Grenoble INP – ENSIMAG
École Nationale Supérieure d'Informatique et de Mathématiques Appliquées

2nd year internship report

Conducted at EDF R&D - OSIRIS department

Incorporation of stochastic demands into the daily optimization program of electricity production at EDF

keywords : stochastic optimization, nonsmooth optimization, electricity production

Lebbe Nicolas [1]

2A – MMIS specialization

From June 22, 2015 to September 11, 2015 (3 months)

EDF R&D
1, avenue du Général de Gaulle
BP 408
92141 Clamart

Internship supervisor
Wim van Ackooij [2] (**EDF**)
Co-supervisor
Jérôme Malick [3] (**INRIA**)
Ensimag reviewer
Franck Hetroy

1. nicolas.lebbe@ensimag.grenoble-inp.fr
2. wim.van-ackooij@edf.fr
3. jerome.malick@inria.fr

Contents

1 Introduction

1.1 Workspace

I did my second-year internship within **EDF** at the **R&D** division and more specifically in the **OSIRIS** (**O**ptimisation, **SI**mulation, **RI**sques et **S**tatistiques pour les marchés de l'énergie) department which principally works on the creation of software to optimize (the costs of) electrical production. (on both theoretical and practical side)

Various aspects of energy management (investment, nuclear maintenance, scheduling, unit-commitment, ...) have lead to the development of mathematical models, and associated software for decision making. Globally, the time horizon is split as short, medium and long term planification. My internship focuses on the first one.

1.2 Context

Short-term production management aims at computing a production schedule in limited time (less than half an hour) that must meet the operational constraints of production units (hydroelectric complex, power plant, thermal power station ...) and meet the energy demand (which is approximately known in advance) at the lowest possible cost.

In this context, **EDF** now studies how to account for uncertainty on their models. Indeed, the electricity produced by renewable energy (unlike nuclear or thermal power) is sensitive to the weather, so it is desirable to take into account different scenarios of weather when calculating the production schedules. The goal is to still provide reliable resources, while taking into account renewable energy at best.

1.3 Topic

The purpose of my internship is to model an approach taking into account the uncertainties in decision making, to implement an algorithm to solve this model efficiently and finally to compare this algorithm with those already existing.

For the technical parts of this document, I will assume that the reader is familiar with standard techniques of numerical optimization such as *Lagrangian dualization* [8].

Note : My internship included a significant theoretical mathematical part. In order to simplify my report, demonstrations and details of calculations are only attached in appendices.

2 Definition and motivation

2.1 Starting point : « unit-commitment »

In energy management, a key problem known as « unit-commitment » deals with finding a minimal cost production schedule that satisfies the operational constraints of the production units and that meets customer load as closely as possible.

Each day, the **EDF** decision algorithm must establish in less than 30 minutes the production schedule of each unit for the next 96 half hour (= 2 days).
To achieve this goal, we know a-priori energy demand of the next two days, discretized by steps of 30 minutes (i.e., for each time t multiple of 30 minutes for the next two days, we almost know the quantity of energy that we will have to produce). This « load » is produced by sophisticated forecasting software.

Mathematically, « unit-commitment » refers to the following optimization problem :

$$\begin{array}{rl} \min\limits_{x\in\mathbb{R}^{n\times T}} & f(x) \\ s.t. & x \in X^1 \cap X^2 \end{array} \tag{1}$$

Where :

- $n \in \mathbb{N}$ is the number of units (hydraulics, thermal ... ~ 200) and $T \in \mathbb{N}$ the number of time discretizations. ($T = 2\text{days} \times \frac{24\text{hours}}{30\text{minutes}} = 96$)
- $x = (x_1, ..., x_n) \in \mathbb{R}^{n\times T}$ is the schedule of each thermal and hydraulics units (a.k.a. the decision vector). For all i from 1 to n, $x_i \in \mathbb{R}^T$ (from now on I will use the notation $x_{i,t}$ for the production schedule at time t of the i-th unit but I insist on the fact that x is a vector, not a matrix. Computationally we have $x_{i,t} = x_{i\times T+t}$).
- $X^1 = \prod_{i=1}^n X_i^1$ is the set of production constraints which are local to each unit.
  For example, a nuclear power plant can only produce some levels of electrical power spaced by a certain amount. Starting-up a thermal unit takes some time and there is a daily constraints on the number of times a unit can be switch on. Hydraulic valleys also have a lot of constraints such as the flow bounds represented in the Figure 1.
- X^2 represent the constraints on demand. For example if we want to limit the gap between $D \in \mathbb{R}^T$ the (exact) known demand and the production schedule, X^2 could be defined as the polyhedral set $\{x : \underline{d} \leq D - Ax \leq \overline{d}\}$ where $\underline{d}, \overline{d} \in \mathbb{R}^T$ are tolerances and $A \in \mathcal{M}_{T,n\times T}(\mathbb{R})$ a matrix such that for all t, $(Ax)_t = \sum_{i=1}^n x_{i,t}$[1].
- $f : \mathbb{R}^{n\times T} \to \mathbb{R}$ is the cost of production for a given schedule and is defined by

$$f : x \quad \mapsto \quad \sum_{i=1}^{n} f_i(x_i)$$

  with the linear functions $f_i : \mathbb{R}^T \to \mathbb{R}$ which give the cost for unit i to produce x_i.
  An important point is that we have the possibility to solve (approximately) the following sub-problem for each unit :

$$\begin{array}{rl} \min\limits_{x_i} & f_i(x_i) + \text{easy-term}(x_i) \\ s.t. & x \in X_i^1 \end{array}$$

The easy term is linear or convex quadratic of the form $x_i \mapsto \langle u, x_i\rangle + c\,\|x_i - a\|^2$.

[1] The matrix form of A is $A = \underbrace{(I_T, \ldots, I_T)}_{n \text{ times}}$

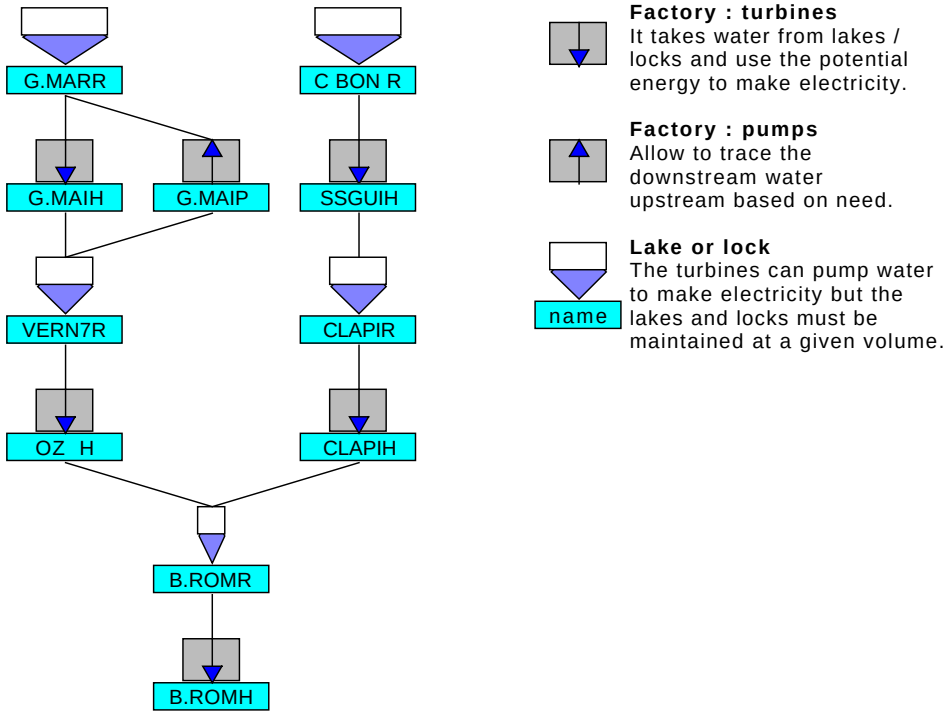


FIGURE 1 – Graphical representation of an hydraulic valley : there are lakes and locks which are linked with factories which can produce electricity with turbines (thus the water goes from a water pool to a lower one) and factories which pump water to an upper lake or lock, while consuming energy.

2.2 Stochastic demands : « two-stage unit-commitment »

As mentioned in the introduction, the purpose of my internship is to add the consideration of uncertainty in energy demand. The D mentioned in the definition of X_2 is now a set $\mathcal{D}$ which contains possible scenarios of demands.

To this end, rather than putting the constraint of production in the domain of x, we can instead penalize the distance to the demand : this means accepting a production schedule not meeting exactly the demand but « at a certain price ».
To this end, we will have to use a function $\psi(d;\cdot)$ which will (supposing that the demand is d) give a penalization cost $\psi(d;x)$ for the production schedule x. An example of such function could be the one represented in Figure 2.

Now we can easily consider more than one possibility of demand. In fact, if we model uncertainty as a finite set of possible demands $\mathcal{D}$ (with an associated distribution of probability), we can now use a penalization function like :

1. $\Psi_{\mathbb{E}}(x) = \mathbb{E}(\psi(d;x))$. (to consider the « average cost »)
2. $\Psi_{\mathbb{P}}(x) = \mathbb{P}(\psi(d;x) > \epsilon, d \in \mathcal{D})$. (a sort of Value at Risk)
3. $\Psi(x) = \sup_{d \in D} \psi(d;x)$. (we always consider the worst case : « robust optimization »)

During my internship, I only studied the last one. At EDF, ψ is a six-segments function and the penalization cost is summing over all time $t \in \{1,\dots,T\}$. Mathematically, this leads to the

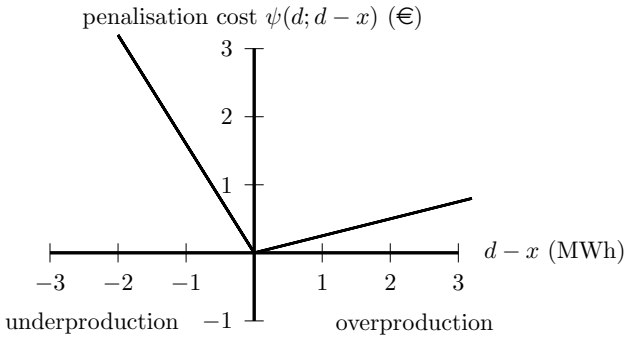


FIGURE 2 – A simple one-dimensional two-segments penalization function.
Here we define ψ as $\psi(d;x) = \sum_{t=1}^{T} \psi_t(d;x)$ with $\psi_t(d;x) = \max(\alpha(d_t - x_t), \beta(d_t - x_t))$ with $\alpha < 0$ and $\beta > 0$ and $|\alpha| > |\beta|$. In this definition of $\psi(d;x)$ we see that underproduction will cost more than overproduction, this reflects the fact that if we have more electricity than requested, we can sell this. And in the other case, we will have to buy electricity on the market at high prices.

following definition of Ψ.
(this definition may seem complex but it is not important for the comprehension of the next sections)

$$\Psi(x) = \sup_{d \in \mathcal{D}} \sum_{t=1}^{T} \max_{i=1,\dots,6} (a_i(d_t - (Ax)_t) + b_i), \tag{2}$$

where $a_i, b_i \in \mathbb{R}$ and $A \in \mathcal{M}_{T,n\times T}(\mathbb{R})$.

Having this in mind we can now define the so-called (2-stage) robust optimization model I have studied :

$$\boxed{\begin{array}{cc} \min\limits_{x \in \mathbb{R}^{n\times T}} & f(x) + \Psi(x) \\ s.t. & x \in X^1 \end{array}} \tag{3}$$

As we can see, we now have an optimization problem which contains itself another optimization problem ($\Psi(x) = \sup_{d\in\mathcal{D}} \dots$). This is the principle of the « two-stage » optimization : in the « first-stage » we minimize other the set X^1 and with the obtained schedule we observe which d is selected during the « second-stage » by maximizing other the uncertainty set $\mathcal{D}$. Computing $\psi(\cdot; x)$ is then seen as the recourse action.

Depending on the shape of the set $\mathcal{D}$ it is possible to establish an **oracle** for the function Ψ, i.e, a numerical algorithm that computes the value of and a sub-gradient of Ψ at x.

2.3 The uncertainty set $\mathcal{D}$ and link with two-stage optimization

Here the second-stage is simple and explicit via Ψ, which differs from other approaches (the one of [14] in particular). Note that $\mathcal{D}\text{om}(\Psi) = \mathbb{R}^{n\times T}$; this is said that complete recourse decisions exist with respect to market conditions, which is a strong assumption (not met in practice).

There are at least three options for the uncertainty set $\mathcal{D} \subseteq \mathbb{R}^T$:

1. the set $\mathcal{D}$ has an infinite cardinal and is defined as a band around the average demand : $\mathcal{D} = \{d \in \mathbb{R}^T, \max_{t=1\dots T} |d_t - D_t| \leq k\}$ with D the average load and $k > 0$. In this case, it is possible to provide an explicit description of $\Psi(x)$, giving a simpler penalization function[2].

[2] see A.2 for details of calculations

Pros : allows to consider a very large number of scenarios.
Cons : some scenarios are not realistic because this set suppresses the temporal dependency.

2. the set $\mathcal{D}$ is a very specific set (see Figure 3 for a graphical representation), constructed following a so-called "state-space represented" model [9]. At each time step $t \in \tau$, we set up a set of nodes E_t containing both a value for D_t and a weight w_t. This set of nodes is assumed to be sorted according to increasing values for D_t. Using this set of nodes we set up a graph (G, V), $G = \bigcup_{t\in\tau} E_t$, wherein V connects all nodes in E_t to those in E_{t+1} that are not further apart than H, i.e., node $i \in E_t$ is connected to node $j \in E_{t+1}$ iff their local labels $|\mathcal{L}(i) - \mathcal{L}(j)| \leq H$. At least one of the nodes in E_t is assumed to have a zero weight and H is assumed to be such that each node is connected to the zero node. The set $\mathcal{D}$ is now given by all paths in (G, V) satisfying $\sum_{t\in\tau} w_t \leq W_{\texttt{max}}$ for a given maximum budget of uncertainty $W_{\texttt{max}}$. As a consequence, $|\mathcal{D}|$ is huge, but computing the sup over $\mathcal{D}$ amounts to applying a simple 1-dimensional dynamic programming principle.
Pros : more realistic scenarios and cover a wide spectrum of possible energy demand.
Cons : the temporal dependency has been artificially introduced with the budget uncertainty and may not be statistically relevant.

3. the set $\mathcal{D}$ is a finite set of possible demands, with $|\mathcal{D}|$ small (say between 50-250), as in [14]. In this case we could consider tackling the problem (3) with a « frontal » approach by expliciting $|\mathcal{D}|$ constraints in (3) and using the algorithm of the deterministic case. We could also just use a scenario that reaches the sup to make up an oracle.
Pros : simple computation.
Cons : the set may not cover a sufficient number of energy demands.

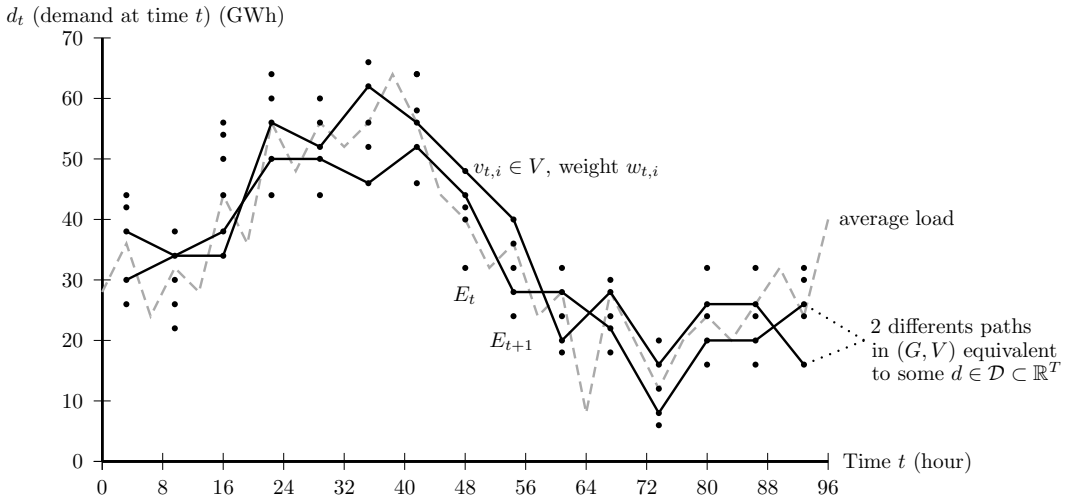


FIGURE 3 – graphical representation of the construction of the set $\mathcal{D}$ defined by M. Minoux in [14]

In the last two cases, the set $\mathcal{D}$ is finite, we can easily maximize over it, and therefore we have a (cheap) exact oracle for Ψ[3], as formalized in the next lemma. (a proof is given in appendix A.1)

Lemma 1 (Oracle for Ψ) *The function Ψ defined by (2) is convex for all x and we have an explicit expression of a subgradient of Ψ at x with a $\bar{d}_x \in \mathcal{D}$ achieving the sup.*

[3] We don't have an explicit value for Ψ, only a numerical algorithm to compute Ψ at a given x.

3 Theory for the non-regularized algorithm

3.1 Cutting-plane method for the master problem – (what was almost implemented)

Following [9], we propose a cutting-plane based method to solve (3). In contrast with [9] though, solving the approximated master-problems is difficult.

At iteration k, assume given $x^1, ..., x^k$, points (previously generated) at which the oracle has returned $\Psi(x^\ell)$ and $g^\ell \in \partial\Psi(x^\ell)$. This allows us to build the Kelley cutting-plane model $\check{\Psi}_k$ of Ψ defined as :

$$\check{\Psi}_k(x) := \max_{\ell \leq k} \left\{ \Psi(x^\ell) + \left\langle g^\ell, x - x^\ell \right\rangle \right\},$$

Since Ψ is convex it follows that $\Psi(x) \geq \check{\Psi}_k(x)$ for all $x \in X^1$.
In the cutting-plane method, the next iterate would then be given by the k-approximated master-problem (which has the same numerical complexity as a deterministic unit-commitment problem)

$$\boxed{\begin{array}{ll} \min\limits_{x \in \mathbb{R}^{n \times T}} & f(x) + \check{\Psi}_k(x) \\ s.t. & x \in X^1. \end{array}} \tag{4}$$

We attack this problem by moving to the Lagrangian dual space of

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^{n \times T}, r \in \mathbb{R}} & f(x) + r \\ s.t. & x \in X^1, \quad \check{\Psi}_k(x) \leq r. \end{array} \tag{5}$$

For a given Lagrange multiplier $\mu \in [0,1]^k$, this yields the following dual function :

$$\boxed{\begin{array}{ll} \max\limits_{\mu \in \mathbb{R}^k} & \Theta_k(\mu) \\ s.t. & \sum\limits_{i=1}^{k} \mu_i = 1, \end{array}} \tag{6}$$

where Θ_k is defined in Lemma 2 below.

Lemma 2 (Oracle for the k-th dual function) *The concave dual function can be expressed, for all $\mu \in \mathbb{R}^k$ such that $\sum_{\ell=1}^k \mu_\ell = 1$, as*

$$\Theta_k(\mu) := \min_{x \in X^1} f(x) + \left\langle \sum_{\ell=1}^{k} \mu_\ell g^\ell, x \right\rangle + \sum_{\ell=1}^{k} \mu_\ell \left(\Psi(x^\ell) - \left\langle g^\ell, x^\ell \right\rangle \right). \tag{7}$$

Thus solving the n sub-problems

$$\min_{x_i \in X_i^1} f_i(x_i) + \left\langle \sum_{\ell=1}^{k} \mu_\ell g_i^\ell, x_i \right\rangle$$

gives the value $\Theta_k(\mu)$, a sub-gradient

$$\left(\Psi(x^1) + \left\langle g^1, x(\mu) - x^1 \right\rangle, \cdots, \Psi(x^k) + \left\langle g^k, x(\mu) - x^k \right\rangle \right)^{\mathsf{T}} \in \partial\Theta_k(\mu)$$

and an associated primal solution $x(\mu) = (x_1(\mu), \ldots, x_n(\mu))$ which lies in X^1.

Although the optimization problem (5) is linear, it is impossible to solve it due to the huge number of variables. The dualization process will allow us to decompose the problem into smaller coordinated sub-problems.

Given that the f_i are linear, the sub-problems are problems of the general form $\min b^\mathsf{T} x_i + c$ with $Ax_i \leq d$ with x_i a mixed-integer vector so they are « easily » solved using the `CPLEX` solver.
Consequently, the dual function can then be maximized by a state-of-the-art bundle method ; see e.g., the textbook[6]. Note that we retain the best primal iterate denoted x^{k+1} which lies in X^1. This has the advantage of not needing any primal recovery heuristic but it is not optimal (see section 4.2 for further discussion).

In conclusion, solving the following maximization program will give us a primal feasible schedule x^{k+1} which allow us to add a new cut (i.e, define $\check{\Psi}_{k+1}$).

Algorithm 1. Cutting-plane (solve (3))

Step 1 (Initialization) Generate at least three initial and different elements $x \in X^1$ and use these to build an initial model for Ψ[4]. Set $k = 3$ and the stopping tolerance $\delta_{\texttt{stop}}$.

Step 2 (Lagrangian dual) Use a bundle method to solve (6). Over the course of the iterations of this bundle method save the iterate that produces the lowest value for $f(x) + \check{\Psi}_k(x)$. Let μ^{k+1} be the computed dual optimal solution and x^{k+1} the best primal iterate found.

Step 3 (Oracle call) Call the oracle for Ψ to compute a value and sub-gradient at x^{k+1}, enriching the model for Ψ used in (4).

Step 4 (Stopping test) Check if

$$\frac{\overbrace{(f+\Psi)}^{\text{real value}} - \overbrace{(f+\check{\Psi}_k)}^{\text{approximation}}}{(f+\Psi)}(x^{k+1}) \leq \delta_{\texttt{stop}}$$

$$\Leftrightarrow$$

$$\left(\Psi - \check{\Psi}_k\right)(x^{k+1}) \leq \delta_{\texttt{stop}}\,(f+\Psi)\,(x^{k+1})$$

and stop the algorithm if this holds.
Let $k = k + 1$ otherwise and **return in Step 1**.

This algorithm does not fit in the recent development of inexact non-smooth optimization methods (see e.g. [3] and reference therein) where the inexactness is the one of the oracle. Here the oracle is cheap and exact but the k-approximating master-problem is solved inexactly. The convergence of this algorithm follows from the usual rationale ; however the accuracy of the final iterate - though estimated - is not controlled explicitly due to the non-convexity of X^1. (see B.1 for convergence proof)

Numerical experiments show that in practice the number of iterations k needed to converge is huge (see for example Figure 1). Also the observed gap at the stopping test oscillates enormously ! Only a quadratic stabilization would be able to fix the problem of huge number of iterations. This is the subject of my internship ; preliminary material about it is gathered in the next section.

Before moving to stabilizing the algorithm, let us briefly discuss the warm-starting I implemented to reduce the computing time.

[4] We need these initial cuts to ensure that the model has a minimum which is not infinite.

3.2 Warm-starting the Lagrangian dual of the master problem

The bundle method solving the $(k+1)$-dual problem can be warm-started (a.k.a hot-started) using previously computed information.

$$\begin{aligned}
\text{With } c(x,k) &= \left(\Psi(x^\ell) + \left\langle g^\ell, x - x^\ell \right\rangle\right)_{l=1\ldots k} \text{ we have} \\
\Theta_{k+1}([\mu;0]) &= \min_{x\in X^1} f(x) + [\mu;0]^\mathsf{T} c(x,k+1) \\
&= \min_{x\in X^1} f(x) + \mu^\mathsf{T} c(x,k) = \Theta_k(\mu)
\end{aligned}$$

and we have a similar transfer for the sub-gradients :

$$\forall \ell \in \{1,\ldots,L\} \text{ (L the number of cuts) and } \forall \lambda \in [0,1]^{k+1}$$

$$\begin{aligned}
\Theta_{k+1}(\lambda) &\leq f\left(x(\mu^\ell)\right) + \lambda^\mathsf{T} c\left(x(\mu^\ell), k+1\right) \\
&= \underbrace{f\left(x(\mu^\ell)\right) + [\mu^\ell;0]^\mathsf{T} c\left(x(\mu^\ell), k+1\right)}_{=\ \Theta_k(\mu^\ell) \text{ by definition of } x(\mu^\ell)} + \left(\lambda - [\mu^\ell;0]\right)^\mathsf{T} c\left(x(\mu^\ell, k+1)\right) \\
&= \Theta_{k+1}\left([\mu^\ell;0]\right) + \left\langle c\left(x(\mu^\ell), k+1\right), \lambda - [\mu^\ell;0]\right\rangle
\end{aligned}$$

And therefore as Θ_{k+1} is concave :

$$\forall \ell = \{1,\ldots,L\},\ \left(\Psi(x^i) + \left\langle g^i, x(\mu^\ell) - x^i\right\rangle\right)_{i=1\ldots k+1} \in \partial\Theta_{k+1}([\mu^\ell;0])$$

So the linearizations of Θ_k can be transformed into linearizations of Θ_{k+1}. The bundle method at $k+1$ can start with a rich bundle – and from $[\mu^L;0]$. In practice, warm-starting enabled us to drastically reduce the number of cuts for the maximization of the Lagrangian dual.

4 My contributions

4.1 Quadratic algorithm – bundle method

Following standard techniques [6, 11], we will regularize the previous algorithm with the help of a quadratic stabilization term in the objective function. At iteration k, for a candidate solution $\hat{x}^k$, we consider the following k-problem instead of (4)

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^{n\times T}} & f(x) + \check{\Psi}_k(x) + \dfrac{1}{2t_k}\left\|x - \hat{x}^k\right\|_2^2 \\ s.t. & x \in X^1. \end{array} \tag{8}$$

This new term can be interpreted as a penalization on the distance from $\hat{x}^k$ to the next iterate. This will constrain the algorithm to produce :

- either a point far from the best program already found $\hat{x}^k$ with a significantly smaller cost.
- or a point near $\hat{x}^k$ with quite a similar cost (thus allowing us to refine the model locally).

As before we move to the Lagrangian dual with respect to the implicit coupling constraints in $\check{\Psi}_k$, which lead to the following equivalent of Lemma 2

Lemma 3 (Oracle for the k-th dual function) *The dual function expression is defined for all $\mu \in \mathbb{R}^k$ such that $\sum_{\ell=1}^k \mu_\ell = 1$ by*

$$\Theta_k^B(\mu) := \min_{x \in X^1} f(x) + \frac{1}{2t_k} \left\| x - \hat{x}^k \right\|_2^2 + \left\langle \sum_{\ell=1}^k \mu_\ell g^\ell, x \right\rangle + \sum_{\ell=1}^k \mu_\ell \left(\Psi(x^\ell) - \left\langle g^\ell, x^\ell \right\rangle \right). \quad (9)$$

Thus we have to solve the n sub-problems

$$\min_{x_i \in X_i^1} f_i(x_i) + \left\langle \sum_{\ell=1}^k \mu_\ell g_i^\ell - \frac{1}{t_k} \hat{x}_i^k, x_i \right\rangle + x_i^\mathsf{T} \left(\frac{1}{2t_k} I \right) x_i$$

Which gives the same sub-gradient as for the cutting-plane

$$\left(\Psi(x^1) + \left\langle g_1^1, x(\mu) - x^1 \right\rangle, \cdots, \Psi(x^k) + \left\langle g_k^k, x(\mu) - x^k \right\rangle \right)^\mathsf{T} \in \partial \Theta_k^B(\mu)$$

Thus, we now suggest the following regularized cutting plane algorithm :

Algorithm 2. Regularized Cutting-plane (solve (3))

Step 1 (Initialization) Generate at least three initial and different elements $x \in X_1$ and use these to build an initial model for Ψ. Set $k = 3$, the stopping tolerance $\delta_{\texttt{stop}}$ and let $\hat{x}^3$ be one of these iterates.

Step 2 (Lagrangian dual) Use a bundle method to maximize (9) with the additional constraint $\sum_{\ell=1}^k \mu_\ell = 1$. Save the iterate that produces the lowest value for $f(x) + \frac{1}{2t_k} \left\| x - \hat{x}^k \right\|_2^2 + \check{\Psi}_k(x)$.

Step 3 (Oracle call) Call the oracle for Ψ to compute a value and sub-gradient at x^{k+1}, enriching the model for Ψ.

Step 4 (Descent condition) Let the predicted decrease $\delta^k := f(\hat{x}^k) + \Psi(\hat{x}^k) - f(x^{k+1}) - \check{\Psi}_k(x^{k+1})$ and check if the observed decrease is at least a fraction $m > 0$ of δ^k :

$$f(x^{k+1}) + \Psi(x^{k+1}) \leq f(\hat{x}^k) + \Psi(\hat{x}^k) - m\delta^k. \quad (10)$$

If (10) holds, then declare a serious step and let $\hat{x}^{k+1} = x^{k+1}$. Else declare a null step.

Step 5 (Stopping test) Check if

$$\delta^k \leq \delta_{\texttt{stop}} \quad (11)$$

and stop the algorithm if this holds.
Let $k = k + 1$ otherwise and **return in Step 1**.

Regarding the choice of t_k, a simple constant value could be sufficient. But there may be a problem because of the approximate resolution of the subproblems. In fact, this approximation may cause a « noise » and lead to a negative δ_k. In order to get rid of this noise, we decide to implement an idea of Kiwiel from the paper [7]. This idea consist on changing the order of magnitude of t_k when the noise is detected. I.e, if the following test is true just before the stopping test (Step 4) :

$$\delta^k < \frac{1}{2t_k} ||\hat{x}^k - x^{k+1}||_2^2,$$

then $t_k \leftarrow 10 \times t_k$.

4.2 Primal recovery

An important point to focus on is the primal recovery. In fact, after maximizing Θ_K (resp. Θ_K^B) we have a Lagrangian multiplier μ_{opt} and an $x(\mu_{opt})$ which is the argmin of the min in the definition of $\Theta_K(\mu_{opt})$ (resp. $\Theta_K^B(\mu_{opt})$).
The vector $x(\mu_{opt})$ is a feasible primal solution (i.e., $x(\mu_{opt}) \in X^1$) but the question is : can we get an even better x (in the sense that $f(x)+\Psi(x)$ is lower, which mean having a smaller duality gap) at a low computational cost ?

4.2.1 Pseudo-program

As suggested in [4], in the case of a convex program, the primal-optimal feasible schedule is obtained through (bi-)dual information given during the maximization of $\Theta_k(\mu)$ (or Θ_K^B).
With L the number of cuts, α_ℓ the dual multipliers of the last quadratic problem solved by the bundle method while maximizing $\Theta_k(\mu)$ and $x(\mu^\ell)$ the argmin in $\Theta_k(\mu^\ell)$ we define the pseudo-program as follows :

$$x^{pp} \quad = \quad \sum_{\ell=1}^{L} \alpha_\ell x(\mu^\ell).$$

I would like to clarify again that this pseudo-program can only be used on units which have a convex technical constraint set X_i^1.

4.2.2 Hybrid

In the problem that we are currently dealing with, the program is not convex.
In fact, we could still use the pseudo-program for the hydraulic units, because they have convex feasible set. So remembering that the x_i are independents over X^1, we can use the pseudo-program for the hydraulic units and another schedule for the thermal ones.
The obvious choice is to choose in the $x(\mu^\ell)$ we already computed because we know that they are feasible. We define $x^{hyb,\ell}$ by :

$$\begin{aligned} \forall \ell, \quad \forall i \text{ hydraulic }, \quad & x_i^{hyb,\ell} = x_i^{pp} \\ \forall i \text{ thermal }, \quad & x_i^{hyb,\ell} = x(\mu^\ell)_i \end{aligned}$$

This in mind, the best $x^{hyb,\ell}$ is the one which minimizes $f(x) + \check{\Psi}_k(x)$.

4.2.3 Takriti & Birge

We can still go a little bit further exploiting that $X^1 = \prod_{i=1}^n X_i^1$.
We still use the pseudo-program as in the previous section but each thermal schedule is selected among $\left\{x(\mu^\ell)\right\}_{\ell=1}^{L}$ in such a way as to minimize the objective function. The resulting schedule is obviously better but this technique leads to the resolution of a mixed integer (quadratic) program which may be not trivial[5].

I refer the reader who wants to see the details of the calculations to Appendix C that formally demonstrates the determination of the program to solve for this heuristic. (Although based on the work of Takriti & Birge [12], I had to fully reprove this heuristic in our case)

4.3 Implementation

To implement these algorithms and to compare them, I had to continue the development of a program that my tutor had begun. One of the first things I did when arriving at my internship was to understand what had already been done and create a « more flexible » architecture for

[5] To reduce cpu time, it is possible to limit the x to the $x(\mu^\ell)$ which have a dual multiplier $\alpha_\ell \neq 0$.

this software in order to add new features easily. I set up the architecture shown in Figure 8 (Appendix D).

As in any numerical optimization program, we find the classic pattern containing the main function performing general iterations, the type of algorithm used, the oracle and the production units to optimize.

In terms of development, the programming was done in `C++` on a Linux environment and I had to use the following libraries :

- `Eigen[5]`, a linear algebra library to manipulate the vectors and matrix.
- The professional solver `IBM CPLEX[1]` to efficiently solve the linear/quadratic sub-problems.
- Given that there are a lot of sub-problems which are all independent, it's a good thing to parallelize there resolutions. To achieve this goal I simply used the `OpenMP[10]` library.

5 Results and analysis

The tests described in the following sections were performed on a `Linux` PC with a `4-core Intel® Xeon® Processor E3-1240`. The real datasets were collected during 2013 and contains 157 units (106 Thermal Units and 41 Hydro Valleys).

5.1 Duality gap

Foremost, we will analyze if the heuristics of section 4.2 reduce the observed duality gap. For this, we first compare the heuristics on simple convex data because in that case the pseudo-schedule gives the optimal primal solution at each iterations. (Fig. 4)

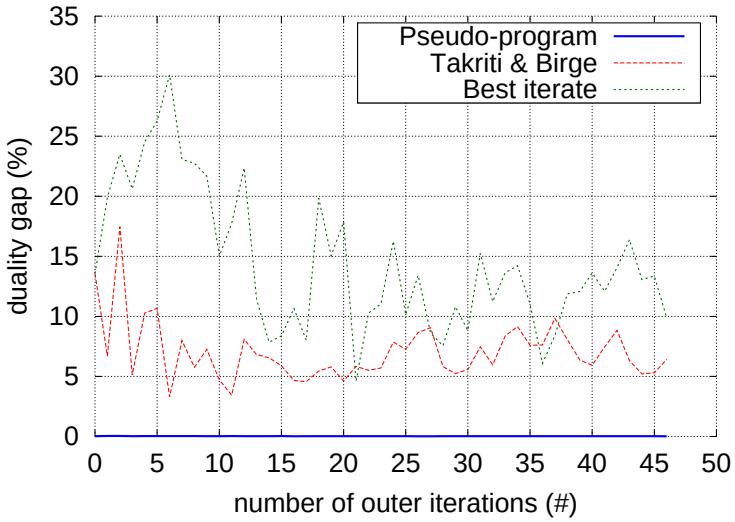


FIGURE 4 – Duality gap for 3 heuristics using cutting-plane on simple (convex) data.

As shown, using the pseudo-schedule we have a duality gap of nearly 0.0% and the Takriti & Birge heuristic seems to give a much smaller duality gap than the simple best-iterate heuristic.

Now let's move on to real datasets where we can't use the pseudo-schedule (details at section 4.2.1).

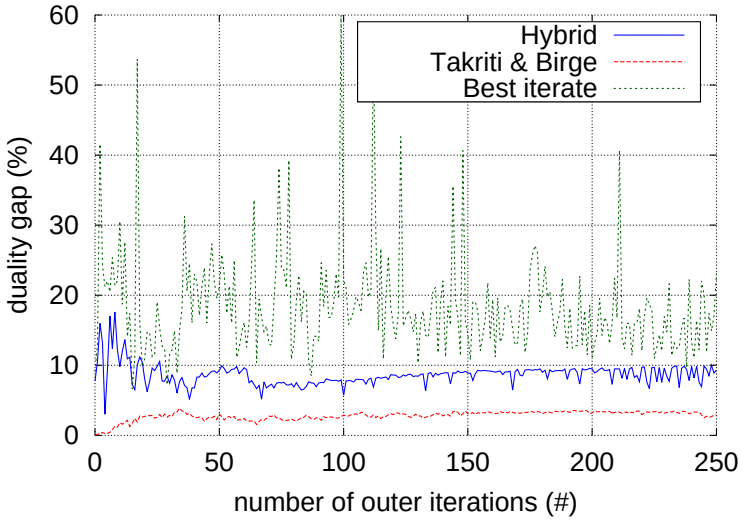


FIGURE 5 – Duality gap for three heuristics using cutting-plane on real data.

On Figure 5, clearly, the Takriti & Birge heuristic gives the smallest duality gaps, much smaller than those of the best-iterates and hybrid heuristics. We also see on this graph the importance of primal recovery heuristics : the programs given by the best-iterate have a duality gap which can go up to more than 50%, which is almost unusable in practice.

5.2 Bundle intern – number of resolutions

Since the speed of the algorithm almost exclusively depends on the number of sub-problems solved, the criterion used to verify that the regularized algorithm is more interesting than the non-regularized one is the number of calls to `CPLEX` to solve the sub-problems.

It is important to notice that to solve the sub-problems, we have solve a MIQP (Mixed Integer Quadratic Program) problem for the regularized algorithm instead of a MILP (Mixed Integer Liner Program) for the non-regularized method. This change causes a huge difference in terms of resolution time of a single iteration.
In reality however, **EDF** does not use `CPLEX` to solve the sub-problems but a dynamic programming-based algorithm. Therefore, in practice, solving a MILP or a MIQP has the same cpu cost and thus we are only concerned with the number of sub-problem resolutions.

We compare the two algorithms on 5 real datasets using hot-start (introduced in section 3.2), Takriti & Birge heuristic (cf. section 4.2.3) and Minoux's $\mathcal{D}$ (explained in section 2.3 and Figure 3). The results are summarized in the table 1.

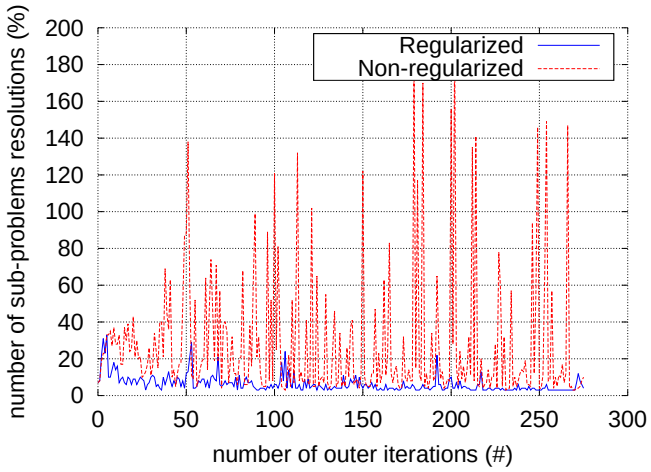
As we can see, the regularization improve significantly the program by decreasing strongly the number of calls to `CPLEX`.

Looking at the successive iterates given by the two algorithm (Figure 6), we easily see the « levels » characteristics of the regularized method and the much lower number of calls to `CPLEX`. This difference is amplified by the use of warm-starting which is more interesting for regularized algorithm (this was also observed in [13]).

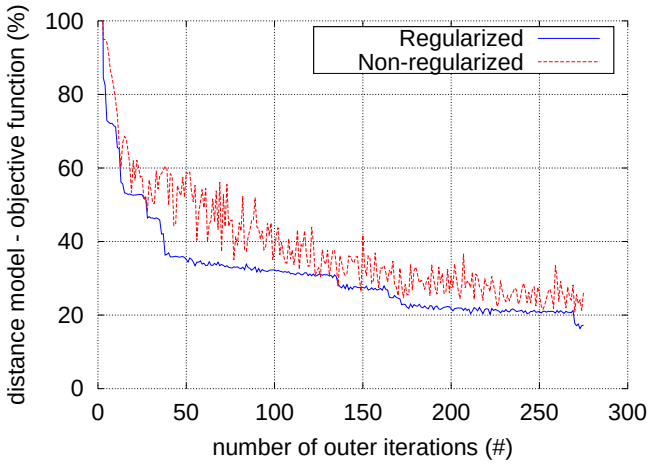
	Non-regularized			Regularized		
Date	Gap (%)	Ext. It. (#)	**Int. It.** (#)	Gap (%)	Ext. It. (#)	**Int. It.** (#)
14/01/2013	5.0	92	1117	5.0	45	393
14/01/2013	0.1	2333	3467	0.1	254	601
19/03/2013	22.0	250	7060	20.5	250	1636
12/05/2013	32.0	300	9876	29.0	300	2757
24/10/2013	20.0	539	13768	5.0	1320	4429
09/12/2013	9.0	250	19799	9.0	250	18495

TABLE 1 – (Some computations were stopped before convergence due to a time limit)

Gap : final gap between the model and the real objective function.
Int. It. : number of calls to `CPLEX` to solve all the sub-problems.
Ext. It. : number of calls to the oracle of Ψ.



(a) Number of calls to `CPLEX`



(b) Duality gap

FIGURE 6 – Successive iterates for both algorithm on 19/03/2013 dataset

5.3 Final schedule compared with average load

In this subsection I will present an interesting practical result : how production matches load. Indeed, as explained in section 2.3, the uncertainty set $\mathcal{D}$ is defined around the average demand. Knowing this, it is interesting to examine the decisions taken by the algorithm concerning the energy demand satisfaction for different definitions of $\mathcal{D}$.

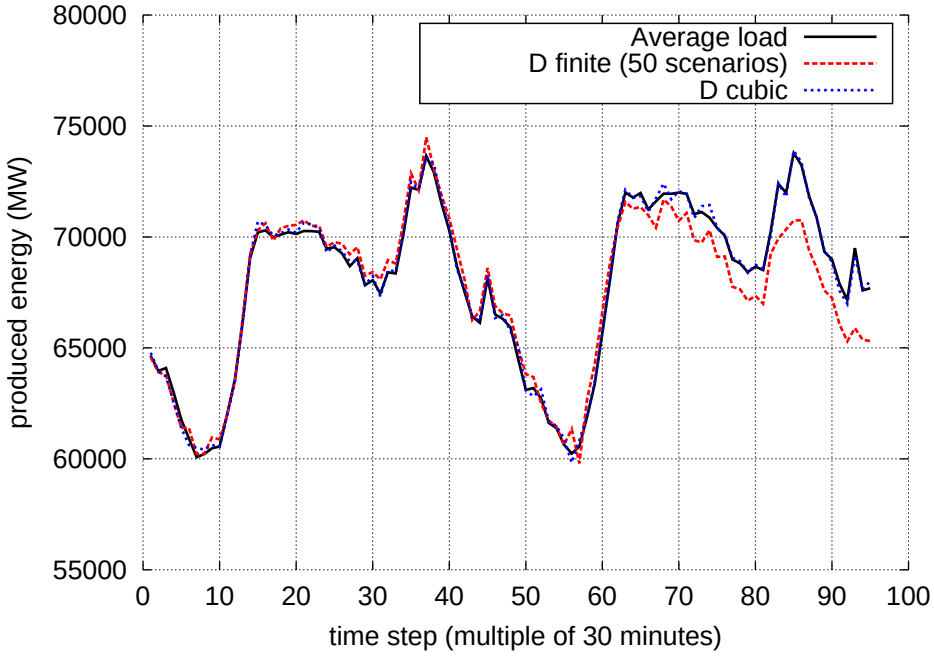


FIGURE 7 – Production of energy for 2 programs compared to the average load ($\delta = 5.0\%$)

As we can see, the scheduled production is around the average load but with some variations. What is interesting to notice is that using the cubic $\mathcal{D}$ the algorithm generates a schedule which is really close to the demand unlike the $\mathcal{D}$ composed of a finite number of scenarios which has a maximum difference with respect to the average load of nearly 1500Mw during the second day ! This is explained by the fact that the penalization function Ψ at **EDF** gives a smaller penalization cost for the second day because the associated schedule is not used.

6 Conclusion

6.1 Internship results

As a conclusion, I have implemented a cutting-plane algorithm and a regularized version for the suggested two-stage robust unit commitment model.
I also optimize these two algorithms with mathematical methods such as warm-starting or primal recovery heuristics.
To ensure the correct functioning of the algorithms, I have also implemented the real oracle Ψ used at **EDF** and tested on different real data and different scenarios $\mathcal{D}$.

The results obtained with the regularized method show that the regulation term clearly improves the convergence speed. The experiments on primal recovery heuristic indicates that the Takriti & Birge heuristic is the best to use in practice : although more expensive than the other heuristics, it gives us in less than a minute a primal feasible iterate with a duality gap less than 5%.

Prospects :

- Even if the Takriti & Birge heuristic gives good result, it could be interesting to see if we could come up with a new heuristic which lower even more the observed duality gap.
- Another axis is to push even further the study of the uncertainty set in order to choose the definition of $\mathcal{D}$ that best meet reality.
- And finally, I briefly talk about the choice of t_k at the end of section 4.1. Implementing a more precise t_k adaptation method could allow the regularized algorithm to make better iterations and greatly improve the convergence speed.

6.2 Personal side

In a more personal way, during this internship I had the opportunity to learn and manipulate the state-of-the-art of numerical optimization methods, whether in the mathematical or numerical aspects.

One of the things I noticed during this internship was the difficulty of debugging. Not for the code itself but for the mathematical algorithms. Even when the program compiles, is executed without errors and gives a result that seems consistent, we still can not be sure of the correct implementation of the algorithm. It is often after several tests on different data sets that eventually we get to be aware of potential problems in the code. I think this was the biggest difficulty I encountered.

Acknowledgments

I would like to thank my tutor Wim van Ackooij for all his advice and his answers to the (many) questions I asked him. I would also like to express my very great appreciation to Jérôme Malick for the assistance given throughout this three months but also for allowing me to get this internship.

Références

[1] IBM ILOG CPLEX Optimizer. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`, Last 2010.

[2] F. H. Clarke. *Optimisation and Nonsmooth Analysis*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1987.

[3] W. de Oliveira and C. Sagastizábal. Level bundle methods for oracles with on demand accuracy. *Optimization Methods and Software*, 29(6) :1180–1209, 2014.

[4] L. Dubost, R. Gonzalez, and C. Lemaréchal. A primal-proximal heuristic applied to french unitcommitment problem. *Mathematical programming*, 104(1) :129–151, 2005.

[5] G. Guennebaud, B. Jacob, et al. Eigen v3. `http://eigen.tuxfamily.org`, 2010.

[6] J.B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms II*. Number 306 in Grundlehren der mathematischen Wissenschaften. Springer-Verlag, 2nd edition, 1996.

[7] K.C. Kiwiel. Bundle methods for convex minimization with partially inexact oracles. *To appear in Comp. Opt. Appl*, 2012.

[8] J. Malick. Numerical optimization. ENSIMAG – University Lecture, 2014.

[9] M. Minoux. Two-stage robust optimization, state-space representable uncertainty and applications. *RAIRO-Operations Research*, 48 :455–475, 2014.

[10] OpenMP Architecture Review Board. OpenMP application program interface version 3.0. `http://www.openmp.org/mp-documents/spec30.pdf`, May 2008.

[11] A. Ruszczyński and A. Shapiro. *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*. Elsevier, Amsterdam, 2003.

[12] S. Takriti and J.R. Birge. Using integer programming to refine lagrangian-based unit commitment solutions. *IEEE Transactions on Power Systems*, 15(1) :151–156, 2000.

[13] W. van Ackooij. Decomposition approaches for block-structured chance-constrained programs with application to hydro-thermal unit commitment. *Mathematical Methods of Operations Research*, 80(3) :227–253, 2014.

[14] W. van Ackooij and J. Malick. Decomposition algorithm for large-scale two-stage unitcommitment. *Draft submitted, available on `http://www.optimization-online.org/DB_HTML/2015/04/4873.html`*, pages 1–26, 2014.

A Proofs of some specific points

A.1 Oracle for Ψ

Convexity of Ψ (2) :
First, $x \mapsto Ax$ is linear. Secondly, the functions $\psi_{i,t}(d;\cdot) : x \mapsto a_i(d_t - (Ax)_t) + b_i$ are affine so convex. Finally, the max / sup / sum of convex functions is also convex.
Thus, $\psi_t(d;\cdot) = \max_{i=1\dots6} \psi_{i,t}(d;\cdot)$ is convex and so does :

$$\begin{array}{rcccl} f_d & : & \mathbb{R}^n & \to & \mathbb{R} \\ & & x & \mapsto & \displaystyle\sum_{t=1}^{T} \psi_t(d, Ax) \end{array} \tag{12}$$

Consequently, we found the expected result for Ψ.

Expression of the subgradient :
$\mathcal{D}$ is finite so there is a $\bar{d}_x$ achieving the sup for each x.
Let $g_t \in \partial\psi_t(\bar{d}_x, Ax)$, then $g = \displaystyle\sum_{t=1}^{T} g_t$ is a subgradient of (12) at x (i.e. $\Psi(x)$).
$f_{\bar{d}_x}$ is convex so we have $\forall y \in \mathbb{R}^{n\times T}$:

$$\begin{array}{rcl} f_{\bar{d}_x}(x) + g^{\mathsf{T}}(y - x) & \leq & f_{\bar{d}_x}(y) \\ \Rightarrow \Psi(x) + g^{\mathsf{T}}(y - x) & \leq & f_{\bar{d}_x}(y) \leq \sup_{d\in\mathcal{D}} f_d(y) = \Psi(y) \end{array}$$

Therefore, since Ψ is convex, $g \in \partial\Psi(x)$
Using linear composition rule, we finally come to the following explicit expression for the subgradient :

$$A^{\mathsf{T}} v \in \partial\Psi(x) \quad \text{with} \quad v_i = a_j,\ j \text{ such that } \psi_{j,t}(\bar{d}_x, Ax) = \psi_t(\bar{d}_x, Ax)$$

A.2 « cubic » $\mathcal{D}$

Let $\mathcal{D} = \{x, \|x - \mu\|_\infty \leq k\} = [\mu_1 - k, \mu_1 + k] \times \cdots \times [\mu_T - k, \mu_T + k]$ with $\mu \in \mathbb{R}^T$ and $k > 0$.
Then :

$$\begin{array}{rcl} \Psi(x) & = & \displaystyle\sum_{t=1}^{T} \max_{d_t\in[\mu_t-k,\mu_t+k]} \psi_t(d; Ax) \quad \text{(note : the sup is a max because } \psi \text{ continuous and } \mathcal{D} \text{ compact)} \\ & = & \displaystyle\sum_{t=1}^{T} \max_{d_t\in[\mu_t-k,\mu_t+k]} \max_{i\in\{1\dots m\}} a_i(d_t - (Ax)_t) + b_i \\ & = & \displaystyle\sum_{t=1}^{T} \max_{i\in\{1\dots m\}} \left(b_i - a_i (Ax)_t + \max_{d_t\in[\mu_t-k,\mu_t+k]} a_i d_t \right) \quad \text{(because } \{1\dots\text{m}\} \text{ is finite)} \\ & = & \displaystyle\sum_{t=1}^{T} \max_{i\in\{1\dots m\}} \left(b_i - a_i (Ax)_t + a_i \left\{ \begin{array}{lll} \mu_t - k & \text{if} & a_i < 0 \\ \mu_t + k & \text{if} & a_i \geq 0 \end{array} \right. \right) \end{array}$$

B Convergence of the algorithm

B.1 Cutting plane method

Proposition 4 (Convergence of cutting-plane method) *Let $\delta_{\texttt{stop}} > 0$. Then the cutting-plane algorithm (Algo. 1) terminates with an approximate solution of (3) : more precisely, there exists an iteration k such that the stopping test of Step 3 is active for x^{k+1} and the value of $f+\Psi$ at x^{k+1} is at most at $(\delta_{\texttt{stop}} + f(x^{k+1}) + \check{\Psi}(x^{k+1}) - \Theta_k(\mu^{k+1}))$ of the optimal value of (3).*

Proof. Let prove it by contradiction.
Suppose that there is a $\delta_{\texttt{stop}} > 0$ such that for all k we have $\frac{\Psi(x^{k+1}) - \check{\Psi}_k(x^{k+1})}{f(x^{k+1}) + \Psi(x^{k+1})} > \delta_{\texttt{stop}}$.
First we minorate the denominator by $(f+\Psi)(\bar{x}) = k$ with $\bar{x}$ the optimal value. (we suppose the existence of $\bar{x}$)
Then for all k and $l \leq k$:

$$\Psi(x^{k+1}) - \check{\Psi}_k(x^{k+1}) \geq \Psi(x^{k+1}) - \Psi(x^{\ell}) - \left\langle g^{\ell}, x^{k+1} - x^{\ell} \right\rangle > k\delta_{\texttt{stop}}$$

Using Cauchy-Schwarz inequality :

$$\left|\Psi(x^{k+1}) - \Psi(x^{\ell}) - \left\langle g^{\ell}, x^{k+1} - x^{\ell} \right\rangle\right| < \left|\Psi(x^{k+1}) - \Psi(x^{\ell})\right| + \|g^{\ell}\|\|x^{k+1} - x^{\ell}\| \tag{13}$$

But X^1 compact, so :

1. Ψ is convex on X^1 so locally Lipschitz for each $x \in X^1$. By compactness, Ψ is Lipschitz on all X^1 and using proposition 2.1.2 of [2] there is a constant $C > 0$ such that $\|g^{\ell}\| \leq C$ for all $\ell \in \{1, \ldots, L\}$ because $g^{\ell} \in \partial\Psi(x^{\ell})$ with $x^{\ell} \in X^1$.

2. There is a convergent subsequence $\left(x^{\sigma(k)}\right)_k$ of $\left(x^k\right)_k$ such that $x^{\sigma(n)} \to x \in X^1$.
   Taking $k_n + 1 = \sigma(n)$ and $\ell_n = \sigma(n-1)$ we have on the one hand $\|x^{k_n+1} - x^{\ell_n}\| \xrightarrow[n\to\infty]{} 0$, and on the other hand with the continuity of Ψ : $\Psi(x^{k_n+1}) - \Psi(x^{\ell_n}) \xrightarrow[n\to\infty]{} 0$.

In conclusion, the term to the right of (13) tends to 0 as n approaches infinity which is in contradiction with the original assumption.
Let say x^* is the optimal value of (3), then we have :

$$\underbrace{\Theta_k(\mu^{k+1})}_{\text{weak duality}} < (f+\Psi)(x^*) < \underbrace{(f+\Psi)(x^{k+1})}_{\text{definition of } x^*} \leq \underbrace{\delta_{\texttt{stop}} + (f+\check{\Psi})(x^{k+1})}_{\text{stopping test}}$$
$$\Rightarrow 0 < (f+\Psi)(x^{k+1}) - (f+\Psi)(x^*) \leq \delta_{\texttt{stop}} + (f+\check{\Psi})(x^{k+1}) - \Theta_k(\mu^{k+1})$$

□

B.2 Regularized cutting-plane

Proposition 5 (Convergence of regularized cutting-plane method) *The regularized cutting-plane algorithm (Algo. 2) end in a finite number of iterations. In other words, for all $\delta_{\texttt{stop}} > 0$ there is a $k \in \mathbb{N}$ such that $\delta^k \leq \delta_{\texttt{stop}}$.*

Proof. By contradiction.
Assume that $\forall k,\ \delta^k = (f+\Psi)(\hat{x}^k) - \left(f + \check{\Psi}_k\right)(x^{k+1}) > \delta_{\texttt{stop}}$ for a given $\delta_{\texttt{stop}} > 0$.

- case 1 : there are an infinite number of serious iterations.

Let K the infinite set of indices of serious steps ; we have :

$$\begin{aligned} & \forall k \in K, \quad (f+\Psi)(\hat{x}^k) \leq (f+\Psi)(x^k) - m\delta_{\texttt{stop}} \\ \Rightarrow \quad & \forall k \in K, \quad (f+\Psi)(\hat{x}^k) \leq (f+\Psi)(x^1) - (k+1)m\delta_{\texttt{stop}} \end{aligned}$$

For k large enough, this contradicts the fact that $f+\Psi$ is bounded from below in the compact X.

- case 2 : there are a finite number of serious iterations $\left(\hat{x}^1, \cdots, \hat{x}^K = \hat{x}\right)$

With $\hat{x}$ the last stability center. As in the proof of cutting-plane method, we have for all k and i large enough with $i < k$:

$$\delta_{\texttt{stop}} < (f+\Psi)(\hat{x}^k) - (f+\Psi)(x^i) + \Lambda\|x^{k+1} - x^i\| \quad \text{with } \forall\ (x,g) \in X^1 \times \partial f(x),\ \|g\| \leq \Lambda \tag{14}$$

$$X^1 \text{ compact } \Rightarrow \quad \exists \text{ infinite index set K and } \bar{x} \in X^1 \quad x^{k'} \xrightarrow[\substack{k' \in K \\ k' \mapsto +\infty}]{} \bar{x}$$

By passing to the limit in (14) for $i, k+1 \in K$ with the continuity of f we get :

$$0 < \delta_{\texttt{stop}} \leq (f+\Psi)(\hat{x}) - (f+\Psi)(\bar{x}) \tag{15}$$

Observe now that we have :

$$(f+\Psi)(\bar{x}) \geq \liminf_{k+1 \in K} \left(f + \check{\Psi}_k\right)(x^{k+1})$$

This allows to conclude as follows : Since we only have null iterates, by definition we have

$$(f+\Psi)(\hat{x}) - (f+\Psi)(x^{k+1}) \leq m\left((f+\Psi)(\hat{x}) - \left(f + \check{\Psi}_k\right)(x^{k+1})\right)$$

In particular for $k+1 \in K$ and then passing to the lim sup, we get :

$$\begin{aligned} (f+\Psi)(\hat{x}) - (f+\Psi)(\bar{x}) \quad &\leq \quad m\left((f+\Psi)(\hat{x}) - \liminf_{k+1 \in K} \left(f + \check{\Psi}_k\right)(x^{k+1})\right) \\ \Rightarrow (1-m)\left((f+\Psi)(\hat{x}) - (f+\Psi)(\bar{x})\right) \quad &\leq \quad 0 \\ \Rightarrow (f+\Psi)(\hat{x}) - (f+\Psi)(\bar{x}) \quad &\leq \quad 0 \quad \text{since } m \text{ is chosen } 0 < m < 1 \end{aligned}$$

This contradicts (15). Conclusion, in the second cases, we reach a contradiction.
Then we have that $\exists k, \delta^k \leq \delta_{\texttt{stop}}$ □

C Obtaining Takriti & Birge's heuristic

Considering that there is already K cuts, p intern bundle iterations gives and :

- $x(\mu) = (x(\mu_i))_{i=1\dots p}$ (each x in $M_{n,T}(\mathbb{R})$ with $n = n_T + n_H$ number of units)
- $\{1 \dots n\} = I_T \cup I_H$ with I_T (I_H) the set of index for thermal (hydraulic) units. We suppose by rearranging the terms that $I_T = \{1 \dots n_T\}$ and $I_H = \{n_T + 1 \dots n\}$
- $f(x(\mu)) = (f(x(\mu_i)))_{i=1\dots p}$.
- $\widetilde{x} = (\widetilde{x}_{i,j})_{\substack{i=1\dots n \\ j=1\dots T}}$ pseudo-schedule.

we introduce $p \times n_T$ binary variables $z = (z_{i,j})$ and thus we solve the problem :

$$\left\{\begin{array}{rl} \min\limits_{z} & f(x) + \check{\Psi}_k(x) \\ \forall j \in I_T & x_j = \sum\limits_{i=1}^{p} z_{i,j} x(\mu_i)_j \\ \forall j \in I_H & x_j = \widetilde{x}_j \\ \forall j \in \{1 \dots n_T\} & \sum\limits_{i=1}^{p} z_{i,j} = 1 \\ \forall i,j & z_{i,j} \in \{0,1\} \end{array}\right. \xrightarrow{\text{new variable } r} \left\{\begin{array}{rl} \min\limits_{z,r} & f(x) + r \\ \check{\Psi}_k(x) & \leq r \\ \forall j \in I_T & x_j = \sum\limits_{i=1}^{p} z_{i,j} x(\mu_i)_j \\ \forall j \in I_H & x_j = \widetilde{x}_j \\ \forall j \in \{1 \dots n_T\} & \sum\limits_{i=1}^{p} z_{i,j} = 1 \\ \forall i,j & z_{i,j} \in \{0,1\}, r \in \mathbb{R} \end{array}\right.$$

And the condition $\check{\Psi}_k(x) \leq r$ is equivalent to the K conditions : $\forall \ell \in \{1 \dots K\}$

$$\Psi(x^\ell) + < g^\ell, x - x^\ell > \quad \leq \quad r$$

Which are equivalent to :

$$\Psi(x^\ell) + \sum_{j \in I_H} \sum_{t=1}^{T} g_{j,t}^\ell (\widetilde{x}_{j,t} - x_{j,t}^\ell) + \sum_{j \in I_T} \sum_{t=1}^{T} g_{j,t}^\ell (\sum_{i=1}^{p} z_{i,j} x(\mu_i)_{j,t} - x_{j,t}^\ell) \quad \leq \quad r$$

Rearranging the terms we get the final inequality :

$$\sum_{j \in I_T} \sum_{i=1}^{p} z_{i,j} \left(\sum_{t=1}^{T} g_{j,t}^\ell x(\mu_i)_{j,t} \right) - r \quad \leq \quad \sum_{j=1}^{n} \sum_{t=1}^{T} g_{j,t}^\ell x_{j,t}^\ell - \Psi(x^\ell) - \sum_{j \in I_H} \sum_{t=1}^{T} g_{j,t}^\ell \widetilde{x}_{j,t}$$

And the objective function is equal to :

$$f(x) + r \quad = \quad \sum_{j \in I_H} f_j(\widetilde{x}_j) + \sum_{j \in I_T} \sum_{i=1}^{p} z_{i,j} f_j\left(x(\mu_i)_j\right) + r$$

Which leads to the following program :

$$\left\{\begin{array}{rl} & \sum\limits_{j \in I_H} f_j(\widetilde{x}_j) + \min\limits_{z,r} \quad \sum\limits_{j \in I_T} \sum\limits_{i=1}^{p} z_{i,j} f_j\left(x(\mu_i)_j\right) + r \\ \forall \ell \in \{1 \dots K\} \sum\limits_{j \in I_T} \sum\limits_{i=1}^{p} z_{i,j} \left(\sum\limits_{t=1}^{T} g_{j,t}^\ell x(\mu_i)_{j,t} \right) - r & \leq \sum\limits_{j=1}^{n} \sum\limits_{t=1}^{T} g_{j,t}^\ell x_{j,t}^\ell - \Psi(x^\ell) - \sum\limits_{j \in I_H} \sum\limits_{t=1}^{T} g_{j,t}^\ell \widetilde{x}_{j,t} \\ \forall j \in \{1 \dots n_T\} & \sum\limits_{i=1}^{p} z_{i,j} = 1 \\ \forall i,j & z_{i,j} \in \{0,1\}, r \in \mathbb{R} \end{array}\right.$$

The problem is that solving such minimization program with $p \times n_T$ binary variable is highly time consuming. To solve this problem, we decide to only consider $x(\mu_i)$ of the i-th bundle intern iteration which have a dual multiplier $\alpha_i \geq \epsilon$.

For the regularized cutting plane we have to add the term $\frac{1}{2t_k}\|x - \hat{x}_k\|_2^2$ to the objective function.
Expanding the norm and replacing x by its expression with $z_{i,j}$ we get the following expression :

$$\begin{aligned}\frac{1}{2t_k}\|x-\hat{x}_k\|_2^2 &= \frac{1}{2t_k}\|x\|_2^2 - \frac{1}{t_k}x^\mathsf{T}\hat{x}_k + \frac{1}{2t_k}\|\hat{x}_k\|_2^2 \\ &= \frac{1}{2t_k}\|\widetilde{x}_H\|_2^2 + \frac{1}{2t_k}\|\hat{x}_k\| - \frac{1}{t_k}\widetilde{x}_H^\mathsf{T}\hat{x}_{H,k} - \frac{1}{t_k}\left(\sum_{i=1}^{p} z_{i,j}x(\mu_i)_j\right)_{j\in I_T}^\mathsf{T}\hat{x}_{T,k} + z^\mathsf{T}\frac{1}{2t_k}Az\end{aligned}$$

With A the $p \times n_T, p \times n_T$ square matrix such that

$$\begin{aligned}z^\mathsf{T}Az &= \left\|\left(\sum_{i=1}^{p} z_{i,1}x(\mu_i)_1, \ldots, \sum_{i=1}^{p} z_{i,I_T}x(\mu_i)_{n_T}\right)\right\|_2^2 \\ &= \sum_{j\in I_T}\sum_{t=1}^{T}\left(\sum_{i=1}^{p} z_{i,j}x(\mu_i)_{j,t}\right)^2 = \sum_{\substack{j\in I_T \\ i_1,i_2\in\{1\ldots p\}}} z_{i_1,j}z_{i_2,j}x(\mu_{i_1})_j^\mathsf{T}x(\mu_{i_2})_j\end{aligned}$$

According that $z = (z_{1,1}, \ldots, z_{p,1}, \ldots, z_{1,n_T}, \ldots, z_{p,n_T})$, the expression of A is finally :

$$A = \begin{pmatrix} M_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & M_{n_T} \end{pmatrix} \quad \text{and} \quad M_j = \left(x(\mu_{i_1})_j^\mathsf{T}x(\mu_{i_2})_j\right)_{i_1,i_2\in\{1\ldots p\}} \in M_p(\mathbb{R})$$

D Program's architecture

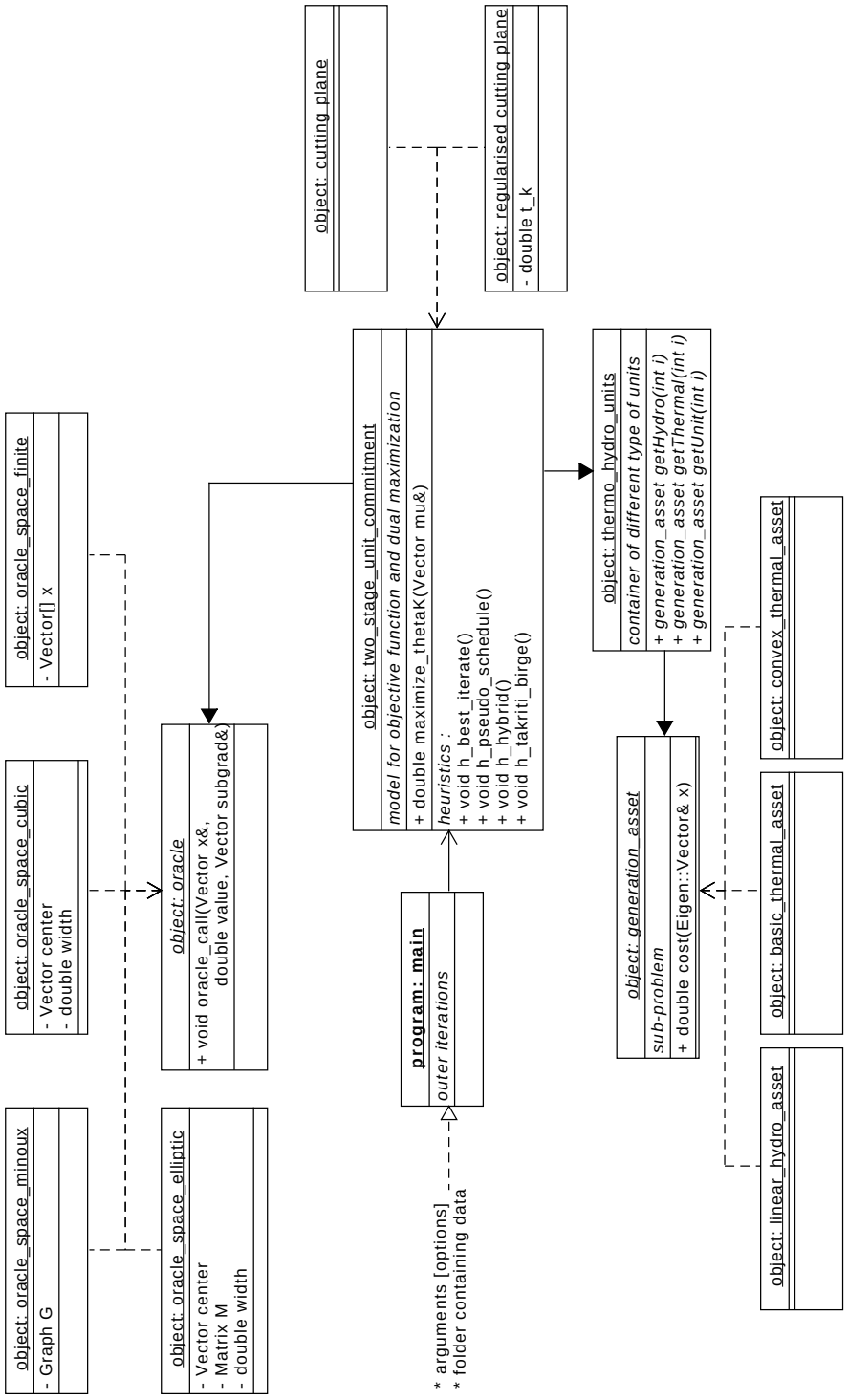


FIGURE 8 – pseudo-uml diagram showing the program files organization