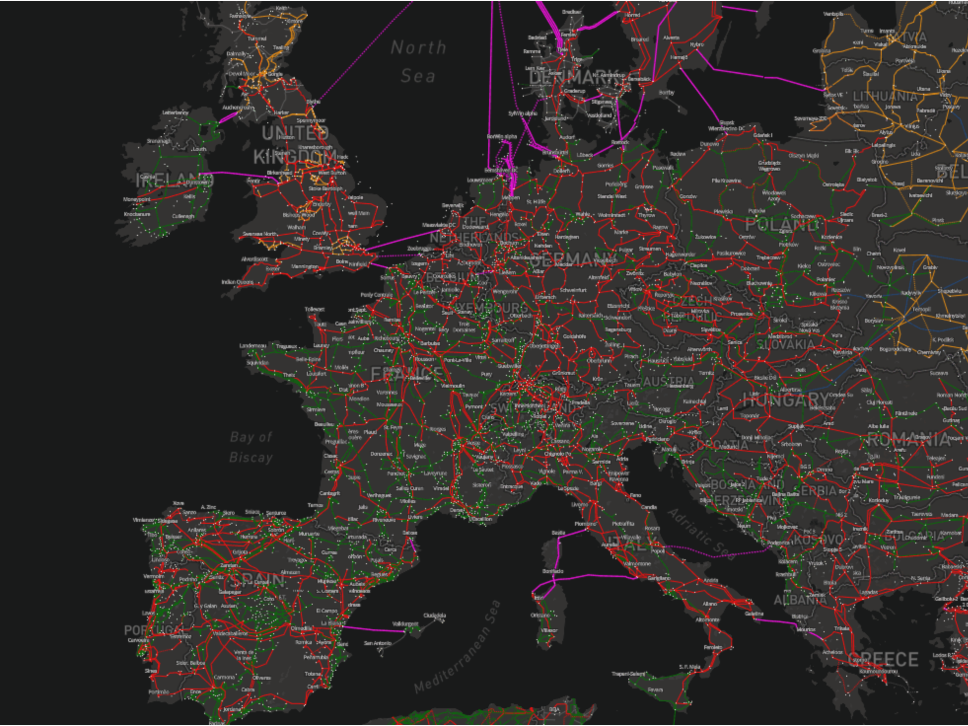
Optimization of Energy Production and Transport

A stochastic decomposition approach

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

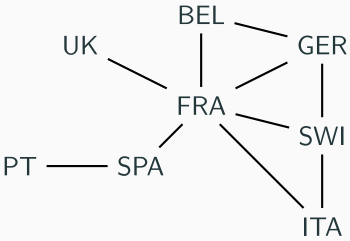
GdR MOA — October 18, 2017

ENSTA ParisTech — ENPC ParisTech — EDF Lab — Efficacity


North Sea
UNITED KINGDOM
IRELAND
DENMARK
LITHUANIA
THE NETHERLANDS
GERMANY
POLAND
LUXEMBOURG
CZECH REPUBLIC
SLOVAKIA
FRANCE
AUSTRIA
HUNGARY
ROMANIA
CROATIA
Bay of Biscay
SERBIA
Adriatic Sea
KOSOVO
ITALY
SPAIN
ALBANIA
PORTUGAL
Mediterranean Sea
GREECE

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 or monthly time steps).
- Large-scale problem (8 countries).

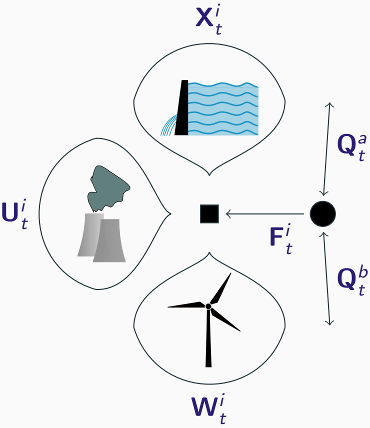
[1]But the framework remains valid for smaller energy management problems.

Lecture outline

Modelling

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 dynamic $x_{t+1}^i = f_t^i(x_t^i, u_t^i, w_t^i)$ 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:

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

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]

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

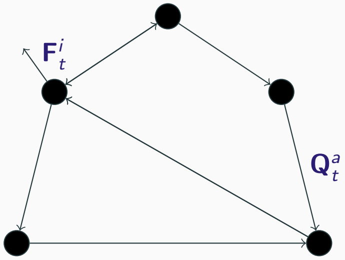
$$\begin{aligned} \text{s.t.} \quad & \mathbf{X}^i_{t+1} = f^i_t(\mathbf{X}^i_t, \mathbf{U}^i_t, \mathbf{F}^i_t, \mathbf{W}^i_{t+1}) , \\ & \mathbf{X}^i_t \in \mathcal{X}^{i,\mathrm{ad}}_t , \quad \mathbf{U}^i_t \in \mathcal{U}^{i,\mathrm{ad}}_t , \\ & \mathbf{U}^i_t \preceq \mathcal{F}_t , \end{aligned}$$

The last equation is the measurability constraint, where $\mathcal{F}_t$ is the σ-field generated by the noises $\{\mathbf{W}^i_\tau\}_{\tau=1\dots t}$ 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:

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

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

Resolution methods

Where are we heading to?

The problem P has:

- N nodes (with $N = 8$);
- T time steps (with $T = 12$ or $T = 52$);
- N independent random variables per time step t: $\mathbf{W}_t^1, \cdots, \mathbf{W}_t^N$.

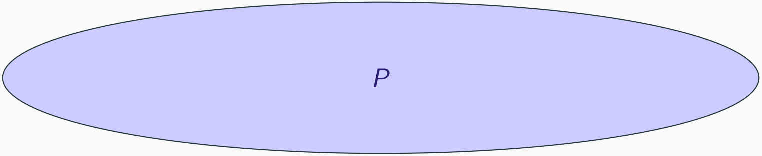
We aim to solve the problem numerically. We suppose that for all t, $\mathbf{W}_t^i$ is a discrete random variable, with support size $\mathfrak{n}_{bin}$. Thus, the random variable

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

has a support size $\mathfrak{n}_{bin}^N$ (because of the independence).

First idea: solving the whole problem inplace!

Write the problem and solve it!



But ...

- N nodes and T time steps.
- Non-anticipativity constraint: we ought to formulate the problem on a tree (Stochastic Programming approach)

$$\text{number of nodes} = (\mathfrak{n}_{bin}^{N})^{T} = \mathfrak{n}_{bin}^{NT} ,$$

giving a complexity in $\mathcal{O}(\mathfrak{n}_{bin}^{NT})$.

The problem is not tractable ...

Second idea: decomposition with Dynamic Programming

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

P :

The complexity reduces to $\mathcal{O}(T\mathfrak{n}_{bin}^N)$. We use Dynamic Programming to compute the value functions $V_1, \cdots, V_T$.

But ...

- N nodes: curse of dimensionality
- 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

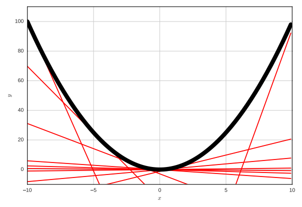
Dynamic Programming

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



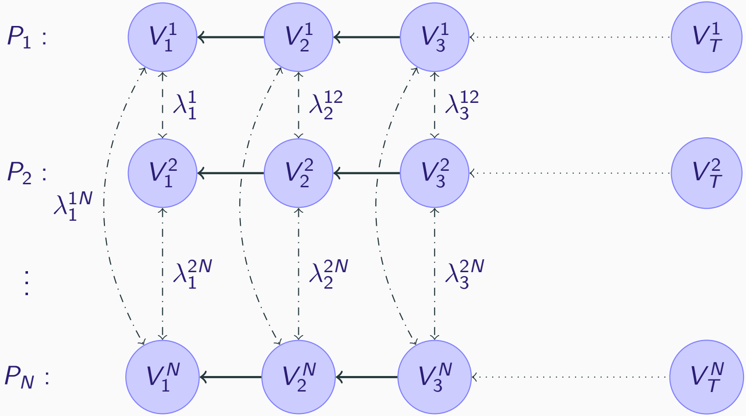
- Convex value functions V_t are approximated as a supremum of a finite set of affine functions
- Affine functions (=cuts) are computed during forward/backward passes, till convergence

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

- SDDP makes an extensive use of LP/QP solver

Third idea: spatial decomposition

We decompose the problem time by time and node by node to obtain $N \times T$ decomposed subproblems:



The complexity reduces to $\mathcal{O}(NT\mathfrak{n}_{bin})$! But ...

How to compute the different λ?

Introducing decomposition methods

The decomposition/coordination methods we want to deal with are iterative algorithms involving the following ingredients.

- Decompose the global problem in several subproblems of smaller size by dualizing the constraint $A\mathbf{Q} + \mathbf{F} = 0$,
- Coordinate at each iteration the subproblems using the price λ.

$$A\mathbf{Q} + \underbrace{\mathbf{F}}_{\textit{allocation}} = 0 \quad \rightsquigarrow \underbrace{\boldsymbol{\lambda}}_{\textit{price}}$$

- Solve the subproblems using Dynamic Programming, taking into account the price transmitted by the coordination.

Approximating the subproblems

In both cases, the subproblems encompass a new "noise", that is, the price multiplier $\boldsymbol{\lambda}_t^{(k)}$, which may be correlated in time.
The white noise assumption fails.

Dynamic Programming cannot be used for solving the subproblems.

Approximating the subproblems

In both cases, the subproblems encompass a new “noise”, that is, the price multiplier $\boldsymbol{\lambda}_t^{(k)}$, which may be correlated in time.
The white noise assumption fails.

Dynamic Programming cannot be used for solving the subproblems.

In order to overcome this difficulty, we use a trick that involves approximating the new noise $\boldsymbol{\lambda}_t^k$ by its conditional expectation w.r.t. a chosen random variable $\mathbf{Y}_t$.

Approximating the subproblems

In both cases, the subproblems encompass a new "noise", that is, the price multiplier $\boldsymbol{\lambda}_t^{(k)}$, which may be correlated in time.
The white noise assumption fails.

Dynamic Programming cannot be used for solving the subproblems.

In order to overcome this difficulty, we use a trick that involves approximating the new noise $\boldsymbol{\lambda}_t^k$ by its conditional expectation w.r.t. a chosen random variable $\mathbf{Y}_t$.

Assume that the process $\mathbf{Y}$ has a given dynamics:

$$\mathbf{Y}_{t+1} = h_t(\mathbf{Y}_t, \mathbf{W}_{t+1}) \ .$$

If noises $\mathbf{W}_t$'s are time independent, then $(\mathbf{X}_t^i, \mathbf{Y}_t)$ is a valid state for the i-th subproblem and Dynamic Programming applies.

Price decomposition

The production and transport optimization problem writes

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

The decomposition scheme consists in dualizing the constraint, and then in approximating the multiplier $\boldsymbol{\lambda}$ by its conditional expectation w.r.t. $\mathbf{Y}$. This trick leads to the following problem

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

It is not difficult to prove that this dual problem is associated to the following relaxed primal problem:

$$\min_{\mathbf{Q},\mathbf{F}} J_{\mathfrak{P}}[\mathbf{F}] + J_{\mathfrak{T}}[\mathbf{Q}] \qquad \text{s.t.} \quad \mathbb{E}\big(A\mathbf{Q} + \mathbf{F} \;\big|\; \mathbf{Y}\big) = 0 \,,$$

and hence provides a lower bound of $(\mathcal{P})$.

A dual gradient-like algorithm

Applying the Uzawa algorithm to the dual problem

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

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 transport problem

The transport subproblem

$$\min_{\mathbf{Q}} \; J_{\mathfrak{T}}[\mathbf{Q}] + \left\langle \mathbb{E}\left(\boldsymbol{\lambda}^{(k)} \;\middle|\; \mathbf{Y}\right), A\mathbf{Q} \right\rangle ,$$

writes in a detailled manner

$$\min_{\mathbf{Q}} \sum_{t=0}^{T-1} \mathbb{E}\Big(\sum_{a \in \mathcal{A}} c_t^a(\mathbf{Q}_t^a) + \left\langle A^\top \mathbb{E}\left(\boldsymbol{\lambda}_t^{(k)} \;\middle|\; \mathbf{Y}_t\right), \mathbf{Q}_t \right\rangle \Big) .$$

This minimization subproblem is evidently decomposable in time (t by t) and in space (arc by arc), leading to a collection of easy to solve subproblems.

Decomposing the production problem

The production subproblem

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

evidently decomposes node by node

$$\min_{\mathbf{F}^i} J^i_{\mathfrak{P}}[\mathbf{F}^i] + \left\langle \mathbb{E}\left(\boldsymbol{\lambda}^{i,(k)} \;\middle|\; \mathbf{Y}\right), \mathbf{F}^i \right\rangle ,$$

hence a stochastic optimal control subproblem for each node i:

$$\begin{aligned}
\min_{\mathbf{X}^i,\mathbf{U}^i,\mathbf{F}^i} \; & \mathbb{E}\left(\sum_{t=0}^{T-1} \left(L^i_t(\mathbf{X}^i_t, \mathbf{U}^i_t, \mathbf{F}^i_t, \mathbf{W}_{t+1}) + \left\langle \mathbb{E}\left(\boldsymbol{\lambda}^{i,(k)}_t \;\middle|\; \mathbf{Y}_t\right), \mathbf{F}^i_t \right\rangle \right) + K^i(\mathbf{X}^i_T) \right) \\
\text{s.t.} \; & \mathbf{X}^i_{t+1} = f^i_t(\mathbf{X}^i_t, \mathbf{U}^i_t, \mathbf{F}^i_t, \mathbf{W}_{t+1}) \\
& \mathbf{U}^i_t \preceq \mathcal{F}_t \; .
\end{aligned}$$

Solving the production subproblems by DP

Assuming that

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

Dynamic Programming applies for production subproblems:

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

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

- DADP implementation is more elaborated:

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

  We use a crude relaxation:

  - We choose $\mathbf{Y} = 0$. We denote $\underline{\lambda}^{(k)} = \mathbb{E}\left(\boldsymbol{\lambda}^{(k)}\right)$. The update becomes

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

  - Unfortunately, we do not know the Lipschitz constant of the derivative!
  - And the problem is not even strongly convex ...

We compare three algorithms for gradient ascent

- Quasi-Newton (BFGS): To ensure strong convexity, we add a quadratic term to the cost: $\hat{L}_t^i(.) = L_t^i(.) + u^\top Qu$, with $Q \succ 0$. The update is:

$$\underline{\lambda}^{(k+1)} = \underline{\lambda}^{(k)} + \rho^{(k)} \, \hat{\mathbb{E}}\big\{A\mathbf{Q}^{(k+1)} + \mathbf{F}^{(k+1)}\big\} \; .$$

- Alternating Direction Method of Multipliers (ADMM): we add an augmented Lagrangian to solve the problem. The update becomes

$$\underline{\lambda}^{(k+1)} = \underline{\lambda}^{(k)} + \frac{\tau}{2} \, \hat{\mathbb{E}}\big(A\mathbf{Q}^{(k+1)} + \mathbf{F}^{(k+1)}\big) \; .$$

- Stochastic Gradient Descent (SGD):

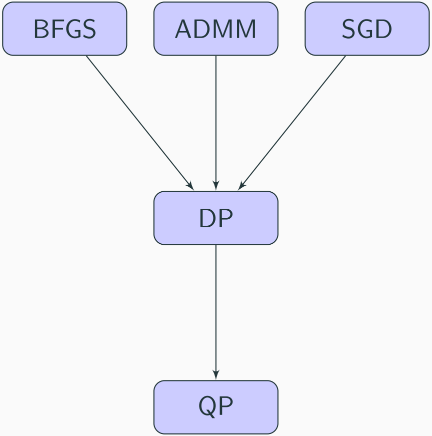
$$\underline{\lambda}^{(k+1)} = \underline{\lambda}^{(k)} + \frac{1}{1+k} \, \big(A\mathbf{Q}^{(k+1)} + \mathbf{F}^{(k+1)}\big)(\omega) \; .$$

	BFGS	ADMM	SGD
ρ	line search	$\rho^{(k)} \to \tau$	$1/(1+k)$
MC size	100-1000	100-1000	1
software	L-BFGS-B[3]	self	self

[3] The famous implementation of [Zhu et al, 1997]

Double, double toil and trouble

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



- Global problem 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 subproblem P_i

$$J_{\mathfrak{P}}(\mathbf{F}^i) = \min_{\mathbf{X}^i,\mathbf{U}^i,\mathbf{F}^i} \mathbb{E}\Big(\sum_{t=0}^{T-1} \big(L_t^i(\mathbf{X}_t^i, \mathbf{U}_t^i, \mathbf{F}_t^i, \mathbf{W}_{t+1}) + \langle \mathbb{E}(\boldsymbol{\lambda}_t^{i,(k)} \mid \mathbf{Y}_t), \mathbf{F}_t^i \rangle \big) + K^i(\mathbf{X}_T^i) \Big)$$
$$\text{s.t.} \quad \mathbf{X}_{t+1}^i = f_t^i(\mathbf{X}_t^i, \mathbf{U}_t^i, \mathbf{F}_t^i, \mathbf{W}_{t+1})$$

- DP subproblem V_t^i

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

Results — Monthly

Compute Bellman value functions at monthly time steps ($T = 12$).

$\mathfrak{n}_{bin}$	1	2	5
SDDP value	5.048	5.203	$+\infty$
SDDP time	0.5"	87"	$+\infty$
BFGS value	5.088	5.202	5.286
BFGS time	18"	49"	161"
ADMM value	5.087	5.201	5.288
ADMM time	14"	49"	66"
SGD value	5.088	5.202	5.292
SGD time	37"	66"	130"

- SDDP does not converge if $\mathfrak{n}_{bin} = 5$.
- If $\mathfrak{n}_{bin} = 1$, SDDP is better than DADP because of the discretization scheme used in Dynamic Programming.
- BFGS and ADMM compute a gradient with Monte-Carlo ...
- BFGS does not solve the original problem (strong convexification)

Results — Weekly

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''
SGD value	9.411	9.679	9.971
SGD time	194''	281''	712''

- The longer the horizon, the slower SDDP is.
- Here, BFGS is penalized by line-search, as it uses an approximated gradient
- SGD works quite well compared to BFGS and ADMM: these two algorithms are penalized by the Monte-Carlo computation of the gradient.

Multipliers convergence

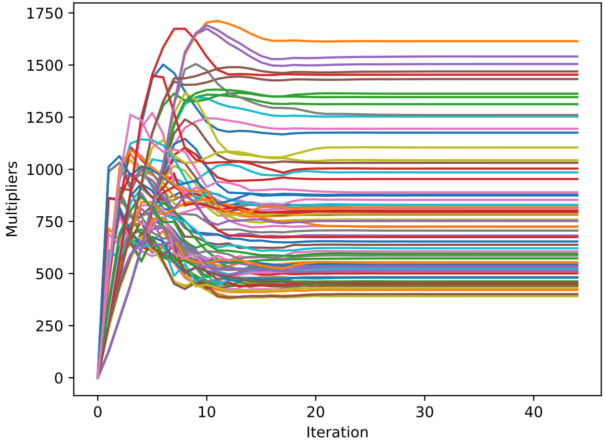


Figure 1: Convergence of multipliers with BFGS ($T = 12$, $\mathfrak{n}_{bin} = 1$).

SDDP convergence

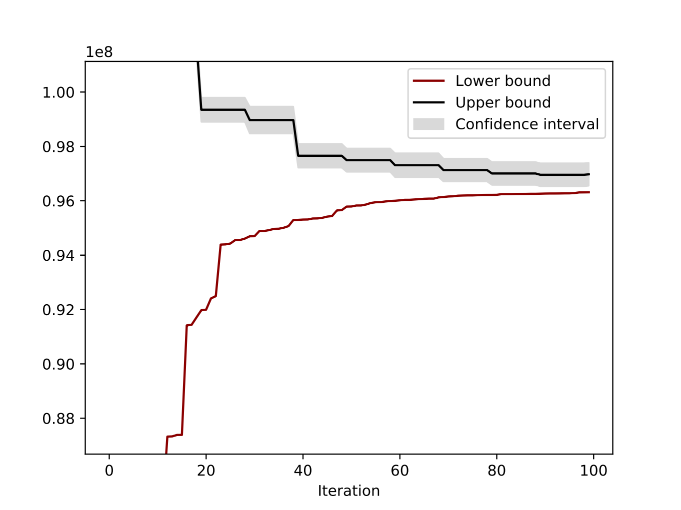


Figure 2: Convergence of SDDP's upper and lower bounds ($T = 52$, $\mathfrak{n}_{bin} = 2$).

Conclusion

Conclusion

Conclusion

- A survey of different algorithms, mixing spatial and time decomposition.
- DADP works well with the crude relaxation $\mathbf{Y} = 0$, and even beats SDDP if $\mathfrak{n}_{bin} \geq 2$.
- 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. Carpentier et G. Cohen.
Décomposition-coordination en optimisation déterministe et stochastique.
En préparation, Springer, 2016.

P. Girardeau.
Résolution de grands problèmes en optimisation stochastique dynamique.
Thèse de doctorat, Université Paris-Est, 2010.

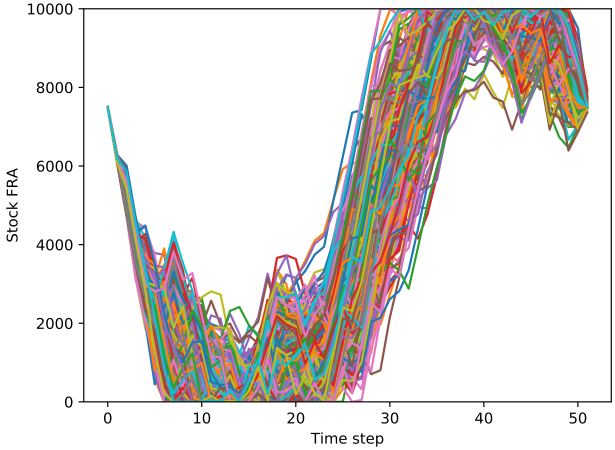
V. Leclère.
Contributions aux méthodes de décomposition en optimisation stochastique.
Thèse de doctorat, Université Paris-Est, 2014.

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.

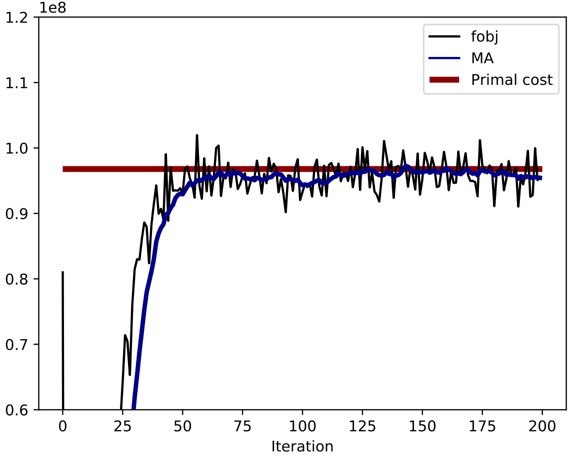
Zhu, Ciyou and Byrd, Richard H and Lu, Peihuang and Nocedal, Jorge.
L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization .
ACM Transactions on Mathematical Software (TOMS), 23-4, 1997.

Dams trajectory



SGD convergence

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



ADMM convergence

Plotting the logarithm of the norm of the primal residual with $T = 52$ and $\mathfrak{n}_{bin} = 5$.