Optimal energy management and stochastic decomposition

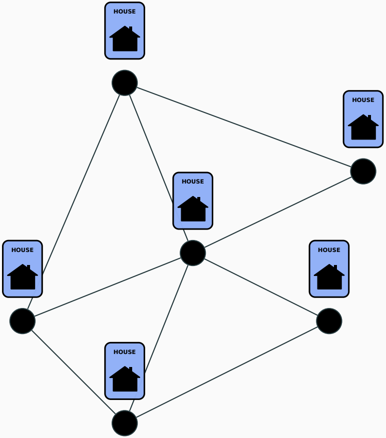
F. Pacaud — P. Carpentier — J.P. Chancelier — M. De Lara

JuMP-dev workshop, 2018

ENPC ParisTech — ENSTA ParisTech — Efficacity

Motivation

We consider a *peer-to-peer* community,
where different buildings exchange energy



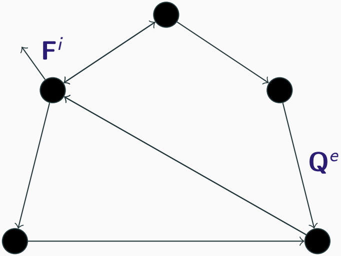
Lecture outline

- We will formulate a large scale (stochastic) optimization problem
- We will apply decomposition algorithm on it
- We will put emphasis on the numerical side
  (built on top of JuMP!)

Nodal decomposition of a network optimization problem

Modeling flows between nodes

Graph $G = (\mathcal{V}, \mathcal{E})$



- $\mathbf{Q}_t^e$ flow through edge e,
- $\mathbf{F}_t^i$ flow imported at node i

Let A be the *node-edge* incidence matrix

At each time $t \in [\![0, T-1]\!]$, Kirchhoff current law couples nodal and edge flows

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

Writing down the nodal problem

We aim at minimizing the nodal costs over the nodes $i \in \mathcal{V}$

$$J^i_{\mathcal{V}}(\mathbf{F}^i) = \min_{\mathbf{X}^i, \mathbf{U}^i} \mathbb{E}\Big[\sum_{t=0}^{T-1} \underbrace{L^i_t(\mathbf{X}^i_t, \mathbf{U}^i_t, \mathbf{W}_{t+1})}_{\text{instantaneous cost}} + K^i(\mathbf{X}^i_T) \Big]$$

subject to, for all $t \in [\![0, T-1]\!]$

i) The nodal dynamics constraint (for battery and hot water tank)

$$\mathbf{X}^i_{t+1} = g^i_t(\mathbf{X}^i_t, \mathbf{U}^i_t, \mathbf{W}_{t+1})$$

ii) The non-anticipativity constraint (future remains unknown)

$$\sigma(\mathbf{U}^i_t) \subset \sigma(\mathbf{W}_0, \cdots, \mathbf{W}_t)$$

iii) The load balance equation (production + import = demand)

$$\Delta^i_t(\mathbf{X}^i_t, \mathbf{U}^i_t, \mathbf{F}^i_t, \mathbf{W}_{t+1}) = 0$$

Transportation costs are decoupled in time

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

$$J_{\mathcal{E}}^e(\mathbf{Q}) = \mathbb{E}\Big(\sum_{t=0}^{T-1} l_t^e(\mathbf{Q}_t^e)\Big)$$

Global optimization problem

The *nodal cost* $J_{\mathcal{V}}$ aggregates the costs at all nodes i

$$J_{\mathcal{V}}(\mathbf{F}) = \sum_{i \in \mathcal{V}} J_{\mathcal{V}}^i(\mathbf{F}^i)$$

and the *edge cost* $J_{\mathcal{E}}$ aggregates the edges costs at all time t

$$J_{\mathcal{E}}(\mathbf{Q}) = \sum_{e \in \mathcal{E}} J_{\mathcal{E}}^e(\mathbf{Q}^e)$$

The global optimization problem writes

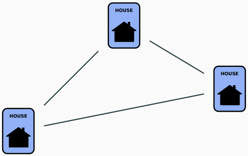
$$V^\sharp = \min_{\mathbf{F},\mathbf{Q}} J_{\mathcal{V}}(\mathbf{F}) + J_{\mathcal{E}}(\mathbf{Q})$$
$$\text{s.t. } A\mathbf{Q} + \mathbf{F} = 0$$

What do we plan to do?

- We have formulated a multistage stochastic optimization problem on a graph
- We will handle the coupling Kirchhoff constraints by two decomposition methods
  - Price decomposition
  - Resource decomposition
- We will show the scalability of decomposition algorithms (We solve problems up to 48 buildings)

Resolution methods

Price decomposition decomposes the global problem with a *price process* $\boldsymbol{\lambda}$



Three levels of hierarchy

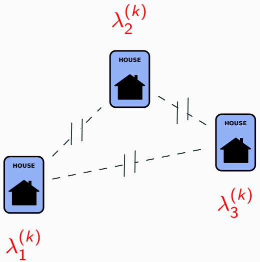
1. The *central planner* fixes a price $\boldsymbol{\lambda}$ so as to optimize global cost
2. The *nodal managers* manage buildings to decrease local costs
3. *Nodal value functions* are computed locally, time steps by time steps

The central planner has to find optimal coordination process

- The central planner aims to find the optimal price process $\boldsymbol{\lambda}$

$$\max_{\boldsymbol{\lambda}} \underline{V}(\boldsymbol{\lambda}) := \min_{\mathbf{F},\mathbf{Q}} J_P(\mathbf{F}) + J_T(\mathbf{Q}) + \langle \boldsymbol{\lambda} , A\mathbf{Q} + \mathbf{F} \rangle$$

- Let $\boldsymbol{\lambda}^{(k)}$ be a given price
  The global function $\underline{V}(\boldsymbol{\lambda}^{(k)})$ decomposes w.r.t. nodes and arcs



$$\min_{\mathbf{F}} J_P(\mathbf{F}) + \langle \boldsymbol{\lambda}^{(k)} , \mathbf{F} \rangle = \min_{\mathbf{F}^1, \cdots, \mathbf{F}^N} \sum_{i=1}^{N} J_P^i(\mathbf{F}^i) + \langle \boldsymbol{\lambda}^i , \mathbf{F}^i \rangle$$

$$= \sum_{i=1}^{N} \underbrace{\min_{\mathbf{F}^i} \left\{ J_P^i(\mathbf{F}^i) + \langle \boldsymbol{\lambda}^i , \mathbf{F}^i \rangle \right\}}_{\text{local problem}}$$

- Once subproblems solved by each *nodal managers*,
  she updates the price with the oracle $\nabla \underline{V}(\boldsymbol{\lambda}^{(k)})$

$$\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \rho \nabla \underline{V}(\boldsymbol{\lambda}^{(k)})$$

Managing buildings in each node

At each building $i \in [\![1, N]\!]$, the nodal manager

- Receives a price $\boldsymbol{\lambda}^i$ from the central planner and build the nodal problem

$$\underline{V}^i(\boldsymbol{\lambda}^i) = \min_{\mathbf{F}^i} J_P^i(\mathbf{F}^i) + \left\langle \boldsymbol{\lambda}^i \, , \mathbf{F}^i \right\rangle$$

  which rewrites as a Stochastic Optimal Control problem

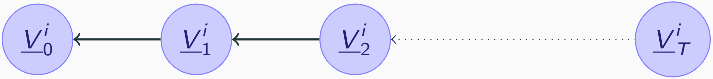
$$\begin{aligned} \underline{V}^i(\boldsymbol{\lambda}^i) = \min_{\mathbf{X}^i, \mathbf{U}^i, \mathbf{F}^i} \; & \mathbb{E}\Big[\sum_{t=0}^{T-1} L_t^i(\mathbf{X}_t^i, \mathbf{U}_t^i, \mathbf{W}_{t+1}^i) + \left\langle \boldsymbol{\lambda}_t^i \, , \mathbf{F}_t^i \right\rangle + K^i(\mathbf{X}_T^i) \Big] \\ \text{s.t. } & \mathbf{X}_{t+1}^i = f_t^i(\mathbf{X}_t^i, \mathbf{U}_t^i, \mathbf{W}_{t+1}^i) \\ & \sigma(\mathbf{U}_t^i) \subset \sigma(\mathbf{W}_0^i, \cdots, \mathbf{W}_t^i) \\ & \Delta_t^i(\mathbf{X}_t^i, \mathbf{U}_t^i, \mathbf{F}_t^i, \mathbf{W}_{t+1}) = 0 \end{aligned}$$

- Solves $\underline{V}^i$ by Dynamic Programming

- Estimates by Monte Carlo the local gradient
  by simulating the optimal flow $(\mathbf{F}^i)^\sharp = (\mathbf{F}_0^i, \cdots, \mathbf{F}_{T-1}^i)^\sharp$

$$\nabla \underline{V}^i(\boldsymbol{\lambda}^i) = \mathbb{E}\big[(\mathbf{F}^i)^\sharp\big] \in \mathbb{R}^T$$

Nodal value functions compute by Dynamic Programming



If the price process $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_0, \cdots, \boldsymbol{\lambda}_{T-1})$ is Markovian, then

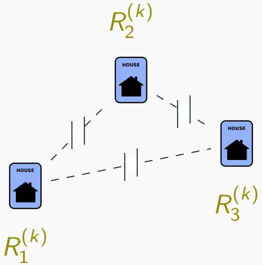
- We are able to compute value functions $\{\underline{V}_t^i\}$ by backward recursion
- At each time step, we solve the local one-step DP problem

$$\underline{V}_t^i(x_t^i) = \min_{u_t^i, f_t^i} \sum_{s=1}^{|\mathbb{W}_{t+1}^i|} p_s \Big(L_t(x_t^i, u_t^{i,s}, \mathbf{W}_{t+1}^{i,s}) + \langle \lambda_t^{i,s}, f_t^{i,s} \rangle + \underline{V}_{t+1}^i\big(f_t^i(x_t^i, u_t^{i,s}, \mathbf{W}_{t+1}^{i,s})\big) \Big)$$

  that decomposes on all atoms

- DP one-step problem formulates as LP or QP problem!

How about resource allocation?



- We fix allocations $\mathbf{R}$ rather than prices $\boldsymbol{\lambda}$ and solve

$$\min_{\mathbf{R}} \overline{V}(\mathbf{R}) := \overline{V}_P(\mathbf{R}) + \overline{V}_T(\mathbf{R})$$

with

$$\overline{V}_P(\mathbf{R}) = \min_{\mathbf{F}} \; J_P(\mathbf{F}) \quad \text{s.t. } \mathbf{F} - \mathbf{R} = 0 \qquad\qquad \overline{V}_T(\mathbf{R}) = \min_{\mathbf{Q}} \; J_T(\mathbf{Q}) \quad \text{s.t. } A\mathbf{Q} + \mathbf{R} = 0$$

- We must ensure that $\mathbf{R}_t \in im(A)$, that is

$$\mathbf{R}_t^1 + \cdots + \mathbf{R}_t^N = 0$$

- The update step becomes

$$\mathbf{R}^{(k+1)} = \mathrm{proj}_{im(A)}\left(\mathbf{R}^{(k)} - \rho \nabla \overline{V}(\mathbf{R}^{(k)})\right)$$

We obtain lower and upper bounds

Theorem

- *For all multipliers* $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_0, \cdots, \boldsymbol{\lambda}_{T-1})$
- *For all allocations* $\mathbf{R} = (\mathbf{R}_0, \cdots, \mathbf{R}_{T-1})$ *such that*

$$\mathbf{R}_t^1 + \cdots + \mathbf{R}_t^N = 0$$

we have

$$\underline{V}(\boldsymbol{\lambda}) \leq V^\sharp \leq \overline{V}(\mathbf{R})$$

Proof.

Next thursday!

Deducing two admissible global control policies

Once value functions $\underline{V}_t^i$ and $\overline{V}_t^i$ computed, we define

- the global price policy

$$\underline{\pi}_t(x_t^1, \cdots, x_t^N, w_{t+1}) \in \operatorname*{arg\,min}_{u_t, f_t, q_t} \sum_{i=1}^{N} L_t^i(x_t^i, u_t^i, w_{t+1}) + \underline{V}_{t+1}^i(x_{t+1}^i)$$

$$\text{s.t. } x_{t+1}^i = g_t^i(x_t^i, u_t^i, w_{t+1}) \ , \ \ \forall i \in [\![1, N]\!]$$

$$\Delta_t^i(x_t^i, u_t^i, f_t^i, w_{t+1}^i) \ , \ \ \forall i \in [\![1, N]\!]$$

$$Aq_t + f_t = 0$$

- the global resource policy

$$\overline{\pi}_t(x_t^1, \cdots, x_t^N, w_{t+1}) \in \operatorname*{arg\,min}_{u_t, f_t, q_t} \mathbb{E}\Big[\sum_{i=1}^{N} L_t^i(x_t^i, u_t^i, w_{t+1}) + \overline{V}_{t+1}^i(x_{t+1}^i)\Big]$$

$$\text{s.t. } x_{t+1}^i = g_t^i(x_t^i, u_t^i, w_{t+1}) \ , \ \ \forall i \in [\![1, N]\!]$$

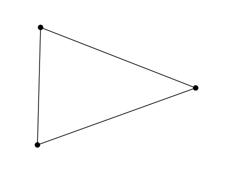
$$\Delta_t^i(x_t^i, u_t^i, f_t^i, w_{t+1}^i) \ , \ \ \forall i \in [\![1, N]\!]$$

$$Aq_t + f_t = 0$$

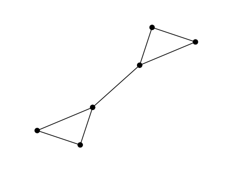
Numerical results on urban microgrids

We consider different urban configurations

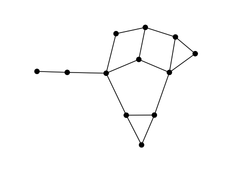
3-Nodes

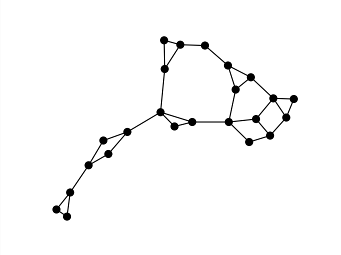


6-Nodes

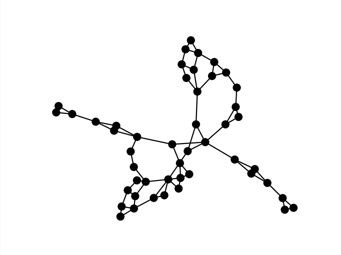


12-Nodes





24-Nodes



48-Nodes

Problem settings

- One day horizon at 15mn time step: $T = 96$
- Weather corresponds to a sunny day in Paris *(June 28th, 2015)*
- We mix three kind of buildings
  1. Battery + Electrical Hot Water Tank
  2. Solar Panel + Electrical Hot Water Tank
  3. Electrical Hot Water Tank

  and suppose that all consumers are commoners sharing their devices

Algorithms inventory

Nodal decomposition

- Encompass price and resource decompositions
- Resolution by Quasi-Newton (BFGS) gradient descent

$$\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \rho^{(k)} W^{(k)} \nabla \underline{V}(\boldsymbol{\lambda}^{(k)})$$

- BFGS iterates till no descent direction is found
- Each nodal subproblem solved by SDDP (quickly converge)
- Oracle $\nabla \underline{V}(\boldsymbol{\lambda})$ estimated by Monte Carlo ($N^{scen} = 1,000$)

SDDP

We use as a reference the good old SDDP algorithm

- Noises $\mathbf{W}_t^1, \cdots, \mathbf{W}_t^N$ are independent node by node
(total support size is $|supp(\mathbf{W}_t^i)|^N$.) Need to resample the support!
- Level-one cut selection algorithm (keep 100 most relevant cuts)
- Converged once gap between UB and LB is lower than 1%

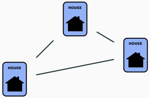
Building problems on the fly

We use metaprogramming to build `AbstractStochasticProgram` on the fly

Build node problem dynamically:

```
In [ ]:
exphouse = Expr[]
for dev in house.devices
    # parse device's dynamics as Expr
    dyn = parsedevice(dev, xindex, uindex, house.time.δt, params)
    push!(exphouse, dyn...)
    xindex += nstates(dev)
    uindex += ncontrols(dev)
end

eval(:((t, x, u, w) -> $exphouse))
```

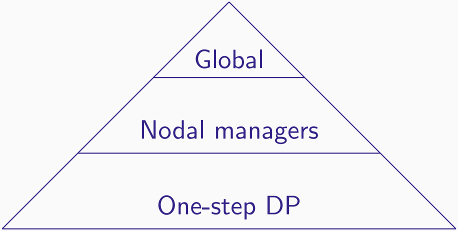


Then build global problem dynamically:

```
In [ ]:
expgrid = Expr(:vect)
for node in pb.nodes
    # parse dynamics of nodes
    ex = parsebuilding(node, xindex, uindex, node.time.δt, params)
    push!(expgrid.args, ex...)
    xindex += nstocks(node)
    uindex += ncontrols(node)
end

eval(:((t, x, u, w) -> $expgrid))
```

Each level of hierarchy has its own algorithm



L-BFGS (IPOPT)

SDDP (StochDynamicProgramming)

QP (Gurobi)

All glue code is implemented in Julia 0.6 with JuMP 0.18

Special thanks to all JuliaOpt folks!

Fortunately, everything converge nicely!

Illustrating convergence for **12-Nodes** problem

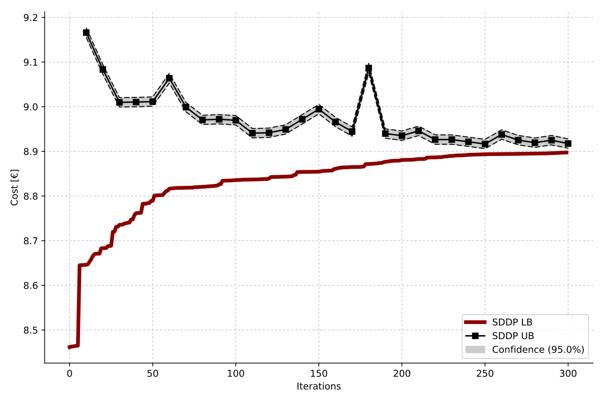


Figure 1: SDDP convergence, upper and lower bounds

Fortunately, everything converge nicely!

Illustrating convergence for **12-Nodes** problem

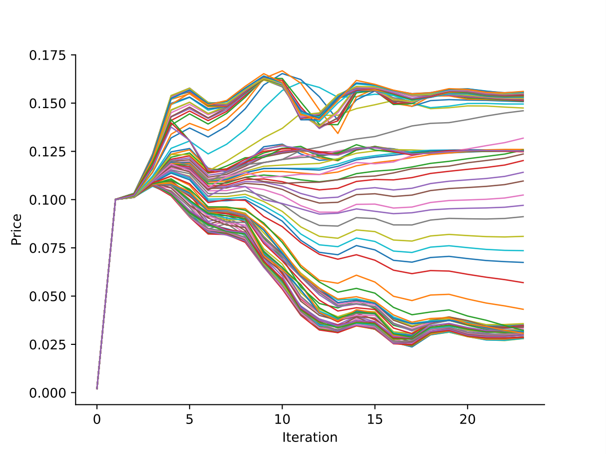


Figure 1: DADP convergence, multipliers for **Node-1**

Upper and lower bounds on the global problem

	Graph	**3-Nodes**	**6-Nodes**	**12-Nodes**	**24-Nodes**	**48-Nodes**
State dim.	$\lvert\mathbb{X}\rvert$	4	8	16	32	64
SDDP	time	1'	3'	10'	79'	453'
SDDP	LB	2.252	4.559	8.897	17.528	33.103
Price	time	6'	14'	29'	41'	128'
Price	LB	2.137	4.473	8.967	17.870	33.964
Resource	time	3'	7'	22'	49'	91'
Resource	UB	2.539	5.273	10.537	21.054	40.166

Upper and lower bounds on the global problem

	Graph	**3-Nodes**	**6-Nodes**	**12-Nodes**	**24-Nodes**	**48-Nodes**
State dim.	$\lvert\mathbb{X}\rvert$	4	8	16	32	64
SDDP	time	1'	3'	10'	79'	453'
SDDP	LB	2.252	4.559	8.897	17.528	33.103
Price	time	6'	14'	29'	41'	128'
Price	LB	2.137	4.473	8.967	17.870	33.964
Resource	time	3'	7'	22'	49'	91'
Resource	UB	2.539	5.273	10.537	21.054	40.166

- For the **24-Nodes** problem

$$
\begin{array}{ccccccc}
\underline{V}_0[sddp] & \leq & \underline{V}_0[price] & \leq & V^\sharp & \leq & \overline{V}_0[resource] \\
17.528 & \leq & 17.870 & \leq & V^\sharp & \leq & 21.054
\end{array}
$$

Upper and lower bounds on the global problem

	Graph	**3-Nodes**	**6-Nodes**	**12-Nodes**	**24-Nodes**	**48-Nodes**
State dim.	$\|\mathbb{X}\|$	4	8	16	32	64
SDDP	time	1'	3'	10'	79'	453'
SDDP	LB	2.252	4.559	8.897	17.528	33.103
Price	time	6'	14'	29'	41'	128'
Price	LB	2.137	4.473	8.967	17.870	33.964
Resource	time	3'	7'	22'	49'	91'
Resource	UB	2.539	5.273	10.537	21.054	40.166

- For the **24-Nodes** problem

$$\begin{array}{ccccccc} \underline{V}_0[sddp] & \leq & \underline{V}_0[price] & \leq & V^\sharp & \leq & \overline{V}_0[resource] \\ 17.528 & \leq & 17.870 & \leq & V^\sharp & \leq & 21.054 \end{array}$$

- For the biggest instance, Price Decomposition is 3.5x as fast as SDDP

Policy evaluation by Monte Carlo simulation

Graph	**3-Nodes**	**6-Nodes**	**12-Nodes**	**24-Nodes**	**48-Nodes**
SDDP policy	2.26 ± 0.006	4.71 ± 0.008	9.36 ± 0.011	18.59 ± 0.016	35.50 ± 0.023
Price policy	2.28 ± 0.006	4.64 ± 0.008	9.23 ± 0.012	18.39 ± 0.016	34.90 ± 0.023
Gap	-0.9 %	+1.5%	+1.4%	+1.1%	+1.7%
Resource policy	2.29 ± 0.006	4.71 ± 0.008	9.31 ± 0.011	18.56 ± 0.016	35.03 ± 0.022
Gap	-1.3 %	0.0%	+0.5%	+0.2%	+1.2%

Price policy beats SDDP policy and resource policy

$$
\begin{array}{ccccccc}
V^\sharp & \leq & C[price] & \leq & C[resource] & \leq & C[sddp] \\
V^\sharp & \leq & 18.39 & \leq & 18.56 & \leq & 18.59
\end{array}
$$

Conclusion

Conclusion

- We have presented two algorithms that decompose, spatially then temporally, a global optimization problem under coupling constraints

- On this case study, decomposition beats SDDP for large instances ($\geq$ 24 nodes)
  - In time (3.5x faster)
  - In precision ($>$ 1% better)

- Extension?
  - Move from nodal to zonal decomposition
  - Parallelization (towards a spatial parallelization scheme for SDDP)
  - Test other decomposition schemes (operator splitting)