Optimization of Energy Production and Transport

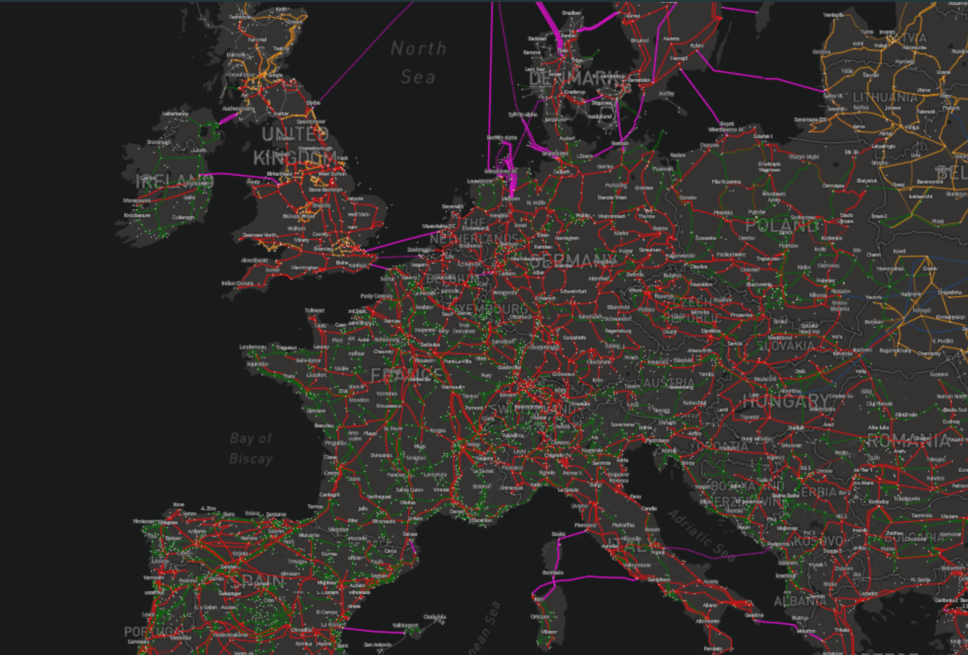
A stochastic decomposition approach

P. Carpentier, J.-P. Chancelier, A. Lenoir, F. Pacaud

PGMO Days — November 14, 2017

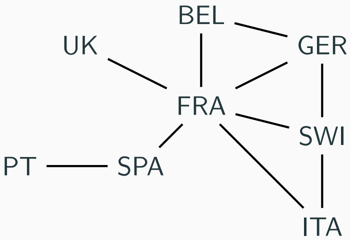
ENSTA ParisTech — ENPC ParisTech — EDF Lab — Efficacity

Managing the network at European scale



Motivation

An energy production and transport optimization problem on a grid modeling energy exchange across European countries.[1]



- Stochastic dynamical problem
- Discrete time formulation (weekly time steps)
- Large-scale problem (8 countries)

[1] But the framework remains valid for smaller energy management problems.

Lecture outline

Modeling

Resolution methods

- Stochastic Programming
- Time decomposition
- Spatial decomposition

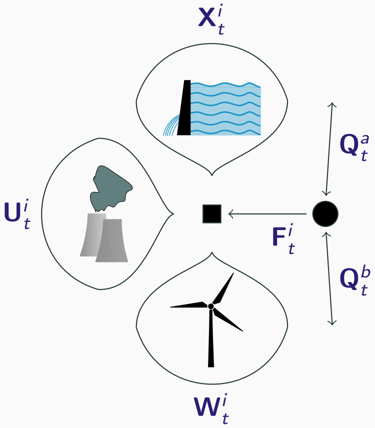
Numerical implementation

Conclusion

Modeling

Production at each node of the grid

At each node i of the grid, we formulate a production problem on a discrete time horizon $[\![0, T]\!]$, involving the following variables at each time t:



- $\mathbf{X}_t^i$: state variable
  (dam volume)
- $\mathbf{U}_t^i$: control variable
  (energy production)
- $\mathbf{F}_t^i$: grid flow
  (import/export from the grid)
- $\mathbf{W}_t^i$: noise
  (consumption, renewable)

Writing the problem for each node

For each node $i \in [\![1, N]\!]$:

- The dynamics $x_{t+1}^i = f_t^i(x_t^i, u_t^i, w_t^i)$ writes

$$x_{t+1}^i = x_t^i + \underbrace{a_t^i}_{\text{inflow}} - \underbrace{p_t^i}_{\text{turbinate}} - \underbrace{s_t^i}_{\text{spillage}} .$$

- The load balance (supply = demand) gives

$$\underbrace{p_t^i}_{\text{turbinate}} + \underbrace{g_t^i}_{\text{thermal}} + \underbrace{r_t^i}_{\text{recourse}} + \underbrace{f_t^i}_{\text{grid flow}} = \underbrace{d_t^i}_{\text{demand}} .$$

Writing the problem for each node

For each node $i \in [\![1, N]\!]$:

- The dynamics $x_{t+1}^i = f_t^i(x_t^i, u_t^i, w_t^i)$ writes

$$x_{t+1}^i = x_t^i + \underbrace{a_t^i}_{\text{inflow}} - \underbrace{p_t^i}_{\text{turbinate}} - \underbrace{s_t^i}_{\text{spillage}} .$$

- The load balance (supply $=$ demand) gives

$$\underbrace{p_t^i}_{\text{turbinate}} + \underbrace{g_t^i}_{\text{thermal}} + \underbrace{r_t^i}_{\text{recourse}} + \underbrace{f_t^i}_{\text{grid flow}} = \underbrace{d_t^i}_{\text{demand}} .$$

Thus, we explicit w_t^i and u_t^i

$$w_t^i = (a_t^i, d_t^i) , \quad u_t^i = (p_t^i, s_t^i, g_t^i, r_t^i) .$$

We pay to use the thermal power plant and we penalize the recourse:

$$L_t^i(x_t^i, u_t^i, f_t^i, w_t^i) = \underbrace{\alpha_t^i (g_t^i)^2 + \beta_t^i g_t^i}_{\text{quadratic cost}} + \underbrace{\kappa_t^i r_t^i}_{\text{recourse penalty}} .$$

A stochastic optimization problem decoupled in space

At each node i of the grid, we have to solve a stochastic optimal control subproblem depending on the grid flow process $\mathbf{F}^i$:[2]

$$J^i_{\mathfrak{P}}[\mathbf{F}^i] = \min_{\mathbf{X}^i, \mathbf{U}^i} \mathbb{E}\Big(\sum_{t=0}^{T-1} L^i_t(\mathbf{X}^i_t, \mathbf{U}^i_t, \mathbf{F}^i_t, \mathbf{W}^i_{t+1}) + K^i(\mathbf{X}^i_T) \Big) ,$$

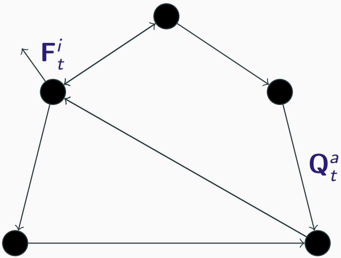
$$\begin{aligned} \text{s.t.} \quad & \mathbf{X}^i_{t+1} = f^i_t(\mathbf{X}^i_t, \mathbf{U}^i_t, \mathbf{F}^i_t, \mathbf{W}^i_{t+1}) , \\ & \mathbf{X}^i_t \in \mathcal{X}^{i,\mathrm{ad}}_t , \quad \mathbf{U}^i_t \in \mathcal{U}^{i,\mathrm{ad}}_t , \\ & \mathbf{U}^i_t \preceq \mathcal{F}_t , \end{aligned}$$

The last equation is the measurability constraint, where $\mathcal{F}_t$ is the σ-field generated by the noises $\{\mathbf{W}^i_\tau\}_{\tau=1\ldots t, i=1\ldots N}$ up to time t.

[2] The notation $J^i_{\mathfrak{P}}[\cdot]$ means that the argument of $J^i_{\mathfrak{P}}$ is a *random variable*.

Modeling exchanges between countries

The grid is represented by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$. At each time $t \in [\![0, T-1]\!]$ we have:



- a flow $\mathbf{Q}_t^a$ through each arc a, inducing a cost $c_t^a(\mathbf{Q}_t^a)$, modeling the exchange between two countries
- a grid flow $\mathbf{F}_t^i$ at each node i, resulting from the balance equation

$$\mathbf{F}_t^i = \sum_{a \in input(i)} \mathbf{Q}_t^a - \sum_{b \in output(i)} \mathbf{Q}_t^b$$

A transport cost decoupled in time

At each time step $t \in [\![0, T-1]\!]$, we define the transport cost as the sum of the cost of the flows $\mathbf{Q}_t^a$ through the arcs a of the grid:

$$J_{\mathfrak{T},t}[\mathbf{Q}_t] = \mathbb{E}\Big(\sum_{a \in \mathcal{A}} c_t^a(\mathbf{Q}_t^a)\Big) \ ,$$

where the c_t^a's are easy to compute functions (say quadratic).

Kirchhoff's law

The balance equation stating the conservation between $\mathbf{Q}_t$ and $\mathbf{F}_t$ rewrites in the following matrix form:

$$A\mathbf{Q}_t + \mathbf{F}_t = 0 \ ,$$

where A is the node-arc incidence matrix of the grid.

The overall production transport problem

The *production cost* $J_{\mathfrak{P}}$ aggregates the costs at all nodes i:

$$J_{\mathfrak{P}}[\mathbf{F}] = \sum_{i \in \mathcal{N}} J^i_{\mathfrak{P}}[\mathbf{F}^i] \; ,$$

and the *transport cost* $J_{\mathfrak{T}}$ aggregates the costs at all time t:

$$J_{\mathfrak{T}}[\mathbf{Q}] = \sum_{t=0}^{T-1} J_{\mathfrak{T},t}[\mathbf{Q}_t] \; .$$

The compact production-transport problem formulation writes:

$$\begin{aligned} \min_{\mathbf{Q},\mathbf{F}} \quad & J_{\mathfrak{P}}[\mathbf{F}] + J_{\mathfrak{T}}[\mathbf{Q}] \\ \text{s.t.} \quad & A\mathbf{Q} + \mathbf{F} = 0 \; . \end{aligned} \tag{$\mathcal{P}$}$$

Resolution methods

Where are we heading to?

The problem $\mathcal{P}$ has:

- N nodes (with $N = 8$);
- T time steps (with $T = 52$);
- N independent random variables per time step t: $\mathbf{W}_t^1, \cdots, \mathbf{W}_t^N$.

We aim to solve the problem numerically. We suppose that for all t, $\mathbf{W}_t^i$ is a discrete random variable, with support size $\mathfrak{n}_{bin}$. We denote by

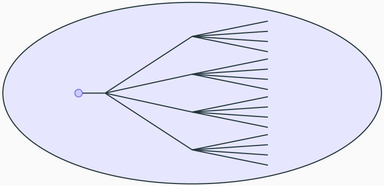
$$\mathbf{W}_t = (\mathbf{W}_t^1, \cdots, \mathbf{W}_t^N) ,$$

the global random variable at time t.

First idea: solving the whole problem inplace!

Write the problem and solve it!

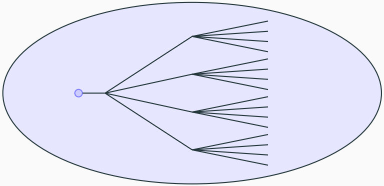
$\mathcal{P}$:



First idea: solving the whole problem inplace!

Write the problem and solve it!

$\mathcal{P}$:



But ...

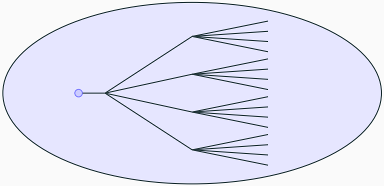
- $N = 8$ nodes and $T = 52$ time steps.
- Non-anticipativity constraint: we ought to formulate the problem on a tree (Stochastic Programming approach)
- We suppose that $\mathbf{W}_t^1, \cdots, \mathbf{W}_t^N$ are space independent. The support size of $\mathbf{W}_t$ is equal to $\mathfrak{n}_{bin}^N$ ...

$$\text{number of nodes} \quad \propto \quad (\mathfrak{n}_{bin}^N)^T$$

First idea: solving the whole problem inplace!

Write the problem and solve it!

$\mathcal{P}$:



But ...

- $N = 8$ nodes and $T = 52$ time steps.
- Non-anticipativity constraint: we ought to formulate the problem on a tree (Stochastic Programming approach)
- We suppose that $\mathbf{W}_t^1, \cdots, \mathbf{W}_t^N$ are space independent. The support size of $\mathbf{W}_t$ is equal to $\mathfrak{n}_{bin}^N$ ...

$$\text{number of nodes} \quad \propto \quad (\mathfrak{n}_{bin}^N)^T$$

$\mathfrak{n}_{bin}$	1	2	5
n leafs	1	$\approx 10^{125}$	$\approx 10^{290}$

Second idea: Dynamic Programming

We assume that the noise $\mathbf{W}_0, \cdots, \mathbf{W}_T$ are independent.
We decompose the problem time step by time step $\rightarrow$ T subproblems

$$\mathcal{P}: \quad V_1 \leftarrow V_2 \leftarrow V_3 \leftarrow \cdots \cdots V_T$$

Second idea: Dynamic Programming

We assume that the noise $\mathbf{W}_0, \cdots, \mathbf{W}_T$ are independent.
We decompose the problem time step by time step $\rightarrow$ T subproblems

$$\mathcal{P}: \quad V_1 \longleftarrow V_2 \longleftarrow V_3 \leftarrow \cdots\cdots \; V_T$$

We use Dynamic Programming to compute the value functions $V_1, \cdots, V_T$.

But ...

- N nodes: curse of dimensionality (8 decoupled stocks dynamics).
- Still a support size $\mathfrak{n}_{bin}^N$ for $\mathbf{W}_t$

We use Stochastic Dual Dynamic Programming to solve the problem with $N = 8$ dimensions.

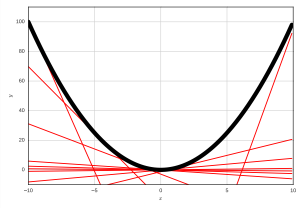
A brief recall on Stochastic Dynamic Programming

Dynamic Programming

We compute value functions with the backward equation:

$$V_T(x) = K(x)$$
$$V_t(x_t) = \min_{u_t} \mathbb{E}\Big[\underbrace{L_t(x_t, u_t, \mathbf{W}_{t+1})}_{\text{current cost}} + \underbrace{V_{t+1}\Big(f(x_t, u_t, \mathbf{W}_{t+1})\Big)}_{\text{future costs}} \Big]$$

Stochastic Dual Dynamic Programming



- Convex value functions V_t are approximated as a supremum of a finite set of affine functions
- Affine functions (=cuts) are computed during forward/backward passes, till convergence

$$\widetilde{V}_t(x) = \max_{1 \leq k \leq K} \{\lambda_t^k x + \beta_t^k\} \leq V_t(x)$$

- SDDP makes an extensive use of LP/QP solver

A brief recall on Stochastic Dynamic Programming

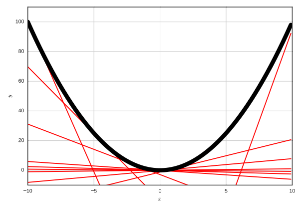
Dynamic Programming

We compute value functions with the backward equation:

$$V_T(x) = K(x)$$

$$V_t(x_t) = \min_{u_t} \mathbb{E}\Big[\underbrace{L_t(x_t, u_t, \mathbf{W}_{t+1})}_{\text{current cost}} + \underbrace{V_{t+1}\Big(f(x_t, u_t, \mathbf{W}_{t+1})\Big)}_{\text{future costs}}\Big]$$

Stochastic Dual Dynamic Programming



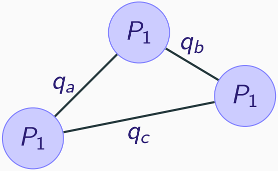
- Convex value functions V_t are approximated as a supremum of a finite set of affine functions
- Affine functions (=cuts) are computed during forward/backward passes, till convergence

$$\widetilde{V}_t(x) = \max_{1 \le k \le K} \{\lambda_t^k x + \beta_t^k\} \le V_t(x)$$

- SDDP makes an extensive use of LP/QP solver

However, SDDP still has to deal with a noise $\mathbf{W}_t$ with a support size $\mathfrak{n}_{bin}^N$...

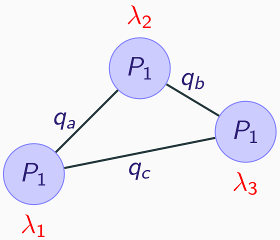
Introducing decentralized decomposition methods



$$\min_{\mathbf{Q},\mathbf{F}} \quad J_{\mathfrak{P}}[\mathbf{F}] + J_{\mathfrak{T}}[\mathbf{Q}] \qquad (\mathcal{P})$$

$$\text{s.t.} \quad A\mathbf{Q} + \mathbf{F} = 0 \qquad .$$

Introducing decentralized decomposition methods

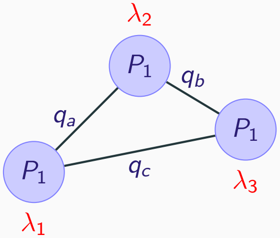


$$\min_{\mathbf{Q},\mathbf{F}} \quad J_{\mathfrak{P}}[\mathbf{F}] + J_{\mathfrak{T}}[\mathbf{Q}] \qquad (\mathcal{P})$$

$$\text{s.t.} \;\; A\mathbf{Q} + \mathbf{F} = 0 \quad \rightsquigarrow \underbrace{\lambda}_{price} .$$

Once the price λ is fixed, we can decompose the problem $\mathcal{P}$ in 3 independent subproblems $\mathcal{P}_1, \cdots, \mathcal{P}_3$.

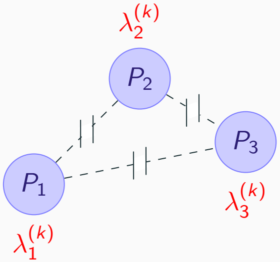
Introducing decentralized decomposition methods



$$\min_{\mathbf{Q},\mathbf{F}} \quad J_{\mathfrak{P}}[\mathbf{F}] + J_{\mathfrak{T}}[\mathbf{Q}] \qquad (\mathcal{P})$$

$$\text{s.t.} \quad A\mathbf{Q} + \mathbf{F} = 0 \quad \rightsquigarrow \underbrace{\lambda}_{price} .$$

Once the price λ is fixed, we can decompose the problem $\mathcal{P}$ in 3 independent subproblems $\mathcal{P}_1, \cdots, \mathcal{P}_3$.



Dual decomposition:

- Fix a voltage $\lambda^{(k)}$
- Decouple the problem **node by node**
- Solve $P_1, \cdots, P_3$ by Dynamic Programming and get an outflow $\mathbf{F}$
- Solve transport problem and get flow $\mathbf{Q}$
- Update λ with:

$$\lambda^{(k+1)} = \lambda^{(k)} + \rho \times \underbrace{(A\mathbf{Q} + \mathbf{F})}_{=0 \text{ if equilibrium}}$$

Recalling the original problem

$$\min_{\mathbf{Q},\mathbf{F}} \quad J_{\mathfrak{P}}[\mathbf{F}] + J_{\mathfrak{T}}[\mathbf{Q}] \tag{$\mathcal{P}$}$$

$$\text{s.t.} \;\; A\mathbf{Q} + \mathbf{F} = 0 \; .$$

where

- $J_{\mathfrak{P}}(\mathbf{F}) = \sum_{i=1}^{N} J_{\mathfrak{P}}^{i}(\mathbf{F}^{i})$ with

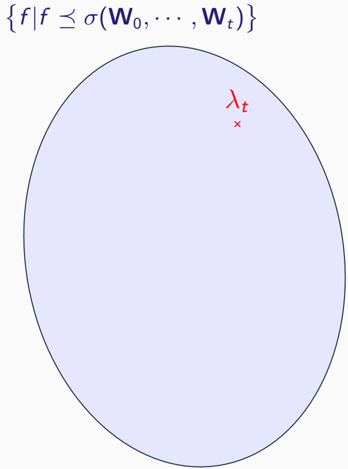
$$J_{\mathfrak{P}}^{i}[\mathbf{F}^{i}] = \min_{\mathbf{X}^{i},\mathbf{U}^{i}} \; \mathbb{E}\Big(\sum_{t=0}^{T-1} L_t^i(\mathbf{X}_t^i, \mathbf{U}_t^i, \mathbf{F}_t^i, \mathbf{W}_{t+1}^i) + K^i(\mathbf{X}_T^i) \Big) \; ,$$

$$\text{s.t.} \quad \text{lot of constraints}$$

- $\mathbf{F} = \mathbf{F}_0, \cdots, \mathbf{F}_{T-1}$ is a process,
- so is $\mathbf{Q} = \mathbf{Q}_0, \cdots, \mathbf{Q}_{T-1}$.

$\rightsquigarrow$ λ appears to be also a time process ...

Decomposition appears more complicated than expected



$\lambda^{(k)} = (\lambda_1^{(k)}, \lambda_2^{(k)}, \cdots, \lambda_T^{(k)})$ is a processus, correlated in time:

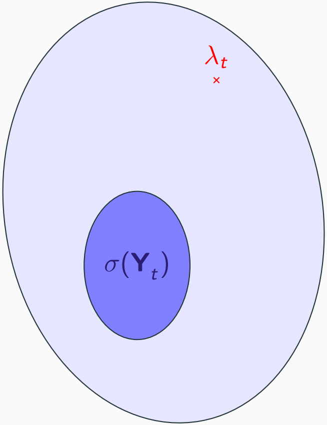
- $\lambda_t^{(k)}$ depends on past history

$$\lambda_t^{(k)} = \phi_t(\mathbf{W}_0, \cdots, \mathbf{W}_t) \ldots$$

- ... and $\lambda^{(k)}$ is a "noise" in the subproblems $P_1, \cdots, P_N$

Decomposition appears more complicated than expected

$\{f | f \preceq \sigma(\mathbf{W}_0, \cdots, \mathbf{W}_t)\}$



$\lambda^{(k)} = (\lambda_1^{(k)}, \lambda_2^{(k)}, \cdots, \lambda_T^{(k)})$ is a processus, correlated in time:

- $\lambda_t^{(k)}$ depends on past history

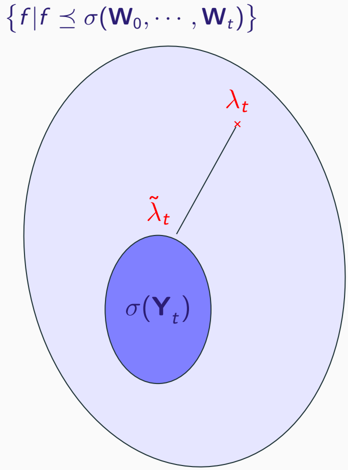
$$\lambda_t^{(k)} = \phi_t(\mathbf{W}_0, \cdots, \mathbf{W}_t) \ldots$$

- ... and $\lambda^{(k)}$ is a "noise" in the subproblems $P_1, \cdots, P_N$

We use a relaxation to overcome this issue:

- We introduce an information process $\mathbf{Y}_t$, whose dynamics is known

Decomposition appears more complicated than expected



$\lambda^{(k)} = (\lambda_1^{(k)}, \lambda_2^{(k)}, \cdots, \lambda_T^{(k)})$ is a processus, correlated in time:

- $\lambda_t^{(k)}$ depends on past history

$$\lambda_t^{(k)} = \phi_t(\mathbf{W}_0, \cdots, \mathbf{W}_t) \dots$$

- ... and $\lambda^{(k)}$ is a "noise" in the subproblems $P_1, \cdots, P_N$

We use a relaxation to overcome this issue:

- We introduce an information process $\mathbf{Y}_t$, whose dynamics is known
- We approximate $\lambda_t^{(k)}$ by its conditional expectation w.r.t. $\mathbf{Y}_t$

$$\tilde{\lambda}_t^{(k)} = \mathbb{E}(\lambda_t^{(k)} | \mathbf{Y}_t)$$

Price decomposition

The production and transport optimization problem writes

$$\min_{\mathbf{Q},\mathbf{F}} J_{\mathfrak{P}}[\mathbf{F}] + J_{\mathfrak{T}}[\mathbf{Q}] \qquad \text{s.t.} \quad A\mathbf{Q} + \mathbf{F} = 0 \, . \tag{$\mathcal{P}$}$$

The decomposition scheme consists in:

1. dualizing the constraint,
2. approximating the multiplier $\boldsymbol{\lambda}$ by its conditional expectation w.r.t. $\mathbf{Y}$.

This trick leads to the following problem

$$\max_{\boldsymbol{\lambda}} \min_{\mathbf{Q},\mathbf{F}} \; J_{\mathfrak{P}}[\mathbf{F}] + J_{\mathfrak{T}}[\mathbf{Q}] + \left\langle \mathbb{E}(\boldsymbol{\lambda} \mid \mathbf{Y}) \, , A\mathbf{Q} + \mathbf{F} \right\rangle \, .$$

A dual gradient-like algorithm

Applying the Uzawa algorithm to the dual problem

$$\max_{\boldsymbol{\lambda}} \min_{\mathbf{Q},\mathbf{F}} \quad J_{\mathfrak{P}}[\mathbf{F}] + J_{\mathfrak{T}}[\mathbf{Q}] + \left\langle \mathbb{E}\left(\boldsymbol{\lambda} \mid \mathbf{Y}\right), A\mathbf{Q} + \mathbf{F} \right\rangle ,$$

leads to a decomposition between production and transport:

$$\mathbf{F}^{(k+1)} \in \arg\min_{\mathbf{F}} J_{\mathfrak{P}}[\mathbf{F}] + \left\langle \mathbb{E}\left(\boldsymbol{\lambda}^{(k)} \mid \mathbf{Y}\right), \mathbf{F} \right\rangle , \qquad \text{Production}$$

$$\mathbf{Q}^{(k+1)} \in \arg\min_{\mathbf{Q}} J_{\mathfrak{T}}[\mathbf{Q}] + \left\langle \mathbb{E}\left(\boldsymbol{\lambda}^{(k)} \mid \mathbf{Y}\right), A\mathbf{Q} \right\rangle , \qquad \text{Transport}$$

$$\mathbb{E}\left(\boldsymbol{\lambda}^{(k+1)} \mid \mathbf{Y}\right) = \mathbb{E}\left(\boldsymbol{\lambda}^{(k)} \mid \mathbf{Y}\right) + \rho\, \mathbb{E}\left(A\mathbf{Q}^{(k+1)} + \mathbf{F}^{(k+1)} \mid \mathbf{Y}\right) . \qquad \text{Update}$$

Decomposing the production problem

The production subproblem

$$\min_{\mathbf{F}} J_{\mathfrak{P}}[\mathbf{F}] + \left\langle \mathbb{E}\left(\boldsymbol{\lambda}^{(k)} \mid \mathbf{Y}\right), \mathbf{F} \right\rangle ,$$

evidently decomposes node by node

$$\min_{\mathbf{F}^i} J_{\mathfrak{P}}^i[\mathbf{F}^i] + \left\langle \mathbb{E}\left(\boldsymbol{\lambda}^{i,(k)} \mid \mathbf{Y}\right), \mathbf{F}^i \right\rangle ,$$

hence a stochastic optimal control subproblem for each node i:

$$\begin{aligned} \min_{\mathbf{X}^i,\mathbf{U}^i,\mathbf{F}^i} \ & \mathbb{E}\left(\sum_{t=0}^{T-1} \Big(L_t^i(\mathbf{X}_t^i, \mathbf{U}_t^i, \mathbf{F}_t^i, \mathbf{W}_{t+1}) + \left\langle \mathbb{E}\left(\boldsymbol{\lambda}_t^{i,(k)} \mid \mathbf{Y}_t\right), \mathbf{F}_t^i \right\rangle \Big) + K^i(\mathbf{X}_T^i) \right) \\ \text{s.t.} \ & \mathbf{X}_{t+1}^i = f_t^i(\mathbf{X}_t^i, \mathbf{U}_t^i, \mathbf{F}_t^i, \mathbf{W}_{t+1}) \\ & \mathbf{U}_t^i \preceq \mathcal{F}_t \ . \end{aligned}$$

Solving the production subproblems by DP

Assuming that

- the process $\mathbf{W}$ is a white noise,
- the process $\mathbf{Y}$ follows a dynamics $\mathbf{Y}_{t+1} = h_t(\mathbf{Y}_t, \mathbf{W}_{t+1})$,

Then $(\mathbf{X}_t, \mathbf{Y}_t)$ is a valid state to apply Dynamic Programming:

$$
\begin{aligned}
V_T^i(x, y) &= K^i(x) \\
V_t^i(x, y) &= \min_{u,f} \mathbb{E}\Big(L_t^i(x, u, f, \mathbf{W}_{t+1}) \\
&\qquad\qquad + \left\langle \mathbb{E}\left(\boldsymbol{\lambda}_t^{i,(k)} \;\middle|\; \mathbf{Y}_t = y\right), f \right\rangle + V_{t+1}^i(\mathbf{X}_{t+1}^i, \mathbf{Y}_{t+1}) \Big) \\
&\quad \text{s.t.} \quad \mathbf{X}_{t+1}^i = f_t^i(x, u, f, \mathbf{W}_{t+1}) \;, \\
&\qquad\qquad \mathbf{Y}_{t+1} = h_t(y, \mathbf{W}_{t+1}) \;.
\end{aligned}
$$

Where are we heading to?

- Solving directly the problem is not numerically tractable
- SDDP allows to solve the problem, but still has to deal with a noise $\mathbf{W}_t$ with size $\mathfrak{n}_{bin}^N$ ...
- Price decomposition allows to decompose the problem in N independent subproblems

Now, we aim to compare numerically SDDP and DADP.

Numerical implementation

Our stack is deeply rooted in Julia language

- Modeling Language: `JuMP`
- Open-source SDDP Solver: `StochDynamicProgramming.jl`
- LP/QP Solver: `Gurobi 7.02`

https://github.com/JuliaOpt/StochDynamicProgramming.jl

Implementation of SDDP and DADP

- Implementing SDDP is straightforward
  (but still a noise $\mathbf{W}_t$ with size $\mathfrak{n}_{bin}^N$...)

Implementation of SDDP and DADP

- Implementing SDDP is straightforward
(but still a noise $\mathbf{W}_t$ with size $\mathfrak{n}_{bin}^{N}$...)

- DADP is more elaborated. The difficulty lies in the update scheme:

$$\mathbb{E}\big(\boldsymbol{\lambda}^{(k+1)} \mid \mathbf{Y}\big) = \mathbb{E}\big(\boldsymbol{\lambda}^{(k)} \mid \mathbf{Y}\big) + \rho\, \mathbb{E}\big(A\mathbf{Q}^{(k+1)} + \mathbf{F}^{(k+1)} \mid \mathbf{Y}\big) .$$

We use a crude relaxation: $\mathbf{Y} = 0$. Denoting $\underline{\lambda}^{(k)} = \mathbb{E}\big(\boldsymbol{\lambda}^{(k)}\big)$,
the update becomes

$$\underline{\lambda}^{(k+1)} = \underline{\lambda}^{(k)} + \underbrace{\rho}_{\text{Update step}} \underbrace{\mathbb{E}\big(A\mathbf{Q}^{(k+1)} + \mathbf{F}^{(k+1)}\big)}_{\text{Monte Carlo}} .$$

Implementing gradient ascent

- Gradient ascent is too slow ...
- We try to implement accelerated gradient ascent[3] but ...
  - Unfortunately, we do not know the Lipschitz constant of the derivative!
  - The line-search kills the performance of gradient ascent...

[3] described in the seminal paper of Nesterov

Implementing gradient ascent

- Gradient ascent is too slow ...
- We try to implement accelerated gradient ascent[3] but ...
    - Unfortunately, we do not know the Lipschitz constant of the derivative!
    - The line-search kills the performance of gradient ascent...

To overcome this issue, we use Quasi-Newton (BFGS): the update becomes

$$\underline{\lambda}^{(k+1)} = \underline{\lambda}^{(k)} + \rho^{(k)} W^{(k)}\ \hat{\mathbb{E}}\left\{ A\mathbf{Q}^{(k+1)} + \mathbf{F}^{(k+1)} \right\} .$$

- We exploit the strong-convexity,
- The line-search is penalized by inexact gradient (especially near convergence where the algorithm requires precision)

[3] described in the seminal paper of Nesterov

Adding an augmented Lagrangian

Let first introduce the augmented Lagrangian corresponding to the relaxed problem:

$$\mathcal{L}(\mathbf{F},\mathbf{Q},\boldsymbol{\lambda}) = J_{\mathfrak{P}}(\mathbf{F}) + J_{\mathfrak{T}}(\mathbf{Q}) + \left\langle \boldsymbol{\lambda}, \mathbb{E}(A\mathbf{Q}+\mathbf{F}|\mathbf{Y})\right\rangle + \frac{\rho}{2}\left\|\mathbb{E}(A\mathbf{Q}+\mathbf{F}|\mathbf{Y})\right\|^2 .$$

If a saddle point exists, the problem is equivalent to:

$$\max_{\boldsymbol{\lambda}} \min_{\mathbf{F},\mathbf{Q}} \; \mathcal{L}(\mathbf{F},\mathbf{Q},\boldsymbol{\lambda}) .$$

Adding an augmented Lagrangian

Let first introduce the augmented Lagrangian corresponding to the relaxed problem:

$$\mathcal{L}(\mathbf{F},\mathbf{Q},\boldsymbol{\lambda}) = J_{\mathfrak{P}}(\mathbf{F}) + J_{\mathfrak{T}}(\mathbf{Q}) + \langle \boldsymbol{\lambda}, \mathbb{E}(A\mathbf{Q}+\mathbf{F}|\mathbf{Y})\rangle + \frac{\rho}{2}\left\|\mathbb{E}(A\mathbf{Q}+\mathbf{F}|\mathbf{Y})\right\|^2 .$$

If a saddle point exists, the problem is equivalent to:

$$\max_{\boldsymbol{\lambda}} \min_{\mathbf{F},\mathbf{Q}} \ \mathcal{L}(\mathbf{F},\mathbf{Q},\boldsymbol{\lambda}) .$$

ADMM solves iteratively the subproblems $J_{\mathfrak{P}}$ and $J_{\mathfrak{T}}$, and updates the multiplier $\boldsymbol{\lambda}$ with a constant step-size ρ:

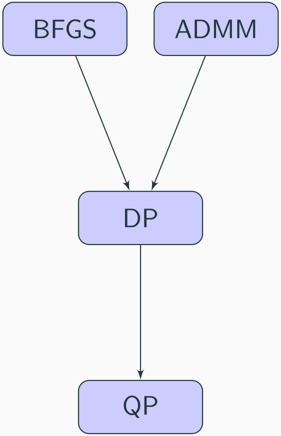
$$\mathbf{F}^{(k+1)} = \arg\min_{\mathbf{F}} J_{\mathfrak{P}}(\mathbf{F}) + \left\langle \boldsymbol{\lambda}^{(k)}, \mathbf{F}\right\rangle + \frac{\rho}{2}\left\|\mathbb{E}\left(A\mathbf{Q}^{(k)}\right) + \mathbf{F}\right\|^2$$

$$\mathbf{Q}^{(k+1)} = \arg\min_{\mathbf{Q}} J_{\mathfrak{T}}(\mathbf{Q}) + \left\langle \boldsymbol{\lambda}^{(k)}, A\mathbf{Q}\right\rangle + \frac{\rho}{2}\left\|A\mathbf{Q} + \mathbb{E}\left(\mathbf{F}^{(k+1)}\right)\right\|^2$$

$$\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \rho\mathbb{E}(A\mathbf{Q}^{(k+1)} + \mathbf{F}^{(k+1)}) .$$

Double, double toil and trouble

Digesting the stochastic caldron, between time and space ...



- Global problem $\mathcal{P}$

$$\min_{\mathbf{Q},\mathbf{F}} \quad J_{\mathfrak{P}}[\mathbf{F}] + J_{\mathfrak{T}}[\mathbf{Q}]$$
$$\text{s.t.} \quad A\mathbf{Q} + \mathbf{F} = 0 \ .$$

- Decomposed production subproblem $\mathcal{P}_i$

$$\min_{\mathbf{F}^i} J_{\mathfrak{P}}(\mathbf{F}^i) + \langle \lambda^{i,(k)} \ , \mathbf{F}^i \rangle$$

- DP subproblem V_t^i

$$V_t^i(x, y) = \min_{u,f} \mathbb{E}\Big(L_t^i(x, u, f, \mathbf{W}_{t+1}) + \langle \mathbb{E}(\boldsymbol{\lambda}_t^{i,(k)} \mid \mathbf{Y}_t = y) \ , f \rangle + V_{t+1}^i(\mathbf{X}_{t+1}^i, \mathbf{Y}_{t+1}) \Big)$$

SDDP convergence

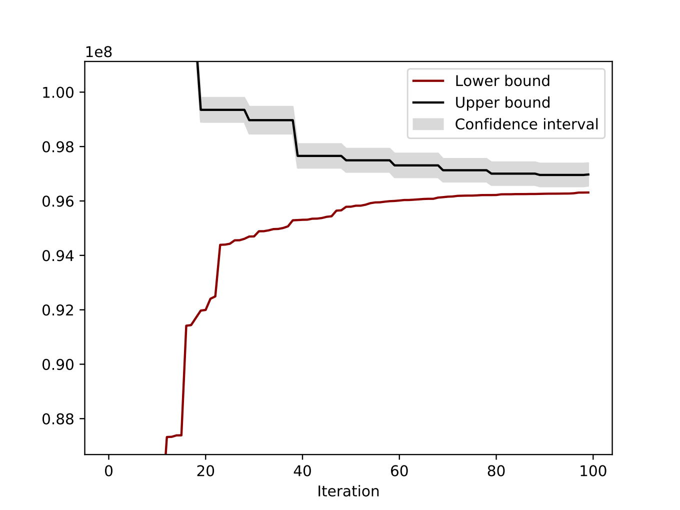


Figure 1: Convergence of SDDP's upper and lower bounds ($T = 52$, $\mathfrak{n}_{bin} = 2$).

Multipliers convergence

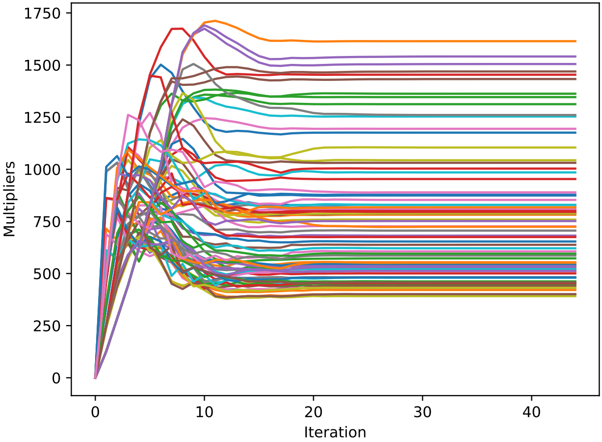


Figure 2: Convergence of multipliers with BFGS ($T = 52$, $\mathfrak{n}_{bin} = 2$).

ADMM convergence

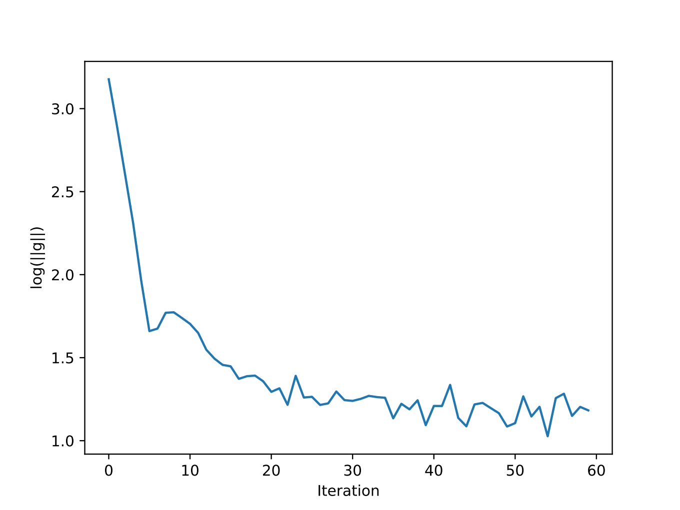


Figure 3: Convergence of ADMM, plotting the logarithm of the norm of the primal residual ($T = 52$, $\mathfrak{n}_{bin} = 2$).

Results — Weekly time steps

Compute Bellman value functions at weekly time steps ($T = 52$).

$\mathfrak{n}_{bin}$	1	2	5
SDDP value	9.396	9.687	$+\infty$
SDDP time	8''	928''	$+\infty$
BFGS value	9.411	9.687	9.974
BFGS time	69''	157''	575''
ADMM value	9.404	9.682	9.984
ADMM time	65''	326''	643''

- SDDP does not converge if $\mathfrak{n}_{bin} = 5$.

Results — Weekly time steps

Compute Bellman value functions at weekly time steps ($T = 52$).

$\mathfrak{n}_{bin}$	1	2	5
SDDP value	9.396	9.687	$+\infty$
SDDP time	8''	928''	$+\infty$
BFGS value	9.411	9.687	9.974
BFGS time	69''	157''	575''
ADMM value	9.404	9.682	9.984
ADMM time	65''	326''	643''

- SDDP does not converge if $\mathfrak{n}_{bin} = 5$.
- If $\mathfrak{n}_{bin} = 1$, results of SDDP, BFGS and ADMM are almost equivalent.

Results — Weekly time steps

Compute Bellman value functions at weekly time steps ($T = 52$).

$\mathfrak{n}_{bin}$	1	2	5
SDDP value	9.396	9.687	$+\infty$
SDDP time	8''	928''	$+\infty$
BFGS value	9.411	9.687	9.974
BFGS time	69''	157''	575''
ADMM value	9.404	9.682	9.984
ADMM time	65''	326''	643''

- SDDP does not converge if $\mathfrak{n}_{bin} = 5$.
- If $\mathfrak{n}_{bin} = 1$, results of SDDP, BFGS and ADMM are almost equivalent.
- BFGS and ADMM compute a gradient with Monte-Carlo ...

Results — Weekly time steps

Compute Bellman value functions at weekly time steps ($T = 52$).

$\mathfrak{n}_{bin}$	1	2	5
SDDP value	9.396	9.687	$+\infty$
SDDP time	8''	928''	$+\infty$
BFGS value	9.411	9.687	9.974
BFGS time	69''	157''	575''
ADMM value	9.404	9.682	9.984
ADMM time	65''	326''	643''

- SDDP does not converge if $\mathfrak{n}_{bin} = 5$.
- If $\mathfrak{n}_{bin} = 1$, results of SDDP, BFGS and ADMM are almost equivalent.
- BFGS and ADMM compute a gradient with Monte-Carlo ...
- Here, BFGS is penalized by line-search, and stops earlier if no search direction is found.

Conclusion

Conclusion

Conclusion

- A survey of different algorithms, mixing spatial and time decomposition.
- DADP works well with the crude relaxation $\mathbf{Y} = 0$.
- SDDP does not converge in a finite time if $\mathfrak{n}_{bin} = 5$.
- We had a lot of troubles to deal with approximate gradients!

Perspectives

- Find a proper information process $\mathbf{Y}$.
- Improve the integration between SDDP and DADP.
- Test other decomposition schemes (by quantity, by prediction).

P. Girardeau.

Résolution de grands problèmes en optimisation stochastique dynamique.

Thèse de doctorat, Université Paris-Est, 2010.

V. Leclère.

Contributions aux méthodes de décomposition en optimisation stochastique.

Thèse de doctorat, Université Paris-Est, 2014.

A. Lenoir and P. Mahey.

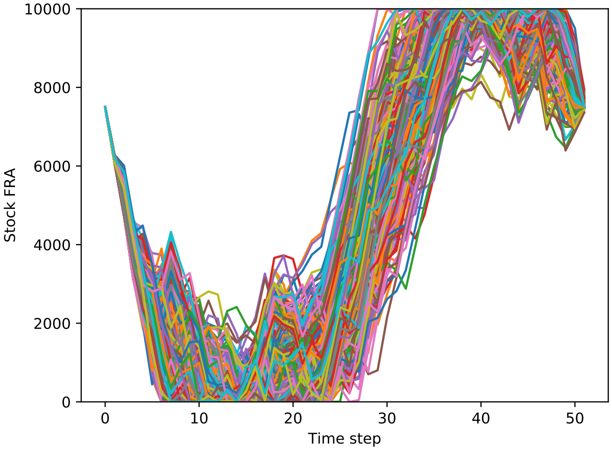
A survey of monotone operator splitting methods and decomposition of convex programs.

RAIRO Operations Research 51, 17-41, 2017.

Philippe Mahey, Jonas Koko, Arnaud Lenoir and Luc Marchand.

Coupling decomposition with dynamic programming for a stochastic spatial model for long-term energy pricing problem.

Dams trajectory



SGD convergence

Plotting the convergence with $T = 52$ and $\mathfrak{n}_{bin} = 2$.