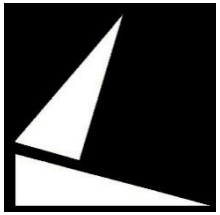
July 6th, 2016
DTU Lyngby, Denmark



EES-UETP Uncertainty in Electricity Markets and System Operations

An asynchronous distributed subgradient algorithm for solving stochastic unit commitment

Anthony Papavasiliou, Ignacio Aravena
Center for Operations Research and Econometrics
Université catholique de Louvain, Belgium

Outline

- Motivation
- Deterministic and stochastic unit commitment
- Lagrange relaxation and the subgradient method
- Synchronous distributed algorithm
- Asynchronous block-coordinate descent method
- Asynchronous distributed algorithm
    - Dual iterations and lower bound computation
    - Primal solution recovery
- Case study: Central Western Europe
    - High performance computing implementation and results
- Conclusions

Motivation

- Continental Europe leading in renewable energy integration, 82 GW of solar PV and 108 GW of wind power
- Renewable energy integration levels will continue to increase worldwide following environmental targets
- Questions:
    - Are current practices for scheduling energy and reserves effective under high renewable integration levels?
    - Are proposed stochastic models solvable in operational time frames?

Deterministic unit commitment

- Schedules production based on renewable infeed forecast, while reserving additional capacity to handle forecast errors

$$
\begin{aligned}
\min_{p,r,u,v} \quad & \sum_{g \in G} \left(NLC_g \, \mathbf{1}^T \boldsymbol{u}_g + SUC_g \, \mathbf{1}^T \boldsymbol{v}_g + c_g(\boldsymbol{p}_g) \right) \\
\text{s.t.} \quad & \sum_{g \in G} p_{g,t} \geq \hat{D}_t, \quad \forall t \in T \\
& \sum_{g \in G} r_{g,t} \geq R, \quad \forall t \in T \\
& (\boldsymbol{p}_g, \boldsymbol{r}_g, \boldsymbol{u}_g, \boldsymbol{v}_g) \in \mathcal{D}_g^R, \quad \forall g \in G
\end{aligned}
$$

Stochastic unit commitment

- Schedules production based on the possible realizations of renewable infeed (explicitly modelling uncertainty)
- Problem size grows proportionally to the number of scenarios

$$
\begin{aligned}
\min_{\substack{p,u,v \\ w,z}} \quad & \sum_{s\in S} \pi_s \sum_{g\in G} \Big(NLC_g \, \mathbf{1}^T \boldsymbol{u}_{g,s} + SUC_g \, \mathbf{1}^T \boldsymbol{v}_{g,s} + c_g\big(\boldsymbol{p}_{g,s}\big) \Big) \\
\text{s.t.} \quad & \sum_{g\in G} p_{g,s,t} \geq D_{s,t}, \quad \forall t \in T, s \in S \\
& \big(\boldsymbol{p}_{g,s}, \boldsymbol{u}_{g,s}, \boldsymbol{v}_{g,s}\big) \in \mathcal{D}_{g,s}, \quad \forall g \in G, s \in S \\
& \big(\boldsymbol{w}_g, \boldsymbol{z}_g\big) \in \mathcal{D}_g^{wz}, \quad \forall g \in G \\
& \boldsymbol{u}_{g,s} = \boldsymbol{w}_g, \ \boldsymbol{v}_{g,s} = \boldsymbol{z}_g, \quad \forall g \in G, s \in S
\end{aligned}
$$

Lagrange relaxation

- Stochastic unit commitment can be dualized by associating Lagrange multipliers $\boldsymbol{\mu}, \boldsymbol{\nu}$ to non-anticipativity constrains, leading to the following dual function

$$
\begin{aligned}
g(\boldsymbol{\mu},\boldsymbol{\nu}) = \min_{\substack{p,u,v \\ w,z}} \quad & \sum_{s\in S}\pi_s \sum_{g\in G}\Big((NLC_g\,\mathbf{1}+\boldsymbol{\mu}_{g,s})^T\boldsymbol{u}_{g,s} + (SUC_g\,\mathbf{1}+\boldsymbol{\nu}_{g,s})^T\boldsymbol{v}_{g,s} + c_g\big(\boldsymbol{p}_{g,s}\big)\Big) - \\
& \sum_{g\in G}\left(\left(\sum_{s\in S}\pi_s\boldsymbol{\mu}_{g,s}\right)^T\boldsymbol{w}_g + \left(\sum_{s\in S}\pi_s\boldsymbol{\nu}_{g,s}\right)^T\boldsymbol{z}_g\right) \\
\text{s.t.} \quad & \sum_{g\in G} p_{g,s,t} \geq D_{s,t}, \quad \forall t\in T, s\in S \\
& \big(\boldsymbol{p}_{g,s},\boldsymbol{u}_{g,s},\boldsymbol{v}_{g,s}\big)\in\mathcal{D}_{g,s}, \quad \forall g\in G, s\in S \\
& \big(\boldsymbol{w}_g,\boldsymbol{z}_g\big)\in\mathcal{D}_g^{wz}, \quad \forall g\in G
\end{aligned}
$$

Lagrange relaxation (cont.)

- The dual function $g(\boldsymbol{\mu}, \boldsymbol{\nu})$ has the following properties:
  - $g(\boldsymbol{\mu}, \boldsymbol{\nu})$ is concave
  - $g(\boldsymbol{\mu}, \boldsymbol{\nu})$ is non-differentiable (piecewise linear)
  - $g(\boldsymbol{\mu}, \boldsymbol{\nu})$ is a lower bound for the optimal solution of the stochastic unit commitment problem
  - $g(\boldsymbol{\mu}, \boldsymbol{\nu})$ is decomposable, $g(\boldsymbol{\mu}, \boldsymbol{\nu}) = f_0(\boldsymbol{\mu}, \boldsymbol{\nu}) + \sum_{s \in S} f_s(\boldsymbol{\mu}_s, \boldsymbol{\nu}_s)$, where

$$f_0(\boldsymbol{\mu}, \boldsymbol{\nu}) = \min_{w,z} \quad \sum_{g \in G} \left(\left(-\sum_{s \in S} \pi_s \boldsymbol{\mu}_{g,s} \right)^T \boldsymbol{w}_g + \left(-\sum_{s \in S} \pi_s \boldsymbol{\nu}_{g,s} \right)^T \boldsymbol{z}_g \right)$$
$$\text{s.t.} \quad \left(\boldsymbol{w}_g, \boldsymbol{z}_g \right) \in \mathcal{D}_g^{wz}, \quad \forall g \in G$$

and

Lagrange relaxation (cont.)

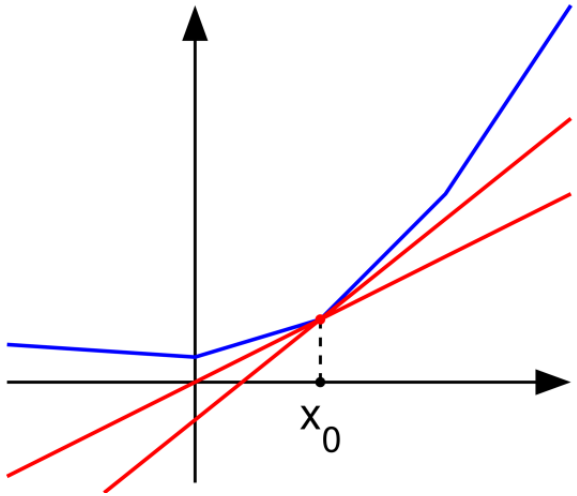
$$
\begin{aligned}
f_s(\boldsymbol{\mu}_s, \boldsymbol{\nu}_s) = \min_{p,u,v} \quad & \pi_s \sum_{g \in G} \Big((NLC_g \, \mathbf{1} + \boldsymbol{\mu}_{g,s})^T \boldsymbol{u}_g + (SUC_g \, \mathbf{1} + \boldsymbol{\nu}_{g,s})^T \boldsymbol{v}_g + c_g(\boldsymbol{p}_g) \Big) \\
\text{s.t.} \quad & \sum_{g \in G} p_{g,t} \geq D_{s,t}, \quad \forall t \in T \\
& (\boldsymbol{p}_g, \boldsymbol{u}_g, \boldsymbol{v}_g) \in \mathcal{D}_{g,s}, \quad \forall g \in G
\end{aligned}
$$

- Evaluation of $g(\boldsymbol{\mu}, \boldsymbol{\nu})$ is amenable to distributed computing
- Maximizing $g(\boldsymbol{\mu}, \boldsymbol{\nu})$ (***dual problem***) can lead to a tight lower bound on stochastic unit commitment
- Primal recovery heuristics can be used to compute *good* solutions for stochastic unit commitment, starting from approximate solutions to the dual problem

Solving the dual problem: subgradient method

- Let's assume that we want to minimize $h(\boldsymbol{x})$, which is a convex and non-differentiable function
- Recall: a sub-gradient $\boldsymbol{d}$ of $h(\boldsymbol{x})$ at $\boldsymbol{x}_0$ is a vector such that

$$\boldsymbol{d}^T(\boldsymbol{x} - \boldsymbol{x}_0) \leq h(\boldsymbol{x}) - h(\boldsymbol{x}_0) \; \forall \boldsymbol{x}$$



- Let $\boldsymbol{x}^k$ be the current iterate and $\boldsymbol{d}^k$ a subgradient of $h(\boldsymbol{x})$ at $\boldsymbol{x}^k$. A subgradient iteration is performed as:

$$\boldsymbol{x}^{k+1} := \boldsymbol{x}^k - \alpha_k \boldsymbol{d}^k$$

where α_k is a suitable step size (decreasing, constant, dynamic)

Subgradient method applied to $g(\mu, \nu)$

- Minimizing $-g(\boldsymbol{\mu}, \boldsymbol{\nu})$ is equivalent to maximize $g(\boldsymbol{\mu}, \boldsymbol{\nu})$ ($-g(\boldsymbol{\mu}, \boldsymbol{\nu})$ is convex and non-differentiable)
- A subgradient of $-g(\boldsymbol{\mu}, \boldsymbol{\nu})$ at $(\boldsymbol{\mu}^k, \boldsymbol{\nu}^k)$ can be computed as

$$\boldsymbol{d}_{\boldsymbol{\mu}_s}^k = \pi_s(\boldsymbol{w}^* - \boldsymbol{u}_s^*), \qquad \forall s \in S$$

$$\boldsymbol{d}_{\boldsymbol{\nu}_s}^k = \pi_s(\boldsymbol{z}^* - \boldsymbol{v}_s^*), \qquad \forall s \in S$$

  where $(\boldsymbol{w}^*, \boldsymbol{z}^*)$ are obtained from the evaluation of $f_0(\boldsymbol{\mu}^k, \boldsymbol{\nu}^k)$ and $(\boldsymbol{u}_s^*, \boldsymbol{v}_s^*)$ are obtained from the evaluation of $f_s(\boldsymbol{\mu}_s^k, \boldsymbol{\nu}_s^k)$
- A subgradient update on $(\boldsymbol{\mu}^k, \boldsymbol{\nu}^k)$ requires the independent evaluation of f_0 and $f_s \rightarrow$ parallelization

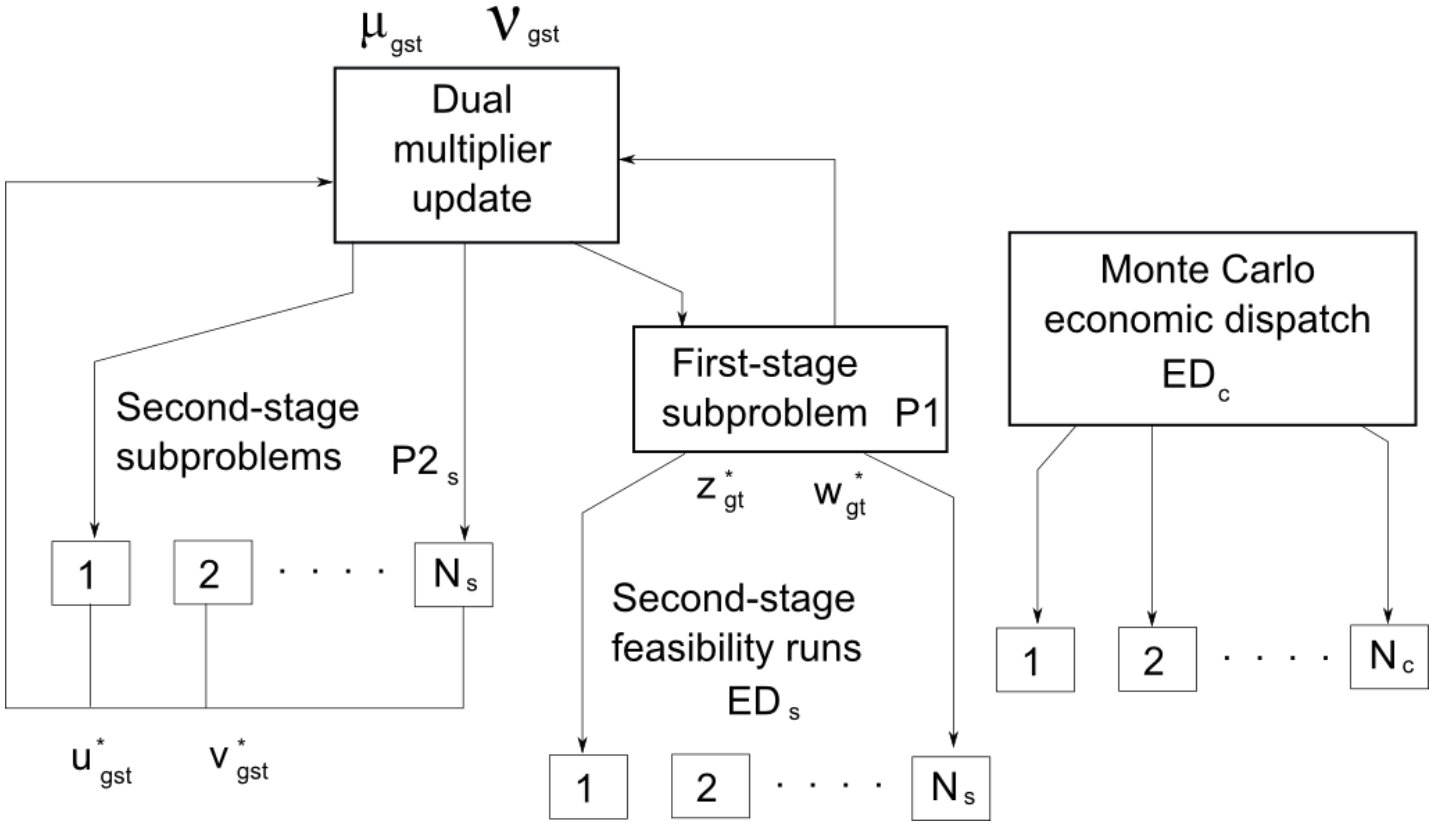
Synchronous distributed algorithm for stochastic UC

Let $k = 0$, $\boldsymbol{\mu}^0 = 0$, $\boldsymbol{\nu}^0 = 0$, $LB = -\infty$, $UB = +\infty$

1. Evaluate (in parallel) $f_0(\boldsymbol{\mu}^k, \boldsymbol{\nu}^k)$ and $f_s(\boldsymbol{\mu}_s^k, \boldsymbol{\nu}_s^k), \forall s \in S \rightarrow \boldsymbol{d}$
2. Let $LB := \max\{LB, f_0(\boldsymbol{\mu}^k, \boldsymbol{\nu}^k) + \sum_s f_s(\boldsymbol{\mu}_s^k, \boldsymbol{\nu}_s^k)\}$
3. Perform primal recovery (in parallel) and update UB.
4. If $UB - LB \leq \epsilon$, terminate. Otherwise, continue.
5. Make subgradient update: $(\boldsymbol{\mu}^{k+1}, \boldsymbol{\nu}^{k+1}) := (\boldsymbol{\mu}^k, \boldsymbol{\nu}^k) - \alpha_k \boldsymbol{d}$
6. Let $k := k + 1$ and go to 1

- Two steps of the algorithm can be parallelized
- Note that each iteration includes synchronization points, e.g. 2

Implementation



Synchronous distributed algorithm for stochastic UC

- Why are synchronization points undesirable?

  They can lead to **inefficient** use of distributed computing infrastructure
- How bad can this be for stochastic UC?

  We have seen instances where the evaluation of f_s for one scenario can take up **75 times** more than the rest!
- For MILP problems, **balanced size does not imply balanced solution times**
- Alternative: use asynchronous versions of the subgradient algorithm, Bertsekas & Tsitsiklis (1989), Tseng, (2001), Nedić et al., (2001), Kiwiel, (2004), Liu et al., (2014)

Block-coordinate descent method

- Performs updates on subsets of variables (*block-coordinate*)
- Useful alternative whenever computing the subgradient with respect to a subset of variables is considerably cheaper than computing a full subgradient
- Serial algorithm: maximizing $g(\boldsymbol{\mu}, \boldsymbol{\nu})$, starting from $k = 1$
  1. Select a scenario $s^k \in S$ (sequentially or randomly)
  2. Evaluate $f_0(\boldsymbol{\mu}^k, \boldsymbol{\nu}^k)$ and $f_s\left(\boldsymbol{\mu}_{s^k}^k, \boldsymbol{\nu}_{s^k}^k\right) \rightarrow \boldsymbol{d}_{s^k}$
  3. Make update: $\left(\boldsymbol{\mu}_{s^k}^{k+1}, \boldsymbol{\nu}_{s^k}^{k+1}\right) := \left(\boldsymbol{\mu}_{s^k}^k, \boldsymbol{\nu}_{s^k}^k\right) - \alpha_k \boldsymbol{d}_{s^k}$

     $\left(\boldsymbol{\mu}_r^{k+1}, \boldsymbol{\nu}_r^{k+1}\right) := \left(\boldsymbol{\mu}_r^k, \boldsymbol{\nu}_r^k\right) \ \forall r \in S - \{s^k\}$
  4. Let $k := k + 1$ and go to 1.

Asynchronous distributed block-coordinate descent method

- Assume you have as many processors as block-coordinates.

  Ideally, at all times, each processor should be performing a block-coordinate update.
- Associate each block-coordinate s with a processor m_s.
- Asynchronous algorithm:

  At each processor m_s, starting from $k(s) = 1$

  1. Get current $\left(\boldsymbol{\mu}_r^{k(s)}, \boldsymbol{\nu}_r^{k(s)}\right) := \left(\boldsymbol{\mu}_r^{k(r)}, \boldsymbol{\nu}_r^{k(r)}\right), \forall r \in S - \{s\}$
  2. Evaluate $f_0\left(\boldsymbol{\mu}^{k(s)}, \boldsymbol{\nu}^{k(s)}\right)$ and $f_s\left(\boldsymbol{\mu}_s^{k(s)}, \boldsymbol{\nu}_s^{k(s)}\right) \rightarrow \boldsymbol{d}_s$
  3. Make update: $\left(\boldsymbol{\mu}_s^{k(s)+1}, \boldsymbol{\nu}_s^{k(s)+1}\right) := \left(\boldsymbol{\mu}_s^{k(s)}, \boldsymbol{\nu}_s^{k(s)}\right) - \alpha_k^s \boldsymbol{d}_s$
  4. Let $k(s) := k(s) + 1$ and go to 1.

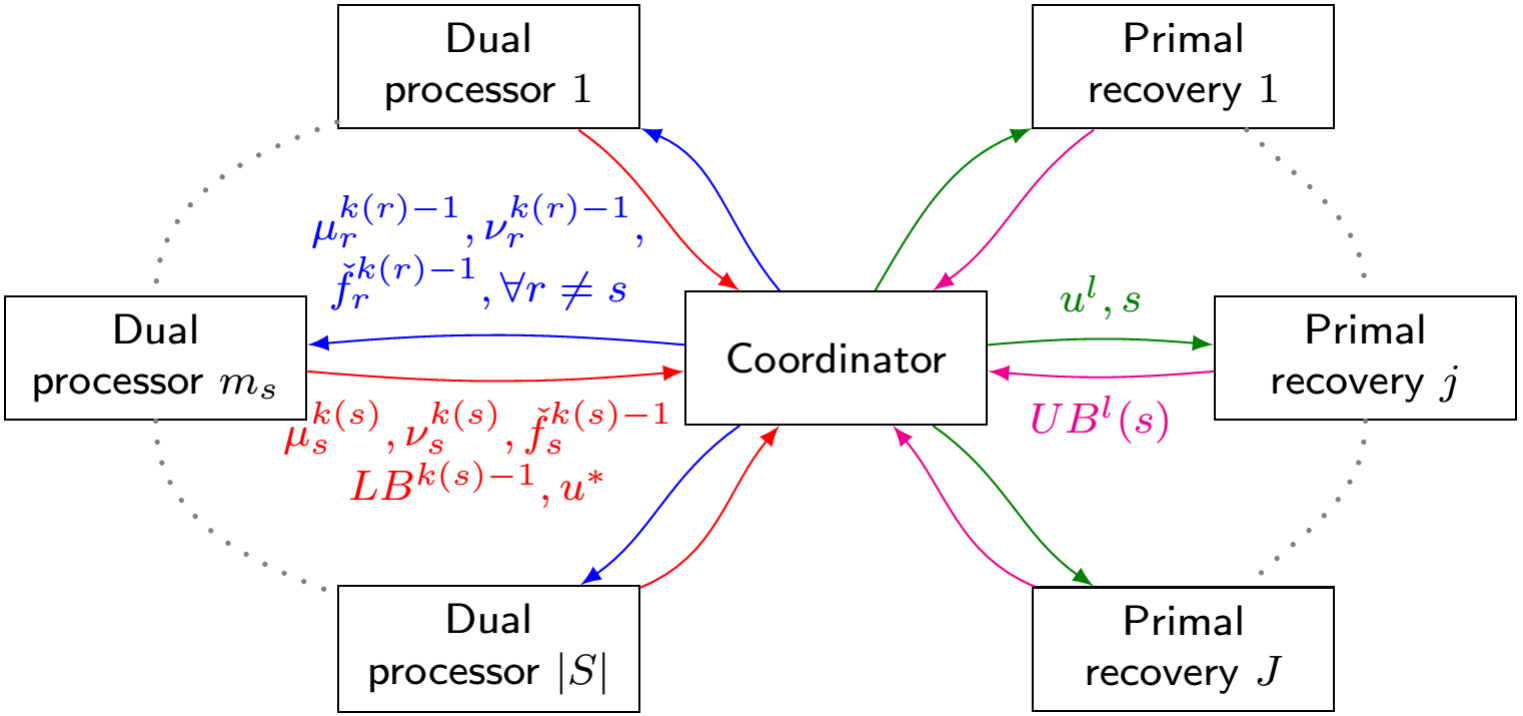
Asynchronous distributed block-coordinate descent method

- Asynchronous algorithm fully exploits distributed computing infrastructure, but…
- Remark 1: *Get current…*

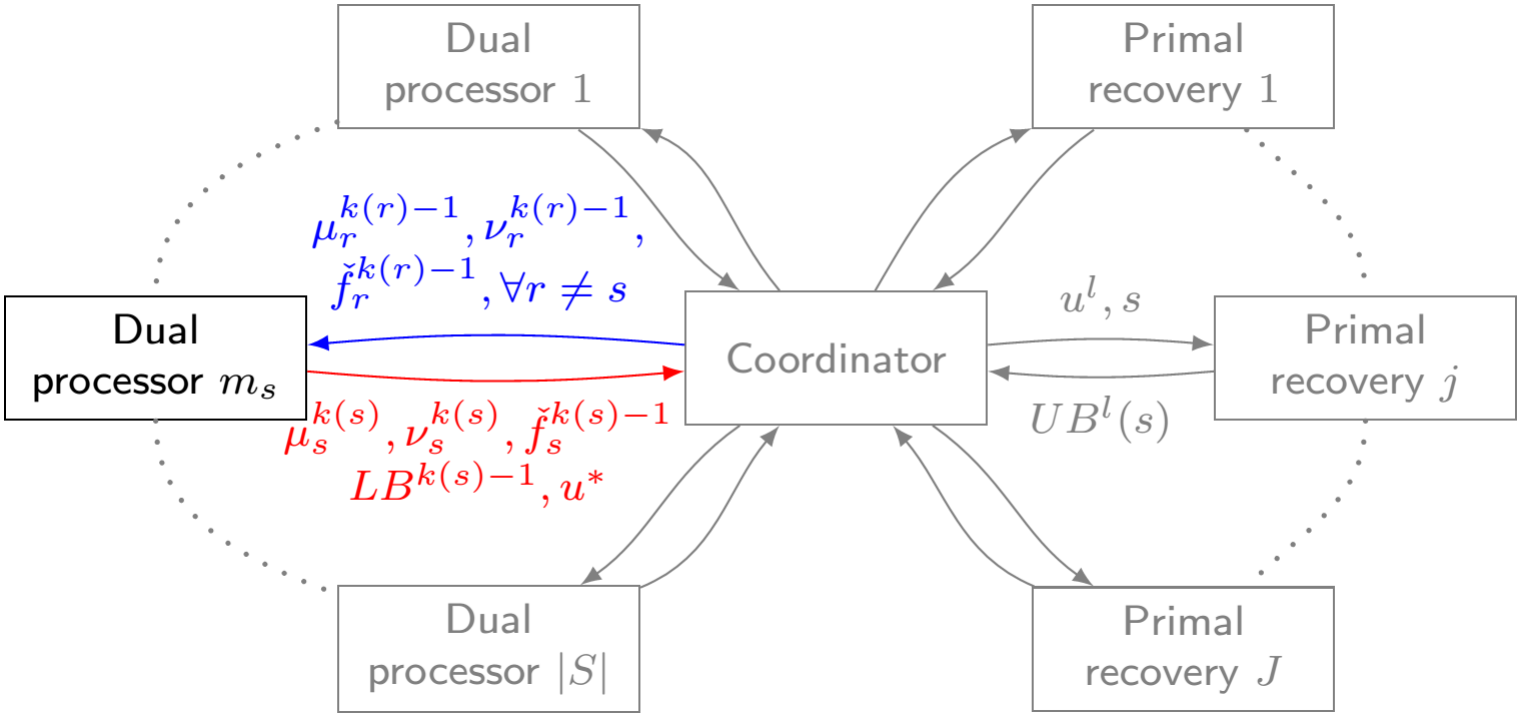
  Requires communications between processors. Expensive if the number of processors is large.
- Remark 2: α_k^s (k is the global iterator count, i.e. $k \coloneqq \sum_{s \in S} k(s)$)

  Some block-coordinates will be updated less frequently than others. Step sizes need to compensate for this difference: larger step sizes for less frequently updated block-coordinates.
- Remark 3: Value of $g\left(\boldsymbol{\mu}^{k(s)}, \boldsymbol{\nu}^{k(s)}\right)$?

Asynchronous distributed algorithm for stochastic UC



Asynchronous algorithm: Dual process



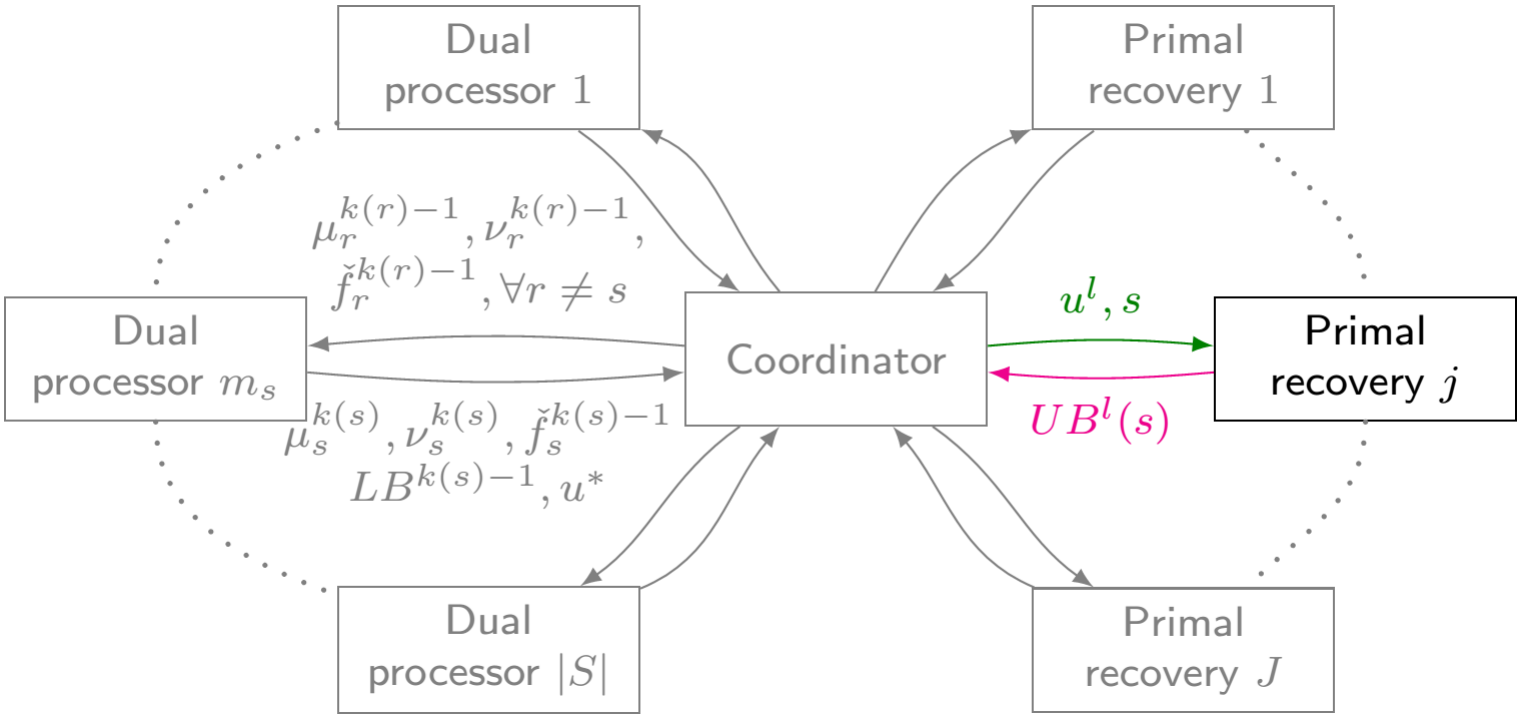
Asynchronous algorithm: Dual process

1. Evaluate $\check{f}_s^{k(s)} := f_s\left(\boldsymbol{\mu}_s^{k(s)}, \boldsymbol{\nu}_s^{k(s)}\right) \rightarrow \boldsymbol{u}^*$
2. Gather $\boldsymbol{\mu}_r^{k(r)-1}, \boldsymbol{\nu}_r^{k(r)-1}, \check{f}_r^{k(r)-1}$ $\forall r \in S - \{s\}$, from *Coordinator*

   Let $\bar{\boldsymbol{\mu}} := \left(\boldsymbol{\mu}_{s_1}^{k(s_1)-1}, \ldots, \boldsymbol{\mu}_s^{k(s)}, \ldots, \boldsymbol{\mu}_{r_{|S|}}^{k(r_{|S|})-1}\right)$

   $\bar{\boldsymbol{\nu}} := \left(\boldsymbol{\nu}_{s_1}^{k(s_1)-1}, \ldots, \boldsymbol{\nu}_s^{k(s)}, \ldots, \boldsymbol{\nu}_{r_{|S|}}^{k(r_{|S|})-1}\right)$
3. Evaluate $f_0(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\nu}}) \xrightarrow{(\text{using } \boldsymbol{u}^*)} \bar{\boldsymbol{d}}_s$
4. Compute $LB^{k(s)} := g(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\nu}}) = f_0(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\nu}}) + \check{f}_s^{k(s)} + \sum_{r \neq s} \check{f}_r^{k(r)-1}$
5. Make update $\left(\boldsymbol{\mu}_s^{k(s)+1}, \boldsymbol{\nu}_s^{k(s)+1}\right) := \left(\boldsymbol{\mu}_s^{k(s)}, \boldsymbol{\nu}_s^{k(s)}\right) - \alpha_k^s \bar{\boldsymbol{d}}_s$
6. Let $k(s) := k(s) + 1$
7. Send $\boldsymbol{\mu}_s^{k(s)}, \boldsymbol{\nu}_s^{k(s)}, \check{f}_s^{k(s)-1}, LB^{k(s)-1}, \boldsymbol{u}^*$ to *Coordinator* and go to 1.

Asynchronous algorithm: Primal recovery



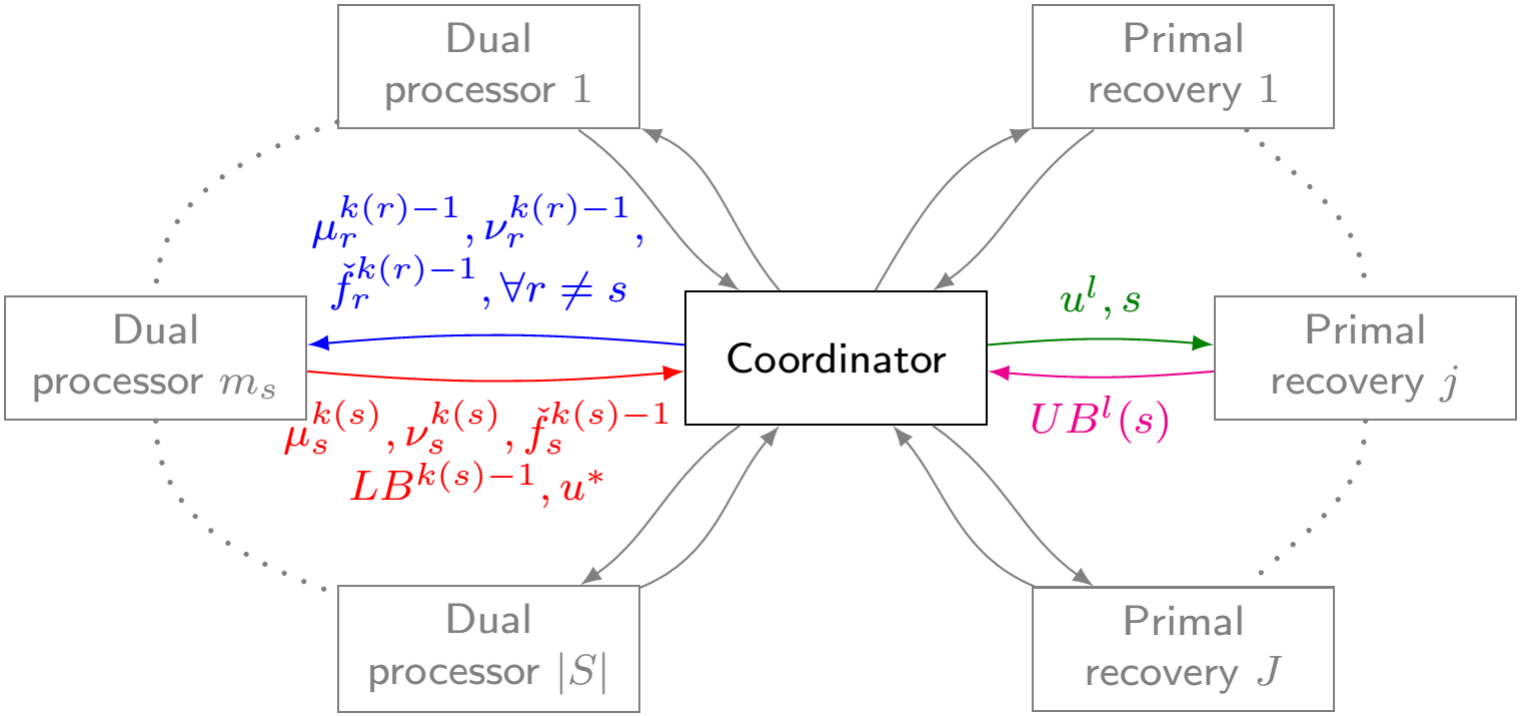
Asynchronous algorithm: Primal recovery

1. Receive *assignment* $\boldsymbol{u}^l, s_i$ from *Coordinator*
2. Determine $\boldsymbol{v}^l$ and solve 2nd stage problem for s_i,

$$\begin{aligned} \boldsymbol{p}^* := \arg\min_{p} \quad & \sum_{g \in G} c_g(\boldsymbol{p}_g) \\ \text{s.t.} \quad & \sum_{g \in G} p_{g,t} \geq D_{s_i,t}, \quad \forall t \in T \\ & (\boldsymbol{p}_g, \boldsymbol{u}_g^l, \boldsymbol{v}_g^l) \in \mathcal{D}_{g,s_i}, \quad \forall g \in G \end{aligned}$$

3. Compute $UB^l(s_i) := \pi_{s_i} \sum_{g \in G} \left(NLC_g \mathbf{1}^T \boldsymbol{u}_g^l + SUC_g \mathbf{1}^T \boldsymbol{v}_g^l + c_g(\boldsymbol{p}_g^*) \right)$
4. Send $UB^l(s_i)$ to *Coordinator* and wait for next *assignment*

Asynchronous algorithm: Coordinator



Asynchronous algorithm: Coordinator

Let: $k(s) = 1,\ \boldsymbol{\mu}_s^0 = \boldsymbol{\nu}_s^0 = \boldsymbol{\mu}_s^1 = \boldsymbol{\nu}_s^1 = \mathbf{0},\ \check{f}_s^0 = -\infty, \forall s \in S$

$\mathcal{L}_u = \emptyset,\ LB = -\infty, UB = +\infty$

1. *Launch Dual process* s with $(\boldsymbol{\mu}_s^1, \boldsymbol{\nu}_s^1)\ \forall s \in S$
2. *Assign* as many queued $(\boldsymbol{u}^l, s_i)$ to *Primal recovery* processes as possible, *remove* them from $\mathcal{L}_u$ and *wait* for next *event*
3. If *receive* $\boldsymbol{\mu}_s^{k(s)}, \boldsymbol{\nu}_s^{k(s)}, \check{f}_s^{k(s)-1}, LB^{k(s)-1}, \boldsymbol{u}^*$ then,
   - i. *Store* $\boldsymbol{\mu}_s^{k(s)}, \boldsymbol{\nu}_s^{k(s)}, \check{f}_s^{k(s)-1}$ and *add* $\{(\boldsymbol{u}^*, r)\ \forall r \in S\}$ to $\mathcal{L}_u$
   - ii. Update lower bound $LB := \max\{LB, LB^{k(s)-1}\}$ and go to 4.

   Otherwise (received $UB^l(s)$),
   - i. If $UB^l(s)$ is available $\forall s \in S$ then, update upper bound $UB := \min\{UB, \sum_{s \in S} UB^l(s)\}$ and go to 4. Otherwise, go to 2.
4. If $UB - LB \leq \epsilon$, terminate. Otherwise, go to 2.

Asynchronous algorithm: lower bound initialization

- Recall: evaluation of f_s requires to solve a UC problem,

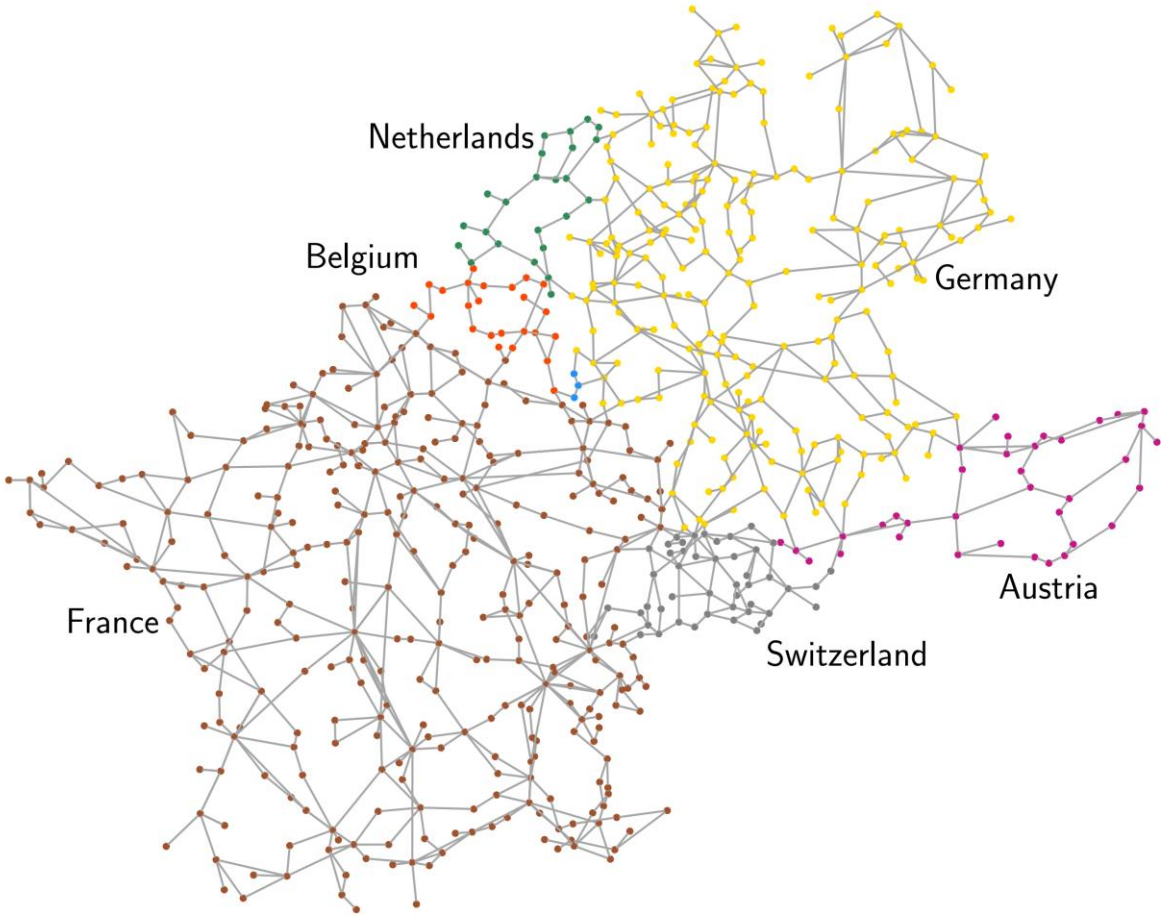
$$f_s(\boldsymbol{\mu}_s, \boldsymbol{\nu}_s) = \min_{p,u,v} \quad \pi_s \sum_{g \in G} \left((NLC_g \, \mathbf{1} + \boldsymbol{\mu}_{g,s})^T \boldsymbol{u}_g + (SUC_g \, \mathbf{1} + \boldsymbol{\nu}_{g,s})^T \boldsymbol{v}_g + c_g(\boldsymbol{p}_g) \right)$$

$$\text{s.t.} \quad \sum_{g \in G} p_{g,t} \geq D_{s,t}, \quad \forall t \in T$$

$$(\boldsymbol{p}_g, \boldsymbol{u}_g, \boldsymbol{v}_g) \in \mathcal{D}_{g,s}, \quad \forall g \in G$$

- Evaluation of f_s can take up to 75 times more for certain scenarios than others → **one scenario** can delay the computation of the first "full" lower bound
- We can use a relaxation of UC to compute a fast lower bound during the first iteration ($k(s) = 1 \; \forall s \in S$)
  - Linear relaxation of UC
  - Sequence of OPF problems

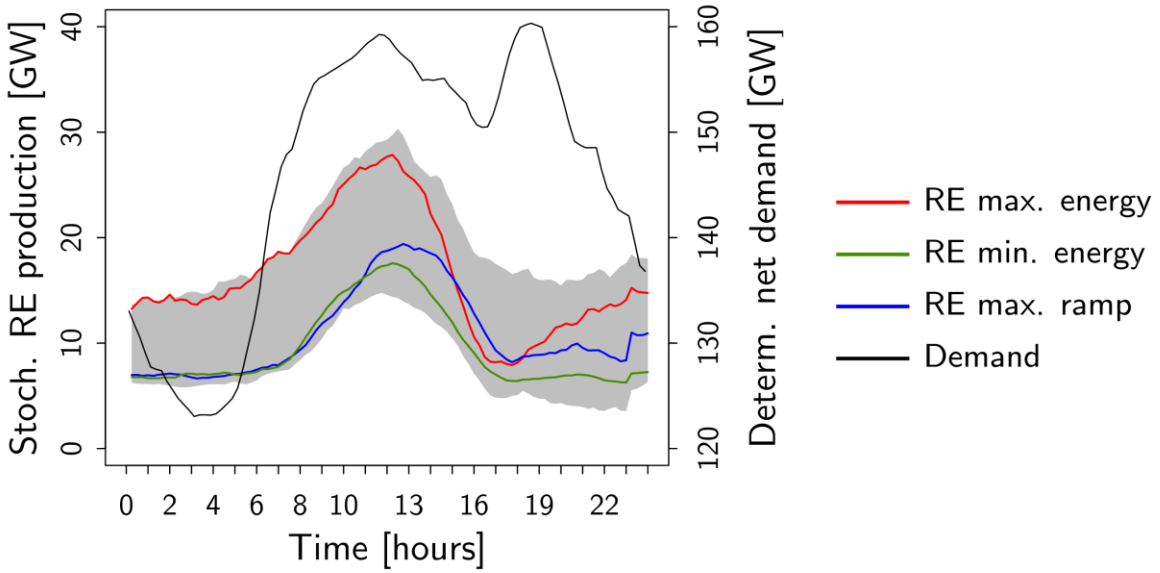
Case study: Central Western European system



CWE grid model of Hutcheon and Bialek (2013): 7 countries, 679 nodes, 1073 lines

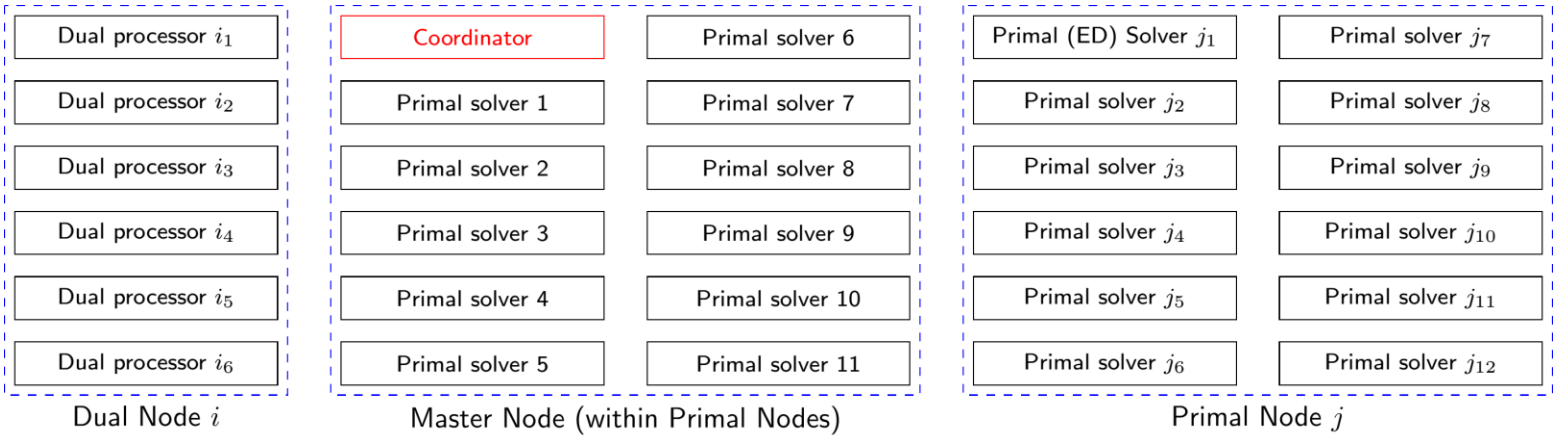
Case study: Central Western European system

- 656 thermal generators: 85 GW nuclear, 40 GW CHP, 99 GW slow, 14 GW fast and 10 GW aggregated
- 47.3 GW of solar PV power and 51.2 GW of wind power
- Multi-area renewable production and demand with 15’ time resolution



Implementation details

- Asynchronous algorithm implemented in Mosel (using *mmjobs* to manage communications) calling on Xpress
- Lawrence Livermore National Laboratory Sierra cluster: 23,328 cores on 1,944 nodes, 2.8 Ghz, 24 GB/node
- Using 10 nodes:
  - 5 nodes for *Dual processes* (6 per node)
  - 5 nodes to *Primal recovery* (12 per node)



Instances

- Comparing deterministic UC (Determ2R and Determ3R) with stochastic UC (Stoch30, Stoch60 and Stoch120) for an industrial scale model.
- 8 instances (day types) per model: 2 per season, weekday/weekend

Model	Scenarios	Variables	Constraints	Integers
Determ2R	1	570.432	655.784	9.552
Determ3R	1	636.288	719.213	9.552
Stoch30	30	13.334.400	16.182.180	293.088
Stoch60	60	26.668.800	32.364.360	579.648
Stoch120	120	53.337.600	64.728.720	1.152.768

Running times

Solution statistics over 8 instances.

Model	Nodes	Init.	Running time [hours]	Worst final gap [%]
Determ2R	1		1,9 (0,6 – 4,2)	0,95
Determ3R	1		9.4* (6,3 – 10,0)	4,91
Stoch30	10	LP	1,1 (0,7 – 2,2)	0,93
	10	OPF	0,8 (0,3 – 1,8)	1,00
Stoch60	10	LP	3,2 (1,1 – 8,4)	1,00
	10	OPF	1,5 (0,6 – 4,7)	0,97
Stoch120	10	LP	6,1* (1,6 – 10,0)	1,68
	10	OPF	3,0* (0,6 – 10,0)	1,07

Termination criteria: 1% optimality or 10 hours wall-time.

* Average running time computed considering models hitting wall-time.

Running times: optimality vs. wall-time

Solution statistics over 8 instances.

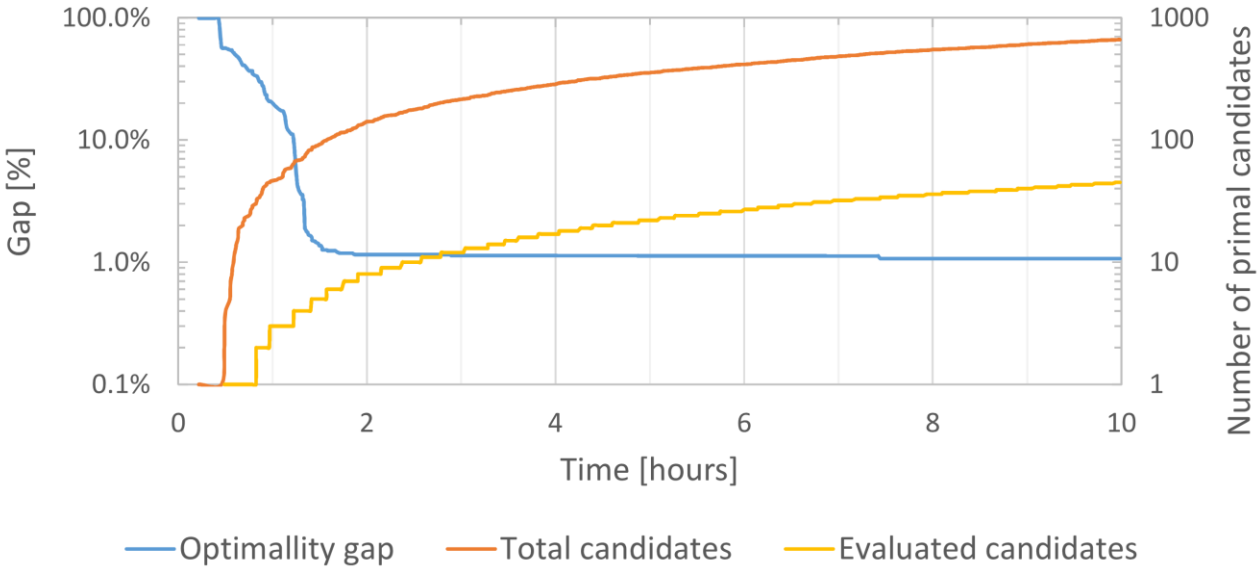
Model	Init.	Worst gap [%]			
		1 hour	2 hours	4 hours	8 hours
Stoch30	LP	7,59	1,02	0,93	
	OPF	1,90	1,00		
Stoch60	LP	23,00	5,32	5,22	4,50
	OPF	4,60	1,57	1,03	0,97
Stoch120	LP	70,39	31,66	4,61	1,87
	OPF	46,69	27,00	1,42	1,07

- Lower bound initialization using sequential OPF observed to be very effective, sometimes avoiding to evaluate f_s for hard scenarios
- Asynchronous stochastic UC algorithm capable of achieving acceptable optimality gaps within operational time frames

Room for improvement: evaluation of primal candidates

Bounds and primal candidates.

Stoch120 (OPF), summer weekday (worst case).



- Valuable computational resources spent in **detailed** evaluation of sub-optimal candidates.
- Future research: pre-screening and pruning of primal candidates

Conclusions and perspectives

- Asynchronous algorithms and high performance computing have the potential to solve stochastic UC within the same time frame required to solve deterministic UC using a state-of-the-art MILP solver
- Sequential OPF initialization provides fast lower bounds, significantly reducing running times
- Obtaining good primal candidates from first iterations drastically accelerates the convergence of the algorithm
- Future extensions:
  - Pruning and scoring primal candidates
  - Dynamical queue management for dual and primal processes
  - Multi-stage stochastic UC

Thank you.

Contact:

Anthony Papavasilou, anthony.papavasilou@uclouvain.be

Ignacio Aravena, ignacio.aravena@uclouvain.be