Les Cahiers du GERAD ISSN: 0711–2440

Coupling decomposition with dynamic programming with application to an energy model

L. Marchand, P. Mahey,
J.-P. Dussault

G–2018–39

June 2018

Coupling decomposition with dynamic programming with application to an energy model

Luc Marchand [a]

Philippe Mahey [b]

Jean-Pierre Dussault [a,c]

[a] *Université de Sherbrooke, Sherbrooke (Québec), Canada J1K 2R1*

[b] *Université Clermont-Auvergne, 63178 Aubière, France*

[c] *GERAD, Montréal (Québec), Canada, H3T 2A7*

luc.marchand@usherbrooke.ca
philippe.mahey@isima.fr
jean-pierre.dussault@usherbrooke.ca

June 2018
Les Cahiers du GERAD
G–2018–39

Abstract: This paper targets a stochastic energy management problem. We first decouple the stochasticity of the global scenarios to local scenarios. Then, we use spatial decomposition through ADMM and time decomposition through Dynamic Programming to find approximate optimal values. The algorithms are validated on toy problems and a convergence discussion about the coupling of ADMM and Dynamic Programming is started.

1 Introduction

The main idea of this article is to apply a large stochastic programming model to mid-to-long term energy management. Three types of couplings make this model particularly hard to solve, namely :

- coupling with respect to time,
- coupling with respect to the scenarios via the non-anticipativity constraints,
- coupling with respect to space connecting the differents zones together.

Those couplings make the problem hard as the number of time steps and zones increase. Also, if a discretization of the stochastic process is used (i.e. a scenario tree is built), the amount of scenarios increases exponentially with the number of time steps, making direct solving intractable. In order to be able to work on such a model, scenarios randomness will be relaxed to zonal scenarios.

1.1 Literature review on mid-term management problem

A lot of work was done regarding relaxations of this problem. Focussing on a single zone, works in that field go back to the 50s, see Wallace and Fleten's survey [20]. Important papers and algorithms in this field include Birge's Bender's Decomposition application to the model [3], followed by Pereira and Pinto's SDDP algorithm (1985, 1988, 1991) [14].

SDDP has been in use since for multireservoir systems. However, those systems are highly dependant of one another and usually have a small indecomposable number of dams (and control variables). Our model aggregates multireservoir systems as a single reservoir, but considers a system of many zones, possibly a large number of them. This makes the problem larger in the amount of control variables and motivates exploration of methods other than direct SDDP's application. Our objective is to use Dynamic Programming's versatility to handle stochasticity and couple it with an Operator Splitting Method, namely ADMM.

Dynamic Programming [1] is a classical way to deal with multistage problems. It enables us to split the problem in multiple (hopefully) small problems through state variables discretization. Efficient stochastic programming methods spans from this method, namely Stochastic Dual Dynamic Programming presented before and Approximate Dynamic Programming (ADP) [15]. A main difficulty of basic Dynamic Programming methods follows from the discretization : as the number of state variables grows, computation time increases exponentially. This is known as the *curse of dimensionnality*, which motivates the decomposition of the state variables through ADMM (or the use of refined Dynamic Programming algorithms such as ADP).

Operator splitting methods are well known to solve convex optimisation problems. Back in [10], Lions and Mercier showed the possibility of solving some coupled convex optimisation problems through finding the zero of the sum of two monotone operators. Many methods spanned from this, between others Spingarn's method [19], Gabay's ADMM [3] and many other proximal methods, surveyed in [13]. Additionnal details on different operator splitting algorithms can be found in Lenoir and Mahey's survey [9].

In Section 2 we will present a quick overview of multistage stochastic programming problems, then apply it to our hydro-thermal energy management model, as presented in [11]. In Section 3, we will elaborate on the difficulties of the model, then give insights into potential algorithms to solve it. Some numerical results will follow in Section 4.

2 Stochastic multizonal model

We start by giving an abstract model to linear multistage stochastic programming problems :

$$\begin{aligned} \min_{x_t} \ & \mathbb{E}_\xi \sum_{t=1}^{T} c_t x_t(\xi_t) \\ \text{s.t.} \ & A x_t(\xi_t) = b(\xi_t) \quad \text{almost surely} \\ & x_t(\xi_t) \in X \end{aligned}$$

where ξ_t are random processes and X gives bounds on the control or state variables x_t. It is worth noting that x_t will also be stochastic, following from the random processes ξ_t. This kind of problem requires optimizing an expectation which makes significantly harder the solving of the problem (see [7] for an intuition of why this happens). If only $b(\xi_t)$ is stochastic, then the problem is convex by its linearity [4].

An important remark here is that solving this kind of problem directly without discretizing the random variables is hard. A way to solve this is therefore to discretize ξ_t, building a scenario tree in the process. The amount of scenarios rises exponentially with the number of time steps ; as a quick example, the scenario tree of a 10 time steps problem with 5 realizations of the random events at each time steps make a scenario tree of the order of 10 millions of scenarios, clearly intractable to solve directly.

2.1 General formulation of the model

We consider a set Z of spatially-splitted zones (such as geographical zones) where there is a random demand in a ressource (for example electric energy) for a duration. This time line is split in T time steps indexed by $t \in 0 \dots T$. The variable $d_{z,t}(\xi_{z,t})$ defines demand of zone z at time t and depends on the random realization $\xi_{z,t}$. This demand will have to be satisfied through resource generation, either from costly sources $p_{z,t}$ (like thermal electric generation) which have a piecewise linear cost $c_{z,t}$ or from free (renewable energy) sources $u_{z,t}$ which are free to use, but limited in capacity through the state $x_{z,t}$. This free resource's production capacity is filled through another random variable $i_{z,t}(\xi_{z,t})$, which can for example be seen as hydro influx in the dam from hydrology patterns.

The random sources are held in the demand variable $d_{z,t}(\xi_{z,t})$ and free sources inflow $i_{z,t}(\xi_{z,t})$. The control variables $p_{z,t}, u_{z,t}$ and state variable $x_{z,t}$ will therefore depend from those variables, also making them random. To simplify the notation, we will use the ξ_t for the random realizations of all zones at time t and ξ_z for the random realizations of zone z for all time steps.

The decision variables are all splitted for each zone. What makes the problem hard is the interconnectivity between the different zones. The zones can import and export missing or extra resource from each other through a set of transportation lines $e = (z, z')$. These connections can be seen as a directed graph with each zone being a node and each arc being a transportation line $e \in E$. The amount of resource transported in the line e at time step t is a decision variable denoted $f_{e,t}$ with extra cost l_e for the transport.

We will work on a decision-hazard strategy, meaning decision at time t is made before realization of the random events at time t. The taken decisions need to satisfy non-anticipativity constraints, which means the current decision should not depend on future random realisations since they should not be known at this current time step.

This is the stochastic energy management model in the probability space on which we will work (bold variables are random). From here on, we will consider the resource being energy, produced from aggregated thermal sources and hydro power.

$$\min_{\mathbf{p},\mathbf{x},\mathbf{u},\mathbf{f}} \mathbb{E}_\xi \left[\sum_{\tau=1}^{T} \left(\sum_{z\in Z} c_{z\tau}\mathbf{p}_{z\tau} + \sum_{e\in E} l_{e\tau}\mathbf{f}_{e\tau} \right) + \sum_{z\in Z} \Psi_z(\mathbf{x}_{zT}) \right] \tag{1}$$

$$\text{s.t. } \mathbf{u}_{z\tau} + \mathbf{p}_{z\tau} - \sum_{e\in z^+} \mathbf{f}_{e\tau} = \mathbf{d}_{z\tau}(\xi_{z\tau}) - \sum_{e\in z^-} \mathbf{f}_{e\tau}, \qquad \forall z \in Z, \tau \in [1,T] \tag{2}$$

$$\mathbf{x}_{z,\tau} = \mathbf{x}_{z,\tau-1} - \mathbf{u}_{z\tau} + \mathbf{i}_{z\tau}(\xi_{z\tau}), \qquad \forall z \in Z, \tau \in [1,T] \tag{3}$$

$$\mathbf{x} \in X, \mathbf{u} \in U, \mathbf{p} \in P, \mathbf{f} \in F \tag{4}$$

with $\mathbf{p}, \mathbf{x}, \mathbf{u} \in \mathbb{R}^{T\times|Z|}, \mathbf{f} \in \mathbb{R}^{|E|\times T}$ and non-anticipative, meaning present decision cannot take into account future random realizations (more on this will be presented in Section 2.2). The sets X, U, P, F define box constraints on the variables. The decision variable $\Psi_z(\mathbf{x}_{zT})$ is a penalty on the last state's level in order to keep the final level of renewable energy storage not too low.

This problem has some properties which make its study interesting. First of all, with multiple time steps, zones and random realizations, it gets really large. The high number of zones will make the state variable's dimension very large, discarding direct application of Dynamic Programming tools. The high number of total scenarios makes the application of a sampling algorithm as SDDP also pretty hard or not precise enough.

A first thing we could try out is to split the problem into smaller sub-problems. However, recall this model has 3 different couplings presented in Section 1. These couplings make impossible the direct splitting of the problem. We will need to relax some constraints and use decoupling tools. We will therefore aim at decoupling time and space respectively with Dynamic Programming and Operator Splitting schemes. However, to do this, we first need to make sure the randomness can be considered locally.

2.2 Uncertainties and how to handle the random processes

The pure random variables are the demand and the renewable energy inflow. These end up making the problem mainly stochastic in its constraints, which we would want satisfied almost surely. This is hard, so instead for now we will simply discretize the random processes into multiple realizations for each time step with a certain probability of realization. The grouping of a realization at each time step for zone z is called a scenario, noted $\xi_z(\nu)$. We will note N_t the number of possible random processes at time t; therefore the index ν_t take values between 1 and N_t. The span of these different random events make a scenario tree, which gets large as the different parameters Z and T get larger.

A scenario combines the random events occuring in all zones for each time step. As a simple example, consider a problem of two time steps with two zones, each with three possible realizations of each random variable. A typical scenario would be :

Time 1 Zone 1 has low demand and renewable inflow realization,
Zone 2 has high demand and mid inflow,
Time 2 Zone 1 has high demand and high inflow,
Zone 2 has high demand and low inflow.

In this particular case, if all events are equiprobable, we would have $3^{2\times2\times2} = 6561$ different scenarios, giving this particular scenario probability 1/6561. This being said, the number of realizations of each random process here is only 3 (low, mid or high), so definitely small compared to what would be ideal in order to get a good representation of the underlying probability law. Despite its small nature compared to our end goal, this exemple is already large, although still tractable.

It is worth noting that this model has a *deterministic equivalent* problem, where the expectation is replaced by the weighted sum of each possible scenario by their respective probabilities. However, such a formulation requires to take into account another property of the problem, namely the presence of so-called non-anticipativity constraints. In words, those constraints require any decision for two scenarios with the same past to be the same. It means we have no knowledge on future random realizations at a certain time step. It therefore prevents us from splitting the problem on the scenarios and simply solve each scenario independantly of the others. More on this can be found in Rockafellar and Wets's article [18].

3 Model reformulation and relaxations

There will be need of some relaxations to keep on working on the model as is. We'll mainly relax the stochastic constraints in order to be able to work with them properly, then use splitting methods to handle the other couplings.

3.1 Relaxation on stochasticity

The model's stochastic processes were originally considered markovian processes. This makes the analysis of each time step particularly hard since it does not only depend on the state x_t, but also on the last random realization that happened at time $t-1$. In order to avoid this, we will consider time-independant random events and, therefore, decision at time t only depends on the current state x_t. It will enable us to easily use Dynamic Programming tools, between others.

3.2 Relaxation from global scenarios to local scenarios

Considering all random realizations for each zone significantly increases the amount of scenarios. To avoid that, one can try merging constraints on the same variables for scenarios with the same local random realizations. This will split the randomness from scenarios to all zones to small local scenarios applicable on each zone separately. Doing so, one will also be able to merge some variables together as well as decouple the randomness of the demand constraints between the zones. This kind of aggregation can be found in [21] and [2].

The idea is therefore to aggregate the demand constraints (2) through the same zonal scenarios. As an exemple, taking a small toy problem with 2 zones, 2 time steps and 2 random realizations, we get 16 possible scenarios. After this relaxation, the scenarios will only be seen locally so optimisation will be done on 4 scenarios for zone 1 and 4 scenarios for zone 2. Although we can still find the 16 scenarios by taking the product of those 4×4 scenarios, optimisation is not done on the complete scenario tree and variables are aggregated to local scenarios. This kind of relaxation makes sense since while optimizing a zonal problem, a certain zone should not be able to "see" what random occurences happen in the other zones.

Looking back at those merged constraints, with i defining scenarios with the same local random event realization :

$$\sum_{i=1}^{2}(\mathbf{u}^i_{z,t} + \mathbf{p}^i_{z,t} - \sum_{e \in z^+} \mathbf{f}^i_{e,t}) = \sum_{i=1}^{2}(\mathbf{d}^i_{z,t} - \sum_{e \in z^-} \mathbf{f}^i_{e,t})$$

we observe that the coupled constraints only differ through the variables $f_{e,t}$. Additionnally, the decision variable u^1 becomes indissociable from u^2, since both are meant to consider local randomness of the other zone which we can not see anymore. It is then safe to aggregate those variables together through $u^1 = u^2$. The same can be said about the decision variable p, hence taking $p^1 = p^2$. It is worth noting that, through the aggregation of the u variables, we will also aggregate the dynamic constraints (3), hence also aggregating the state variables $x^i_{z,\tau}$.

Up to now, we aggregated some demand (and dynamic) constraints and some u, p and x variables. Following from [21], these two aggregations should end up respectively as a relaxation and a restriction. Also, we should be able to deduce lower bounds if we implement the constraints relaxation alone and upper bounds if we implement the variables aggregation alone.

This relaxation and restriction allows us to have, at a certain time step, zonal independance of the randomness. This also means the variable f will not be considered stochastic anymore. We can therefore use stochastic programming algorithms on each zone if we end up able to decouple the zones.

3.3 Zonal decomposition

The objective here is to look at strategies to decouple the zones from each others in order to simplify the problem. With the stochastic events now local, the coupling between the different zones is entirely determined by the transfer variables f within the coupling constraints

$$\mathbf{u}_{z\tau} + \mathbf{p}_{z\tau} - \sum_{e \in z^+} f_{e\tau} = \mathbf{d}_{z\tau} - \sum_{e \in z^-} f_{e\tau}, \forall z \in Z, \tau \in [1, T]$$

A way to handle this would be to use convex operator splitting algorithms such as the Alternate Direction Method of Multipliers (ADMM, [5]). This will allow the decomposition of the problem in two steps : the local subproblems and the transfer problem.

Formally, the actual problem can be written:

$$\min_{\mathbf{p},\mathbf{x},\mathbf{u},f} \mathbb{E}\left[\sum_{\tau=1}^{T}\left(\sum_{z\in Z} c_{z\tau}\mathbf{p}_{z\tau} + \sum_{e\in E} l_{e\tau} f_{e\tau}\right) + \sum_{z\in Z}\Psi_z(\mathbf{x}_{zT})\right]$$

$$\begin{aligned}
\text{s.t. } & \mathbf{u}_{z\tau} + \mathbf{p}_{z\tau} - \sum_{e\in z^+} f_{e\tau} = \mathbf{d}_{z\tau} - \sum_{e\in z^-} f_{e\tau}, && \forall z\in Z, \tau\in[1,T]\\
& \mathbf{x}_{z,\tau} = \mathbf{x}_{z,\tau-1} - \mathbf{u}_{z\tau} + \mathbf{i}_{z\tau}, && \forall z\in Z, \tau\in[1,T]\\
& x\in X, u\in U, p\in P, f\in F
\end{aligned}$$

We decouple the f variables using a zonal aggregation of those transfer with a new variable q. We therefore add the constraint $\sum_{e\in z^-} f_{e\tau} - \sum_{e\in z^+} f_{e\tau} = q_{z,\tau}$ and replace the zonal demand constraint using the variable q instead of f yielding the following model:

$$\min_{f,\mathbf{q}} \sum_{\tau=1}^{T}\sum_{e\in E} l_{e\tau} f_{e\tau} + \min_{\mathbf{p},\mathbf{x},\mathbf{u}} \mathbb{E}\left[\sum_{\tau=1}^{T}\sum_{z\in Z} c_{z\tau}\mathbf{p}_{z\tau} + \sum_{z\in Z}\Psi_z(\mathbf{x}_{zT})\right]$$

$$\begin{aligned}
\text{s.t. } & \mathbf{u}_{z\tau} + \mathbf{p}_{z\tau} + q = \mathbf{d}_{z\tau}, && \forall z\in Z, \tau\in[1,T]\\
& \sum_{e\in z^-} f_{e\tau} - \sum_{e\in z^+} f_{e\tau} = q_{z,\tau} && \\
& \mathbf{x}_{z,\tau} = \mathbf{x}_{z,\tau-1} - \mathbf{u}_{z\tau} + \mathbf{i}_{z\tau}, && \forall z\in Z, \tau\in[1,T]\\
& x\in X, u\in U, p\in P, f\in F
\end{aligned}$$

which, in turn, can be decomposed as :

$$\min_{\mathbf{q}} L(q) + \sum_{z\in Z} F_z(q_z) \tag{5}$$

with

$$F_z(q_z) = \min_{\mathbf{p},\mathbf{x},\mathbf{u}} \mathbb{E}\left[\sum_{\tau=1}^{T} c_{z\tau}\mathbf{p}_{z\tau} + \Psi_z(\mathbf{x}_{zT})\right] \tag{6}$$

$$\begin{aligned}
\text{s.t. } & \mathbf{u}_{z\tau} + \mathbf{p}_{z\tau} + q_z = \mathbf{d}_{z\tau}, && \forall z\in Z, \tau\in[1,T]\\
& \mathbf{x}_{z,\tau} = \mathbf{x}_{z,\tau-1} - \mathbf{u}_{z\tau} + \mathbf{i}_{z\tau}, && \forall z\in Z, \tau\in[1,T]\\
& x\in X, u\in U, p\in P
\end{aligned}$$

$$L(q) = \min_{f} \sum_{\tau=1}^{T}\sum_{e\in E} l_{e\tau} f_{e\tau} \tag{7}$$

$$\begin{aligned}
\text{s.t. } & \sum_{e\in z^+} f_{e\tau} - \sum_{e\in z^+} f_{e\tau} = q\\
& f\in F.
\end{aligned}$$

For conciseness, we rewrite the constraint of (7) using the Arc-Node matrix A, making this constraint of the form $Af = q$.

With L and F being convex, the form taken by (5) suggests the use of operator splitting algorithms. Indeed, with $L(q)$ and $F_z(q_z)$ associated with convex stochastic minimisation problems, the first order optimality conditions of this problem take the form $0 \in \partial_z L(q) + \partial F_z(q_z)$ with $\partial_z L(q)$ a subdifferential on q_z and $\partial F_z(q_z)$ also a subdifferential, both being montonone operators (see [16] for details on operator splitting

Algorithm 1 Alternating Direction Method of Multipliers algorithm

```
Step 1. Zonal subproblems
    Compute $q_z^{k+1}$ as solution of
        $\min_{q_z} \quad F_z(q_z) + \langle \pi_z^k, q_z \rangle + \frac{\lambda}{2}\|q_z - (Af^k)_z\|^2$
Step 2. Network subproblem
    Compute $f^{k+1}$ as solution of
        $\min_{f} \quad L(f) - \langle \pi^k, Af \rangle + \frac{\lambda}{2}\|q^{k+1} - (Af)\|^2$
Step 3. Dual update:
        $\pi^{k+1} = \pi^k + \lambda(q^{k+1} - Af^{k+1})$
```

schemes applied to optimisation problems). The current formulation enables us to use the ADMM algorithm to decompose the problem using the q variables as in-between for the two problems and the $Af = q$ constraint as Augmented Lagrangian penalty. The resulting procedure is as written in Algorithm 1.

It is pertinent to notice that the use of Augmented Lagrangian methods will result in the cost function to be now piecewise quadratic instead of piecewise linear. This, however, does not affect the convexity of the problem.

Although we proposed the use of ADMM, other operator splitting methods could be considered, such as other proximal-type methods [12] [17] or the Separable Augmented Lagrangian Algorithm [6].

3.4 Time decomposition

The sub-problems resulting from the zonal decomposition are still too large to be solved directly. Indeed, even though the problem's size significantly decreased, it is still large as T gets larger and with an exponentially-increasing amount of scenarios. As we increase the amount of scenarios, the number of non-anticipativity constraints also increases drastically. Also, the zonal problems are still coupled through time, making the decision space roughly as large as the number of time steps. We will need another way to find zonal solutions, ideally with enough precision so the ADMM decomposition method can return with a good solution.

A way to solve such a problem is to use Dynamic Programming. Doing so, we end up finding the solution to many small almost trivial problems through the discretization of the state space for each time step. This method therefore considers non-anticipativity constraints without having to deal with them explicitly, as well as solve a single problem per discretization of the state space for each time steps instead of solving as many problems as scenarios leading to those time steps.

The zonal subproblems we want to solve are, with $z \in Z$ fixed :

$$\min_{\mathbf{p},\mathbf{x},\mathbf{u}} \mathbb{E}_{\xi_z} \left[\sum_{\tau=1}^{T} \left(c_{z\tau}\mathbf{p}_{z\tau} + \langle \pi_{z\tau}^k, q_{z\tau} \rangle + \frac{\lambda}{2}\|q_{z\tau} - (Af_\tau^k)_z\|^2 \right) + \Psi_z(\mathbf{x}_{zT}) \right]$$

$$\begin{aligned} \text{s.t. } & \mathbf{u}_{z\tau} + \mathbf{p}_{z\tau} + q_{z\tau} = \mathbf{d}_{z\tau}, && \forall z \in Z, \tau \in [1,T] \\ & \mathbf{x}_{z,\tau} = \mathbf{x}_{z,\tau-1} - \mathbf{u}_{z\tau} + \mathbf{i}_{z\tau}, && \forall z \in Z, \tau \in [1,T] \\ & x \in X, u \in U, p \in P. \end{aligned}$$

We can rewrite those problems in the following form to use Dynamic Programming :

$$V_T(x_T) = \sum_z \Psi_z(x_{zT})$$

for $t = T-1, \ldots 0$

$$V_t(x_t) = \min_{u_t \in U_t} \frac{1}{N_{t+1}} \sum_{\nu_{t+1}=1}^{N_{t+1}} W_t(x_t, u_t, \nu_{t+1})$$

for $v_{t+1} = 1, \dots, N_{t+1}$

$$W_t(x_t, u_t, \nu_{t+1}) = \begin{cases} \min_{p_{t+1}, f_{t+1}} & \sum_z c_z(p_{z,t+1}(\nu_t + 1)) + \sum_e l_{e,t+1}(f_{e,t+1}(\nu_{t+1})) + V_{t+1}(x_{t+1}) \\ \text{s.t.} & u_{zt} + p_{z,t+1} + \sum_{e \in z^+} f_{e,t+1} - \sum_{e \in z^-} f_{e,t+1} = d_{z,t+1}(\nu_{t+1}), \quad \forall z \\ & x_{z,t+1} = x_{xt} - u_{zt} + i_{z,t+1}(\nu_{t+1}), \quad \forall z \\ & p_{z,t+1} \in P_z, x_{z,t+1} \in X_z, \quad \forall z \end{cases}$$

Using Dynamic Programming gives an approximate solution. The sharpness of the discretization of the state space should play a big role as we want to merge it with ADMM. Other methods are to be considered in order to solve this problem. Namely, we could use Benders-type methods (as SDDP [14] or Nested Benders Decomposition [3]) or Progressive Hedging [18]. Our actual focus will however be on Dynamic Programming.

The main problem coming from using Dynamic Programming is due to the discretization's sharpness. It would be interesting to see if an adaptive Dynamic Programming Algorithm, which sharpens the discretization near the last iteration's solution and coarsens it far from it, makes things better. Interresting topics on this would be the convergence properties when coupled with ADMM as well as how to find a good enough solution in order to start sharpening around it.

4 Numerical results

The end goal of this modelisation is to solve the global problem doing the following steps:

1. decouple the randomness between zones by aggregating contraints,
2. decouple the zones from each others with ADMM,
3. solve zonal sub-problems with Dynamic Programming.

To validate these steps separately and easily, we will use some small scale toy problems. These toy problems will be defined in the next subsection, followed by the implementation and validations on decoupling the randomness. The section will be completed by illustrating the combined algorithm based on ADMM and Dynamic Programming.

4.1 Toy problems

We will define simple toy problems here to get insights about how each component of the algorithms behave. With small problems, it is also possible to compute a solution to the deterministic equivalent problem, which would be impossible for a larger problem.

The idea being to see how each algorithm or relaxation works, we will present four different toy problems. The first one aims to evaluate the effect of randomness aggregation to local random events. This problem, noted TP1, will be as follows :

TP1 12 global scenarios, all equiprobable,

- 2 zones, 2 time steps, 2 random occurences per time step per zone,
- zero probability for (ξ_{11}^2, ξ_{21}^1), making it 12 scenarios instead of 16,
- modeled with its deterministic counterpart.

The local aggregation will take those 12 scenarios and merge them to 8 zonal scenarios, 4 per zone, as shown in Figure 1.

Next, we will validate Dynamic Programming's interaction with ADMM. In order to do this, we simulate scenarios that are already decoupled locally. Many toy problems will be used to do this, each noted TPx, with x being the toy problem's number.

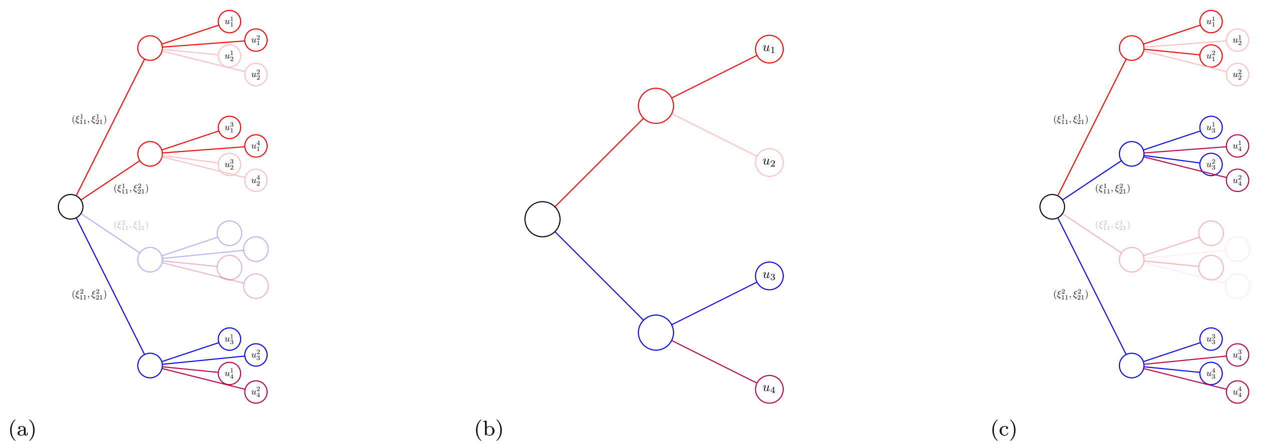

Figure 1: Global Scenario Tree for Toy Problem 1 with local aggregation for both zones. The colors represent each local scenario, and two global scenario sharing the same color on the last node are locally aggregated together. We can observe the different u_j^i aggregated together to a single variable u_j for each zone. Graph a) represents the aggregation from the global scenario tree for zone 1, where graph c) represents the aggregation from the global scenario tree for zone 2. Graph b) represents the zonal random events of a zone, hence the result of the aggregation.

TP2 2 time steps

- 2 zones
- 4 random events (2 demands, 2 inflow).

TP3 Has the aim of evaluating the effect of an increasing amount of zones.

- 5 time steps
- 4 zones
- 4 random events.

TP4 Has the aim of evaluating the effect of an increasing amount of time steps.

- 10 time steps
- 2 zones
- 4 random events.

These toy problems will be used as followed.

- The validations on decoupling the randomness are done with TP1.
- The validations on the interraction between ADMM and Dynamic Programming are accomplished on TP2, TP3 and TP4.

We recall here that we use a decision-hazard policy. Although this may have a reduced effect when T is large, our toy problems do not have many time steps and, therefore, this may have a significant impact on optimal solutions.

4.2 Preliminary results on the Global Scenarios to local scenarios relaxation

This toy problem uses scenario generation data provided from Électricité de France (EDF). As explained in Section 4.1, this validation aims to see on small scale problems whether or not the aggregation of randomness to local scenarios have a significant impact on the costs. The scenario aggregations are illustrated on Figure 1.

Our objective here is to verify the aggregation's difference in cost. We will start our analysis by solving, on many instances, the original problem's deterministic equivalent form through Matlab's linprog. A typical results table is available in Appendix A. Notice that the non-anticipativity constraints need to be hardcoded and are identified with (red/green/brown)-colored fonts. The different local randomness are also identified with background colors, although no aggregation on global scenarios are done.

Next comes computing solutions of the aggregated form of each instance we solved without relaxation. Tables 3 and 4 in the appendix section give the local scenarios and their given solutions, again using colors to identify their local randomness and variables restricted with non-anticipative constraints. Injecting those solutions in the original global scenarios with transfers yields Table 5. For conciseness, Table 1 sums up scenario total costs for the instance where transfers vary.

Table 1: Cost Comparison Problem with varying transfers.

Cost comparison	Scen.1	Scen.2	Scen.3	Scen.4	Scen.5	Scen.6	Scen.7	Scen.8	Scen.9	Scen.10	Scen.11	Scen.12
Without Aggregation (10^4)	3.5813	3.6084	3.6732	3.7003	3.7466	3.7736	3.9756	4.0027	4.1541	4.1812	4.3832	4.4102
With Aggregation (10^4)	3.7112	3.7382	3.7454	3.7725	3.7536	3.7807	3.7535	3.7806	4.2752	4.3022	4.2751	4.3021

Comparing the two results sets with different instances of the toy problem gave interesting insights. We can notice that :

- "trivial" transfer problems give exact total cost for each global scenario.
- when transfers vary from zone to zone, we get that each scenario ends up with a different cost than the non-relaxed problem, with the same expected total cost.

First, the "trivial" transfers happen when it is particularly easier for a certain zone to satisfy its demand at a low-cost rate compared to the other zone, making it always advantageous for the low-cost rate zone to produce the maximum possible transfer toward the high-cost rate zone. For such problems, the exact total costs for each scenario can be interpreted as global random scenarios naturally falling back to local ones, significantly reducing its random nature. This actually makes sense, as the main difference between global scenarios and local ones are the merged constraints:

$$\sum_{i=1}^{2}(\mathbf{u}^i + \mathbf{p}^i - \sum_{e\in z^+} \mathbf{f}_e^i) = \sum_{i=1}^{2}(\mathbf{d}^i - \sum_{e\in z^-} \mathbf{f}_e^i)$$

The variables u^i, p^i and d^i are all the same $\forall i = 1\dots 2$. The possible "communication" with non-local scenarios are done through the f variable. In a case where the $f^i = f$ are all the same, this aggregated constraint simply becomes :

$$2(\mathbf{u} + \mathbf{p} - \sum_{e\in z^+} \mathbf{f}_e) = 2(\mathbf{d} - \sum_{e\in z^-} \mathbf{f}_e).$$

Secondly, when transfers vary, our relaxation becomes an approximation of the optimal cost per scenario and optimal controls vary a little. However, an interesting conclusion is that expected total cost over every scenario is still the same! The reason why this is true is the fact that the cost function is not affected by the random events (see [21] for more bounding results with the aggregated stochastic models).

4.3 Preliminary results on ADMM-Dynamic Programming coupling

This section first presents preliminary results about the ADMM-Dynamic Programming Coupling. The results will consider already-relaxed local scenarios. The toy problems aim to distinguish how close to optimality the solutions output from the coupled algorithms are. Also, we want to see how the convergence of this coupling is limited and the effect of different parameters of the problem.

The results on the second toy problem are rather straightforward. With a deterministic equivalent formulation of the problem, Matlab's Linprog gets, for a certain instance, a total cost solution of 1.0286×10^5. Using the ADMM decomposition scheme to compute the transfers but using Quadprog to solve the zonal subproblems gave a total cost of 1.0291×10^5, so a relative error of 0.04%, concluding that for small enough problems, the decomposition scheme doesn't affect that much the final cost. Lastly, using an intuitive implementation of Dynamic Programming to solve the subproblems, we end up with a solution of 1.0442×10^5, so a relative error around 1.5% . The problem using Dynamic Programming takes a lot more time to compute than ADMM coupled with quadprog, but it is hoped the computation time of using Dynamic Programming will increase linearly instead of exponentially. Other instances of the toy problem gave similar results.

Next we validate the ADMM-Dynamic Programming coupled algorithm on many instances in order to see if a solution is found with larger problems. First, we take a five time steps and four zones problem. With a basic discretization of the state space, convergence seemed limited as the algorithm oscillates between four different solutions (see Tables 6 and 7). Increasing drastically the discretization's sharpness of the state space (from 51 to 201) seemed to help reduce the amount of solutions between which the algorithm ends up oscillating, although oscillation was still observed between 2 solutions (see Table 8). Second and lastly, we took a ten time steps problem with two zones in order to see if this oscillating effect was still present with less zones, but more time steps (this effect did not occur on any generated problem of 2 zones and 5 time steps). The algorithm still ended up oscillating, this time between two solutions. Slightly increasing the discretization's sharpness (51 to 81 states) made the algorithm converge to a single solution.

It is worthwhile to say that computation time before ending up oscillating seemed to increase about linearly with the parameters that were changed. An exception to this was the discretization's sharpness, which seemed to slow down more than linearly the Dynamic Programming algorithm. This makes sense, since increasing the problem's discretization makes the amount of states that Dynamic Programming must pass through larger, but it also gives more choices of possible hydro productions, which are computed for each discretization of the state space for each time step for each local scenario.

5 Conclusion

This technical report gives a general large multi-zonal stochastic model of an energy management problem on which the coupling of ADMM and Dynamic Programming might find solutions. Different relaxations from global scenarios to local scenarios are validated for small toy problems, and solutions seem to be of exact total expected cost. More precisely, for a certain subset of these problems where transfers are trivial, the relaxation yields exact costs for each scenario. The ADMM-Dynamic Programming based algorithm is then roughly validated, confirming on such toy problems the linear increase in computation time of Dynamic Programming as the number of zones and time steps get larger. A difficulty with the coupled algorithm is confirmed in which convergence is limited by Dynamic Programming's discretization sharpness, leading the coupled algorithm to oscillate between a set of solutions.

Future works includes the implementation of an efficient adaptive Dynamic Programming algorithm which will sharpen its discretization near the last found optimal solution while coarsing it far from the last found optimal solution. It would be interresting to test it against Benders-type methods, such as SDDP (see [8] for a first application of SDDP to a similar model). An extensive analysis of the convergence of the ADMM-Dynamic Programming scheme should theorically be possible, giving bounds depending on the algorithm's discretization sharpness. Those bounds could then be used to give an adapted ADMM ending criterion, making the algorithm stop before it starts oscillating and giving "good enough" results through the found bounds.

Appendix A

Table 2: Solution of Toy Problem 1 without relaxation (constraints aggregation). Background colors higlights local randomness for each zone at each time step. Colored font highlights enforced non-anticipativity constraints : for exemple for a zone and time step, green font variables are linked and forced equal.

Variable	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6	Scenario 7	Scenario 8	Scenario 9	Scenario 10	Scenario 11	Scenario 12
p zone 1 time 1	26.8860				26.8860				27.5125			
u zone 1 time 1	6.6078				6.6078				6.6078			
x zone 1 time 1	281.2500				281.2500				281.2500			
p zone 2 time 1	67.6672				68.3151				68.3151			
u zone 2 time 1	3.5405				3.5405				3.5405			
x zone 2 time 1	125.0000				125.0000				125.0000			
Transfers 2->1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
transfers 1->2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
p zone 1 time 2	21.2596	23.7662	26.8788	28.8542	27.0059	28.8410	27.0053	28.8415	24.4905	27.3556	27.8387	29.0502
u zone 1 time 2	6.8432	6.8432	6.8432	6.8432	6.8432	6.8432	6.8432	6.8432	6.7798	6.7798	6.7798	6.7798
x zone 1 time 2	281.3299	281.3299	281.3299	281.3299	281.3299	281.3299	281.3299	281.3299	281.2965	281.2965	281.2965	281.2965
p zone 2 time 2	62.3748	68.4920	63.7011	71.4325	61.1980	67.6703	61.1545	70.5362	67.8234	67.8241	70.4629	70.4659
u zone 2 time 2	2.8716	2.8716	2.8716	2.8716	2.7591	2.7591	2.7591	2.7591	2.7591	2.7591	2.7591	2.7591
x zone 2 time 2	124.7406	124.7406	124.7406	124.7406	124.6813	124.6813	124.6813	124.6813	124.6813	124.6813	124.6813	124.6813
transfers 2->1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
transfers 1->2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
x_T	281.3815	281.2500	281.3815	281.2500	281.3815	281.2500	281.3815	281.2500	281.3815	281.2500	281.3815	281.2500
	125.2786	125.2786	125.0000	125.0000	125.2786	125.2786	125.0000	125.0000	125.2786	125.2786	125.0000	125.0000
Cost Zone 1 (10^3)	44.871	48.296	44.871	48.296	44.871	48.296	44.871	48.296	46.977	50.402	46.977	50.402
Cost Zone 2 (10^3)	78.121	78.121	80.681	80.681	79.733	79.733	82.293	82.293	79.733	79.733	82.293	82.293
Total Cost (10^3)	123.04	126.46	125.60	129.02	124.65	128.08	127.21	130.64	126.76	130.18	129.32	132.74

Appendix B

Table 3: Four zonal scenarios given from the constraints aggregation for zone 1. Background colors are local random events and can be linked with those of Table 5.

Zone 1	Scenario 1	Scenario 2	Scenario3	Scenario 4
p time 1	26.2601		26.8866	
u time 1	7.2336		7.2336	
x time 1	281.2500		281.2500	
p time 2	26.1633	27.7378	26.7904	28.8287
u time 2	6.2173	6.2173	6.1540	6.1540
x time 2	280.7040	280.7040	280.6707	280.6707
x_T	281.3815	281.2500	281.3815	281.2500
Cost Zone 1 (10^3)	44.871	48.296	46.977	50.402

Appendix C

Table 4: Four zonal scenarios given from the constraints aggregation for zone 2. Background colors are local random events and can be linked with those of Table 5.

Zone 2	Scenario 1	Scenario 2	Scenario3	Scenario 4
p time 1	67.7577		68.4055	
u time 1	3.4326		3.4326	
x time 1	125.0000		125.0000	
p time 2	65.3429	67.4763	66.0385	68.1719
u time 2	2.9621	2.9621	2.8496	2.8496
x time 2	124.8310	124.8310	124.7718	124.7718
x_T	125.2786	125.0000	125.2786	125.0000
Cost Zone 2 (10^3)	78.121	80.681	79.733	82.293

Appendix D

Table 5: Global scenario simulation with relaxation. The zonal variables here are computed locally and fit with zonal results from Tables 3 and 4. Here again, colored font highlights non-anticipativity constraints. Blank spaces take the same values as those of the same background color at each time step.

Variable	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6	Scenario 7	Scenario 8	Scenario 9	Scenario 10	Scenario 11	Scenario 12
p zone 1 time 1	26.2601				26.2601				26.8866			
u zone 1 time 1	7.2336				7.2336				7.2336			
x zone 1 time 1	281.2500				281.2500				281.2500			
p zone 2 time 1	67.7577				68.4055				68.4055			
u zone 1 time 1	3.4326				3.4326				3.4326			
x zone 1 time 1	125.0000				125.0000				125.0000			
Transfers 2->1	1.0000				1.0000				1.0000			
transfers 1->2	0.0000				0.0000				0.0000			
p zone 1 time 2	26.1633	28.2016			26.1633	28.2016			26.7904	28.8287		
u zone 1 time 2	6.2173	6.2173			6.2173	6.2173			6.1540	6.1540		
x zone 1 time 2	280.7040	280.7040			280.7040	280.7040			280.6707	280.6707		
p zone 2 time 2	65.3429		67.4763		66.0385		68.1719		66.0385		68.1719	
u zone 2 time 2	2.9621		2.9621		2.8496		2.8496		2.8496		2.8496	
x zone 2 time 2	124.8310		124.8310		124.7718		124.7718		124.7718		124.7718	
transfers 2->1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
transfers 1->2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
x_T	281.3815	281.2500	281.3815	281.2500	281.3815	281.2500	281.3815	281.2500	281.3815	281.2500	281.3815	281.2500
	125.2786	125.2786	125.0000	125.0000	125.2786	125.2786	125.0000	125.0000	125.2786	125.2786	125.0000	125.0000
Cost Zone 1 (10^3)	44.871	48.296		48.296	44.871	48.296		48.296	46.977	50.402		
Cost Zone 2 (10^3)	78.121		80.681		79.733		82.293		79.733		82.293	
Total Cost (10^3)	123.04	126.46	125.60	129.02	124.65	128.08	127.21	130.64	126.76	130.18	129.32	132.74

Appendix E

Table 6: Values of Dual Variables of a problem with 4 zones, 5 time steps with an oscillation between 4 solutions. Dynamic Programming discretize the state variable in 51 states. Two of those solutions are explicited here on two separate iteration each. Notice values of the dual variables for iteration 43 and 47 are the same, as well as values of the dual variables of iteration 44 and 48. Pale blue lines are variables that converged, where red lines are oscillating variables as seen in Table 7.

Iteration 43	Iteration 47	Iteration 44	Iteration 48
1.16400000e+03	1.16400000e+03	1.16400000e+03	1.16400000e+03
1.15200000e+03	1.15200000e+03	1.15200000e+03	1.15200000e+03
1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03
1.16400000e+03	1.16400000e+03	1.16400000e+03	1.16400000e+03
1.28377561e+03	1.28377561e+03	1.28377561e+03	1.28377561e+03
1.30000000e+03	1.30000000e+03	1.30000000e+03	1.30000000e+03
1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03
1.21400000e+03	1.21400000e+03	1.25622439e+03	1.25622439e+03
1.28600925e+03	1.28600926e+03	1.28600925e+03	1.28600926e+03
1.30000003e+03	1.30000003e+03	1.30000003e+03	1.30000003e+03
1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03
1.21399999e+03	1.21399999e+03	1.25399072e+03	1.25399071e+03
1.16400000e+03	1.16400000e+03	1.16400000e+03	1.16400000e+03
1.15200000e+03	1.15200000e+03	1.15200000e+03	1.15200000e+03
1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03
1.16400000e+03	1.16400000e+03	1.16400000e+03	1.16400000e+03
1.28580993e+03	1.28580999e+03	1.28580996e+03	1.28581001e+03
1.30000002e+03	1.30000002e+03	1.30000002e+03	1.30000002e+03
1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03
1.21399997e+03	1.21399997e+03	1.25419002e+03	1.25418997e+03

Appendix F

Table 7: Values of Dual Variables for all four solutions of a 4 zones 5 time steps problem with 51 discretizations of the state space. Pale blue lines are variables that converged, where red lines are oscillating variables. Notice values of iteration 46 and 50 are almost the same for all variables.

Iteration 46	Iteration 47	Iteration 48	Iteration 49	Iteration 50
1.16400000e+03	1.16400000e+03	1.16400000e+03	1.16400000e+03	1.16400000e+03
1.15200000e+03	1.15200000e+03	1.15200000e+03	1.15200000e+03	1.15200000e+03
1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03
1.16400000e+03	1.16400000e+03	1.16400000e+03	1.16400000e+03	1.16400000e+03
1.29079025e+03	1.28377561e+03	1.28377561e+03	1.29079024e+03	1.29079024e+03
1.30000000e+03	1.30000000e+03	1.30000000e+03	1.30000000e+03	1.30000000e+03
1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03
1.24920975e+03	1.21400000e+03	1.25622439e+03	1.21400000e+03	1.24920976e+03
1.28126154e+03	1.28600926e+03	1.28600926e+03	1.28126153e+03	1.28126153e+03
1.30000000e+03	1.30000003e+03	1.30000003e+03	1.30000000e+03	1.30000000e+03
1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03
1.25873846e+03	1.21399999e+03	1.25399071e+03	1.21400000e+03	1.25873847e+03
1.16400000e+03	1.16400000e+03	1.16400000e+03	1.16400000e+03	1.16400000e+03
1.15200000e+03	1.15200000e+03	1.15200000e+03	1.15200000e+03	1.15200000e+03
1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03
1.16400000e+03	1.16400000e+03	1.16400000e+03	1.16400000e+03	1.16400000e+03
1.30496381e+03	1.28580999e+03	1.28581001e+03	1.30496378e+03	1.30496376e+03
1.30000001e+03	1.30000002e+03	1.30000002e+03	1.30000001e+03	1.30000001e+03
1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03
1.23503618e+03	1.21399997e+03	1.25418997e+03	1.21399999e+03	1.23503623e+03

Appendix G

Table 8: Values of the Dual Variable for the four last iterations of a 4 zones 5 time steps problem with 201 discretizations of the state space. Pale blue lines are variables that converged. where the red line is the oscillating variable. We observe less oscillating variables than results in Table 7. However, the oscillating variable do so with more amplitude.

Iteration 47	Iteration 48	Iteration 49	Iteration 50
1.27600035e+03	1.27600000e+03	1.27600034e+03	1.27600000e+03
1.26402219e+03	1.26402254e+03	1.26402254e+03	1.26402288e+03
1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03
1.29997746e+03	1.29997746e+03	1.29997712e+03	1.29997712e+03
1.27600000e+03	1.27600034e+03	1.27600000e+03	1.27600033e+03
1.26402087e+03	1.26402087e+03	1.26402121e+03	1.26402121e+03
1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03
1.29997913e+03	1.29997879e+03	1.29997879e+03	1.29997846e+03
1.27600000e+03	1.27600033e+03	1.27600000e+03	1.27600031e+03
1.26402116e+03	1.26402116e+03	1.26402149e+03	1.26402149e+03
1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03
1.29997884e+03	1.29997851e+03	1.29997851e+03	1.29997820e+03
1.27600037e+03	1.27600000e+03	1.27600035e+03	1.27600000e+03
1.26402184e+03	1.26402221e+03	1.26402221e+03	1.26402256e+03
1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03
1.29997779e+03	1.29997779e+03	1.29997744e+03	1.29997744e+03
1.27600036e+03	1.27600000e+03	1.27600034e+03	1.27600000e+03
1.29997713e+03	1.29997713e+03	1.29997679e+03	1.29997679e+03
1.80000000e+03	1.80000000e+03	1.80000000e+03	1.80000000e+03
1.18879633e+03	1.26402287e+03	1.18879669e+03	1.26402321e+03

References

[1] R. E. Bellman. The theory of dynamic programming. Bull. Amer. Math. Soc., 60(6):503–515, 11 1954.

[2] J. R. Birge. Aggregation bounds in stochastic linear programming. Mathematical Programming, 31(1):25–41, Jan 1985.

[3] J. R. Birge. Decomposition and partitioning methods for multistage stochastic programs. Operations Research, 33:989–1007, 1985.

[4] J. R. Birge and F. Louveaux. Introduction to Stochastic Programming. Springer-Verlag, New York, NY, USA, 1997.

[5] D Gabay. Applications of the method of multipliers to variational inequalities. Studies in Mathematics and its Applications, 15, 1983.

[6] O. M. Guèye, J. P. Dussault, and P. Mahey. Separable augmented lagrangian algorithm with multidimensional scaling for monotropic programming. Journal of Optimization Theory and Applications, 127(2):329–345, 2005.

[7] J. Higle. Stochastic Programming : Optimization When Uncertainty Matters. Tutorials in Operations Research, pages 30–53, 2005.

[8] M. Lemery. Stochastic dual dynamic programming applied to a long-term energy production planning problem. Dissertation for Master's Degree, 2017.

[9] A. Lenoir and P Mahey. Operator Splitting and Decomposition of Convex Programs. RAIRO - Operations Research, 51(1):17–41, 2017.

[10] P. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. SIAM Journal on Numerical Analysis, 16(6):964–979, 1979.

[11] P. Mahey, J. Koko, and A. Lenoir. Decomposition methods for a spatial model for long-term energy pricing problems. Math. Methods of Oper. Res., 85:137–153, 2017.

[12] B. Martinet. Détermination approchée d'un point fixe d'une application pseudo-contratante. cas de l'application prox. Comptes Rendus de l'Académie des Sciences, Série A, (274):163–165, 1972.

[13] N. Parikh and S. Boyd. Proximal algorithms. Found. Trends Optim., 1(3):127–239, January 2014.

[14] M.V.F. Pereira and M.V.G. Pinto. Multi-stage Stochastic Optimization applied to Energy Planning. Mathematical Programming, 52(2):359–375, 1991.

[15] W. P. Powell. Approximate Dynamic Programming. John Wiley and Sons, second edition edition, 2011.

[16] R.T. Rockafellar. Convex Analysis. Princeton landmarks in mathematics and physics. Princeton University Press, 1970.

[17] R.T. Rockafellar. Monotone operators and the proximal point algorithm. SIAM Journal on Control and Optimization, 14(5):877–898, 1976.

[18] R.T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. Mathematical Operations Research, 16(1):119–147, 1991.

[19] J. E. Spingarn. Partial inverse of a monotone operator. Applied Mathematics and Optimization, 10(1):247–265, 1983.

[20] S. W. Wallace and S.-E. Fleten. Stochastic programming models in energy, pages 637–677. (Handbooks in Operations Research and Management Science). Elsevier, 2003.

[21] S. E. Wright. Primal-dual aggregation and disaggregation for stochastic linear programs. Mathematics of Operations Research, 19(4):893–908, 1994.