinsically scalable to large-scale generation management problems, and may in principle handle many kinds of uncertainties and practical constraints beyond those that we have considered in our work. We believe that it could also be adapted to other contexts, such as medium-term and long-term generation management problems.

In chapter 4, we have proposed an implementation of the proposed approach and we have evaluated it on a realistically sized hydrothermal electric power generation system under two different settings.

- The first setting consists in learning intra-day recourse strategies predicting the detailed adjustment of all the generation units over the subsequent time-horizon. This learning problem is rather complex since its output space is very high-dimensional; on the other hand, it may be used for online decision making by combining it with a rather direct post-processing stage which merely aims at restoring, unit by unit, some feasibility constraints that would be violated by the predicted schedules.

- The second setting consists in using supervised learning in order to identify among the set of n_{gen} available generating units the best subset - of a priori given size $K < n_{gen}$ - of units to adjust. This leads to a much easier machine learning problem, but calls for a more complex post-processing stage, which has to compute the adjustment of the schedules of the K best units over the remaining time horizon.

Both approaches have been compared carefully on a realistic set of simulations. We found that both lead to promising results in terms of their ability in coping with uncertainties in intra-day operation. In addition, we have also illustrated in

Chapter 4 the versatility of machine learning approaches to extract information from datasets, by studying different combinations of input and output variables, showing its scalability to very high dimensional problems, and illustrating its capacity to identify problem parameters that have significant impact on the decision strategies computed ("important" features of the problem).

In the subsequent sections we discuss more in detail our conclusions and provide directions of further research that seem the most promising at this stage.

5.2 Machine learning problem formulations

Practically, a supervised learning problem is formulated by choosing its input variables (features) and the target output variable that one wants to predict. In the context of generation scheduling, the choice of the input variables concerns only the way we want to encode the information at hand when a decision has to be taken; on the other hand, the choice of the output variables determines more fundamentally the kind of information predicted and hence the kind of analysis and post-processing that is implied.

Below, we first compare the two machine learning formulations that we have investigated and then we discuss some further work about alternative formulations that could be interesting to investigate.

5.2.1 Comparison of the two evaluated formulations

Let us first summarize the results obtained with the formulations of Sections 4.4 and 4.5, which respectively aim at predicting the full generation re-schedule and at identifying a subset of size K of units to adjust.

5.2.1.1 Machine learning problem complexity

Clearly the learning problem of Section 4.4 is much more complex than the one of Section 4.5. On the same computer and using the same learning algorithm it takes about one hour to run the 5-fold CV in the first case and about 5 minutes in the second case. However as stated previously this is not really a limiting factor since the Extra-trees computation may easily be parallelized.

5.2.1.2 Predictive accuracy and optimality

In terms of predictive accuracy, the error seems large with the two formulations, since the output variables predicted by machine learning in the two formulations are quite often away from the optimal adjustments that would be taken in the case where full information would be available at the moment of recourse decision making.

However we should rather analyze how the learned recourse strategies (combined with their post-processing stage) actually influence the adjustment costs, in other words on the incurred cost over the horizon following the recourse decision step. From this point of view, we found that with the formulation of

Section 4.4 the decision making strategies obtained by machine learning are indeed significantly better than the passive adjustment strategy consisting of using the primary and secondary reserves and slack penalization only. We also found, in the context of the formulation of Section 4.5, that the decisions predicted by machine learning where actually often better than those that could be computed by direct optimization (assuming perfect knowledge) with a restricted computing time budget.

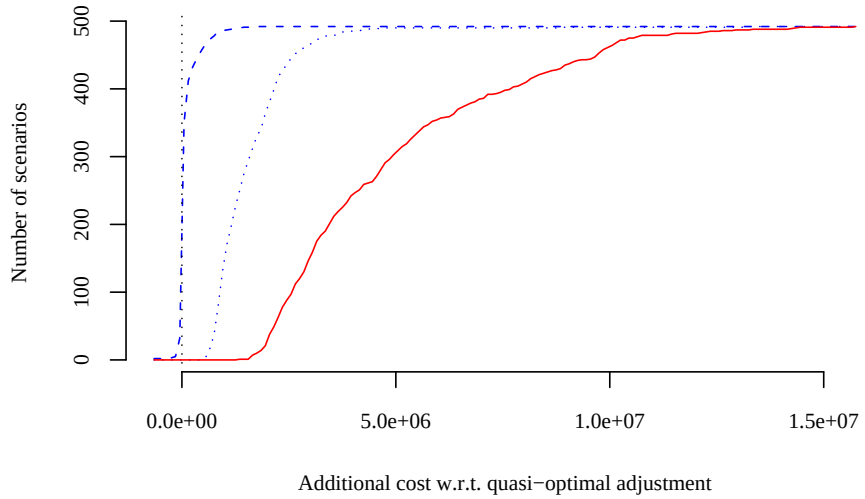


Figure 5.1: Comparison of the adjustment costs of the two formulations.

The adjustment costs of these two machine learning formulations are compared on Figure 5.1 for the common subsample of 500 scenarios used in Section 4.4. The red curve shows, for the sake of comparison, the cost increment with respect to a quasi-optimal adjustment without cardinality constraint when no recourses are computed (passive strategy). The dotted blue curve shows the cost increment incurred with the procedure of Section 4.4 while the blue dashed one shows this information for the procedure of Section 4.5. We see that the latter approach is significantly better in terms of optimality than the former. In addition, we observe that on a few scenarios the procedure of Section 4.5 with cardinality constraint provides actually better schedules than the full reoptimization without cardinality constraint.

5.2.1.3 Post-processing computing requirements

While the computing time for the dataset generation and the machine learning application are incurred offline (say, on the day before), the inference of a prediction and the post-processing needed to enforce constraints and/or determine precise schedules has to be carried out in real-time, just before the opportunity of recourse during the next day.

For this real-time aspect we can conclude that the approach of Section 4.4 clearly outperforms the approach of Section 4.5, since the post-processing com-

puting time of the former is about 10 times smaller than that of the latter. Furthermore, for the approach of Section 4.4 the post-processing lends itself to easy parallel computation and hence can be kept constant even if the system size is scaled up, while in the approach of Section 4.5 the complexity of post-processing will be superlinear with the limit K on the number of adjustable units, which shall inevitably grow with the size of the generation system (K=6 in our experiment, and typically $K = 30$ in the full system containing 150 generation units).

5.2.1.4 Interpretability and gaining of problem knowledge

From this point of view both approaches are essentially complementary, since the first approach allows to find out important factors influencing the precise generation schedules at different instants during the recourse period, while the second one allows to find out which factors have a strong influence on the ranking of each unit in terms of its efficiency to help during the recourse period.

5.2.2 Related work

Our experiments of Section 4.5 can be compared to experiments performed in Ben-Abbes et al. (2010). The authors of that paper draw similar conclusions to ours regarding speed and cost-related performance. The main differences with our work are as follows.

- The generation system of Ben-Abbes et al. (2010) contains 27 generation units (vs. 22 in our test system), but the constraint on the maximal number of allowed adjustments (K) is expressed using the same formulation as in our experiment.
- In Ben-Abbes et al. (2010), the authors impose that the rescheduling problem is solved to optimality when the learning samples are generated (which necessitates an average computing time of one hour for each sample). In our experiments we used "less-optimal" learning examples, and we observe that learned strategies are able to enhance some of the suboptimal schedules of the training set after post-processing.
- In the work of Ben-Abbes et al. (2010), scenarios are made of historical demand perturbations and they do not account for unit outages.
- In Ben-Abbes et al. (2010) individual classifiers are learned using a boosting algorithm (AdaBoost, cf. Freund and Schapire (1996)) and the paper compares the use of several types of weak learners. An interesting aspect is that they also analyze the use of some so called "règles métiers" representative of the rules that are nowadays used by human operators to solve the intra-day rescheduling problem.
- In Ben-Abbes et al. (2010), the cardinality constraint on the maximal number of adjusted units is kept in the post-processing stage when the number of predicted adjustments exceeds the value of K, while our

approach always fully identifies the K units to adjust, thus leading to a simpler optimization problem at the post-processing stage.

5.2.3 Further work

As mentioned in Section 3.4.2 there may be interesting learning formulations lying between the two formulations that we have considered in Sections 4.4 and 4.5. In addition there are also many ways to define the post-processing problem solved to obtain feasible adjustments.

5.2.3.1 Alternative post-processing problem formulations

In the experiments of Section 4.4 we have applied a post-processing that does not try to improve the demand versus generation satisfaction. We may imagine other post-processing formulations that dispatch the slack demand between the adjusted generation units, or which keep a coupling between the generation units and a term penalizing the demand discrepancy in the objective function. One could also keep only the integer part out of the post-processed adjustment obtained according to Section 3.4.2.3, and solve a linear programming version of Formulation 5 to satisfy the demand more accurately.

Again in the context of the experiment of Section 4.4, one could make a different use of the predicted adjustments.

1. Instead of implementing adjustments as obtained in Section 3.4.2.3, which necessitate anyway a post-processing stage, one can also think of generating a simplified instance of Formulation 5 by comparing the output of the recourse strategy to the reference schedule, identifying the regions where significant adjustment may occur, and authorize the modification of the decision variables only in these regions. This would thus result in an instance of Formulation 5 containing less variables and constraints.

2. One may also use the distribution of the predictions among the leaves of the trees in addition to their precise values, and penalize differently the modifications to the predicted schedule in the post-processing stage so as to allow more important modifications when for example the prediction has a higher residual variance.

5.2.3.2 Alternative machine learning problem formulations

In addition to the two formulations that we have investigated, one can imagine many other supervised learning formulations in order to extract useful information from the kind of datasets that we have generated. Such formulations would be obtained by choosing appropriate input features and target output variables. In particular, the following two alternatives seem promising.

1. Rather than learning recourse decisions for each generator, one could target the machine learning problem towards the prediction of the marginal cost curve at the optimum (cf. Appendix C.1.1). In the spirit of Lagrangian relaxation approach, these predicted marginal costs could then

be exploited to decouple the computations of the generation schedules during the post-processing stage. Notice that this idea could be exploited in either of the two formulations that we have studied in this thesis.

2. Another interesting possibility would be to use supervised learning to predict the overall adjustment cost for any scenario, based on features describing the scenario. A learned model which would be sufficiently accurate in this respect could then be exploited in various ways in order to speed-up Monte-Carlo simulations used in order to compare alternative reference schedules, or to compute simply a good estimate of the expected adjustment costs for the next day.

Anticipating on Part II of this thesis, let us already mention here that it is possible to extend the supervised learning framework in order to exploit so-called *censored* datasets where rather than having the exact value of the optimal cost for each scenario, one is given only (upper and/or lower) *bounds* on these costs. In our context, this would thus allow one to exploit datasets generated by using problem relaxations (lower bounds), or by halting the optimization procedure as soon as a given computing time budget is exhausted (upper bounds). This possibility clearly opens the way towards many different possible tradeoffs for combining machine learning and optimization.

5.3 Relation with two-stage stochastic programming

As exposed in Chapter 2, the two-stage scenario-based stochastic programming approach consists in optimizing jointly the first stage decision and the recourses, which leads to the statement of a huge optimization problem comprising decision variables and constraints for a set of scenarios and where the objective is to minimize the average cost over this set of scenarios. Moreover, non-anticipativity constraints need to be imposed in this framework in order to enforce a single first stage decision and to enforce recourse decisions which depend only on the information in $\xi_{[0:t_r-1]}$.

There are similarities and differences between the two-stage stochastic programming paradigm and our learning based approach.

- The aim of stochastic programming is mainly to optimize the first stage decision by hedging it against possible futures. However the number of scenarios that one is able to consider in practical formulations is strongly limited by computational complexity, and hence even if the first stage decisions computed by this approach are useful, the second stage decisions that are computed by it generally do not cover sufficiently well the range of possible future outcomes so that they are not usable in practice to actually take decisions at the second stage without having an additional mechanism to extrapolate them to unseen scenarios.

- On the other hand, the approach that we have exposed considers that the first stage, or reference schedule, is computed beforehand as it is actually the case in the practical context that we have considered (e.g. based

on a single scenario), and rather puts the emphasis on the computation of a practically applicable strategy for the second stage to adjust the reference schedule to the evolving system conditions in an optimal way. In other words, the computations of the first and second stage decisions are decoupled in our approach.

We believe that these two approaches are thus complementary rather than in competition. For example, to obtain a reference schedule allowing inexpensive and feasible adjustments to uncertainties, nothing prevents the use of a stochastic or a robust programming approach for computing the first stage and then to use it as the reference schedule of our approach. Also, the adjustment problems formulated to build datasets exploited by machine learning may themselves be replaced by an optimization over several scenarios in order to improve the robustness of the resulting adjustment decisions. Furthermore, the learned second stage strategies could be exploited in order to help computing the reference schedule.

5.3.1 Related work

Some work has already been done in order to approximate the solutions of a scenario tree based stochastic program in Defourny and Wehenkel (2007) and Defourny et al. (2009), who propose to use supervised learning to generalize the information contained in the solutions associated to a scenario tree. This work differs however from our work since we learn the recourse strategy from a set of single scenario-schedules, not a "schedule tree" corresponding to a scenario tree. Also, while in our work we have focused on the practical context of two-stage intra-day electric power re-scheduling, the work presented in Defourny (2010) focuses rather on the exploitation of supervised learning in order to leverage scenario-tree based multistage stochastic programming methods.

5.3.2 Further work

One could imagine to set up an iterative scheme in order to optimize jointly the first and second stages:

1. start with a given reference schedule (first stage) and a set of scenarios $\xi \in \Xi$, and compute a recourse strategy (second stage) by combining optimization with supervised learning (as in this work),

2. evaluate the combination of the current first and second stage decision making procedures, and stop if the evaluation is sufficiently good according to a criterion C_1, else

3. modify the reference schedule to improve its hedging, given the already computed second stage strategy,

4. compute adjustments of recourse decisions to the new reference schedule for the scenarios $\xi \in \Xi$, and learn a new recourse strategy; go to step 2.

Of course, the difficult parts are

1. to define a criterion C_1 to evaluate the necessity to further improve the first stage and adjust the second stage: we may for example define a threshold on the maximal admissible additional cost between predicted adjustments and quasi-optimal adjustments;

2. to modify in a productive and efficient way the reference schedule: we could for example search in the solution pool provided by the algorithm used to compute the original reference schedule (e.g. Branch-and-Cut);

3. and to assess the quality of the resulting policies at convergence of the overall approach given a particular way to implement steps 1 and 2.

5.4 Robustness to outliers

In the simulation phase we have paid attention to the design of scenarios reflecting as accurately as possible the disturbances impacting the system, since it may be unnecessary to compute adjustments for implausible scenarios, or even counterproductive depending on their influence on the learned policy. This is particularly true in a stochastic programming setting where the first stage must be 'good on average' for all scenarios. This is to be balanced by the fact that the average is weighted and that we may devote less importance to extreme scenarios by assigning them smaller weights. However it is hard to measure a priori which scenarios are actually extreme and how the weights assignment will impact the resulting decision policy.

5.4.1 Further work

Although we do not expect the learned strategy to generalize well to scenarios completely different from those used for training, our feeling is that our learning based approach is able to filter and/or specialize to outlying scenarios without degrading too much the whole adjustment strategy. For example in Figure 4.28, the adjustment cost for the rightmost scenarios remains close to the quasi-optimal adjustments but it does not seem to decrease the performance for scenarios closer to the reference scenario. Note that tree-based methods are well-known for their robustness to outliers (Breiman et al., 1984). However it would be of high interest to study in more details the robustness of our approach in this respect, e.g. by adding or removing several extreme scenarios and analyzing how the distribution of the adjustment costs evolve, so as to gain better insight on the role of the parameters of the learning algorithm in this respect.

5.5 Actively selecting the scenarios to simulate

In our approach we sequentially

1. generate a set of scenarios,

2. optimize the adjustments of the reference schedule for these scenarios,

3. compute a recourse strategy by combining offline determination of decision rules by supervised learning and online post-processing of their predictions by optimization.

We have analyzed the sensitivity of our method with respect to the size of the learning set in Section 4.4 and observed that decreasing the size of the learning set (400 scenarios) by a factor 2 does not degrade a lot the recourse strategy, while decreasing it by a factor 10 has a great influence although the resulting strategy is still better than applying the day ahead strategy without adjustment. This information may give some insight on the number of scenarios that one must generate for this problem, e.g. more than 200. But we have no clue to decide a priori of the number of scenarios to generate, and if we generate an insufficient number of scenarios, of the way to select new scenarios to improve the current strategy.

5.5.1 Further work

We may think about using the quasi-optimally adjusted schedule or the learned recourse strategy to guide the generation of new scenarios, thus to interleave the learning and the dataset generation procedures. As the dataset generation procedure is very costly in our problem (especially the optimization step), this could allow to achieve a better generalization error for a fixed size of the dataset (cf. Cohn et al. (1994)).

If we assume that optimally adjusting the generation schedule to an updated forecast is the best strategy, then one could iteratively

1. evaluate our learned adjustment strategy on the first sample by using cross-validation,

2. sample new points around the areas of the input space where the generalization errors are found to be the highest,

3. compute the optimal adjustments to the reference schedule for these new scenarios,

4. and restart the learning procedure by using the original sample completed by the new scenarios, until the coverage of the input space is satisfactory.

This topic is also related to Section 5.4 since an active learning approach may as well be able to identify scenarios that would degrade the overall performance of the learned recourse strategy. Hence, in addition to generating new scenarios, the approach could help us to throw out such “outliers”.

5.6 Evaluating a strategy in the face of uncertainty

As we do not explicitly take into account the costs of the adjusted schedules during the learning phase of our approach, but instead learn from schedules

having the lowest cost regarding the objective function of the optimization problem, it is hard to provide a bound on the performance for our learned strategy, although it is easy to evaluate empirically the range of costs incurred by our strategies (cf. Figure 4.29) and to compare them to another strategy. Remark that this is also true for stochastic programming based approaches reviewed in Section 2.3.4 which provide an idea of the performance of the hedged decisions for given scenario weights but nevertheless require a simulation phase to assess the value of the method, like in Carpentier et al. (1996).

In the experiments reported in Chapter 4 we have assessed several adjustment strategies assuming that the future behavior of the environment is perfectly predicted by the updated forecasts made at the recourse moment. We have compared under this assumption our recourses computed by the machine learning based procedure to the strategy consisting in re-optimizing the generation schedule deterministically given this forecast, thus putting the later in a situation of a priori being the best possible strategy (modulo the limitations of the optimization procedure).

In practice, however, forecasts re-estimated at the recourse moment are not perfect guesses of the subsequent environment evolution. Hence, the performance assessment of any strategy should take into account this residual uncertainty. Assuming that the further evolution of the system over the recourse interval is perfectly known at the moment of committing the recourse decisions leads indeed to optimistically biased performance assessments. Using our approach to obtain a recourse strategy, whether or not we consider the updated demand forecast as inputs of the problem, our aim is to minimize a generalization error, not the error on a particular scenario assumed to be known perfectly at the moment of taking the recourse decision.

We notice however that the evaluation bias introduced by the assumption of perfect prediction of environment behavior at the recourse moment was present in our evaluations of both the golden standard (quasi-optimal adjustment) and the strategies inferred by machine learning.

5.6.1 Further work

Ideally we should thus evaluate the expected performance on the set of possible futures $\xi_{[t_r:T-1]}$ given the observation of $\xi_{[0:t_r-1]}$. To this end, we may either

1. simulate the set of possible futures and evaluate the average cost of the different strategies on this set;

2. make use of historical data: if we dispose of the day ahead forecast, of the forecast updated at time t_r and of the realization, for several days we may be able to evaluate the value of different adjustment strategies by averaging the results of the strategies on the realization over the set of days considered.

Also, as we do not know the type of risk measure induced by the variance based criterion used to develop the regression trees of our models, i.e. which kind of objective function we do optimize by using this criterion, we believe

that some theoretical analyses of the relation between this criterion and actual performance would be of interest, possibly in order to develop better loss functions to be used at the learning stage.

5.7 Multiple recourse opportunities

Up to now we have considered a single intra-day recourse opportunity, while in practice there are several opportunities of recourse.

At the first opportunity of recourse the state of the system approximately corresponds to the state planned in the schedule computed one day ahead, modulo some non optimized primary and secondary control actions. This is no more true for the next recourse opportunities, since when a recourse opportunity at a later stage occurs some modifications have already (possibly) been made to the day ahead schedule via previous recourse stages and have (possibly) driven the system along a path very different from the path that the day ahead schedule would have yielded without recourse actions. This creates some uncertainty on the state of the system at the recourse opportunities of later stages.

In this setting, a recourse strategy $\pi_{[t+1:T-1]}(\xi_{[0:t]})$ allowing to adjust the generation schedule to the realized scenario up to time t must thus also take into account the effect on the state of the system at the recourse instant that could be implied by recourse decisions at previous stages ($t_r < t$).

Thus, a line of further research will be to investigate approaches allowing to exploit the supervised learning approach in a context where more than a single opportunity of recourse has to be prepared the day ahead. On the other hand, taking recourse actions at earlier time steps, should in this "multistage" context also take into account the effect of future recourses. This leads us to the classical, forward-backward types of approaches, where decisions are iteratively coordinated over the temporal horizon by temporal relaxation procedures, and iterations over the recourse opportunities.

How to appropriately adapt these "forward-backward" approaches in our machine learning based setting is an important line of future research.

5.8 Broader application contexts

Up to now we have considered that recourse decision strategies would be learned everyday for the next day, while in practice it seems also reasonable to consider problems where strategies would be learned in such a way that they are useful for more than a single day.

In order to obtain recourse strategies applicable several days, e.g. for all days of the same cluster as defined in Section 4.2.1, we could modify the dataset generation procedure (Section 4.2) by considering the set Ξ as a set of reference scenarios, thus compute a reference schedule for each of them, and generate adjustments between reference scenarios. The inputs of the learning problem should then be extended with a description of the reference scenarios and of the reference schedule.

Part II

Prior knowledge in supervised learning algorithms

Chapter 6

Preliminary remarks

The optimization formulations presented in Part I, that we use to generate our learning samples, are very complex because of the presence of many operating constraints requiring integer variables, and because the generation units are coupled. We have shown that a learning algorithm which is unaware of the constraints the data is subject to may indeed successfully capture the sensitivity of the solution to the model parameters. Nevertheless this raised our attention on one particular aspect of the relation between machine learning algorithms and optimization algorithms. When we apply a SL algorithm to search in a hypothesis space based on data that satisfies a known set of constraints, can we guarantee that the hypothesis that we select will make predictions that satisfy the constraints? Can we at least benefit of our knowledge of the constraints to eliminate some hypotheses while learning and thus hope that the selected hypothesis has a better generalization error?

We found the paper of Lauer and Bloch (2008b) while surveying the literature on these topics. This paper deals with the incorporation of prior knowledge in SVR. As we opted for tree-based methods in our experiments of Part I, we started working on the adaptation of the work of Lauer and Bloch (2008b) to tree-based models. We propose an optimization formulation to do so in Part II (Part II is to a large extent similar to Cornélusse et al. (2009a)). However we do not study the above questions in the setting of Part I, but from a more general perspective, and then experiment on problems of general interest in the machine learning community. In particular, we show how our formulation may be used to handle censored data and how we can transpose tree-based methods in a semi-supervised learning setting. We reconcile the two parts of this thesis in the concluding chapter when we discuss about future work.

Chapter 7

Regularizing tree-based supervised learning models using non-standard information

Tree-based ensemble methods can be seen as a way to learn a kernel from a sample of input-output pairs. This chapter proposes a regularization framework to incorporate non-standard information in the kernel learning algorithm. In particular, we want to take advantage of additional information, not presented in the form of input-output pairs, but for example as constraints on the value of the outputs at different places of the input space. To this end a generic convex optimization problem is formulated. In Chapter 8 it is then customized into a manifold regularization approach for semi-supervised learning, or as a way to exploit censored output values, or finally as a generic way to exploit prior information about the problem.

7.1 Motivation

In the standard setting, supervised learning aims at inferring a predictive model mapping an input space $\mathcal{X}$ to an output space $\mathcal{Y}$ given a completely labeled sample of input-output pairs. However, in many applications the available output information for a set of input points is incomplete. For example, in the setting of semi-supervised learning the output is simply unknown for a subset of the provided inputs, but under some assumptions taking into account these unlabeled inputs allows the induction of better predictive models. Between these extreme settings, in the context of censored data some outputs are only partially specified for a subset of the given inputs in the form of a range of possible values (e.g. typically a lower bound on the life-time, in the context of survival data). In other contexts, additional prior knowledge about the target problem is given in the form of hard or soft constraints. In all these cases, one would like to exploit all the available information together with the labeled sample of input-output pairs so as to infer better predictive models.

We propose a general framework for the regularization of tree based ensemble models. It exploits a kernel formulation of tree-based predictors and is formulated as a convex optimization problem where the incomplete data and/or prior knowledge is used as extra information to regularize the model.

Semi-supervised learning and learning from censored data fit naturally into this general framework. However, other kinds of information can be used, like prior information on the measurement accuracy on the outputs or specific constraints on output values which must be represented in the model. Relations between input-output pairs can also be imposed as well as some other kinds of structural properties about the problem.

Our convex optimization formulation is presented in Section 7.2, as well as the consequences in terms of problem complexity. Section 7.3 exposes the related work.

7.2 Regularizing an ensemble of regression trees

After an intuitive description of the nature of the problems addressed, the principles of the induction of ensembles of regression trees are recalled and their regularization is formulated as a convex optimization problem which is discussed in terms of modeling capacity and solution complexity.

7.2.1 Nature of the problem

We consider the supervised learning framework, where we typically seek to infer a function $f(\cdot) : \mathcal{X} \to \mathcal{Y}$ from a completely labeled training sample of input-output observations

$$\{(x_i, y_i)\}_{i=1}^{n}.$$

For convenience, we consider the context of regression, where $\mathcal{Y} \subseteq \mathbb{R}^q$, with $q = 1$ unless stated otherwise.

In many cases, additional information can be useful in this inference process. E.g., if for some points $\{(x_i, y_i)\}_{i=n+1}^{n+c}$ the output is censored, for example right censored, we would wish to regularize f such that $f(x_i) \geq y_i$, $\forall i \in \{n+1, ..., n+c\}$. In semi-supervised learning, we have a (typically large) number of input points $\{x_i\}_{i=n+1}^{n+u}$ without their associated outputs. We then would like to exploit regularity assumptions about the input-output relation to bias the learning of the mapping f from the labeled data. But it might also happen that the output targeted by the learning process is a priori known to satisfy constraints for some particular inputs, or even over some given regions of the input space[1] (e.g. "$f(x) \geq Ax + b,$ whenever $Bx \leq c$", cf. Mangasarian et al. (2004)). More generally, the additional information at hand might also entail more complex relations involving input-output pairs at several places (e.g. "$f(x_k) \geq f(x'_k), \forall k = 1, \ldots, K$"). Another example is in multiple or structured output prediction if individual models $f^j(\cdot)$ are fitted for each individual output y^j. E.g., if $\mathcal{Y} \subseteq \mathbb{R}^2$ then we may wish to couple the individual models to better respect the known structure of their output relations, so as to satisfy constraints such as "$f^1(x) \geq f^2(x)$".

[1] In what follows, (in)equality relations are to be understood as component-wise when they apply to vectors.

7.2.2 Tree-based ensemble methods

In this paper, we consider the incorporation of prior knowledge and incompletely labeled samples in the forms suggested in Section 7.2.1 into tree-based supervised learning methods[2].

The general idea of regression trees is to recursively split the training sample with tests based on the input space description x, trying at each split to reduce as much as possible the variation of the output y in the left and right subsamples of learning cases corresponding to that split. The splitting of a node is stopped when the output y is constant in the subsample of this node or when some other stopping criterion is met (e.g., the size of the local subsample is smaller than $n_{min} \in \mathbb{N}$). To each leaf a label is attached so as to minimize the empirical error, which in least squares regression trees is the local subsample average of outputs. Figure 7.1 illustrates a simple regression tree.

While useful for their interpretability, single trees are usually not competitive with other methods in terms of accuracy, essentially because of their high variance. Thus, ensemble methods have been proposed to reduce variance and thereby improve accuracy. In general, these methods induce an ensemble of M diverse trees and then combine their predictions to yield a final prediction as a weighted average of the predictions of the individual trees.

In the following, trees are indexed by the letter t, l_t is the number of leaves in tree t, $l_{t,i}(x)$ is the leaf indicator function[3] and $n_{t,i}$ is the number of training samples reaching leaf i. Denoting by

$$l_t(x) = \left(l_{t,1}(x)/\sqrt{n_{t,1}}, \ldots, l_{t,l_t}(x)/\sqrt{n_{t,l_t}}\right)^\top,$$

the vector of leaf indicator functions of the tree t, the prediction of the output of this tree for an input x may be written as

$$f_t(x) = \sum_{i=1}^{n} y_i l_t^\top(x_i) l_t(x).$$

This formula is indeed equivalent to computing the average label of the sample corresponding to the leaf reached by an input x. Considering an ensemble of M trees, its weighted average prediction

$$f(x) = \sum_{t=1}^{M} w_t f_t(x), \quad w_t \geq 0 \ \forall t \in \{1, \ldots, M\}, \ \sum_{t=1}^{M} w_t = 1$$

may hence also be represented as a scalar product based average. Namely, denoting by

$$l(x) = \left(w_1 l_1^\top(x), \ldots, w_M l_M^\top(x)\right)^\top \in \mathbb{R}^p, \ p = \sum_{t=1}^{M} l_t$$

we have that

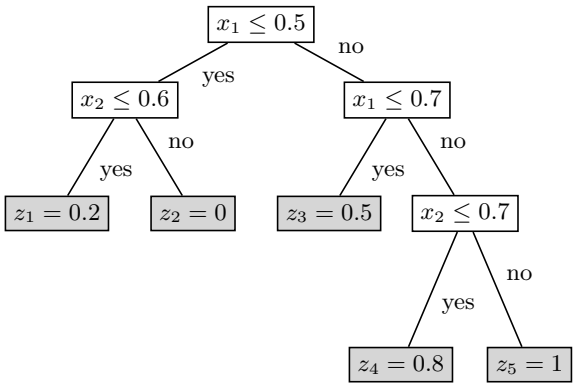
$$f(x) = \sum_{i=1}^{n} y_i l^\top(x_i) l(x).$$

[2]See Appendix A for an overview of tree-based SL methods.

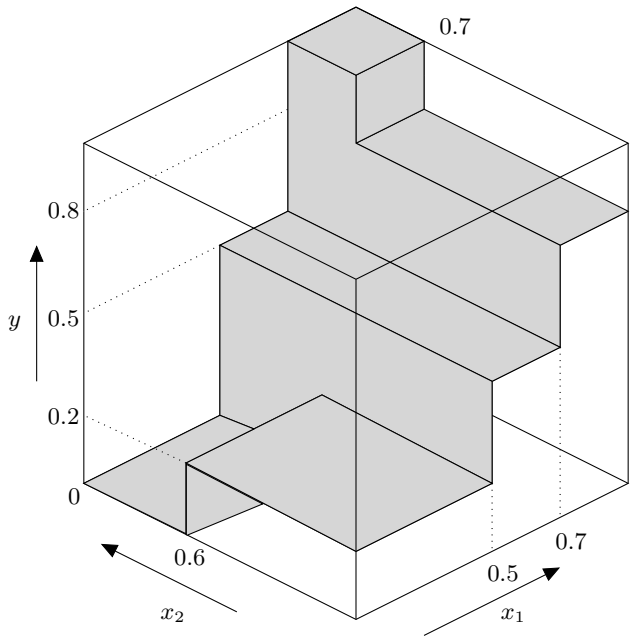
[3]$l_{t,i}(x) = 1$ if x reaches leaf i of tree t, $l_{t,i}(x) = 0$ otherwise.

Equivalently, if we define the kernel $K(x, x') = l^\top(x)l(x')$, thus determined by the structures of the M trees of the ensemble,

$$f(x) = \sum_{i=1}^{n} y_i K(x_i, x).$$



(a) A graphical model.



(b) The 3D corresponding representation.

Figure 7.1: A single tree example with $\mathcal{X} \in \mathbb{R}^2$ and $\mathcal{Y} \in \mathbb{R}$.

7.2.3 Regularization of a tree ensemble model

To incorporate the information contained in the incomplete data and/or prior knowledge, we must modify the model described in Section 7.2.2. In order to remain as generic as possible, we choose not to modify the tree induction algorithm and the way the kernel function K is computed (e.g. bagging (Breiman, 1996), random forests (Breiman, 2001), Extra-Trees (Geurts et al., 2006a), boosting (Freund and Schapire, 1996), etc). Thus we choose not to modify the structure of the trees, i.e. the way splits are selected at their internal nodes, but rather to modify the labels assigned to their leaves. This can be interpreted as a regularization of the model generated by the tree induction algorithm which assigns constant values to regions of the input space, by the correction of these assignments through the resolution of an optimization problem. To this end we consider two possibilities: to modify the vector y of training sample outputs and/or to add a bias to the labels attached to the leaves of the trees (cf. Figure 7.2). The first way allows to correct the y_i values when they are corrupted by noise and the second way arises when we do not want to modify these values.

To formulate the corresponding regularization problem, we introduce a vector of decision variables to denote the leaf biases $\Delta z \in \mathbb{R}^p$, a vector of modifications to the training sample outputs $\Delta y \in \mathbb{R}^n$ and a vector of auxiliary variables $\nu \in \mathbb{R}^n_+$, and we denote by $K \in \mathbb{R}^{n \times n}$ the gram matrix of the training sample, i.e. $K_{ij} = K(x_i, x_j)$, and by L the sample partitioning matrix of the ensemble:[4]

$$L = \begin{pmatrix} l^\top(x_1) & \dots & l^\top(x_n) \end{pmatrix}^\top \in \mathbb{R}^{n \times p}.$$

We also denote by $\Omega(\cdot, \cdot, \cdot) : \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}$ a convex function used to express generically various compromises in terms of regularization, and by $\mathcal{C} \subseteq \mathbb{R}^{n+p+n}$ a convex set used to express hard constraints. Given these notations, we formulate the optimization problem of Formulation 7.

Formulation 7: Tree-based ensemble model regularization.

$$\begin{aligned} \min \quad & \Omega(\Delta y, \Delta z, \nu) && (7.1) \\ \text{s.t.} \quad & -\nu \leq Ky + K\Delta y + L\Delta z - y \leq \nu && (7.2) \\ & (\Delta y, \Delta z, \nu) \in \mathcal{C}. && (7.3) \end{aligned}$$

The inequality constraints (7.2) aim at keeping the prediction error on the training sample low through the definition of the vector ν, which may be penalized in (7.1) and/or constrained in (7.3). The information of the incomplete data or prior knowledge may be expressed in the constraints (7.3) and in the objective (7.1).

[4] Up to some normalization, the line i of matrix L essentially indicates the leaves reached by the sample i.

In Formulation 7, we express the fact that we want to regularize the model by incorporating the information from the labeled training sample, the incomplete data and the prior knowledge, by assuming that these may be expressed by a finite number of constraints on the vectors $\Delta y, \Delta z$ and ν and/or by an appropriate choice of the objective function. In general, a trade-off must however be defined between the regularization induced by the prior knowledge and the incomplete data and the error on the labeled training sample. Notice also that without prior knowledge or incomplete data, this formulation allows to globally (re)optimize the leaf labels so as to minimize the error on the training sample without affecting too much the original model, in a way depending on the definition of Ω. In Chapter 8 we use this formulation to handle various problems and evaluate its interest.

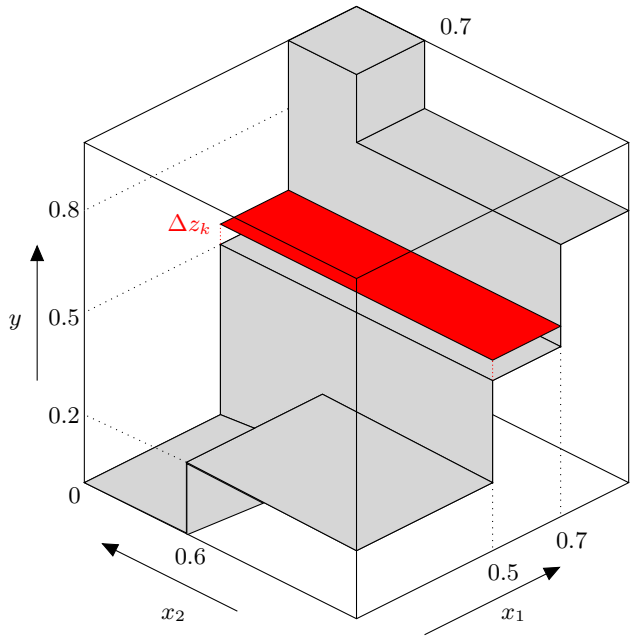
7.2.4 Problem dimensions and computational complexity

Formulation 7 contains $p + 2n$ variables, and $2n$ linear constraints, without taking into account the constraints defining $\mathcal{C}$. For a balanced tree t built from a finite sample size n, the number of leaves l_t is on the order of n/n_{min}, thus $p \approx Mn/n_{min}$. High dimensional problems formulated as LP can be solved in polynomial time. Anyway if the problem is nonlinear but convex it might still be solved in polynomial time. Depending on the complexity of the ensemble of trees on which the optimization problem is formulated, and on the parameter choices, the problem might not be feasible, or might be feasible only at the price of a significant increase of the error on the training sample. This would be the case if there were not enough degrees of freedom in the model to incorporate the incomplete data. A solution would be to penalize constraint (7.3) violations in (7.1).

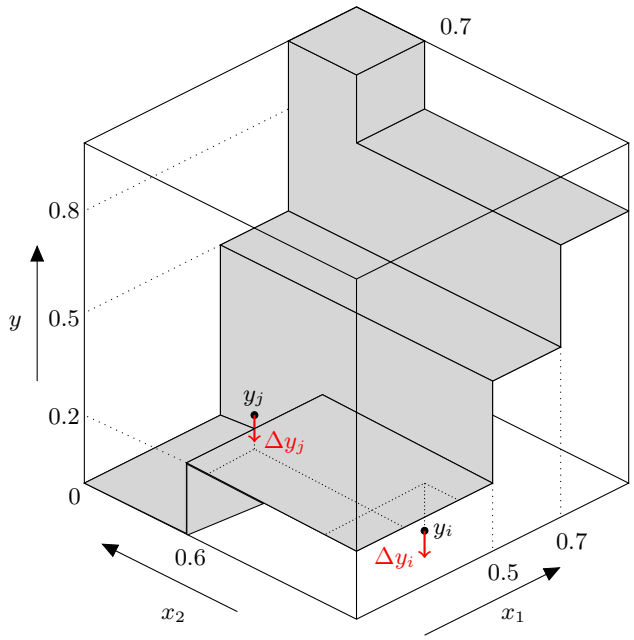
7.3 Related work

Many developments in supervised learning can be considered as the incorporation of (more or less explicit) constraints on the learned input-output map. Model regularization imposes global constraints on the smoothness of input-output maps and semi-supervised learning (Chapelle et al., 2006) imposes local constraints among the predictions at nearby samples derived from similarity measures.

We have focused on the regularization of tree-based models by the incorporation of incomplete data (and possibly other sources of additional prior knowledge about the problem) into predictive models. To the best of our knowledge, there is no related work using tree-based learning algorithms. In support vector machines, the explicit incorporation of constraints has already received a lot of attention (see Lauer and Bloch (2008a,b) and the references therein). The definition of the model derived from these methods as the solution of a convex (quadratic or linear) optimization problem indeed makes the incorporation of regularization terms and additional constraints natural. At first sight, one main advantage of these approaches with respect to tree-based ones is the simultaneous handling of both fitting the training data and satisfying the con-



(a) Adding a bias to the leaves.



(b) Modifying the output of learning samples.

Figure 7.2: Illustration of Formulation 7.

straints, whereas, in our case, the optimization only acts as a corrector for the tree-based learning algorithm. However, our Formulation 7 incorporates the quality of the fitting of the training data, meaning that it could be able to learn a useful model even if the initial tree model is not determined from the training data (but for example randomly built). Furthermore, the tree-based ensemble methods allow to learn a kernel over the input-space from the data in a supervised way, contrary to many approaches which assume that this kernel is given. Additionally, exploiting a learned tree model may take benefit of the main advantages of tree-based methods, such as for example their embedded feature selection mechanism.

Chapter 8

Applications and experimental results

In this chapter we show how the generic reformulation proposed in Chapter 7 may be applied to various problems, namely problems with censored data, semi-supervised learning problems, the problem of predicting the output of a system when we know some of its equilibrium points and a learning problem with an order relation between the outputs.

8.1 Censored data

Censored data arise frequently, e.g. in survival analysis where one is interested in the survival time of patients after the inception of a treatment. In this context, it often arises that people leave the study at a given instant t_0 for reasons independent of the disease, i.e. their survival time is only known to be larger than t_0. Censored data may also appear when a suboptimal solution algorithm is used to compute optimal output labels. For example, in relation with Part I, if we want to predict the cost of an adjusted schedule, we may account for the fact that the optimization algorithm provides us only an upper bound on the cost. Here we try to use this partial information, as Shivaswamy et al. (2007) did using support vector regression for censored data (SVCR). To this end, we learn a tree-based kernel function (using the Extra-Trees algorithm (cf. Appendix A.2) and denoted by ET in the sequel) K on the subset of uncensored data $\{(x_i, y_i)\}_{i=1}^{n}$ and then impose the information contained in the censored data $\{(x_i, y_i)\}_{i=n+1}^{n+c}$ thanks to Formulation 8, a particular case of Formulation 7, where y^c denotes the vector of censored outputs, $\nu^c \in \mathbb{R}_+^c$

Formulation 8: Regularization of a model built from censored data.

$$\begin{aligned}
\min \quad & C_1\|\Delta y\| + C_2\|\Delta z\| + C_3\|\nu\| + C_4\|\nu^c\| && (8.1)\\
\text{s.t.} \quad & -\nu \leq Ky + K\Delta y + L\Delta z - y \leq \nu && (8.2)\\
& -\nu^c \leq K^c y + K^c\Delta y + L^c\Delta z - y^c, && (8.3)
\end{aligned}$$

is a vector of auxiliary variables, $K^c \in \mathbb{R}^{c\times n}$ with $K^c_{ij} = K(x_i, x_j)$, $\forall i \in \{n+1,\dots,n+c\}$ and $\forall j \in \{1,\dots,n\}$, and

$$L^c = \begin{pmatrix} l(x_{n+1}) & \dots & l(x_{n+c}) \end{pmatrix}^\top \in \mathbb{R}^{c\times p}.$$

Constraints (8.3) with the term $C_4\|\nu^c\|$ of Ω imply that for censored objects an excessive prediction is not penalized. We did not use hard constraints here for the censored data for feasibility reasons. Since in survival data the sample outputs are in principle measured with high accuracy, we penalized $\|\Delta y\|$ very strongly, but we could as well have removed these variables from the formulation.

We compared this approach (Table 8.1) to unregularized tree-based ensemble methods (denoted by ET and ET*) and to the SVCR algorithm presented in Shivaswamy et al. (2007). We analyzed the four real life data sets from the R package "survival" on which SVCR is tested in Shivaswamy et al. (2007), and evaluated the error measure

$$MAE = \frac{1}{n+c}\left(\sum_{i=1}^{n} |y_i - f(x_i)| + \sum_{i=n+1}^{n+c} \max(0, y_i - f(x_i))\right)$$

by 5-fold cross-validation. ET was used to compute the gram matrix K of formulation 8. In ET* the censored points are included in the training sample and handled as uncensored points. For SVCR we used a Gaussian kernel. The parameters of these methods are tuned by grid search while using only the training samples. We observe that exploiting the censored data via Formulation 8 may indeed improve significantly the quality of the predictors. This is especially remarkable in the "nwtco" data set where the proportion of censored outputs is very high. We also notice that using regularized tree-based predictors actually outperforms, sometimes quite strongly, the SVCR approach to these problems.

Table 8.1: Comparison of unregularized/regularized tree-based predictors with SVCR (first row gives the percentage of censored data in each data set). The values reported are the average MAE over the 5 folds ± one unit standard deviation; † indicates that SVCR scores are reproduced from Shivaswamy et al. (2007).

	lung (28%)	veteran (7%)	heart (56%)	nwtco (86%)
ET	144 ± 11	85 ± 22	146 ± 57	1888 ± 72
ET*	117 ± 17	84 ± 19	78 ± 13	224 ± 38
ET + Form. 8	113 ± 13	81 ± 34	68 ± 23	98 ± 13
SVCR	144 ± 14	80 ± 28	138 ± 48	476†

8.2 Manifold regularization for semi supervised learning

In this section we show how the method presented in Belkin et al. (2006) may be casted in our formulation to yield semi-supervised and/or transductive

tree learning algorithms. We denote by $\{x_i\}_{i=1}^{n+m}$ the inputs of the data sample, and suppose that we have access to the output labels $\{y_i\}_{i=1}^{n}$ only for the first n of them. In Belkin et al. (2006) a similarity graph $\mathcal{G}$ is inferred from the inputs $\{x_i\}_{i=1}^{n+m}$ (see Luxburg (2007) for a tutorial on this topic):

- inputs $\{x_i\}_{i=1}^{n+m}$ are vertices of the graph,
- the graph $\mathcal{G}$ connects inputs pairs (x_i, x_j) which are similar according to a similarity measure $s(x_i, x_j)$ (e.g. a Gaussian kernel, $\exp(-\|x_i - x_j\|^2/2\sigma^2)$) or close according to a distance measure $d(x_i, x_j)$ (e.g. the euclidian distance).

For instance one can connect the inputs using a k nearest neighbors criterion. The adjacency matrix W of $\mathcal{G}$ is a matrix of dimension $(n+m) \times (n+m)$ having elements w_{ij} defined as follows:

$$w_{ij} = \begin{cases} 1 & \text{if inputs } x_i \text{ and } x_j \text{ are connected,} \\ 0 & \text{otherwise.} \end{cases}$$

Weights may also take non-negative real values (e.g. if the Gaussian kernel is used as similarity measure). We denote by $\mathcal{L}$ a Laplacian matrix of $\mathcal{G}$ over the whole sample. For instance, we may consider the unnormalized graph Laplacian matrix defined as $\mathcal{L} = D - W$, where D is the degree matrix of $\mathcal{G}$, i.e. the matrix having as components $d_{ii} = \sum_j w_{ij}$ and $d_{ij} = 0$ if $i \neq j$.

In essence, one of the methods presented in Belkin et al. (2006) can be described as a SVR formulation (cf. Appendix A.3) in which the objective function includes an additional regularization term based on the graph Laplacian matrix. This additional term penalizes the hypotheses which assign very different outputs to some pairs of inputs although these inputs are very similar according to the similarity graph.

To adapt Formulation 7 to this setting, we suppose that we have an ensemble of M tree structures and the Laplacian matrix $\mathcal{L}$ of a similarity graph computed over the input samples. Our objective is to compute ensemble predictions over the complete (labeled and unlabeled) sample that are "regular" with respect to the similarity graph. To this end we exploit the full set of inputs $\{x_i\}_{i=1}^{n+m}$ in each tree $t \in \{1, \ldots, M\}$ to compute the values $n_{t,i}$, $\forall i = 1, \ldots, l_t$ so as to define the ensemble kernel $K'(\cdot, \cdot)$, from which we compute the gram matrix $K' \in \mathbb{R}^{(n+m)\times(n+m)}$ over $\{x_i\}_{i=1}^{n+m}$:

$$K' = \begin{pmatrix} K'_{\ell\ell} & K'_{\ell u} \\ K'_{u\ell} & K'_{uu} \end{pmatrix},$$

where $K'_{\ell\ell}$ $K'_{\ell u}$ and K'_{uu} are submatrices corresponding respectively to the kernel evaluations between labeled, between labeled and unlabeled, and between unlabeled cases. We use Formulation 9 to compute predictions regularized along the similarity graph. In this formulation $y = (y_\ell^\top, y_u^\top)^\top \in \mathbb{R}^{n+m}$ denotes a vector of output labels obtained by completing the n given labels with m labels equal to zero ($y_u = 0$), and $\Delta y = (\Delta y_\ell^\top, \Delta y_u^\top)^\top$.

Note that, in this formulation, we do not allow adjusting the given output labels (8.6) nor use leaf biases Δz, but in (8.5) the outputs Δy_u contribute to the predictions for the labeled sample $\{x_i\}_{i=1}^{n}$.

Formulation 9: Manifold regularization for semi-supervised learning.

$$\begin{aligned} \min \quad & \|\nu\|_2^2 + C(y+\Delta y)^\top K' \mathcal{L} K' (y+\Delta y) && (8.4)\\ \text{s.t.} \quad & -\nu \leq (K'_{\ell\ell} | K'_{\ell u})\,(y+\Delta y) - y_\ell \leq \nu && (8.5)\\ & \Delta y_\ell = 0, && (8.6) \end{aligned}$$

8.3 Other types of prior knowledge and objectives

Obviously, the formulations 8 and 9 could be merged to handle both types of data in a common formulation. However, Formulation 7 is not limited to the incorporation of incomplete data.

Imposing the equilibrium points of a system. For example, to approximate the dynamics of a nonlinear system with known equilibrium points $(x_j^\star, y_j^\star)\ \forall j \in \mathcal{J}$, it is possible to force the value $f(x)$ for these points by expressing (7.3) as

$$\sum_{i=1}^{n} K(x_i, x_j^\star)(y_i + \Delta y_i) + l(x_j^\star)^\top \Delta z = y_j^\star,\ \forall j \in \mathcal{J}. \tag{8.7}$$

We have observed experimentally that adding such constraints may significantly enhance the accuracy of the inferred model.

Imposing constraints between outputs. We have also used our framework to incorporate other types of prior knowledge, such as relations between different output space dimensions. We consider the problem of simultaneously approximating two scalar functions while knowing that one is greater than the other. As in Lauer and Bloch (2008b), the functions $f_1(x) = \sin(x)$ and $f_2(x) = \sin(x)+0.1$ are approximated from noisy data: they are measured independently with an additive measurement error $\omega \sim \mathcal{N}(0, 0.04)$, for 70 random samples uniformly distributed in the interval $[0, 2\pi]$. We obtain the datasets $\{(x_i, y_i^{(1)})\}_{i=1}^{70}$ and $\{(x_i, y_i^{(2)})\}_{i=1}^{70}$ for functions f_1 and f_2 respectively.

We first learn one model for each function, with 50 trees and $n_{min} = 10$, to obtain the two tree kernels $K^{(1)}$ and $K^{(2)}$ (Figure 8.1(a)).

Then we couple the models using formulation 10 to impose $f_2(x_i) \geq f_1(x_i)$, $\forall i \in \{1, \ldots, n\}$. In that setting we decide to modify only the leaf labels of the datasets. We obtain the results depicted on Figure 8.1(b) when $C = 1$, $p_1 = 2$ and $p_2 = 1$. Note that using the ℓ_1-norm for the training set outputs modification leads to sparse modifications localized near the crossings of the two functions in Figure 8.1(a), in order to satisfy the constraints, as illustrated on the Figure 8.1(b).

Figure 8.1(c) illustrates the use of the additional knowledge $f_2(x_i) = f_1(x_i) + \delta$, $\forall i \in \{1, \ldots, n\}$ and $\delta \geq 0$ in replacement of (8.11), with $p_1 = p_2 = 2$. Table 8.2 reports the mean square error of the three experiments evaluated with

Formulation 10: Multi-outputs.

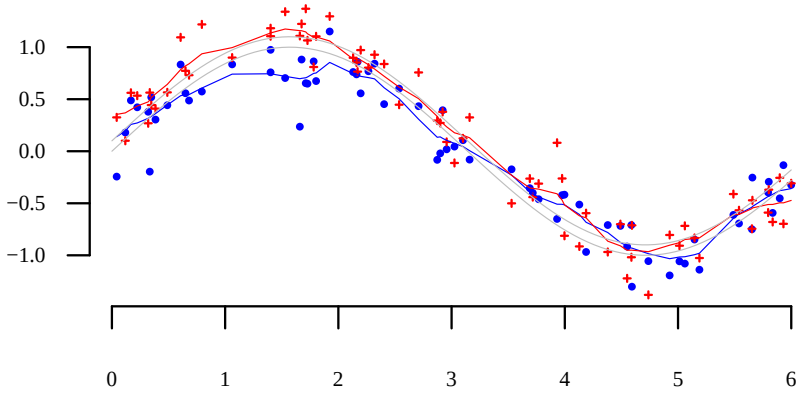
$$\begin{aligned}
\min \quad & \left(\|\nu^{(1)}\|_{p_1} + \|\nu^{(2)}\|_{p_1}\right) + C\left(\|\Delta y^{(1)}\|_{p_2} + \|\Delta y^{(2)}\|_{p_2}\right) && (8.8)\\
\text{s.t.} \quad & -\nu^{(1)} \leq K^{(1)}y^{(1)} + K^{(1)}\Delta y^{(1)} - y^{(1)} \leq \nu^{(1)} && (8.9)\\
& -\nu^{(2)} \leq K^{(2)}y^{(2)} + K^{(2)}\Delta y^{(2)} - y^{(2)} \leq \nu^{(2)} && (8.10)\\
& K^{(1)}y^{(1)} + K^{(1)}\Delta y^{(1)} \leq K^{(2)}y^{(2)} + K^{(2)}\Delta y^{(2)} && (8.11)
\end{aligned}$$

respect to the noiseless functions on a test set of 800 points uniformly distributed in $[0, 2\pi]$.

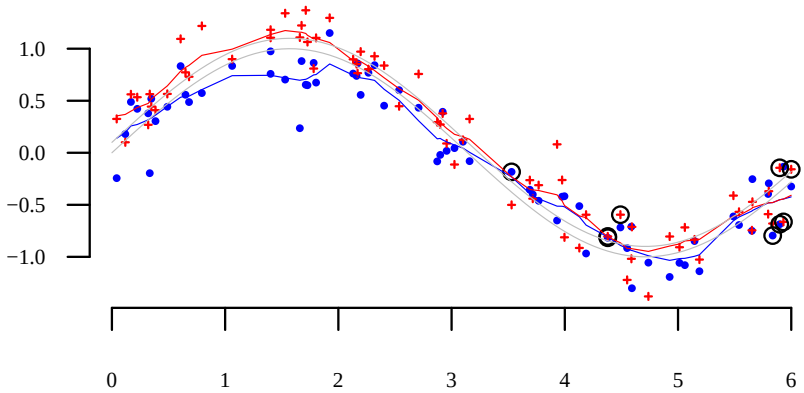
Table 8.2: Mean square error evaluated with respect to the noiseless functions for the multi-outputs problem.

	Figure 8.1(a)	Figure 8.1(b)	Figure 8.1(c)
$\hat{f}_1(x)$	0.020	0.021	0.013
$\hat{f}_2(x)$	0.015	0.012	0.012

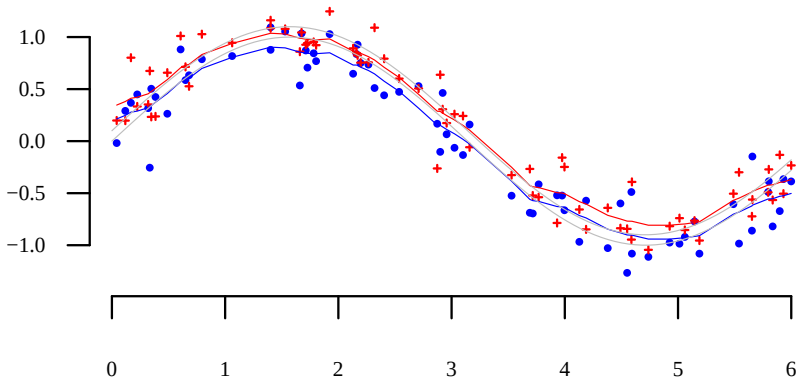
Accounting for measurement error. Also, ϵ-insensitive formulations may be handled by the incorporation of appropriate constraints in the set $\mathcal{C}$. These may be used to inject prior knowledge about the accuracy of the sensors used to measure the output values or to trade-off empirical accuracy with generalization performance.



(a) No prior knowledge.



(b) Imposing $\hat{f}_2(x_i) \geq \hat{f}_1(x_i)$, $\forall i \in \{1, \ldots, n\}$, highlighting using circles the modifications operated on the sample outputs.



(c) Imposing $\hat{f}_2(x_i) = \hat{f}_1(x_i) + \delta$, $\delta \geq 0$, $\forall i \in \{1, \ldots, n\}$. Nearly all the sample outputs are modified, and at the optimum $\delta = 0.133$.

Figure 8.1: The learned functions $\hat{f}_1(x)$ (blue curve) and $\hat{f}_2(x)$ (red curve). The training samples are indicated by blue • ($\sin(x)+\omega$) and red + ($\sin(x)+0.1+\omega$) symbols. The true functions are plotted in gray.

Chapter 9

Conclusion of Part II

We have proposed a generic extension of tree-based ensemble methods which allows to incorporate incomplete data but also prior knowledge about the problem. The framework is based on a convex optimization problem allowing to regularize a tree based ensemble model by adjusting either (or both) the labels attached to the leaves of an ensemble of regression trees or the outputs of the observations of a training sample. It allows to incorporate weak additional information in the form of partial information about output labels (like in censored data or semi-supervised learning) or – more generally – to cope with observations of varying degree of precision, or strong priors in the form of structural knowledge about the sought model. In addition to enhancing the precision by exploiting additional information, the proposed approach may be used to produce models which naturally comply with feasibility constraints which need to be satisfied in many practical decision making problems, especially in contexts where the output space is of high-dimension and/or structured by invariances, symmetries and other kinds of constraints.

Further work will aim at validating these ideas on practical problems and incorporating them within the algorithms used to grow ensembles of trees.

Chapter 10

General conclusion

We first summarize the content of the two parts of this thesis. Then we discuss some lines of further research that may benefit of the contributions of both parts.

10.1 Summary

In this thesis we were interested in the problem of (re)scheduling large electric power generation systems composed of a mixture of thermal and hydroelectric power plants of various types over a short-term horizon.

In Part I, we proposed a way to decouple the day-ahead and intra-day problems and to compute a practically usable recourse strategy for the intra-day rescheduling problem based on (1) the simulation of disturbance scenarios under which the power generation system should be able to operate, (2) the computation of quasi-optimal adjustment of the day-ahead schedule to these scenarios and (3) the application of SL algorithms to these scenario-adjustment pairs. Although it has some practical and conceptual advantages over classical techniques (mixed integer programming algorithms, stochastic programming) as exposed in Chapter 5, a recourse strategy so obtained does not necessarily output feasible decisions (in terms of operating constraints of the generation units) for each possible disturbance scenario. Depending on the type of output we have chosen to predict, we proposed to restore the feasibility of the predictions thanks to a post-processing stage which either projects the predictions on their associated feasibility region or consists in solving a downsized (and slightly different) instance of the original scheduling problem. Indeed the recourse strategies obtained in our experiments were only slightly modified by the post-processing procedure, and look promising in terms of rescheduling costs and response time. We have proposed several lines of further research for this approach in Chapter 5. In particular,

- our approach offers many alternatives for the choice of the output space of the recourse strategy and also for the post-processing stage, and we believe that exploring those alternatives may yield interesting solutions in-between the two that we studied,

- although we considered the day-ahead schedule as given and fixed, we may use the learned strategy to evaluate the expected adjustment cost of any day-ahead strategy, and thus also modify the day-ahead schedule,
- it would be interesting to set up a procedure for evaluating the expected cost of the recourse strategy when a residual uncertainty exists over the recourse horizon (i.e. when the updated forecast is not perfect),
- it would be interesting to extend and analyze the proposed approach to multiple recourse possibilities and/or broaden its applicability to several similar days.

In Part II, we proposed a generic extension of tree-based ensemble methods which allows to incorporate incomplete data and prior knowledge about the problem. The framework is based on a convex optimization formulation to regularize the learned model, and the prior knowledge is incorporated as a set of constraints. We applied it to problems containing censored data, formulated a way to extend tree-based methods to semi-supervised learning, etc. However this research originated from the observation that, in the recourse strategy learning problem of Part I, classical learning algorithms are unable to use the information contained in the constraints of the generation (re)scheduling problem. Thus, when we apply a SL algorithm to search in a hypothesis space based on data that satisfies a known set of constraints, can we guarantee that the hypothesis that we select will make predictions that satisfy the constraints? Or can we benefit of our knowledge of the constraints while learning and improve the generalization accuracy of the selected hypothesis?

10.2 Further work

Regularizing the learned recourse strategy. Given the way we formulated the recourse strategy learning problem, incorporating all the operating constraints of the generation units to regularize the learned recourse strategy and obtain predictions that satisfy all these constraints seems hard to achieve in practice. Even if we impose these constraints for a few scenarios of the learning set, because of the nature of the generation scheduling problem (MILP) and the time decomposition that we operated, we are not ensured of the feasibility of the regularization problem. Furthermore the size of the regularization problem may quickly become excessive when the number of scenarios considered is increased. However, by using other learning formulation decompositions (e.g. unit by unit) and imposing some restrictions on the learned model a priori (e.g. on the outputs participating in the average value assigned to the leaves if we consider a single tree model), it may be possible to incorporate some of the constraints of the generation (re)scheduling problem (e.g. the constraints that are the most often violated in the experiments that we performed, such as the non-convexity of the generation range of thermal units).

But ensuring the satisfaction of the operating constraints of the predicted recourse may not be the most important concern, since we observed experimentally that, when predicting the generation levels of all the units, the effect

of the post-processing stage is marginal. Although it should be experimentally validated, it would maybe be more valuable to improve the demand-generation balance with respect to the updated demand forecast. As in the generation (re)scheduling problem, demand-generation balance may be incorporated as a soft constraint (e.g. in replacement of the penalization of the prediction error) in the regularization problem. This type of constraint is more likely to lead to a beneficial, feasible and tractable regularization problem.

Estimation of the adjustment cost. Another interesting possibility would be to use supervised learning to predict the overall adjustment cost for any scenario, based on features describing the scenario. A learned model which would be sufficiently accurate in this respect could then be exploited in various ways in order to speed-up Monte-Carlo simulations used in order to compare alternative reference schedules, or to compute simply a good estimate of the expected adjustment costs for the next day. In line with the application of Part II of this thesis to exploit censored datasets, we could set up a formulation of the learning problem where rather than having the exact value of the optimal cost for each scenario, one is given only (upper and/or lower) bounds on these costs. In our context, this would thus allow one to exploit datasets generated by using problem relaxations (lower bounds), or by halting the optimization procedure as soon as a given computing time budget is exhausted (upper bounds). This possibility clearly opens the way towards many different possible tradeoffs for combining machine learning and optimization.

Other application domains. Finally, we believe that the ideas investigated in this thesis could be adapted to other problems of electricity generation planning, such as

- stochastic problems arising in a medium-term horizon:
  - the management of nuclear outages for refueling when accounting for uncertainty on demand, generation cost and generation unavailabilities (this problem was proposed by EDF as the ROADEF Challenge in 2010[1]),
  - the valuation of hydroelectric reserves in a hydrothermal generation system (note that this has already received some attention in Aïd et al. (2004) from a RL perspective, and Barty et al. (2010) make use of statistical tools to model the dynamics of the dual variables associated to a demand balance constraint);
  - the handling of CO_2 emissions (Avellà Fluvià et al., 2005),
- or the problem of stochastic capacity expansion planning over long-term horizon, which is defined in Birge and Louveaux (1997, Chapter 1) as the problem of finding optimal levels of investment in various types of power plants to meet future electricity demand, and is sensible to fuel prices and demand variations but also to technological evolutions,

[1] `http://www.roadef.org/content/index.htm`

and thus indeed more generally to optimization problems where one wants to approximate the solution of an optimization algorithm $A_o^z(p)$ assuming some uncertainty on the value of a parameter vector p, as discussed in Chapter 1.1.

Appendices

Appendix A

Supervised learning algorithms

In this appendix we describe the supervised learning algorithms that we have used throughout our research.

A.1 Top-down induction of a regression tree

Tree-based supervised learning is well known for its computational efficiency, interpretability, robustness to outliers, its capability to cope with high-dimensional problems with a large number of input features, and its robustness with respect to irrelevant features and noise. The idea followed by this approach is to recursively split the training set D, thanks to tests on the value of the input features, in several subsets in order to decrease a measure of impurity about the output variable until the subsets are composed of object sufficiently similar in the output space. A classical top-down regression tree induction algorithm works as indicated on Figure A.1.

If we use the mean square error criterion to estimate the accuracy of the tree predictor, defining l_i (step 4) as the mean value of the outputs of the objects constituting a leaf i is optimal with respect to empirical prediction error.

To split a node i (step 5), a test is defined by a feature x^k ($k \in \{1, \ldots, |\mathcal{X}|\}$) and a cut-off value ($v^k$). All the objects satisfying the test $x^k > v^k$ are assigned to the right successor node and the remaining ones are assigned to the left successor node. To find the best test, a score is computed for every input feature and for every possible cut-off value. For regression trees, a typical score measure is the relative decrease of output variance in the two successor nodes with respect to the output variance of the current node,

$$score(D_i, x^k, v^k) = \frac{var\{y|D_i\} - \frac{|D_l|}{|D_i|} var\{y|D_l\} - \frac{|D_r|}{|D_i|} var\{y|D_r\}}{var\{y|D_i\}}. \quad \text{(A.1)}$$

The test with the highest variance reduction is chosen. Notice that, if the mean square error is chosen as error criterion, the split with the highest output variance reduction turns out to be the split that is optimal in terms of the reduction of the empirical prediction error.

It is more complicated to assess if a node should be split (step 3), i.e. to mitigate the complexity of the model. A classical way is to stop splitting when

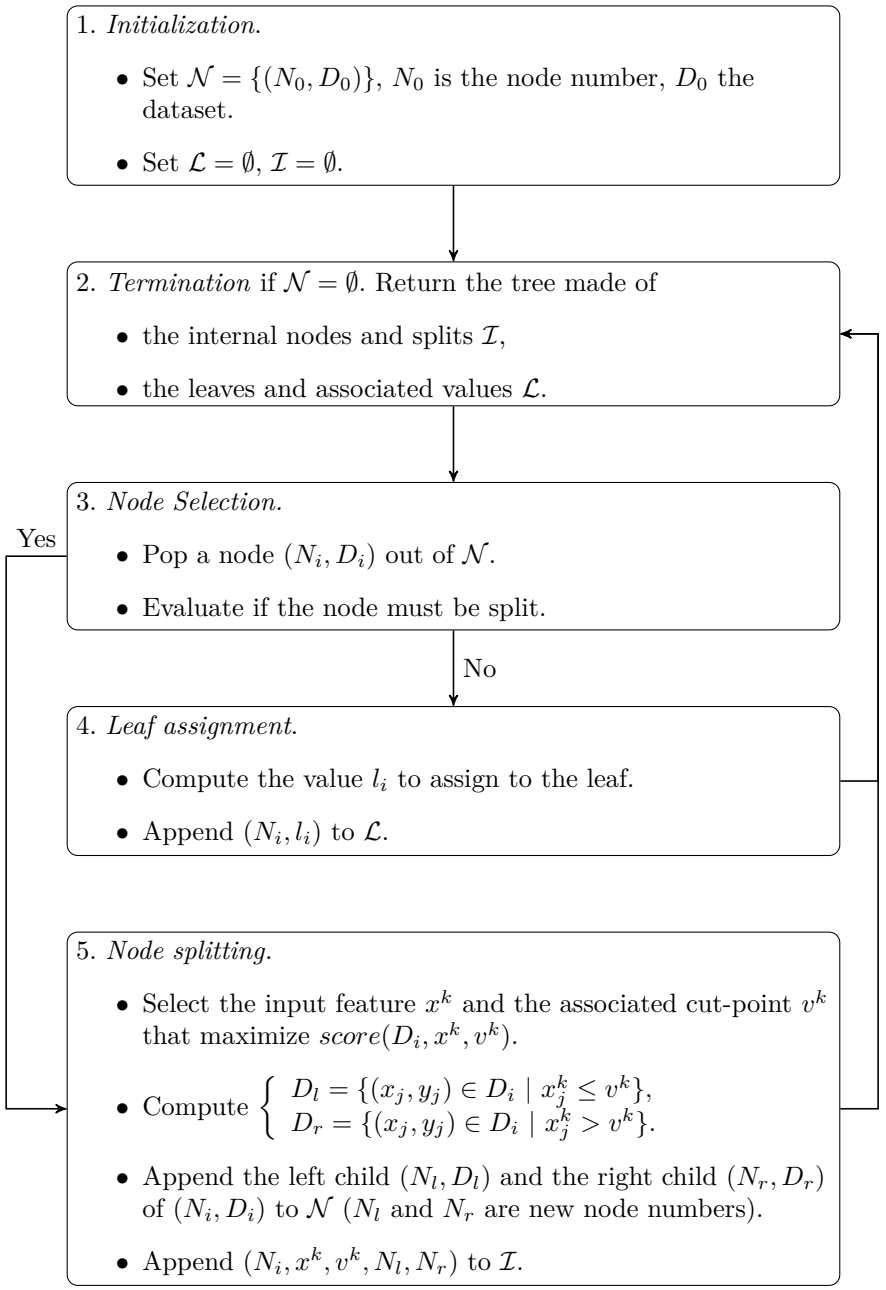


Figure A.1: Top-down induction of a regression tree.

the number of objects in a node is below a threshold value n_{min}, but many other techniques have been developed to identify the tree of optimal complexity (pruning methods). For a more complete description of these pruning algorithms see for example Breiman et al. (1984).

A.2 The Extra-Trees supervised learning method

While the learning of single regression trees is computationally very efficient and often leads to highly interpretable decision rules, it has however been shown that single tree-based methods have a high learning variance (Geurts, 2002), which implies that they are often suboptimal in terms of accuracy, specially on problems where the information is spread among a large number of equally relevant features.

Therefore, tree-based ensemble methods have been introduced to decrease variance and to allow them to cope with very complex tasks such as image, text and time-series classification. The general idea behind these methods is to avoid giving a single tree the capability of modeling the whole training set. This can be achieved either by perturbing the training set, as for example in Tree Bagging (Breiman, 1996) which uses a standard tree induction algorithm to derive the trees from a bootstrap sample, or by perturbing the construction algorithm (e.g. Extra-Trees, random forests (Breiman, 2001)), in order to build from a training set an ensemble of M different trees, and by deriving the hypothesis h by aggregating in some fashion (e.g. by voting or by averaging) the predictions derived from each tree in the ensemble.

A major cause of learning variance of regression trees is the sensitivity of the cut-point value of test nodes to the content of the training set. The main aim of the Extra-Trees method (Geurts et al., 2006a) is to mitigate this behavior by randomly perturbing the structure of the trees, thus decreasing their dependence on the training set. In the Extra-Trees method, each tree is built from the complete original training set. The search for the best test at a node (step 5 in Figure A.1 for the classical tree induction algorithm) is partially randomized; as explained in Table A.1, this algorithm selects K input-features at random and for each input feature, a cut-point at random. It then computes a score for each of the K tests and chooses among these K tests the one that maximizes the score. The algorithm stops splitting a node when the number of elements in this node is less than a parameter n_{min}. The prediction of the ensemble is obtained by averaging the prediction of the M trees.

To summarize, the ensemble construction is guided by the parameters M, K and n_{min}, which have the following effects:

- M (number of trees) determines the strength of the variance reduction of the ensemble model aggregation,
- K (number of candidate tests at each node) determines the strength of the attribute selection process,
- n_{min} (maximal leaf size) determines the strength of averaging output noise.

Table A.1: Procedure to find a test at a node in the Extra-Trees algorithm.

Input: a dataset D.
Output: a test defined by an input variable x^{i_k} and a cut-point v^{i_k}.

1. Select K input-features indices, $\{i_1, ..., i_K\}$, at random, without replacement, among all (non constant) input variables.
2. For all $k' \in \{1, \ldots, K\}$:
   a) compute the maximal and minimal value of $x^{i_{k'}}$ in D, denoted respectively by $v^{i_{k'}}_{max}$ and $v^{i_{k'}}_{min}$,
   b) draw a cut-point $v^{i_{k'}}$ uniformly in $[v^{i_{k'}}_{min}, v^{i_{k'}}_{max}]$,
   c) compute the score $S_{k'} = score(D, x^{i_{k'}}, v^{i_{k'}})$ according to (A.1).
3. Return a test $x^{i_k} \leq v^{i_k}$ such that $k = \arg\max_{k' \in \{1,\ldots,K\}} S_{k'}$.

With respect to classical single trees, the accuracy of this method is in general dramatically increased. Because all the trees are built independently and because the induction procedure is simplified, this algorithm is computationally very efficient. In addition, the Extra-Trees method produces as a byproduct a scoring of the input features in terms of their usefulness to predict the output information. These so-called *variable importances* may be used in practice to analyze the impact of the different features and hence to better understand the problem under consideration. We use these importance measures in our case study to analyze the impact of different features on the recourse decisions. We refer the interested reader to Huynh-Thu et al. (2008) for further information about the computation and nature of these variable importances.

A.3 Support Vector Regression

We provide a brief description of the ε-Support Vector regression algorithm. More details on the solution algorithms and a deeper interpretation of the model and its parameters can be found in Smola and Schölkopf (2003). The idea of this method is to approximate the output value by a linear function of its input features:

$$f(x) = w^T x + b, \qquad w \in \mathcal{X}, b \in \mathbb{R}.$$

The convex optimization problem of Formulation 11 is solved to compute the values of the unknowns w and b. Prediction errors smaller than the parameter $\varepsilon \geq 0$ are hence discarded. As it is typically not possible to find a solution for any value of ε, some non-negative slack variables ξ_i and $\xi_i^\star$ are added to loosen the first two constraints. The second term of the objective function simply penalizes these slack variables, while the first one is a regularization term preventing too large weight vectors and hence limiting the capacity of

Formulation 11: Primal formulation of the ε-SVR problem.

$$\min_{w,b,\xi,\xi^\star} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^\star) \tag{A.2}$$

$$\text{s.t.} \quad y_i - w^\top x_i - b \leq \varepsilon + \xi_i, \quad i = 1, \ldots, N \tag{A.3}$$

$$w^\top x_i + b - y_i \leq \varepsilon + \xi_i^\star, \quad i = 1, \ldots, N \tag{A.4}$$

$$\xi_i, \xi_i^\star \geq 0, \quad i = 1, \ldots, N. \tag{A.5}$$

the function f, or in other words over-fitting of the training data. A trade-off between these two terms is formed through the parameter $C \geq 0$. For a given ε, a too large value of C induces a complex f which may not generalize well to unseen objects, and a too small value might cause a biased approximation function.

To solve this problem for instances where the number of optimization variables – hence the number of features – is large compared to the number of constraints – hence to the number of training objects – it is a good idea to switch to the Lagrangian dual formulation (cf. Formulation 12). The dual variables α_i and $\alpha_i^\star$ are the Lagrange multipliers of the constraints (A.3) and (A.4). The Lagrange multipliers associated to the constraint sets (A.5) can be eliminated during the dualization process. From one of the saddle point conditions used during the dualization, namely $w - \sum_{i=1}^{N}(\alpha_i - \alpha_i^\star)x_i = 0$, we can express $f(x)$ using only the features of the objects of the learning set, the variable b and the dual variables:

$$f(x) = \sum_{i=1}^{N}(\alpha_i - \alpha_i^\star)x_i^T x + b. \tag{A.6}$$

Exploring the complementary slackness conditions, one can gain some insight in the role of the parameter ε. At the optimal solution, the conditions

$$\alpha_i(\varepsilon + \xi_i - y_i + w^T x_i + b) = 0, \quad i = 1, \ldots, N, \tag{A.10}$$

$$\alpha_i^\star(\varepsilon + \xi_i^\star + y_i - w^T x_i + b) = 0, \quad i = 1, \ldots, N, \tag{A.11}$$

$$(C - \alpha_i)\xi_i = 0, \quad i = 1, \ldots, N, \tag{A.12}$$

$$(C - \alpha_i^\star)\xi_i^\star = 0, \quad i = 1, \ldots, N. \tag{A.13}$$

hold. From (A.10) and (A.11), for the objects (x_i, y_i) such that $|f(x_i) - y_i| < \varepsilon$, and thus $\xi_i, \xi_i^\star = 0$, we can conclude that the dual variables α_i and $\alpha_i^\star$ are equal to 0. The parameter ε thus modulates the sparsity of the solution, and the term "Support Vector" qualifies the features vectors x_i corresponding to positive α_i or $\alpha_i^\star$. We can also see that $\alpha_i\alpha_i^\star = 0$, thus either α_i or $\alpha_i^\star$ may be nonzero, if $|f(x_i) - y_i| \geq \varepsilon$. From (A.12) and (A.13), we conclude that $\alpha_i = C$ (resp. $\alpha_i^\star = C$) if and only if $\xi_i > 0$ (resp. $\xi_i^\star > 0$). One can obtain the value of b in (A.6) by further exploiting these complementary slackness conditions.

Formulation 12: Dual formulation of the ε-SVR problem.

$$\max_{\alpha,\alpha^\star} \quad \left\{ -\frac{1}{2} \sum_{i,j=1}^{N} (\alpha_i - \alpha_i^\star)\,(\alpha_j - \alpha_j^\star) x_i^\top x_j \right. \\ \left. -\varepsilon \sum_{i=1}^{N} (\alpha_i + \alpha_i^\star) + \sum_{i=1}^{N} y_i (\alpha_i - \alpha_i^\star) \right\} \tag{A.7}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} (\alpha_i - \alpha_i^\star) = 0, \tag{A.8}$$

$$\alpha_i, \alpha_i^\star \in [0, C], \quad i = 1, \ldots, N. \tag{A.9}$$

Finally, as Formulation 12 and the function f can be formulated in terms of dot products of feature vectors, one can embed the input features in another feature space using a kernel function

$$k(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}.$$

The kernel replaces the dot product in the destination feature space. This permits more combinations than a simple dot product in the original input feature space, at a reduced cost compared to the cost of computing explicitly the dot product if the destination feature space has many dimensions. The conditions under which a kernel corresponds to a dot product in some feature space are covered in details in Schölkopf and Smola (2001, Chapter 2).

Appendix B

Detailed optimization formulations for the problems of Part I

We detail the formulation of the optimization problems used and discussed in Part I. Specifically, Section B.1 covers the day-ahead generation scheduling problem, Section B.2 points out the modifications that are made to the day-ahead problem in order to compute optimal adjustments to a reference schedule for a fixed recourse instant, and Section B.3 details the post-processing used either to ensure the feasibility of some predicted adjustments or to obtain a detailed generation schedule from the prediction of some adjustment indicators.

In this chapter, superscripts must always be understood as clarifying expressions, not as exponents. In particular, t refers to thermal generation units, h to hydroelectric generation plants and p to pumping plants. On the other hand, subscripts t indexes the time, i either a thermal generation unit or a hydroelectric valley and j a sub-entity of a valley, i.e. a reservoir, a generation plant or a pumping plant. The meaning of other subscripts should be clear from the context.

B.1 Generation scheduling problem

This section first describes the model of the components of the generation system: the thermal generation units in section B.1.1 and the hydroelectric valleys in section B.1.2. Then it explains how these components are coupled in section B.1.3 and finally states the objective of the global model – which is to satisfy the requirements as well as possible at the lowest possible costs – in section B.1.4.

B.1.1 Model of the thermal generation units

The following decision variables are defined for each generation unit $i \in \{0, ..., N_t - 1\}$ and each time step $t \in \{0, ..., T - 1\}$:

- $s_{i,t}^t \in \{0, 1\}$ is the On/Off status,
- $p_{i,t}^t \in \mathbb{R}_+$ is the power really generated by the unit,

- $m_{i,t}^{1,t} \in [0, M^{1,max}]$ is the power margin reserved for primary control,
- $m_{i,t}^{2,t} \in [0, M^{2,max}]$ is the power margin reserved for secondary control,
- $s_{i,t}^{t,up} \in \{0,1\}$ indicates whether or not a start-up occurred between time $t-1$ and t.
- $s_{i,t}^{t,down} \in \{0,1\}$ indicates whether or not a shut-down occurred between time steps $t-1$ and t.
- $c_{i,t}^{t} \in \mathbb{R}_+$ is an auxiliary variable representing the cost of a start-up.

When the unit i is started, its generation level must stay above a non-zero minimal level P_i^{min} and below a maximal level P_i^{max}. Binary variables $s_{i,t}^t$ indicating the On/Off state of the unit are thus necessary:

$$\forall i \in \{0 \dots N_t - 1\}, \forall t \in \{0 \dots T-1\}, \\ s_{i,t}^t P_i^{min} \leq p_{i,t}^t \quad \leq \quad s_{i,t}^t P_i^{max}. \tag{B.1}$$

Some power margins must be reserved for the automatic load-frequency regulation of the synchronous network (cf. Section 2.2.1), i.e. for primary control and for secondary control. Both margins are equal to zero when the unit is shut-down, and their sum is smaller than $\min\{P_i^{max} - p_{i,t}^t, p_{i,t}^t - P_i^{min}\}$ otherwise, as expressed by constraints (B.2) and (B.3).

$$\forall i \in \{0 \dots N_t - 1\}, \forall t \in \{0, T-1\}, \\ m_{i,t}^{1,t} + m_{i,t}^{2,t} \quad \leq \quad s_{i,t}^t \; P_i^{max} - p_{i,t}^t \tag{B.2}$$

$$m_{i,t}^{1,t} + m_{i,t}^{2,t} \quad \leq \quad p_{i,t}^t - s_{i,t}^t \; P_i^{min} \tag{B.3}$$

Constraints (B.4) limit the variation of the actual generation level between two successive time steps.

$$\forall i \in \{0 \dots N_t - 1\}, \forall t \in \{0, T-1\}, \\ -\Delta P_i^{t,down} \quad \leq \quad p_{i,t}^t - p_{i,t-1}^t \leq \Delta P_i^{t,up}, \tag{B.4}$$

$$p_{i,-1}^t \quad \doteq \quad P_i^{init} \tag{B.5}$$

The constraints (B.6) and (B.7) define the value of the start-up ($s_{i,t}^{t,up}$) and shut-down ($s_{i,t}^{t,down}$) indicators.

$$\forall i \in \{0 \dots N_t - 1\}, \forall t \in \{0 \dots T-1\}, \\ s_{i,t}^{t,up} - s_{i,t}^{t,down} \quad = \quad s_{i,t}^t - s_{i,t-1}^t \tag{B.6}$$

$$s_{i,t}^{t,up} + s_{i,t}^{t,down} \quad \leq \quad 1 \tag{B.7}$$

These indicators are necessary for stating the minimum up and down times constraints. Technical limitations impose a unit to stay in the same On/Off state for a minimum duration. The following formulation of minimum up and down times was inspired by Carrion and Arroyo (2006).

Constraints (B.8), (B.9) and (B.10) impose a minimum duration $D_i^{min,up}$ in state $s_{i,t}^t = 1$ after a start-up. Let the time step

$$T_i^{init,up} = \min\left\{\max\left\{0, D_i^{min,up} - D_i^{init}U_i^{init}\right\}, T-1\right\}$$

be the minimum between the optimization time horizon and the remainder of $D_i^{min,up}$ if the unit was started before time 0,

$$\forall i \in \{0 \dots N_t - 1\},$$
$$\sum_{t=0}^{T_i^{init,up}-1} s_{i,t}^t = T_i^{init,up}, \quad \text{(B.8)}$$
$$\forall k \in \left\{T_i^{init,up} \dots T - D_i^{min,up} - 1\right\}$$
$$\sum_{t=k}^{k+D_i^{min,up}-1} \left(s_{i,t}^t - s_{i,k}^{t,up}\right) \geq 0, \quad \text{(B.9)}$$
$$\forall k \in \left\{\max\left\{T - D_i^{min,up}, T_i^{init,up}\right\} \dots T-1\right\},$$
$$\sum_{t=k}^{T-1} \left(s_{i,t}^t - s_{i,k}^{t,up}\right) \geq 0. \quad \text{(B.10)}$$

Constraints (B.8) impose that from time 0 to time $T_i^{init,up} - 1$ the unit i cannot be stopped. From time $T_i^{init,up} - 1$ on, constraints (B.9) ensure that if the unit has started operating at time k, i.e. $s_{i,k}^{t,up} = 1$, then it remains started for $D_i^{min,up}$ time steps, and constraints (B.10) does the same for the end of the optimization period.

Similarly, constraints (B.11), (B.12) and (B.13)) impose a minimum duration $D^{down,min}$ in state $s_{i,t}^t = 0$ after a shut-down. Let the time step

$$T_i^{init,down} = \min\left\{\max\left\{0, D_i^{min,down} - D_i^{init}(1 - U_i^{init})\right\}, T-1\right\}$$

be the minimum between the optimization time horizon and the remainder of $D_i^{min,down}$ if the unit was stopped before time 0.

$$\forall i \in \{0 \dots N_t - 1\},$$
$$\sum_{t=0}^{T_i^{init,down}-1} s_{i,t}^t = 0, \quad \text{(B.11)}$$
$$\forall k \in \left\{T_i^{init,down} \dots T - D_i^{min,down} - 1\right\}$$
$$\sum_{t=k}^{k+D_i^{min,down}-1} \left(1 - s_{i,t}^t - s_{i,k}^{t,down}\right) \geq 0, \quad \text{(B.12)}$$
$$\forall k \in \left\{\max\left\{T - D_i^{min,down}, T_i^{init,down}\right\} \dots T-1\right\}$$
$$\sum_{t=k}^{T-1} \left(1 - s_{i,t}^t - s_{i,k}^{t,down}\right) \geq 0. \quad \text{(B.13)}$$

Constraints (B.11) impose that from time 0 to time $T_i^{init,down} - 1$ the unit i cannot be started. From time $T_i^{init,down} - 1$ on, constraints (B.12) ensure that if the unit has stopped operating at time k, i.e. $s_{i,k}^{t,down} = 1$, then it remains stopped for $D_i^{min,down}$ time steps, and constraints (B.13) does the same for the end of the optimization period.

Finally, start-up costs are function of the time the unit has been stopped before being started up. If this time is greater or equal to D_i^{cool}, than a maximum cost is incurred. Otherwise the start-up cost follows the curve whose samples are gathered in the vector C_i^{start}. The constraints (B.14) provide an upper bound on the cost, but as variables $c_{i,t}$ appear with a positive sign in the objective function and that the problem is a minimization, $c_{i,t}$ reflects the true start-up costs at the optimum.

$$\forall i \in \{0 \dots N_t - 1\}, \quad \forall t \in \{0 \dots T - 1\}, \ \forall k \in \{0 \dots D_i^{cool} - 1\},$$

$$c_{i,t}^t \geq C_{i,k}^{start} \left(s_{i,t}^t - \sum_{t'=1}^{k+1} s_{i,t-t'}^t \right), \tag{B.14}$$

$$s_{i,-1}^t \doteq U_i^{init}. \tag{B.15}$$

The parameters appearing in this section are summarized in table B.1.

Table B.1: Parameters related to a thermal unit i.

$D_i^{min,up}$	Minimum up time.
$D_i^{min,down}$	Minimum down time.
D_i^{cool}	Cool down time.
C_i^{start}	A vector containing D^{cool} samples of the start-up cost curve.
C_i^{fixed}	Fixed cost (appearing in the objective function).
C_i^{prop}	Proportional cost (appearing in the objective function).
$\Delta P_i^{t,down}$	Maximum decrease of $p_{i,t}^t$ between time steps t and $t-1$.
$\Delta P_i^{t,up}$	Maximum increase of $p_{i,t}^t$ between time steps t and $t-1$.
U_i^{init}	0 if the unit is down at time -1, 1 otherwise.
D_i^{init}	Time for which the unit is in state U_i^{init}, time step -1 included.
P_i^{init}	Generation level during period -1.

B.1.2 Model of the hydroelectric valleys

Each hydroelectric valley is composed of reservoirs linked by hydroelectric generation and pumping plants (cf. Figure 2.2). Each valley $i \in \{0, N_v - 1\}$ contains $N_{i,r}$ reservoirs, $N_{i,h}$ generation plants, and $N_{i,p}$ pumping plants. Each generation (pumping) plant contains G turbines (pumps). The following variables are defined for each valley i and each time step $t \in \{0 \dots T-1\}$:

- $v_{i,j,t} \in \left[V^{min}_{i,j,t}, V^{max}_{i,j,t}\right]$ is the volume of the reservoir j at the end of the period t,
- $w^s_{i,j,t} \in \mathbb{R}_+$ is the water flow corresponding to the water spilled out of the reservoir j $[m^3/s]$,
- $s^s_{i,j,t} \in \{0, 1\}$ indicates whether or not the spillage of water out of the reservoir j is allowed,
- $w^h_{i,j,t}$ is the water flow traversing the hydroelectric generation plant j $[m^3/s]$,
- $p^h_{i,j,t} \in \mathbb{R}_+$ is the power generated by the hydroelectric generation plant j,
- $m^{1,h}_{i,j,t} \in \mathbb{R}_+$ is the power margin reserved for primary control of the hydroelectric generation plant j,
- $m^{2,h}_{i,j,t} \in \mathbb{R}_+$ is the power margin reserved for secondary control of the hydroelectric generation plant j,
- $s^p_{i,j,t,g} \in \{0, 1\}$ indicates the pumping level g of the pumping plant j,
- $p^p_{i,j,t} \in \mathbb{R}_-$ is the power generated by the pumping plant j.

In a hydroelectric power plant, turbines or pumps must be started in a fixed order. The water flowing through a generation plant j can take any value in the set $[0, W^h_{i,j}]$, but in a pumping plant each pump can operate only at one rate. This results in a piecewise linear power versus water flow curve for generation plants, and in discrete operation levels for pumping plants. Let

$$\begin{aligned} \Theta_{X,Y}(\cdot): \quad & \mathbb{R} \to \mathbb{R}, \\ & x \mapsto y, \end{aligned}$$

be the function parameterized by two vectors X and $Y \in \mathbb{R}^G_+$ and defined on $[0, X^{G-1}]$ which associates to a point x the value of the piecewise linear approximation of points $\{(X_0, Y_0) \dots (X_{G-1}, Y_{G-1})\}$ at the abscissa x (cf. Figure B.1). Then the generation curve of a generation plant can be expressed as,

$$\begin{aligned} &\forall t \in \{0 \dots T-1\}, \\ &\qquad p^h_{i,j,t} \quad = \quad \Theta_{W^h_{i,j}, P^h_{i,j}}\left(w^h_{i,j,t}\right). \end{aligned} \tag{B.16}$$

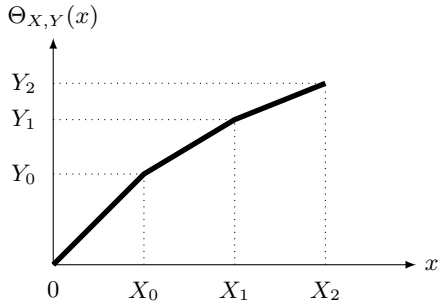


Figure B.1: Representation of $\Theta_{X,Y}(x)$ if X and Y are in $\mathbb{R}^3_+$.

which can be translated in linear expressions using additional binary variables and special ordered sets of type 2 (cf. Appendix C.1.2.4). The margin of power reserved for primary control is a fraction of $p^h_{i,j,t}$,

$$\begin{aligned} &\forall t \in \{0 \dots T-1\}, \\ &\qquad m^{1,h}_{i,j,t} \quad = \quad p^h_{i,j,t} M^{1,h}_{i,j}. \end{aligned} \tag{B.17}$$

The margin of power reserved for secondary control is also defined as a piecewise linear function of the water flowing through the turbine.

$$\begin{aligned} &\forall t \in \{0 \dots T-1\}, \\ &\qquad m^{2,h}_{i,j,t} \quad = \quad \Theta_{W^h_{i,j}, M^{2,h}_{i,j}} \left(w^h_{i,j,t} \right). \end{aligned} \tag{B.18}$$

The discrete operation levels of pumping plants can be modeled by stating that the sets of variables $\left\{ s^p_{i,j,t,0}, \dots, s^p_{i,j,t,G-1} \right\}, \forall t \in \{0 \dots T-1\}$ are special ordered sets of type 1, where the rated operation flows W^p of the plant are used to weight the elements of the set. The power generated by a pumping plant is

$$\begin{aligned} &\forall t \in \{0 \dots T-1\}, \\ &\qquad p^p_{i,j,t} \quad = \quad \sum_{g=0}^{G-1} s^p_{i,j,t,g} P^p_{i,j,g}. \end{aligned} \tag{B.19}$$

Pumping plants do not participate in primary and secondary control. The volume of a reservoir at the end of the period t is equal to its volume at the end of the period $t-1$, plus the integral of the natural inflows and the flows coming from the surrounding plants over one period, minus the integral of the flows taken from the reservoir by the surrounding plants over one period

and the water spilled out.

$$\begin{aligned}
\forall j \in \{0 \dots N_{i,r}-1\}, \quad & \forall t \in \{0 \dots T-1\}, \\
v_{i,j,t} &= v_{i,j,t-1} + D_t \Bigg(W^{nat}_{i,j,t} - w^s_{i,j,t} \\
&+ \sum_{\{k \in \{0 \dots N_{i,h}-1\} | j \in R^{down}(k)\}} w^h_{i,k,t-T^{flow}_k} \\
&+ \sum_{\{k \in \{0 \dots N_{i,p}-1\} | j \in R^{up}_k\}} \sum_{g=0}^{G-1} s^p_{k,t-T^{flow}_k,g} W^p_{k,g} \\
&+ \sum_{\{k \in \{0 \dots N_{i,h}-1\} | j \in R^{down}_k\}} \sum_{\{l \in R^{up}_k\}} w^s_{i,l,t-T^{flow}_k} \\
&- \sum_{\{k \in \{0 \dots N_{i,h}-1\} | j \in R^{up}_k\}} w^h_{i,k,t} \\
&- \sum_{\{k \in \{0 \dots N_{i,p}-1\} | j \in R^{down}_k\}} \sum_{g=0}^{G-1} s^p_{k,t,g} W^p_{k,g} \Bigg), \quad \text{(B.20)}
\end{aligned}$$

where D_t is the duration in seconds of the time period t.

The last two constraints relate to the spillage limits and authorizations.

$$\begin{aligned}
\forall j \in \{0 \dots N_{i,r}-1\}, \quad & \forall t \in \{0 \dots T-1\}, \\
v_{i,j,t} &\geq V^{max}_{i,j,t} s^s_{i,j,t}, \quad \text{(B.21)} \\
w^s_{i,j,t} &\leq s^s_{i,j,t} (W^{nat}_{i,j,t} \\
&+ \sum_{\{k \in \{0 \dots N_{i,h}-1\} | R^{down}_k = j\}} W^{nat}_{i,R^{up}_k,t}). \quad \text{(B.22)}
\end{aligned}$$

The spillage authorization indicator $s^s_{i,j,t}$ is thus equal to 1 if only if the volume of the reservoir j equals its maximum allowed level (B.21). The constraint (B.22) expresses that one cannot spill more than the natural inflows coming directly in reservoir j plus the natural water flows of the upstream reservoirs.

The parameters of all these entities are summarized in table B.2.

B.1.3 Coupling constraints

The coupling of all the generation and pumping plants is achieved through the constraints balancing the load and the generation, and the constraints balancing the ancillary services reserves requirements and the margins actually provided. One slack variable is introduced at each time step $t \in \{0 \dots T-1\}$ to soften each of these constraints as they are sometimes hard to satisfy tightly in practice.

Variables $p^{slack}_t \in \mathbb{R}_+$ represent the absolute value of the mismatch between the demand $P^{req}_t \forall t \in \{0 \dots T-1\}$ and the total generation because a lack as well an excess of generation is penalized (by a coefficient C^{load}). They are defined by constraints (B.23) and (B.24)

Table B.2: Parameters related to hydroelectric valley i.

	Reservoirs
$V^{max}_{i,j,t}$	Maximal volume during time step t.
$V^{min}_{i,j,t}$	Minimal volume during time step t.
$V^{init}_{i,j}$	Volume at time step -1.
$W^{nat}_{i,j,t}$	Natural incoming water flows during time step t.
$C^{water}_{i,j}$	Value of the water stored in the reservoir.
	Hydroelectric plants
$R^{up}_{i,j}$	Set of reservoirs situated directly upstream of the plant.
$R^{down}_{i,j}$	Set of reservoirs situated directly downstream of the plant.
$T^{flow}_{i,j}$	Time needed by the harnessed water to arrive at the next reservoir.
$W^{h}_{i,j,g}$	The values of the water flows delimiting the G operation ranges of the generation plant.
$P^{h}_{i,j}$	The generation levels corresponding to the values of $W^{h}_{i,j}$.
$M^{2,h}_{i,j}$	The amounts of secondary reserve corresponding to the values of $W^{h}_{i,j}$.
$M^{1,h}_{i,j}$	Coefficient of participation to the primary reserve of the generation plant.
$W^{p}_{i,j,g}$	The G admissible set points of the water flows of the pumping plant.
$P^{p}_{i,j,g}$	Power associated to the values of $W^{p}_{i,j}$.

$$
\begin{aligned}
\forall t &\in \{0 \dots T-1\}, \\
p_t^{slack} &\geq \left(\sum_{i=0}^{N_t-1} p_{i,t}^{t} + \sum_{i=0}^{N_v-1} \left(\sum_{j=0}^{N_{i,h}-1} p_{i,j,t}^{h} + \sum_{j=0}^{N_{i,p}-1} p_{i,j,t}^{p} \right) \right) - P_t^{req}, \quad \text{(B.23)} \\
p_t^{slack} &\geq P_t^{req} \\
&\quad - \left(\sum_{i=0}^{N_t-1} p_{i,t}^{t} + \sum_{i=0}^{N_v-1} \left(\sum_{j=0}^{N_{i,h}-1} p_{i,j,t}^{h} + \sum_{j=0}^{N_{i,p}-1} p_{i,j,t}^{p} \right) \right), \quad \text{(B.24)}
\end{aligned}
$$

Variables $m_t^{1,slack} \in \mathbb{R}_+$ and $m_t^{2,slack} \in \mathbb{R}_+$ represent the lack of margin reserved respectively to primary and secondary control (constraints (B.25) and (B.26)), whose requirement are $M_t^{1,req}$ and $M_t^{2,req}$, and are penalized with coefficients $C^{m,1}$ and $C^{m,2}$,

$$
\begin{aligned}
\forall t &\in \{0 \dots T-1\}, \\
m_t^{1,slack} &\geq M_t^{1,req} - \left(\sum_{i=0}^{N_t-1} m_{i,t}^{1,t} + \sum_{i=0}^{N_v-1} \sum_{j=0}^{N_{i,h}-1} m_{i,j,t}^{1,h} \right), \quad \text{(B.25)} \\
m_t^{2,slack} &\geq M_t^{2,req} - \left(\sum_{i=0}^{N_t-1} m_{i,t}^{2,t} + \sum_{i=0}^{N_v-1} \sum_{j=0}^{N_{i,h}-1} m_{i,j,t}^{2,h} \right). \quad \text{(B.26)}
\end{aligned}
$$

An excess of primary and secondary control margins is not penalized explicitly.

B.1.4 Objective function

The objective function decomposes in the costs related to thermal generation, the costs related to the usage of water and the penalization of the slack variables of the coupling constraints.

$$\begin{aligned}\min \sum_{t=0}^{T-1} \Bigg(& \sum_{i=0}^{N_t-1} \Big(D_t C_i^{fixed} s_{i,t}^t + D_t C_i^{prop} p_{i,t}^t + c_{i,t}^t \Big) \\ & + D_t C^{load} p_t^{slack} + D_t C^{m,1} m_t^{1,slack} + D_t C^{m,2} m_t^{2,slack} \Bigg) \\ & + \sum_{i=0}^{N_v-1} \left(\sum_{j=0}^{N_{i,r}-1} C_{i,j}^{water} (v_{i,j,T-1} - V_{i,j}^{init}) \right).\end{aligned} \tag{B.27}$$

B.2 Generation rescheduling problem

Suppose that one starts from the reference schedule solution of Appendix B.1 computed one day ahead $\forall t \in \{0 \dots T-1\}$. In this section we will denote the solution of the day-ahead problem by capital letters. For example, we have

$$\begin{array}{ll} P_{i,t}^t \doteq p_{i,t}^t, \forall i \in \{0, ..., N_t - 1\} & \text{(thermal generation)}, \\ M_{i,t}^{1,t} \doteq m_{i,t}^{1,t} i \in \{0, ..., N_t - 1\} & \text{(thermal primary control reserve)}, \\ M_{i,t}^{2,t} \doteq m_{i,t}^{2,t} i \in \{0, ..., N_t - 1\} & \text{(thermal secondary control reserve)}. \end{array}$$

and similarly for the variables related to hydroelectric valleys, turbines and pumps, as well as for the slack variables of the coupling constraints. We assume that the recourse opportunity occurs at time t_r.

B.2.1 Before the recourse period

From time 0 to time $t_r - 1$, we must use the reference schedule and simulate the action of the ancillary services. To do so, we propose to allow the modification of the schedule on $\{0 \dots t_r - 1\}$ but only around the generation level of the reference schedule, and inside a range defined by the margins devoted to ancillary services in the reference schedule. For simplicity we make no distinction between the use of primary and secondary reserves, and add up both margins. In the new problem, with respect to the problem of Appendix B.1, we consider some additional bounds for the thermal units:

$$\begin{aligned} \forall i \in \{0 \dots N_t - 1\}, \forall t \in \{0 \dots t_r - 1\}, \\ p_{i,t}^t \in [P_{i,t}^t - M_{i,t}^{1,t} - M_{i,t}^{2,t}, P_{i,t}^t + M_{i,t}^{1,t} + M_{i,t}^{2,t}]. \end{aligned} \tag{B.28}$$

Similarly, in each valley i, we consider some additional bounds on the allowed power generation range of the hydroelectric turbines:

$$\begin{aligned} \forall j \in \{0 \dots N_{i,h} - 1\}, \forall t \in \{0 \dots t_r - 1\}, \\ p_{i,j,t}^h \in [P_{i,j,t}^h - M_{i,j,t}^{1,h} - M_{i,j,t}^{2,h}, P_{i,j,t}^t + M_{i,j,t}^{1,h} + M_{i,j,t}^{2,h}]. \end{aligned} \tag{B.29}$$

Although they do not participate to ancillary services, pumping plants must be freely adjustable to ensure the satisfiability of the constraints on the volume of reservoirs.

In the rescheduling problem the reserves of the day-ahead schedule are used during $\{0 \dots t_r - 1\}$ and the recourse aims at adjusting the generation levels so as to satisfy an updated demand curve and to restore the ancillary services reserves. Consequently the objective (B.27) is adapted to penalize the ancillary services requirements only during $\{t_r \dots T-1\}$. In real-time operation, ancillary services regulators act as non-anticipative devices. Here the use of ancillary services is optimized anticipatively since we assume the knowledge of the realized scenario when rescheduling. Indeed, the use of ancillary services could even be optimized to improve the adjustment on $\{t_r \dots T-1\}$. However, note that as the vectors gathering the slack variables representing the load-generation imbalance p_t^{slack} on $\{0 \dots T-1\}$ are penalized with the ℓ_1-norm, no distinction is made between small and large slacks. In addition we add some constraints to state that slack variables representing the load-generation imbalance can only decrease or stay constant with respect to their value in the reference schedule during $\{0 \dots t_r - 1\}$:

$$\forall t \in \{0 \dots t_r - 1\}, \quad p_t^{slack} \in [0, P_t^{slack}]. \tag{B.30}$$

This way the ancillary services reserves are used for their intended purpose.

B.2.2 During the recourse period

If no additional requirement applies during the recourse period, the rescheduling problem is similar to the day-ahead scheduling problem provided that we have correctly modeled the behavior of the system before the recourse time. From time t_r to time $T-1$, the schedule can thus be freely modified to satisfy an updated demand curve and restore the ancillary services reserves.

B.2.2.1 Cardinality constraint

However, as exposed in Section 2.3, the number of thermal units and hydroelectric valleys on which we can act for all the recourse period is limited to an integer K smaller than the total number of units allowed to be adjusted. The problem is thus to select the best subset of units and then to re-optimize their generation schedule to cover the updated demand forecast. This section analyzes how this subset selection problem can be incorporated in the optimization procedure by modifying the original scheduling problem.

One binary variable a_i^t is introduced for each thermal unit $i \in \{0 \dots N_t - 1\}$ and one binary variable a_i^h is introduced for each valley $i \in \{0 \dots N_v - 1\}$. These *adjustment indicators* are used to authorize ($a_i = 1$) or prevent ($a_i = 0$) the adjustment of the corresponding generation units. Two formulations are presented in Appendix B.2.2.2 for modeling these variables. Although a valley is composed of several generation or pumping plants and reservoirs, it is considered as a single entity.

To impose the limitation on the number of adjustments as a strong requirement, one can add the cardinality constraint (B.31) to the generation rescheduling problem.

$$\sum_{i=0}^{N_t-1} a_i^t + \sum_{i=0}^{N_v-1} a_i^h \leq K. \tag{B.31}$$

This is the solution that we have used in Section 4.5.

B.2.2.2 Computing adjustment indicators

First formulation. To model the adjustment indicators of the generation units, the variables $\delta p_{i,t}^t \in \mathbb{R}_+$ and $\delta s_{i,t}^t \in \{0,1\}$, $\forall t \in \{t_r+1,...,T\}$, are introduced for each thermal generation unit, as well as the variables $\delta p_{i,t}^h \in \mathbb{R}_+$, $\forall t \in \{t_r+1,...,T\}$, for each valley. Let H_i^{min} and H_i^{max} be respectively a lower and an upper bound on the power that can be generated in valley i.

With the addition of constraints (B.32) to (B.37) to the problem of Appendix B.1, the adjustment of thermal unit i (resp. valley i) is allowed if $a_i^t = 1$ (resp. $a_i^h = 1$), otherwise the precomputed schedule is imposed:

$$\begin{aligned}
&\forall t \in \{t_r+1,...,T\}, \quad \forall i \in \{1,...,N_t\},\\
&P_i^{min}\delta s_{i,t}^t \leq \delta p_{i,t}^t \leq P_i^{max}\delta s_{i,t}^t, && \text{(B.32)}\\
&\delta s_{i,t}^t \leq a_i^t, && \text{(B.33)}\\
&p_{i,t}^t = P_{i,t}^t(1-a_i^t) + \delta p_{i,t}^t, && \text{(B.34)}\\
&\forall i \in \{1,...,N_v\},\\
&a_i H_i^{min} \leq \delta p_{i,t}^h \leq a_i H_i^{max}, && \text{(B.35)}\\
&p_{i,t}^h = P_{i,t}^h(1-a_i^h) + \delta p_{i,t}^h, && \text{(B.36)}\\
&p_{i,t}^h = \sum_{j=0}^{N_{i,h}-1} p_{i,j,t}^h + \sum_{j=0}^{N_{i,p}-1} p_{i,j,t}^p && \text{(B.37)}
\end{aligned}$$

For the thermal generation part, this set of constraints can be understood as follows:

$$\begin{aligned}
a_i^t = 1 \quad &\Rightarrow \quad \begin{cases} \delta p_{i,t}^t \in \{0\} \cup [P_i^{min}, P_i^{max}] \\ p_{i,t}^t = \delta p_{i,t} \end{cases} && \text{(free adjustment)},\\
a_i^t = 0 \quad &\Rightarrow \quad \begin{cases} \delta p_{i,t}^t = 0 \\ p_{i,t}^t = P_{i,t}^t \end{cases} && \text{(no adjustment)},
\end{aligned}$$

and similarly for the hydroelectric generation part:

$$\begin{aligned}
a_i^h = 1 \quad &\Rightarrow \quad \begin{cases} \delta p_{i,t}^h \in [H_i^{min}, H_i^{max}] \\ p_{i,t}^h = \delta p_{i,t}^h \end{cases} && \text{(free adjustment)},\\
a_i^h = 0 \quad &\Rightarrow \quad \begin{cases} \delta p_{i,t}^h = 0 \\ p_{i,t}^h = P_{i,t}^h \end{cases} && \text{(no adjustment)},
\end{aligned}$$

The duplication of the on/off status variables of the thermal unit i at time t ($s_{i,t}^t$) into $\delta s_{i,t}^t$ may seem unnecessary since when $a_i^t = 1$ then $\delta s_{i,t}^t = s_{i,t}^t$.

However all the other operating constraints of the original model must be satisfied, and $\delta s^t_{i,t} = 0$ when $a^t_i = 0$ even if $P^t_{i,t} > 0$. Thus $\delta s^t_{i,t}$ cannot be used in replacement of $s^t_{i,t}$, e.g., in the minimum up time constraints. Note that the operating constraints of unit i may be dropped when $a^t_i = 0$, since the precomputed schedule is feasible. Thus if an efficient technique for dropping these constraints dynamically when $a^t_i = 0$ were available, then variables $\delta s^t_{i,t}$ would be redundant.

Not adjusting ($a^h_i = 0$) a hydroelectric valley thus means that the global power generation of the valley $p^h_{i,t}$ is not modified with respect to the reference schedule (cf. (B.36)). In this case, the individual generation of the turbines and pumps of this valley may nevertheless be modified (B.37), but this is not an issue since their reference schedule is still feasible and can thus still be implemented.

Second formulation. Another way to model the problem is to replace constraints (B.32) to (B.36) by constraints (B.38) to (B.41).

$$\forall i \in \{1, ..., N_t\},$$

$$a^t_i \geq \frac{p^t_{i,t} - P^t_{i,t}}{P^{max}_i} \tag{B.38}$$

$$a^t_i \geq \frac{P^t_{i,t} - p^t_{i,t}}{P^{max}_i} \tag{B.39}$$

$$\forall i \in \{1, ..., N_v\},$$

$$a^h_i \geq \frac{p^h_{i,t} - P^h_{i,t}}{H^{max}_i - H^{min}_i} \tag{B.40}$$

$$a^h_i \geq \frac{P^h_{i,t} - p^h_{i,t}}{H^{max}_i - H^{min}_i} \tag{B.41}$$

Compared to the first formulation, this formulation does not require the addition of variables other than a_i. However, a_i is equal to 1 when an adjustment is really performed but is potentially also equal to 1 otherwise. a_i being equal to 1 when no adjustment is actually performed is not an issue since we require that at most K units are adjusted, and the optimal solution may use all of these K possibilities to perform adjustments that actually minimize the objective function.

Note that some special branching and cuts generation techniques (cf. Appendix C.1.2) are introduced in DE FARIAS ET AL. (2001) to take into account cardinality constraints (among other types of constraints) without the need to introduce auxiliary variables such as the variables a_i. This is maybe an interesting line to follow to speed up the resolution of the problem. However they are currently not implemented in the commercial solver that we have used.

B.3 Post-processing

B.3.1 From predicted generation levels to feasible adjustments

The post-processing consists in projecting the predicted adjustments of the thermal units ($\hat{P}^t_{i,t}$) onto their feasible domain and to compute the detailed schedule of all the plants of each valley based on the predicted demands for these valleys ($\hat{P}^h_{i,t}$). To this end we instantiate one optimization problem for each unit.

Problems for the valleys. The problem formulated for each valley contains all the constraints of Appendix B.1.2 for the valley under consideration. The ancillary services requirement constraints are discarded and the demand is set to the output of the recourse strategy for that valley. In valley i, a slack variable $p^{h,slack}_{i,t}$ is introduced for each time step to prevent infeasibility:

$$\begin{aligned}
\forall i \in \{0 \dots N_v - 1\}, \quad \forall t \in \quad & \{t_r \dots T-1\}, \\
p^{h,slack}_{i,t} \quad \geq \quad & \left(\sum_{j=0}^{N_{i,h}-1} p^h_{i,j,t} + \sum_{j=0}^{N_{i,p}-1} p^p_{i,j,t} \right) - \hat{P}^h_{i,t}, \qquad \text{(B.42)} \\
p^{h,slack}_{i,t} \quad \geq \quad & \hat{P}^h_{i,t} - \left(\sum_{j=0}^{N_{i,h}-1} p^h_{i,j,t} + \sum_{j=0}^{N_{i,p}-1} p^p_{i,j,t} \right). \qquad \text{(B.43)}
\end{aligned}$$

Problems for the thermal generation units. The problems related to the thermal generation units are formulated in a similar way. We minimize the ℓ_1-norm of the difference between the targeted feasible adjustment and the predicted adjustment, which can be seen as a demand requirement for that unit, and discard the ancillary services requirements.

$$\begin{aligned}
\forall i \in \{0 \dots N_t - 1\}, \quad \forall t \in \quad & \{t_r \dots T-1\}, \\
p^{t,slack}_{i,t} \quad \geq \quad & p^t_{i,t} - \hat{P}^h_{i,t}, \qquad \text{(B.44)} \\
p^{t,slack}_{i,t} \quad \geq \quad & \hat{P}^h_{i,t} - p^t_{i,t}. \qquad \text{(B.45)}
\end{aligned}$$

All the other constraints of Appendix B.1.1 are applied.

Objective functions. For both problem types, the generation costs are also removed of the objective function. Indeed the arbitrage between the different generation units is performed by the learned recourse strategy based on the examples that were used to learn it. Once the problem is decomposed and that a demand is defined for each generation unit, we do not account for the generation costs anymore. This is in a sense comparable to the minimization phase when applying Lagrangian relaxation to solve the problem (cf. Appendix C.1.1), but instead of a demand to satisfy, the Lagrangian multipliers provide the information about the marginal price of the electricity power and generation units having a marginal generation cost higher than this price are not used. Thus

the objective of the problem for valley i is simply

$$\min \sum_{t=t_r}^{T-1} p_{i,t}^{h,slack},$$

and similarly for the thermal problems.

Each problem is thus a MILP, but is much smaller than the global scheduling problem. Once all the problems are solved, the reserves of ancillary services are straightforwardly deduced from the generation levels. The global cost of the adjusted schedule is computed by summing the generation costs, the penalization of the discrepancies between generation and demand and the penalization of the discrepancies between the required and provided reserves for the ancillary services.

B.3.2 From adjustment indicators to feasible adjustments

From the learned recourse strategy we obtain the predictions $\hat{a}_i^t$ of a_i^t for all $i \in \{0 \dots N_t - 1\}$ and $\hat{a}_i^h$ of a_i^h for all $i \in \{0 \dots N_v - 1\}$.

Assuming we predict these values with a regression algorithm, we do not necessarily obtain binary values. A first step is thus to rank the predictions, set the K first ones to 1 and the remaining ones to 0. Then the cardinality constraint (B.31) is dropped since it is satisfied a priori, we formulate a problem according to Appendix B.1 where only K generation units can be freely adjusted on the recourse horizon, and we impose the reference schedule for the remaining generation units.

Appendix C

Related optimization algorithms

C.1 Mixed Integer Linear Programming

The presence of discrete decision variables in the problems described in Chapter 2 and detailed in Appendix B, even though these problems are linear, greatly complicates the resolution process. Solving a continuous relaxation generally results in a solution which violates the integrality constraints, and obtaining a near optimal and feasible solution from this point is far from trivial. There exists an abundant literature about the generation scheduling problem, referred to as the *unit commitment* problem, and about the associated solution techniques (see for example Padhy (2004)). We recall in Sections C.1.1 and C.1.2 the ideas underlying the most widespread algorithms for the type of generation system we consider.

C.1.1 Lagrangian relaxation

One of the most widespread techniques to solve real instances – i.e. containing a large number of generation units and a large number of time steps – or stochastic instances – i.e. resorting to multiple scenarios – of the problem is based on Lagrangian relaxation (Carpentier et al., 1996; Dubost et al., 2005; Merlin and Sandrin, 1983). More complete introductions to Lagrangian relaxation and in particular in the context of integer programming may be found in Lemaréchal (2001); Minoux (2008). One possible usage of Lagrangian relaxation to solve Formulation 5 is to decompose the problem spatially, i.e. unit by unit, by relaxing the coupling constraints (2.2). The latter are included in the objective function (2.1) through the Lagrange multipliers $\lambda_t \in \mathbb{R}^3$. One thus assumes that the problem of finding a feasible schedule for one generation

unit is relatively easy but the coupling constraints are complicating.

$$\mathcal{L}(x_i, u_i, v_i, x_{slack}; \lambda) = \sum_{t=0}^{T} \Bigg(L(x_{slack,t}) + \sum_{i \in I} R_i(x_{i,t}, u_{i,t}) - \lambda_t^\top \big(x_{slack,t} - \sum_{i \in I} C_i x_{i,t} + Y_{req,t} \big) \Bigg) \tag{C.1}$$

For fixed values of the Lagrange multipliers, solving the resulting problem

$$\begin{aligned} g(\lambda) = \min_{x,u,x_{slack}} \quad & \mathcal{L}(x, u, x_{slack}; \lambda) \\ \text{s.t.} \quad & (2.4) \text{ and } (2.5). \end{aligned} \tag{C.2}$$

amounts to solving the problems of scheduling each unit independently, the Lagrangian multipliers playing the role of coordinators. One can thus apply specialized and efficient algorithms to solve the sub-problems (dynamic programming, interior point methods, ...), and solve them in parallel.

The dual function $g(\lambda)$ is piecewise affine and concave since it is a minimum of a finite set of affine functions of λ obtained by fixing the primal variables in (C.1). Its value is always lower or equal to the optimal value of the problem of Formulation 5 since its feasibility domain is larger and the multipliers are freely adjustable (weak duality). The problem

$$\max_{\lambda} g(\lambda)$$

is thus a concave but not necessarily differentiable maximization problem, requiring a carefully designed subgradient algorithm (e.g. the bundle algorithm described in Pellegrino et al. (1996)).

If the optimal solution of the dual corresponds to a primal feasible solution, then this primal solution is optimal. But it is generally not the case in a mixed integer context (no strong duality). In particular in the generation scheduling context, the solution of the relaxed problem associated with the optimal solution of its dual does not satisfy sufficiently well the coupling constraints. Hence the use of a heuristic or a second optimization phase is needed to get a solution satisfying better the relaxed constraints. For example the Augmented Lagrangian (cf. Pellegrino et al. (1996)) technique converges to a primal solution which leads to a lower discrepancy between generation and demand. It consists in adding a second penalization term containing the squared relaxed constraint to (C.1).

C.1.2 Branch-and-cut

As the problem can be stated as a MILP, it is also possible to use general purpose software applicable to this kind of problem. The most widely used algorithm to solve MILP instances in practice is *branch-and-cut*, which combines

the *branch-and-bound* and *cutting-planes* algorithms. Both techniques rely on the continuous relaxation of the MILP that provides a lower bound on the optimal value of the problem. We give a brief introduction to these topics below; see Wolsey (1998) for a deeper introduction.

C.1.2.1 Branch-and-bound

Assume that we want to solve the mixed integer linear problem P defined as

$$\max \left\{ c^\top x | Ax = b, x \in \mathbb{R}_+^K \times \mathbb{Z}_+^L \right\}.$$

Branch-and-bound (Figure C.1) relies on LP relaxations of the problem (see Bertsimas and Tsitsiklis (1997) for a good introduction to linear programming) and on an enumeration procedure on the values of the integer variables. The relaxation

$$\max \left\{ c^\top x | Ax = b, x \in \mathbb{R}_+^{K+L} \right\},$$

is solved at the root node of a tree and provides a first upper bound z_U on the solution of the problem. If the solution of the relaxation satisfies the integrality requirement of the original problem P, then it is also an optimal solution for P. Otherwise, an enumeration of the possible values of the integer variables starts: two new problems containing a restriction on the admissible range of one decision variable x_i are created (e.g either $x_i \leq 2$, or $x_i \geq 3$). Typically one chooses a variable x_i that is fractional in the solution of the relaxation although it is required to be integer in P. One new node is created for each problem and is linked to the root node. We then repeat the same procedure for these nodes (solve the LP-relaxation, analyze the solution, branch), until we have enumerated all the possibilities (all the restrictions on the range of the integer variables). However, each time an integral solution is found we can stop developing the current node because we will not find any better solution if we continue developing the tree under this node, and update the current lower bound z_L if the value of the current solution is greater (initially $z_L = -\infty$). We can also stop developing the current node if the problem is infeasible or if the value of the LP relaxation is not greater than z_L. In the latter case, the value of any integral solution of a successor of the current node will be worse than the current best integral solution. The update of the bound z_U in step 4 is more complicated and depends on the way the nodes of the tree are visited (i.e. depth-first search, breadth-first search or another strategy). Nevertheless the information gathered when solving the intermediate problems allows to compute the integrality gap

$$(z_U - z_L)/z_U,$$

i.e. the maximal relative distance of the current best integer solution to the optimal solution. In practice we can stop the development of the branch-and-bound tree when the integrality gap becomes smaller than a tolerated threshold, or impose a time limitation.

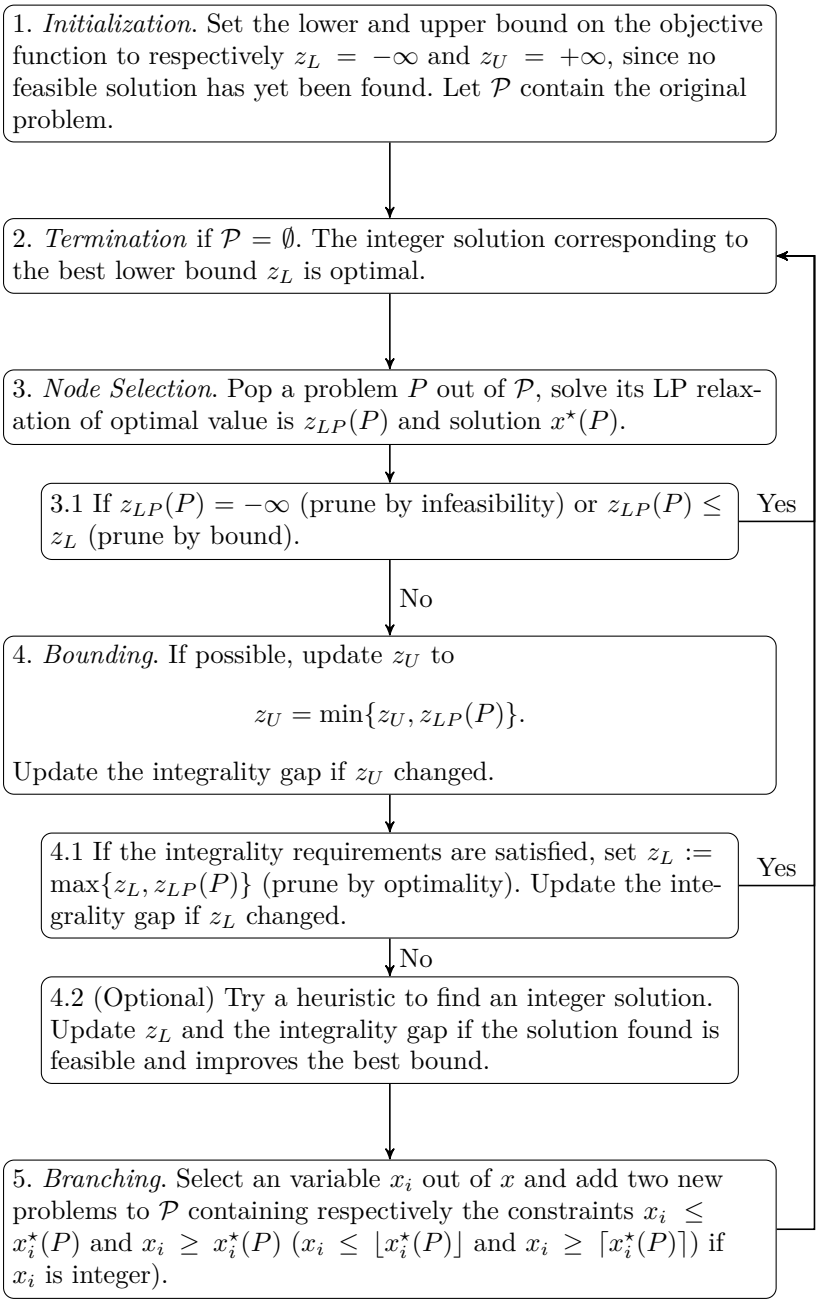


Figure C.1: The branch-and-bound algorithm.

Example 8.
Consider the following problem:

$$\begin{aligned} \max \quad & 5x_1 + 11x_2 && \text{(C.3)} \\ s.t. \quad & x_1 \leq 6 && \text{(C.4)} \\ & x_1 - 3x_2 \geq 1 && \text{(C.5)} \\ & 3x_1 + 2x_2 \leq 19 && \text{(C.6)} \\ & x_1, x_2 \in \mathbb{Z}_+ && \text{(C.7)} \end{aligned}$$

Figure C.2 shows a branch-and-bound tree for this problem.

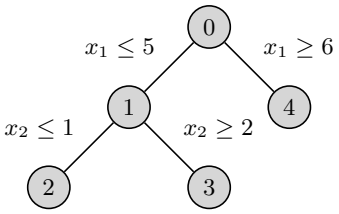
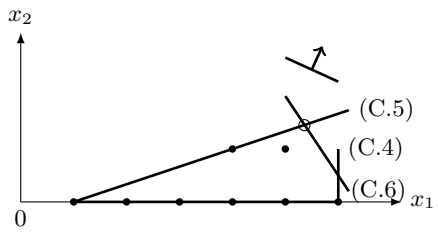
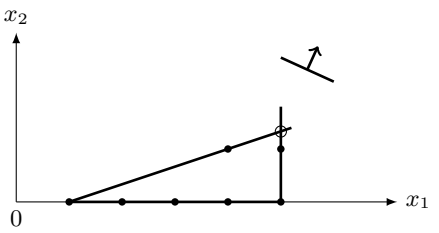


Figure C.2: Branch and bound tree.

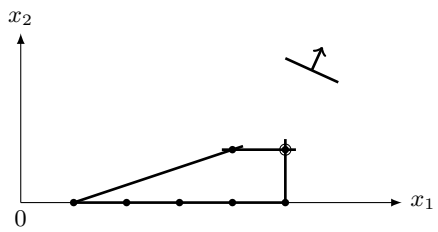
The node numbers correspond to the exploration order. Figure C.3a depicts the root LP relaxation, which provides a fractional solution and a first upper bound $z_U = 42.82$. If one chooses to branch on $x_1 \leq 5$ (Figure C.3b) one gets a better upper bound but still no integer solution, since $x^{\star,1} = (5, 4/3)$. At node 2 the LP relaxation provides an integer solution and a lower bound z_L of 36 (prune by optimality), while



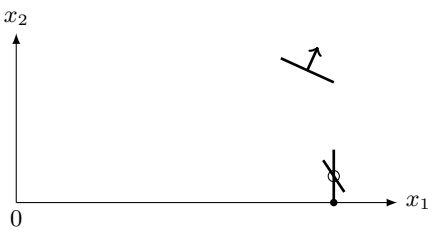
(a) Node 0: $z^{\star,0} \approx 42.82$ and $x^{\star,0} \approx (5.36, 1.45)$.



(b) Node 1: $z^{\star,1} \approx 39.67$ and $x^{\star,1} = (5, 4/3)$.



(c) Node 2: $z^{\star,2} = 36$ and $x^{\star,2} = (5, 1)$.



(d) Node 4: $z^{\star,4} = 35.5$ and $x^{\star,4} = (6, 1/2)$.

Figure C.3: Illustration of branch-and-bound. Subfigures (a) to (d) illustrate the linear relaxations at some nodes of the tree. Integer points inside the feasible region are highlighted by • symbols.

the problem of node 3 has no solution (prune by infeasibility). Coming back to the first branching option on x_1 at node 4, one gets a fractional solution whose value is 35.5. The exploration of the sub-tree attached to this node can thus be stopped (prune by bound) since 35.5 is an upper bound on the best integer solution that may be found in that sub-tree and that an integer solution with a value of 36 was found at another node.

C.1.2.2 The cutting plane algorithm

The cutting plane algorithm iteratively modifies the problem in order to tighten the feasibility region of its LP relaxation towards the convex hull (Figure C.4) of the integer feasible points.

To illustrate why this is a good idea, consider for example the simplex algorithm, which always finds a solution of the LP relaxation[1] lying on one vertex of the polyhedron defined by the constraints. If the polyhedron is the convex hull of the integer feasible points, its vertices are thus integral, and the simplex algorithm finds an integral solution.

The modification of the feasibility region of the LP relaxation is achieved by adding valid inequalities (or cuts). A valid inequality is a constraint inferred from the MILP formulation that cuts off a part of the LP relaxation's feasibility region containing no integer points. The main concern is to identify automatically and efficiently some cuts that define facets of the convex hull, or at least that cut off a "significant" part of the LP relaxation's feasibility region containing no integer points. Many classes of cuts exist (MIR cuts, flow cuts, ...) and can be automatically generated using state of the art solvers.

There is no guarantee of termination of the cutting plane algorithm.

Example 9.
For problem (C.3)–(C.7), it is easy to see from constraints (C.5), (C.4) and the integrality requirement that

$$x_2 \leq 1. \tag{C.8}$$

Then any point of $\mathbb{Z}_+^2$ satisfying (C.6) and (C.8) also satisfies

$$x_1 + x_2 \leq 6. \tag{C.9}$$

Adding (C.6) to (C.8) yields $3x_1 + 3x_2 \leq 20$ and thus $x_1 + x_2 \leq 20/3$. The right hand side of this constraint may be set to $\lfloor 20/3 \rfloor = 6$ since $x_1 + x_2 \in \mathbb{Z}_+$. Taking into account the cuts (C.8) and (C.9), one gets the solution $x^\star = (5, 1)$ directly instead of the solution $x^\star_{LP} \approx (5.36, 1.45)$ of the original LP relaxation.

C.1.2.3 Combining branch-and-bound and cutting planes

Branch-and-cut is a branch-and-bound algorithm which adds cuts before and during the exploration of the tree to obtain better bounds and thus limit the exploration of the tree. The dual simplex can be advantageously used in both the branch-and-bound and cutting planes algorithms, and thus in branch-and-cut, since adding a constraint to the primal problem (a bound or a cut) is

[1] That we suppose feasible and bounded.

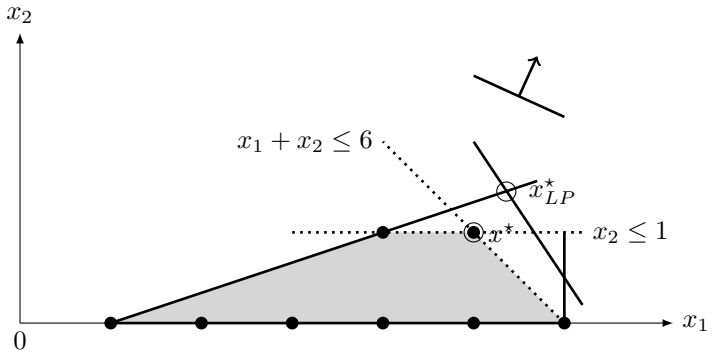


Figure C.4: Illustration of the convex hull (filled in gray) of the feasible integer points of problem (C.3)–(C.7) and of the two cuts (C.8) and (C.9) (dotted lines).

equivalent to adding a new variable to the dual problem. Doing so the optimal solution and basis of the dual problem (without the additional primal constraint) remain feasible, but not necessarily optimal. One can thus start from the solution of the previous problem and find the new solution typically in a few iterations.

Branch-and-Cut is more and more used in the short-term generation planning context because it yields feasible near optimal planning without heuristics, and thanks to the availability of efficient computer implementations such as CPLEX (ILOG, 2007). Note however that some specially structured MILP problems resist to state-of-the art branch-and-cut implementations, meaning that they impose a complete exploration of the tree to guarantee optimality, which is not achievable for moderately large MILP instances. More information about this and other MILP solution algorithms can be found in Louveaux (2004).

C.1.2.4 Special Ordered Sets and specialized branching techniques

Sometimes sets of variables have special properties which can be used to enhance the branch-and-cut algorithm (de Farias et al., 2001). Special Ordered Sets of type 1 (SOS1) arise if among several variables at most one can be nonzero, and are particularly useful if these variables correspond to a given order (e.g. to generation levels). Cardinality constraints are a generalization of SOS1 where at most k ($1 \leq k \leq n$) can be nonzero. Special Ordered Sets of type 2 (SOS2) arise if among several variables at most two adjacent variables can be nonzero and can be used to model piecewise linear functions.

A classical way to express the SOS1 property is to use binary variables x_i, $i = 1, \ldots, n$ and to impose $\sum_{i=1}^{n} x_i \leq 1$. However, if a relaxed solution yields two nonzero components x_l and x_m with $1 \leq l \leq m \leq n$, one can divide the problem in two by choosing an index s such that $l \leq s \leq m$ and creating to nodes, by imposing in the first $x_1 = x_2 = ... = x_s = 0$ and in the second $x_{s+1} = x_{s+2} = ... = x_n = 0$ instead of branching either on $x_l = 0$ or $x_l = 1$, or

on $x_m = 0$ or $x_m = 1$. Cuts can also be derived to tighten the LP formulation.

Both SOS1 and SOS2 are used in the formulation of Appendix B.1.2. Cardinality constraints arise when we want to limit the number of modifications one can make to a schedule to adjust it to a new scenario.

References

M. Avellà Fluvià, K. Boukir and P. Martinetto. Handling a CO_2 reservoir in mid term generation scheduling. In *Proceedings of Power Systems Computation Conference, Liège*. 2005. *(155).*

R. Aïd, V. Grellier, A. Renaud and O. Teytaud. Application de l'apprentissage par renforcement a la gestion du risque. *Journal Electronique d'Intelligence Artificielle*, **6**:1–23, 2004. *(40, 155).*

G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, B. Taskar and S. Vishwanathan, editors. *Predicting Structured Data*. The MIT Press, 2007. *(27).*

K. Barty, P. Carpentier and P. Girardeau. Decomposition of large-scale stochastic optimal control problems. *RAIRO Operations Research*, **44**:167–183, 2010. *(155).*

M. Belkin, P. Niyogi and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, **7**:2399–2434, 2006. *(146, 147).*

A. Ben-Abbes, E. Rachelson and S. Diemer. L'apprentissage au secours de la réduction de dimension pour des problèmes d'optimisation. In *Conférence Francophone sur l'Apprentissage Automatique*. 2010. *(125).*

D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997. *(181).*

J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer, 1997. *(155).*

L. Breiman. Bagging predictors. *Machine Learning*, **24**(2):123–140, 1996. *(141, 161).*

———. Random forests. *Machine Learning*, **45**(1):5–32, 2001. *(141, 161).*

L. Breiman, J. Friedman, R. Olshen and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, 1984. *(129, 161).*

P. J. Brockwell and R. A. Davis. *Introduction to Time Series and Forecasting*. Springer, 1996. *(82).*

L. Busoniu, R. Babuska, B. D. Schutter and D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, 2010. *(29).*

P. Carpentier, G. Cohen, J. Culioli and A. Renaud. Stochastic optimization of unit commitment: a new decomposition framework. *IEEE Transactions on Power Systems*, **11**(2):1067–1073, 1996. *(43, 54, 55, 131, 179).*

M. Carrion and J. Arroyo. A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. *IEEE Transactions on Power Systems*, **21**(3):1371–1378, 2006. *(166).*

C. Carøe and R. Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, **24**(1-2):37 – 45, 1999. *(15, 53, 54, 55).*

O. Chapelle, B. Schölkopf and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006. *(142).*

D. Cohn, L. Atlas and R. Ladner. Improving generalization with active learning. *Machine Learning*, **15**:201–221, 1994. *(130).*

B. Cornélusse, C. Wera and L. Wehenkel. Automatic learning for the classification of primary frequency control behaviour. In *Proc. IEEE Power Tech Conference, Lausanne*, pages 273–278. 2007. *(34).*

B. Cornélusse. *Application of Supervised Learning to very short term decision making for electric power generation.* Master's thesis, Université de Liège, 2008. *(70, 92).*

B. Cornélusse, P. Geurts and L. Wehenkel. Tree based ensemble models regularization by convex optimization. In *International NIPS Workshop on Optimization for Machine Learning, Canada*. 2009a. *(35, 135).*

B. Cornélusse, G. Vignal, B. Defourny and L. Wehenkel. Supervised learning of intra-daily recourse strategies for generation management under uncertainties. In *Proc. IEEE Power Tech Conference, Bucharest*. 2009b. *(34, 49, 75).*

I. de Farias, E. Johnson and G. Nemhauser. Branch-and-cut for combinatorial optimization problems without auxiliary binary variables. *The Knowledge Engineering Review*, **16**(01):25–39, 2001. *(176, 185).*

B. Defourny. *Machine Learning Solution Methods for Multistage Stochastic progamming problems*. Ph.D. thesis, Université de Liège, 2010. *(128).*

B. Defourny, D. Ernst and L.Wehenkel. Bounds for multistage stochastic programs using supervised learning strategies. *Lecture Notes in Computer Science*, **5792**:61–73, 2009. *(128).*

B. Defourny and L. Wehenkel. Projecting decisions induced by a stochastic program on a family of supply curve functions. In *Proc. of Third Carnegie Mellon Conference in Electric Power Systems*. 2007. *(128).*

D. Dentcheva and W. Römisch. Optimal power generation under uncertainty via stochastic programming. *Lecture Notes in Economics and Mathematical Systems*, **458**:22–56, 1998. *(55).*

L. Dubost, R. Gonzalez and C. Lemaréchal. A primal-proximal heuristic applied to the french unit-commitment problem. *Mathematical Programming*, **104**:129–151, 2005. *(179).*

J. Dupacova, G. Consigli and S. W. Wallace. Scenarios for multistage stochastic programs. *Annals of Operations Research*, **100**(1 - 4):25–53, 2000. *(15).*

D. Ernst, P. Geurts and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, **6**:503–556, 2005. *(29).*

D. Ernst, M. Glavic, F. Capitanescu and L. Wehenkel. Reinforcement learning versus model predictive control: a comparison on a power system problem. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, **39**:517 – 529, 2009. *(30).*

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156. 1996. *(125, 141).*

P. Geurts. *Contributions to decision tree induction: bias/variance tradeoff and time series classification.* Ph.D. thesis, University of Liège, Belgium, 2002. *(26, 161).*

P. Geurts, D. Ernst and L. Wehenkel. Extremely randomized trees. *Machine Learning*, **36**(1):3–42, 2006a. *(141, 161).*

P. Geurts, L. Wehenkel and F. d'Alché-Buc. Kernelizing the output of tree-based methods. In *Proceedings of ICML*, pages 345–352. 2006b. *(27).*

R. Gollmer, M. Nowak, W. Römisch and R. Schultz. Unit commitment in power generation - a basic model and some extensions. *Annals of Operations Research*, **96**:167–189, 2000. *(54).*

N. Gröwe-Kuska, K. Kiwiel, M. Nowak, W. Römisch and I. Wegner. Power management under uncertainty by lagrangian relaxation. In *Proceedings of the International Conference Probabilistic Methods Applied to Power Systems*, volume 2. 2000. *(55).*

———. *Decision Making under Uncertainty: Energy and Power*, chapter Power management in a hydro-thermal system under uncertainty by Lagrangian relaxation. Springer-Verlag, 2002. *(55).*

N. Gröwe-Kuska and W. Römisch. *Applications of Stochastic Programming*, chapter Stochastic unit commitment in hydro-thermal power production planning. SIAM, 2005. *(80).*

T. Hastie, R. Tibshirani and J. Friedman. *The Elements of Statistical Learning. Data Mining, Inference and Prediction.* Springer, 2009. *(16, 23).*

H. Heitsch and W. Römisch. Generation of multivariate scenario trees to model stochasticity in power management. In *Proceedings of IEEE Power Tech, St. Petersburg*. 2005. *(55).*

V. Huynh-Thu, L. Wehenkel and P. Geurts. Exploiting tree-based variable importances to selectively identify relevant variables. *JMLR: Workshop and Conference Proceedings*, **4**:60–73, 2008. *(162).*

ILOG. *ILOG CPLEX 11.0 User's manual*, 2007. *(76, 185).*

L. Kaufman and P. Rousseeuw. *Finding Groups on Data: an Introduction to Cluster Analysis.* John Wiley & Sons, 2005. *(18, 81).*

F. Lauer and G. Bloch. Incorporating prior knowledge in support vector machines for classification: A review. *Neurocomputing*, **71**(7-9):1578 – 1594, 2008a. *(142).*

———. Incorporating prior knowledge in support vector regression. *Machine Learning*, **70**:89–118, 2008b. *(135, 142, 148).*

C. Le Goazigo and J. Collet. Générateur de scénarii d'erreurs sur le modèle de prévision moyen-terme. Technical report, EDF, 2006. *(82).*

C. Lemaréchal. *Computational Combinatorial Optimization*, volume 2241/2001 of *Lecture Notes in Computer Science*, chapter Lagrangian Relaxation. Springer, 2001. *(179).*

Q. Louveaux. *Exploring Structure and Reformulations in Different Integer Programming Algorithms.* Ph.D. thesis, Université Catholique de Louvain, 2004. *(185).*

U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, **17**(4):395–416, 2007. *(147).*

J. Löfberg. Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*. Taipei, Taiwan, 2004. *(36).*

J. Maciejowski. *Predictive Control with Constraints.* Prentice Hall, 2002. *(9).*

F. Maes. *Learning in Markov Decision Processes for Structured Prediction.* Ph.D. thesis, Université de Paris VI, Pierre-et-Marie-Curie, 2009. *(27).*

O. Mangasarian, J. Shavlik and E. Wild. Knowledge-based kernel approximation. *Journal of Machine Learning Research*, **5**:1127–1141, 2004. *(138).*

A. Merlin and P. Sandrin. A new method for Unit Commitment at Electicité de France. *IEEE transactions on power apparatus and systems*, **102**(5):1218–1225, 1983. *(179).*

M. Minoux. *Programmation mathématique*. Lavoisier – Tec & Doc, 2008. *(179).*

A. Märkert and R. Schultz. On deviation measures in stochastic integer programming. *Operations Research Letters*, **33**(5):441 – 449, 2005. *(11).*

M. Nowak and W. Römisch. Stochastic lagrangian relaxation applied to power scheduling in a hydro-thermal system under uncertainty. *Annals of Operations Research*, **100**(1-4):251–272, 2000. *(55).*

M. Nowak, R. Schultz and M. Westphalen. A stochastic integer programming model for incorporating day-ahead trading of electricity into hydro-thermal unit commitment. *Optimization and Engineering*, **6**:163–176, 2005. *(55).*

R. Nürnberg and W. Römisch. A two-stage planning model for power scheduling in a hydro-thermal system under uncertainty. *Optimization and Engineering*, **3**(4):355–378, 2002. *(55).*

N. Padhy. Unit commitment — a bibliographical survey. *IEEE Transactions on Power Systems*, **19**(2):1196–1205, 2004. *(179).*

F. Pellegrino, A. Renaud and T. Socroun. Bundle and Augmented Lagrangian Methods for Short-Term Unit Commitment. In *Proceedings of Power Systems Computation Conference*, volume 2, pages 730–739. 1996. *(180).*

T. Poggio, R. Rifkin, S. Mukherjee and P. Niyogi. General conditions for predictivity in learning theory. *Nature*, **428**, 2004. *(24).*

M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994. *(28).*

A. Renaud. Daily generation management at electricite de france: from planning towards real time. *IEEE transactions on Automatic control*, **38**:1080–1093, 1993. *(54, 56).*

R. Rockafellar and R.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, **16**:119–147, 1991. *(15, 53).*

RTE. *Mémento de la sûreté du système électrique*. RTE, 2004. *(43).*

P. Ruiz, C. Philbrick, E. Zak, K. Cheung and P. Sauer. Applying stochastic programming to the unit commitment problem. In *Proceedings of the International Conference Probabilistic Methods Applied to Power Systems*. 2008. *(55).*

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001. *(164).*

R. Schultz and C. Carøe. A two-stage stochastic program for unit commitment under uncertainty in a hydro-thermal power system. In *Preprint SC 98-11, Konrad-Zuse-Zentrum fur Informationstechnik*, pages 98–13. 1998. *(54, 55).*

A. Shapiro, D. Dentcheva and A. Ruszczynski. *Lectures on Stochastic Programming: Modeling and Theory*. SIAM, 2009. *(11).*

P. Shivaswamy, W. Chu and M. Jansche. A support vector approach to censored targets. In *Proceedings of the IEEE International Conference on Data Mining*, pages 655–660. 2007. *(145, 146).*

A. Smola and B. Schölkopf. A tutorial on support vector regression. Technical report, NeuroCOLT2 Technical Report Series, 2003. *(162).*

J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. 1998. *(36).*

S. Takriti, J. R. Birge and E. Long. A stochastic model for the unit commitment problem. *IEEE Transactions on Power Systems*, **11**:1497–1508, 1996. *(53, 54, 55).*

UCTE. Load-frequency control and performance (appendix 1). *UCTE Operation Handbook*, 2004. *(43).*

———. Load-frequency control and performance (policy 1). *UCTE Operation Handbook*, 2009. *(43).*

V. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998. *(22).*

L. A. Wolsey. *Integer Programming*. Wiley-Interscience, 1998. *(181).*

A. Wood and B. Wollenberg. *Power generation, operation and control*. Wiley-Interscience, 1996. *(40).*