Two classical problems of Centralized Energy Production Management

Problem formulation and practical algorithms

October, 9th 2024

Antoine BAMBADE– EDF LAB/OSIRIS/R36

Alex Fauduet– EDF LAB/OSIRIS/R36

Sommaire

1.1 Context

- **Horizon**: « mid term » (i.e., ~years);
- **Setup**:
  - We consider a fleet of nuclear power plants (of different categories);
  - Their refueling shutdowns are already scheduled (for a few years);
- **Question**: how to consume their nuclear stock fuels between shutdowns ?

1.1 Context

- **Horizon**: « mid term » (i.e., ~years);
- **Setup**:
  - We consider a fleet of nuclear power plants (of different categories);
  - Their refueling shutdowns are already scheduled (for a few years);
- **Question**: how to consume their nuclear stock fuels between shutdowns ?

  → **Need**: an indicator (consume now, or later);

1.1 Context

- **Horizon**: « mid term » (i.e., ~years);
- **Setup**:
  - We consider a fleet of nuclear power plants (of different categories);
  - Their refueling shutdowns are already scheduled (for a few years);
- **Question**: how to consume their nuclear stock fuels between shutdowns ?
  - → **Need**: an indicator (consume now, or later);
  - → **Difficulty**: ensure anticipation with flexibility;

1.1 Context

- Horizon: « mid term » (i.e., ~years);
- Setup:
  - We consider a fleet of nuclear power plants (of different categories);
  - Their refueling shutdowns are already scheduled (for a few years);
- Question: how to consume their nuclear stock fuels between shutdowns ?
  - → Need: an indicator (consume now, or later);
  - → Difficulty: ensure anticipation with flexibility;
  - → Same question for other types of « fuel » (water, oil etc.);

1.1 Context

- **Horizon**: « mid term » (i.e., ~years);
- **Setup**:
  - We consider a fleet of nuclear power plants (of different categories);
  - Their refueling shutdowns are already scheduled (for a few years);
- **Question**: how to consume their nuclear stock fuels between shutdowns ?
  - → **Need**: an indicator (consume now, or later);
  - → **Difficulty**: ensure anticipation with flexibility;
  - → **Same question for other types of « fuel »** (water, oil etc.);

How is it used ?

- Energy production management;
  - **Horizon**: « short term » (i.e., ~intra day to a few days);
  - **Merit order principle**: (« smaller usage value is better for production »);
- Coverage options for trading;

1.2 Problem formulation

N nuclear power plants labelled by i characterized by

- s_i^t : stock at discrete time t
- p_i^t : power at discrete time t (discretized values, different régimes)

$\omega \in \Omega$: a possible scenario for the power plants (finite number, M, of them).

- Random prices
- Random outages

$v_{i,\omega}^t$: marginal valorisation of the electric power for i at t.

$g_{i,\omega}^t(s)$: gains for the power plant i, starting from a given stock s at t.

We use at t, as usage value signal, an element of $\partial_s \mathbb{E}[g_{i,\omega}^t(s)]$

1.3 Current algorithms used

- Principle
  - Stock discretization for some range
  - For each plant i at t, compute for each scenario $g^t_{i,\omega}(s)$
  - Using finite difference, compute an element of $\partial_s g^t_{i,\omega}(s)$
  - Assemble an average over scenarii for estimating $\partial_s \mathbb{E}[g^t_{i,\omega}(s)]$

1.3 Current algorithms used

- Principle
  - Stock discretization for some range
  - For each plant i at t, compute for each scenario $g_{i,\omega}^{t}(s)$
  - Using finite difference, compute an element of $\partial_s g_{i,\omega}^{t}(s)$
  - Assemble an average over scenarii for estimating $\partial_s \mathbb{E}[g_{i,\omega}^{t}(s)]$

- How computing each gains ?

1.3 Current algorithms used

- American option valuation detour

Algorithm 1: Longstaff-Schwart algorithm (exact version)

Inputs: M (number of simulations), T (number of time steps), r (unrisky rate), K (strike price),

Initialization:

1) simulate M price trajectories $\{v_\omega^t\}_{\omega,t}$ with T time steps,

2) starting points: $g^T(v_\omega^T) = e^{-rT}\max\{0, K - v_\omega^T\}, \forall\omega$,

for $\omega \in \{\omega_1, ..., \omega_M\}$ **do**

  for $t = T, ..., 1$ **do**

    $h^t(v) = e^{-rt}\max\{0, K - v\}$ (current payoff);

    $C^t(v) \stackrel{\text{def}}{=} \mathbb{E}_\omega(g^{t+1}(v)|v^t = v))$ (continuation value);

    $$g^t(v_\omega^t) = \begin{cases} h^t(v_\omega^t), \text{ (Exercise the option)} & \text{if } h^t(v_\omega^t) \geq C^t(v_\omega^t), \\ g^{t+1}(v_\omega^t), \text{ (Do not exercise))} & \text{otherwise,} \end{cases};$$

  end

end

Return: $\mathbb{E}_\omega(g^1(v_\omega^1))$

Reference: **Francis A. Longstaff, Eduardo S. Schwartz**, *Valuing American Options by Simulation: A Simple Least-Squares Approach* (The Review of Financial Studies) (2001) Vol 14, No 1, pp. 113-147

1.3 Current algorithms used

- American option valuation detour

Algorithm 2: Longstaff-Schwart algorithm (numerical version)

Inputs: M (number of simulations), T (number of time steps), r (unrisky rate), K (strike price), p (number of polynomials of OLS basis expansion chosen)

Initialization:

1) simulate M price trajectories $\{v_\omega^t\}_{\omega,t}$ with T time steps,

2) starting points: $g^T(v_\omega^T) = e^{-rT}\max\{0, K - v_\omega^T\}, \forall\omega,$

for $\omega \in \{\omega_1, ..., \omega_M\}$ **do**

  for $t = T, ..., 1$ **do**

    $h^t(v) = e^{-rt}\max\{0, K - v\}$ (current payoff);

    $C^t(v) \stackrel{\text{def}}{=} \mathbb{E}_\omega(g^{t+1}(v_\omega^{t+1})|v_\omega^t = v))$ (continuation value);

    $\hat{C}^t(v) = \sum_{k=1}^p \beta_k\phi_k(v)$ (OLS estimate);

    $$g^t(v_\omega^t) = \begin{cases} h^t(v_\omega^t), \text{ (Exercise the option)} & \text{if } h^t(v_\omega^t) \geq \hat{C}(v_\omega^t), \\ g^{t+1}(v_\omega^t), \text{ (Do not exercise))} & \text{otherwise,} \end{cases};$$

  end

end

Return: $\frac{1}{M}\sum_{j=1}^M(g^1(v_{\omega_j}^1))$

Reference: **Francis A. Longstaff, Eduardo S. Schwartz**, *Valuing American Options by Simulation: A Simple Least-Squares Approach* (The Review of Financial Studies) (2001) Vol 14, No 1, pp. 113-147

1.3 Current algorithms used

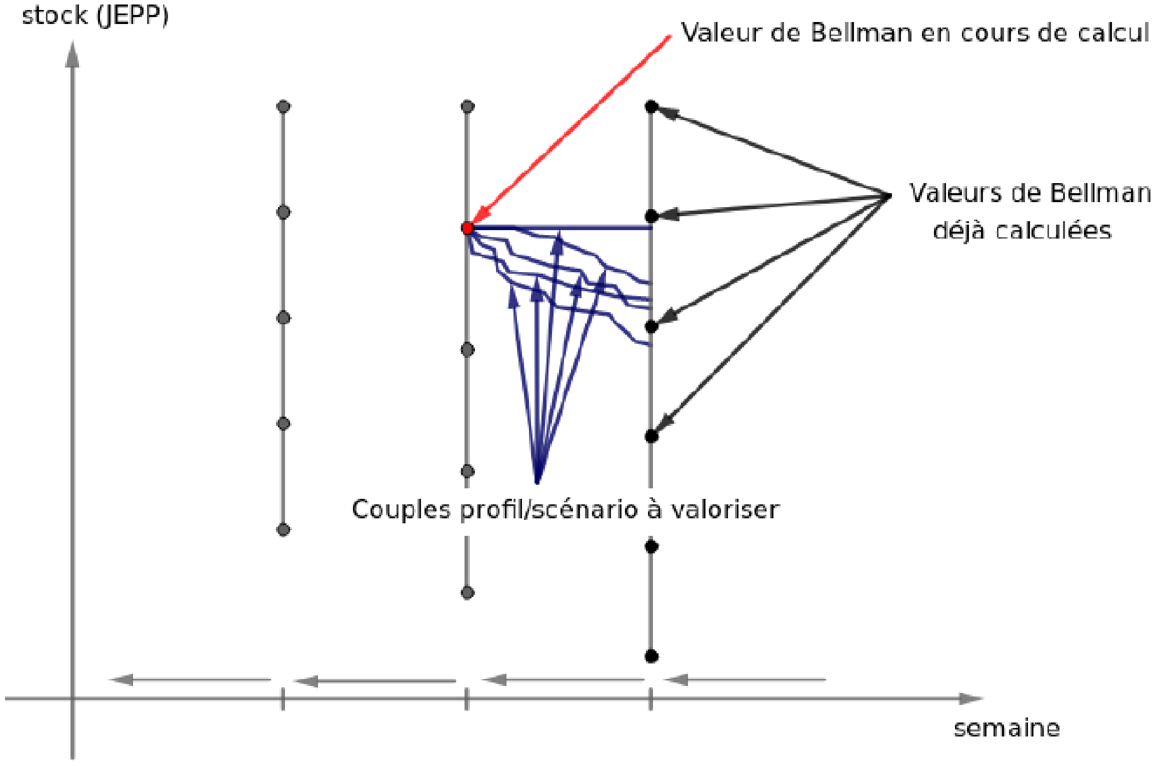
- Backward pass details

Algorithm 3: MOON backward pass (numerical version)

Inputs: M (number of simulations), T (number of time steps), K (number of polynomials of OLS basis expansion chosen), discretized stock $\{s_j\}_j$ (number J), power plant production structure $p(s)$,
1) M price trajectories $\{v^t_\omega\}_{\omega,t}$ with T time steps and power plant outage scenarii,
2) starting points: $g^T(v^T_\omega, s_j)$, $\forall \omega, s_j$,

for $t = T, ..., 1$ **do**
  for $s \in \{s_1, ..., s_J\}$ **do**
    for $\omega \in \{\omega_1, ..., \omega_M\}$ **do**
      $C^t(v, s) \stackrel{\text{def}}{=} \mathbb{E}_\omega(g^{t+1}(v^{t+1}_\omega, s) | v^t_\omega = v))$ (continuation value);
      $\hat{C}^t(v, s) = \sum_{k=1}^{K} \beta_k \phi_k(v, s)$ (OLS estimate);
      $h^t(v, p, s) \stackrel{\text{def}}{=} \sum_\tau p^{t+\tau}(s^{t+\tau}) v^{t+\tau} + \hat{C}^t(v, s) + J(\{p^{t+\tau}(s^{t+\tau})\}_\tau)$ (payoff function);
      $\{p^{t+\tau}_\star(s, v^t_\omega)\}_\tau \in \arg\max_p h^t(v^t_\omega, p, s)$ (production profile)
      $g^t(v^t_\omega, s) = \sum_\tau p^{t+\tau}_\star v^{t+\tau}_\omega + g^{t+1}(v^t_\omega, s) + J(\{p^{t+\tau}_\star(s^{t+\tau})\}_\tau)$;
    end
  end
end

Return: $\{g^t(v^t_\omega, s)\}_{t,\omega,s}$

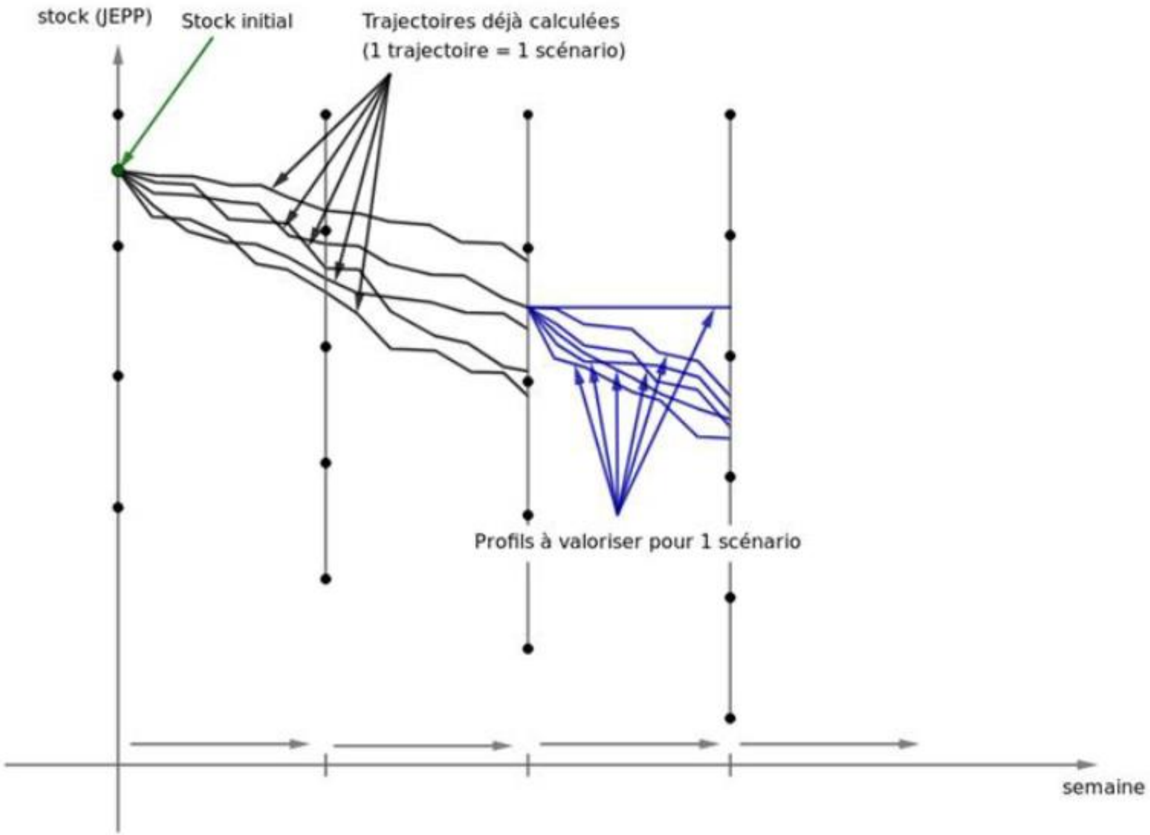


1.3 Current algorithms used

- Forward pass details

Algorithm 4: MOON forward pass (numerical version)

Inputs: M (number of simulations), T (number of time steps), K (number of polynomials of OLS basis expansion chosen), discretized stock $\{s_j\}_j$ (number J), power plant production structure $p(s)$,
1) M price trajectories $\{v^t_\omega\}_{\omega,t}$ with T time steps and power plant outage scenarii,
2) Bellman values on the grid: $\{g^t(v^t_\omega, s)\}_{t,\omega,s}$,
Desired output
1) $g_\omega(s)$: payoff per scenario,
2) $p_\omega(s)$: best production plans for each scenarii (starting form some stock);
for $s \in \{s_1, ..., s_J\}$ **do**
  for $\omega \in \{\omega_1, ..., \omega_M\}$ **do**
    for $t = 1, ..., T$ **do**
      $\hat{h}^t(v, p, s) \stackrel{\text{def}}{=} \sum_\tau p^{t+\tau}(s^{t+\tau})v^{t+\tau} + g^t(v, s) + J(\{p^{t+\tau}(s^{t+\tau})\}_\tau)$ (payoff function);
      $\{p^{t+\tau}_\star(s, v^t_\omega)\}_\tau \in \arg\max_p \hat{h}^t(v, p, s)$ (production profile);
      update $g_\omega(s)$ and stack $p_\omega(s)$;
    end
  end
end
Return: $\{g_\omega(s)\}_{\omega,s}$, $\{p_\omega(s)\}_{s,\omega}$



1.3 Current algorithms used

- Main limitations
  - Not exact calculations of « argmax »;
- Practionners' impact
  - Some « strange » decisions impacting placement of modulations, stock coverage calculation;
  - Mid-term / short-term visions not aligned;

1.3 Current algorithms used

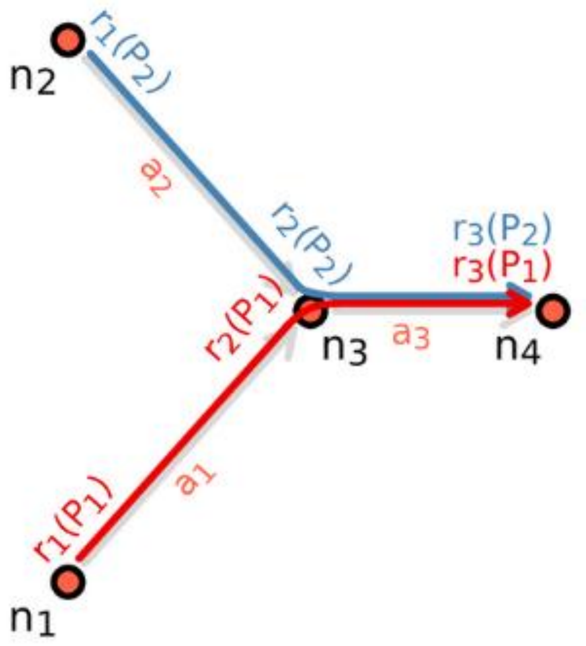
- Main limitations
  - Not exact calculations of « argmax »;

- Practionners' impact
  - Some « strange » decisions impacting placement of modulations, stock coverage calculation;
  - Mid-term / short-term visions not aligned;

- Other attempts
  - DP calculations for sub-problems

1.4 OSAK solver : general principles

Fonctionnement d'OSAK



Evolving resources along two paths

- Notation
  - OSAK solves

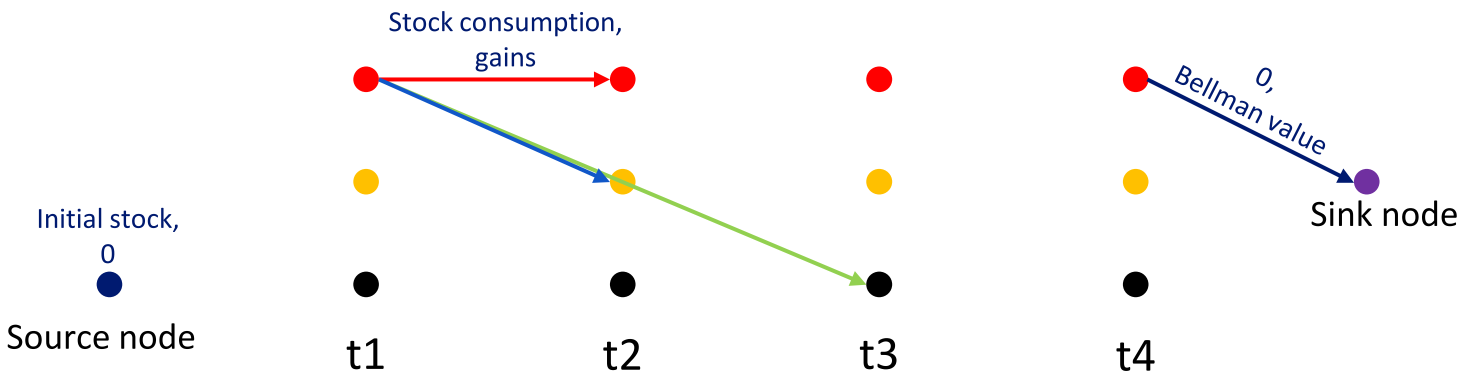
$$\arg\min_{P \in \mathbf{F}} \operatorname{cost}(r(P)).$$

For P a path in an acyclic digraph, with feasible resource *r(P)*, and final cost *cost(r(P))*

  - OSAK uses a dominance rule between the resources partial paths to create a « Pareto front » of paths at each node and eliminate dominate paths
  - Hypothesis:
    - Feasibility and final cost are deacreasing with respect to the resource; i.e. if sub-path *r(P)* dominates *r(Q)* then
      - *Cost(r(P)) < cost(r(Q))*
      - If *r(P)* is infeasible then so is *r(Q)*

Reference: Resource constrained shortest path algorithm for EDF short-term thermal production planning problem, Markus Kruber, Axel Parmentier, Pascal Benchimol

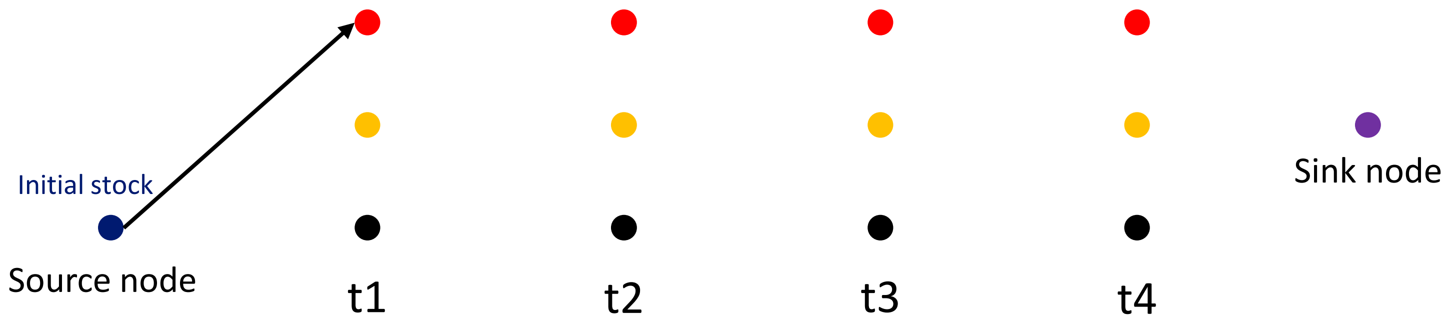
Representing a nuclear plant with OSAK



Representing the problem as a graph:

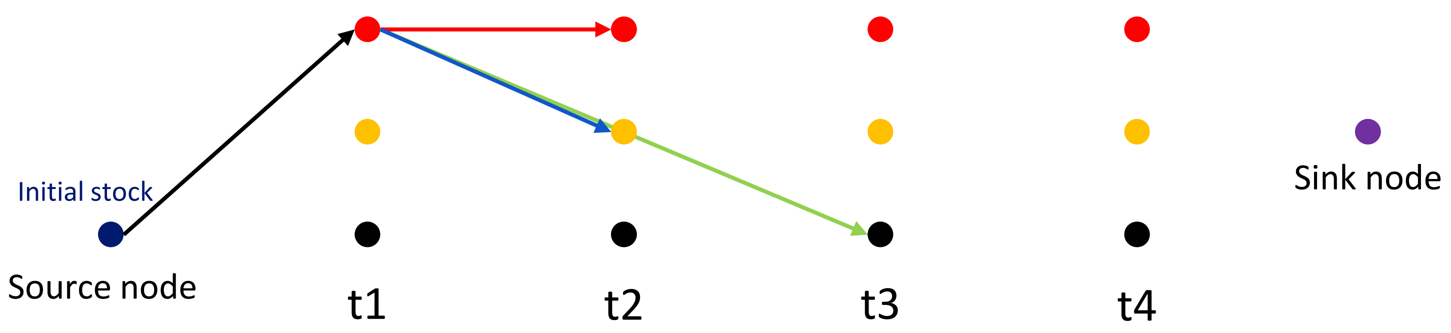
- We start from a given initial stock
- Each node represents a *functionning point* (max power, min power, off) at a given time
- Edges represent a transtion of the power plant production state
  - An edge going to a « higher production » node earns more, but consumes more stock
  - Long edges to off:
    - when the plant turns off, it has to remain so for a minimum duration
- A path represents a feasible production profile for the plant, with its resource (cost, stock) evolving with each edge

Representing a nuclear plant with OSAK



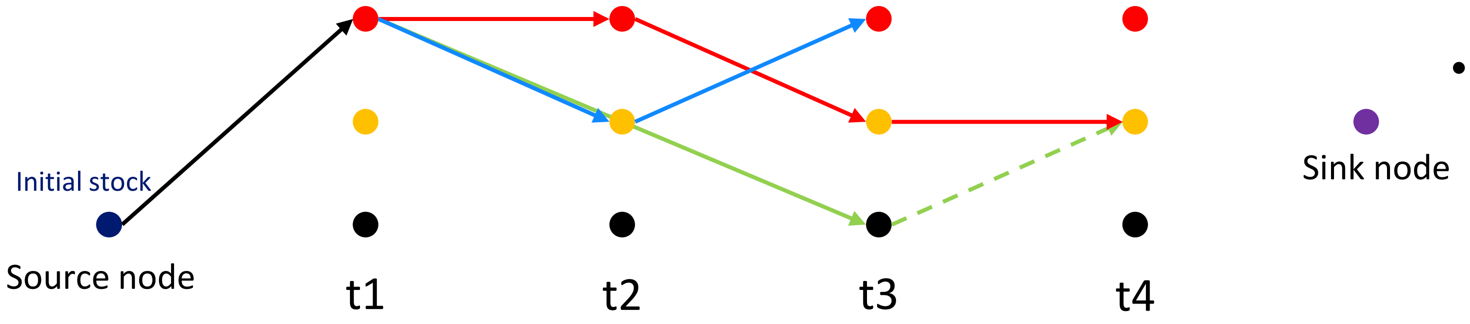
- Select a given sub-path

Representing a nuclear plant with OSAK



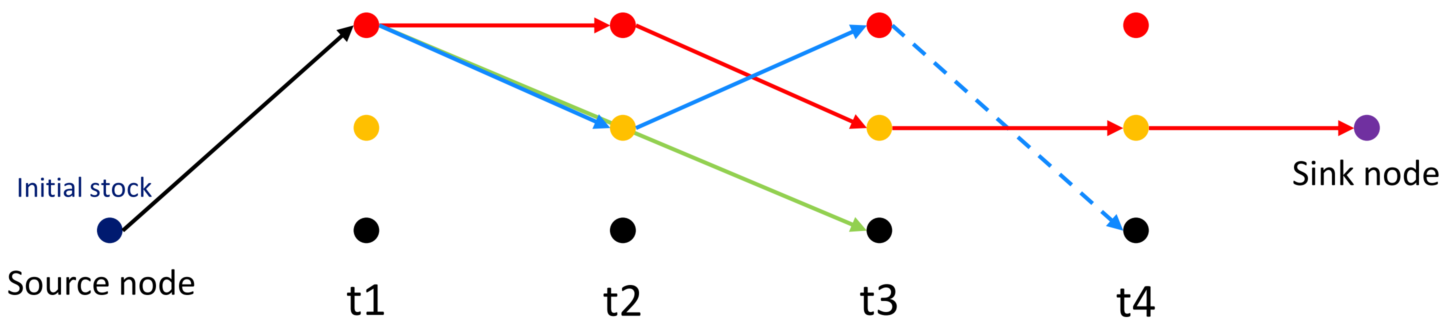
- Select an already constructed sub-path
- Extend it with all possible edges from its ending node
    - Check resource feasibility here
- Add extended sub-paths to the queue

Pruning: dominance



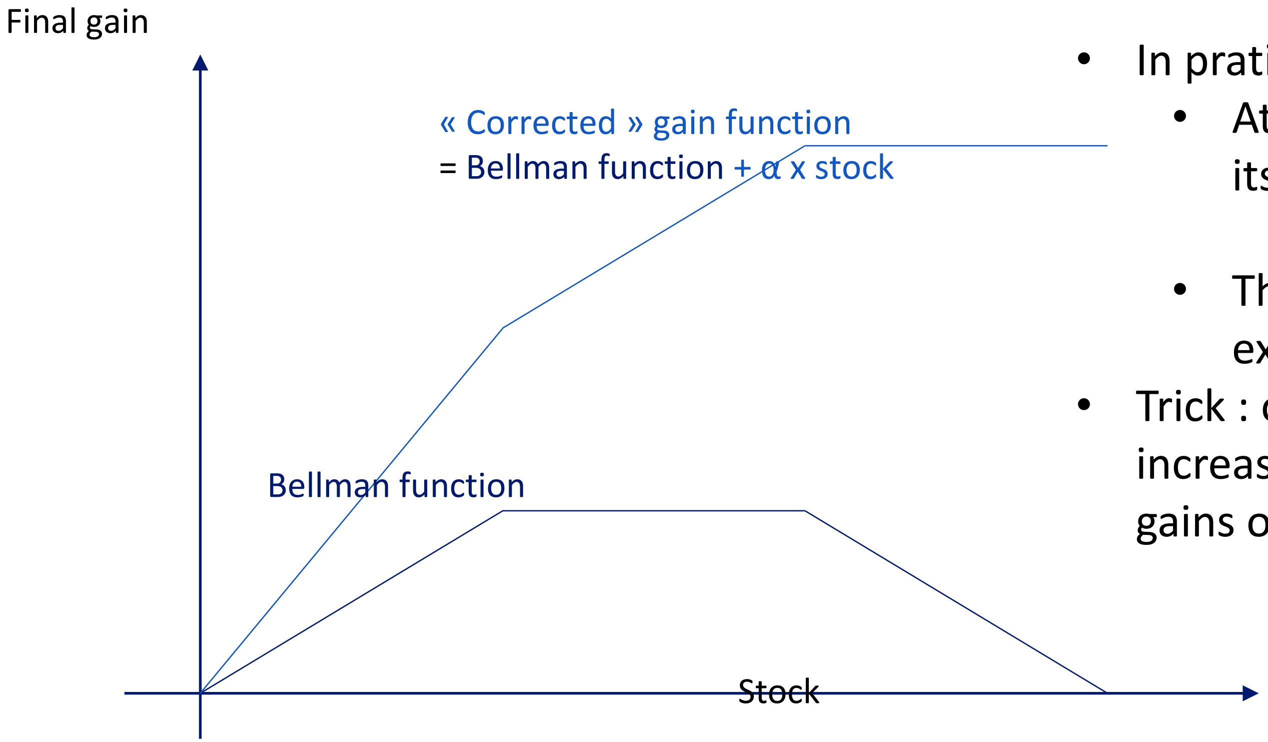
- When two sub-paths end at the same node, compare their resources
- If one *dominates* the other (ex: current gains are higher and current stock is higher)
  - remove the dominated path
  - Otherwise keep both, building a pareto front at that node

Pruning: bounding

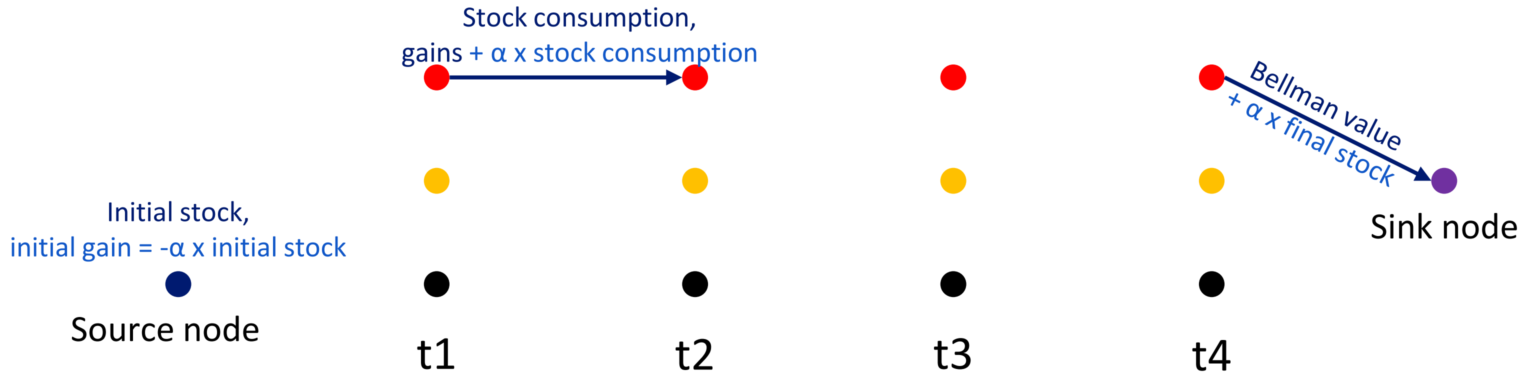


- A path that reached the sink node provides a lower bound to the optimal value
- If we have a relaxed way to extend a sub-path, we can compare it to the lower bound and prune paths that can't beat it
  - Solution : do dynamic programming with poor discretization in pre-processing

A trick for the dominance rule



- In pratice, a higher stock is not always better
  - At the end of the production campaign, a nuclear plant needs its stock to be lower than a threshold
    - Prohibitive penalty added to the final stock
  - That means we loose the domination rule previously explained
- Trick : one can « correct » the final gain function to make it increasing with the stock, and « compensate » in the production gains of edges so that the sum remains the same



Full constraints in the real problem

- We have limited power decreases per day
  - Needs an additional resource, i.e. 3 dimensions : current gains, current stock, and remaining allowed power decrases
- Minimum up time after powering on
  - Needs an additional resource, i.e. 4 dimensions : current gains, current stock, remaining allowed power decrases, and remaining time before the plant can turn back off
- Minimum down time after powering off
  - « Long » edges
- When the stock goes below a certain threshold, only maximum power is possible
- Minimum power and maximum power depend on stock
  - minimum power increases after a point, maximum power decreases afterwards

1.5 Experimental results

What do we compare?

Estimated payoffs (Bellman values) for a nuclear plant,

Usage values and production plans behaviors,

Time calculations,

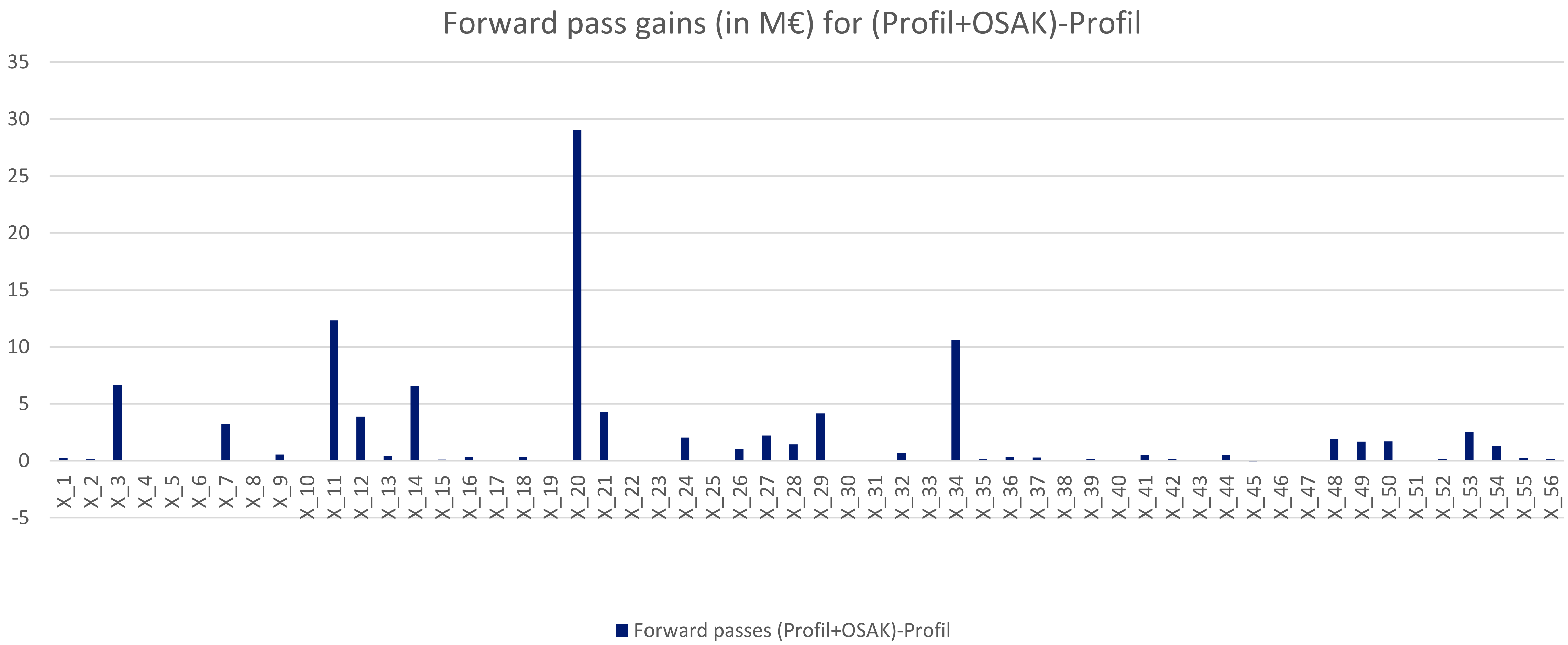
Settings and inputs

- all nuclear plants;
- Stock discretisation: 1 JEPP;
- real production data;

Methods compared

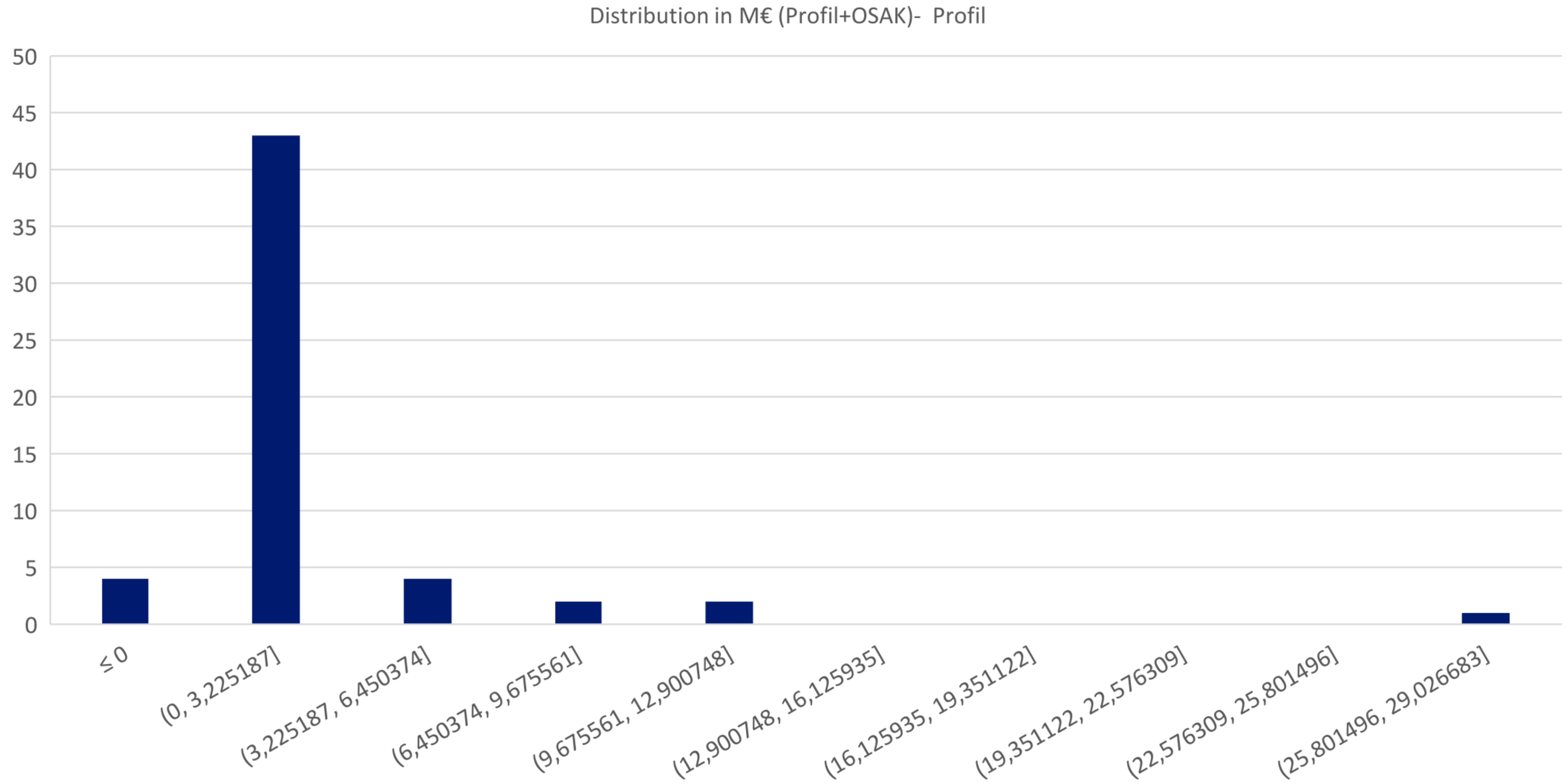
- Reference: « Profil » (backward and forward passes using hardcoded production profils)
- Variant: « Profil+OSAK »: (backward using hard coded profiles; forward pass using OSAK warm started with hard coded production profiles)

1.5 Experimental results



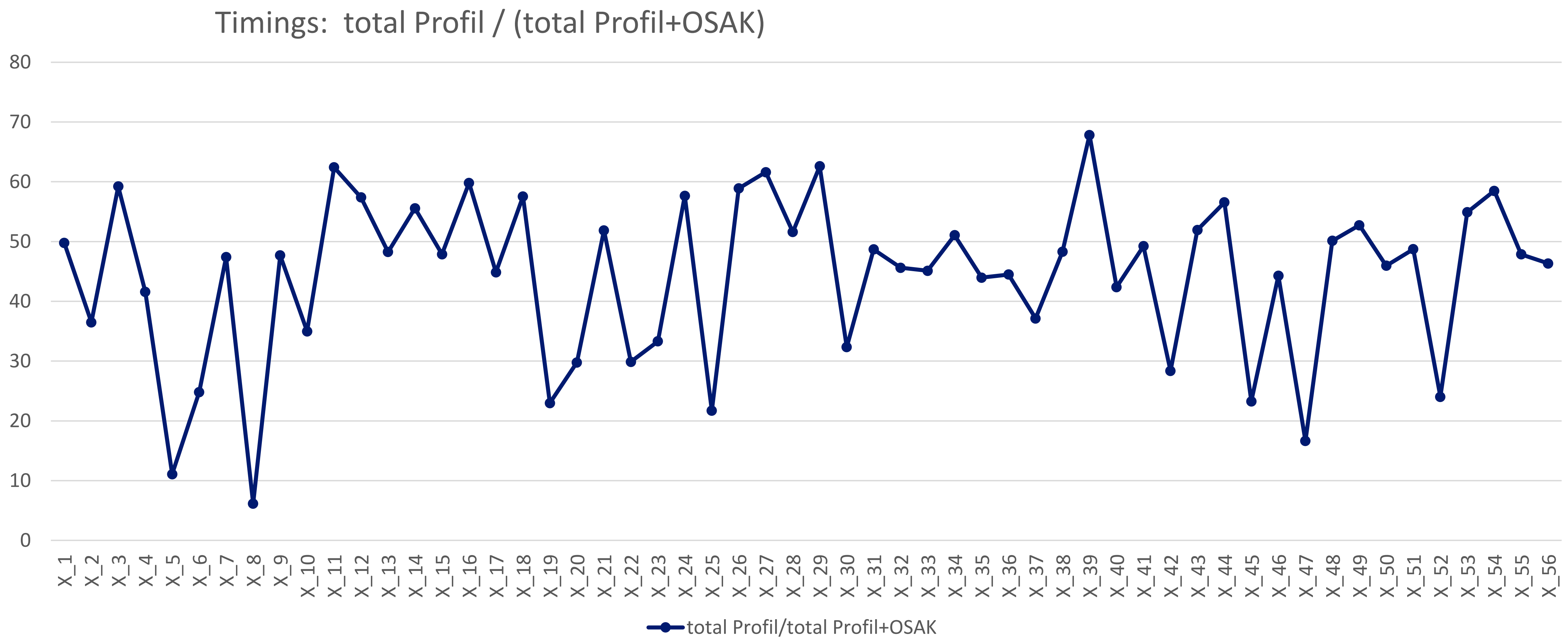
1.5 Experimental results

Principal Mode (78% of gains) : [0, 3M€]
Right hand side distribution (16% of gains) : [3M€, 29M€]
Left hand side distribution (7% of cases): [-0.06M€, 0M€]

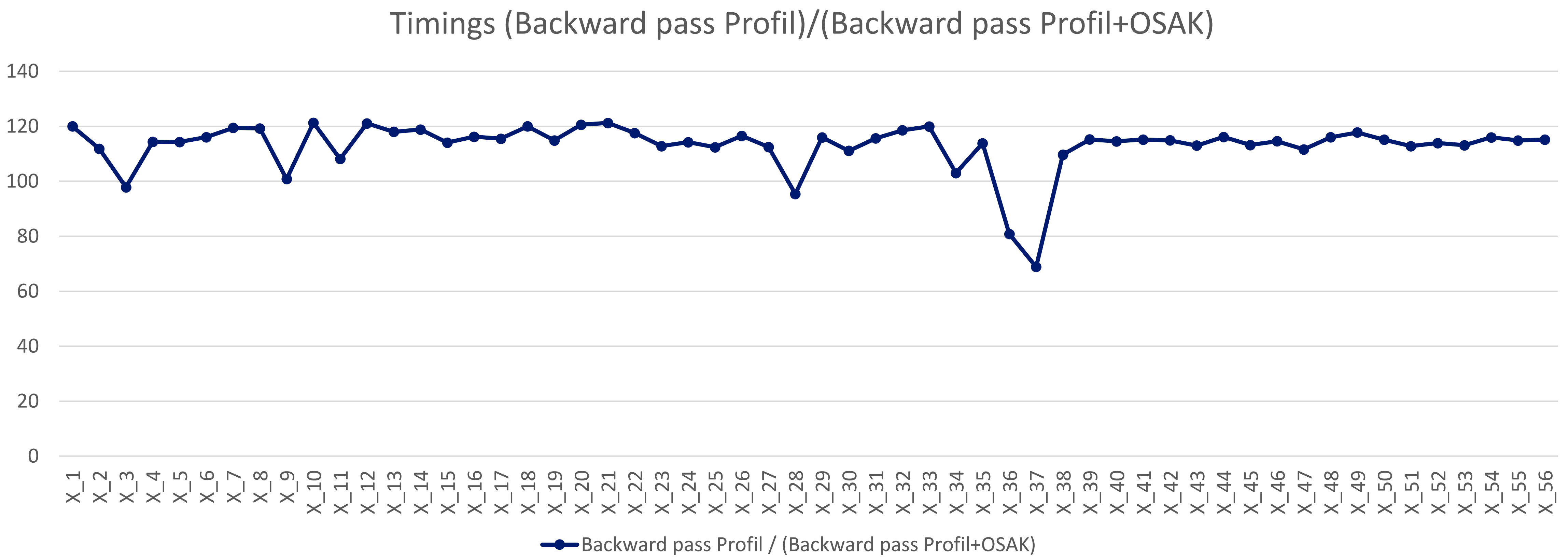


1.5 Experimental results

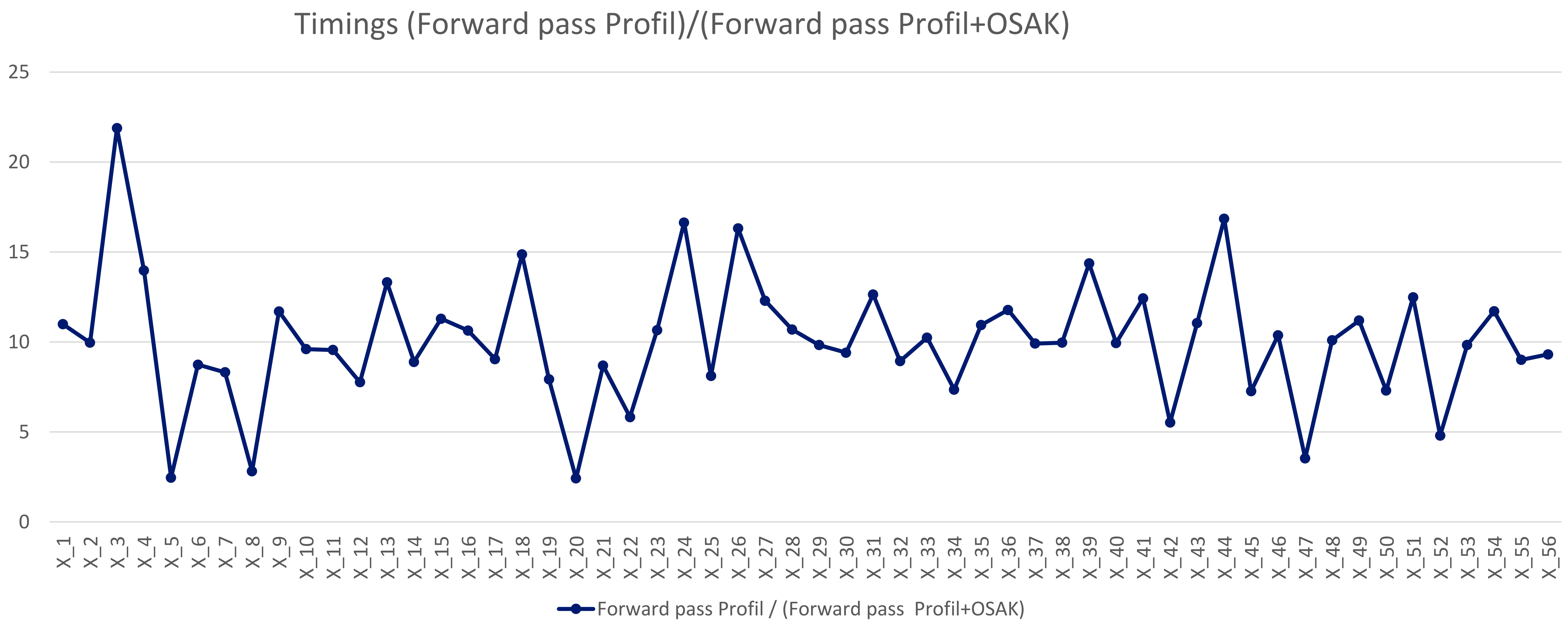
Total time Profil+OSAK : ~6h30
Total time Profil : ~2h00



1.5 Experimental results



1.5 Experimental results



1.6 Conclusions and next steps

Key results

- Better sub-problem resolution provides better decision signals for negligeable supplementary time (about 100M€ for extra 4h compute)
- Results can be explained and production profiles generated can be used;
- Trade-off between timing and gains (ongoing calibration)

Next steps

- Ongoing industrialization
- Solution to be presented to « Grand Trophée de la R&D »
- Optimize the backward pass
- Other formulations ?

2.1 Problem formulation

Unit commitment problem

We wish to solve

$$\min_{p_{i,t},\, t \in T,\, i \in U} \sum_{i \in U, t \in T} C_{i,t}(p_{i,t}),$$

$$\text{tel que } p_{i,t} \in X_i,$$

$$\underbrace{\textstyle\sum_{i \in U} p_{i,t} = d_t}.$$

Cost function (power plant i, time step t)

Operational constraint (power plant i, time step t)

Offer demand equilibrium at time step t

Notations

- U: power plant set
- T: time step set (over 2 days)
- p: power vector (dimension 5)
- d: demand

2.1 Problem formulation

Unit commitment problem

We wish to solve

$$\min_{p_{i,t},\, t\in T,\, i\in U} \sum_{t\in T} \overbrace{\hat{C}_t\left(d_t - \textstyle\sum_{i\in U} p_{i,t}\right)}^{\text{Failure cost at time step t}} + \sum_{i\in U, t\in T} \underbrace{C_{i,t}(p_{i,t})}_{\text{Operational cost (power plant i, time step t)}},$$

$$\text{tel que } \underbrace{p_{i,t} \in X_i}_{\text{Operational constraint (power plant i, time step t)}},$$

Notations

- U: power plant set
- T: time step set (over 2 days)
- p: power vector (dimension 5)
- d: demand

2.1 Problem formulation

Unit commitment problem

We wish to solve

$$\min_{p_{i,t},\, t\in T,\, i\in U} \sum_{t\in T} \underbrace{\hat{C}_t(d_t - \textstyle\sum_{i\in U} p_{i,t})}_{\text{Failure cost at time step t}} + \sum_{i\in U, t\in T} \boxed{C_{i,t}(p_{i,t})},$$

$$\text{tel que } \underbrace{p_{i,t} \in X_i}_{\text{Operational constraint (power plant i, time step t)}},$$

Operational cost (power plant i, time step t)

Algorithm « Apogène »

1. Phase 1
   1. Bundle method
   2. price signal
2. Phase 2
   1. Augmented Lagrangian
   2. First solutions
3. Phase 3
   1. Genetic algorithms
   2. Solution polishing

eDF

2.1 Problem formulation

Unit commitment problem

We wish to solve

$$\min_{p_{i,t},\, t\in T,\, i\in U} \sum_{t\in T} \overbrace{\hat{C}_t(d_t - \textstyle\sum_{i\in U} p_{i,t})}^{\text{Failure cost at time step t}} + \sum_{i\in U, t\in T} C_{i,t}(p_{i,t}),$$

$$\text{tel que } \underbrace{p_{i,t} \in X_i}_{\text{Operational constraint (power plant i, time step t)}},$$

Operational cost (power plant i, time step t)

Algorithm « Apogène »

1. Phase 1
   1. Bundle method
   2. price signal
2. Phase 2
   1. Augmented Lagrangian
   2. First solutions
3. Phase 3
   1. Genetic algorithms
   2. Solution polishing

2.1 Problem formulation

Unit commitment problem

We wish to solve

$$\min_{p_{i,t},\; t\in T,\; i\in U} \sum_{t\in T} \overbrace{\hat{C}_t\Big(d_t - \textstyle\sum_{i\in U} p_{i,t}\Big)}^{\text{Failure cost at time step t}} + \sum_{i\in U, t\in T} C_{i,t}(p_{i,t}),$$

$$\text{tel que } \underbrace{p_{i,t} \in X_i}_{\text{Operational constraint (power plant i, time step t)}},$$

Operational cost (power plant i, time step t)

Frontal solve intractable

2.1 Problem formulation

General principle

Failure cost at time step t

$$\min_{p_{i,t},\, t\in T,\, i\in U} \sum_{t\in T} \boxed{\hat{C}_t(d_t - \textstyle\sum_{i\in U} p_{i,t})} + \sum_{i\in U, t\in T} \boxed{C_{i,t}(p_{i,t})},$$

$$\text{tel que } \boxed{p_{i,t} \in X_i},$$

Operational constraint (power plant i, time step t)

Operational cost (power plant i, time step t)

Frontal solve intractable

Use of a « *decomposition* » *technique*

2.2 Current algorithm used

Phase 2 algorithm

$$\hat{p} \in \underset{\hat{p}_{i,t} \in \hat{X}_i}{\arg\min} \sum_{t \in T} \hat{C}_t\left(d_t - \textstyle\sum_{i \in U} \hat{p}_{i,t}\right),$$

$$p \in \underset{p_{i,t} \in \bar{X}_i}{\arg\min} \sum_{i \in U, t \in T} C_{i,t}(p_{i,t}).$$

We would like to find solution such that $p = \hat{p}$

Notations

- $\hat{X}_i$: a *split* of constraint X_i
- $\bar{X}_i$:*complementary split* of X_i

2.2 Current algorithm used

Phase 2 algorithm

Proximal Jacobian ADMM

$$\hat{p}^{k+1} \in \underset{\hat{p}_{i,t} \in \hat{X}_i}{\arg\min} \sum_{t \in T} \hat{C}_t(d_t - \textstyle\sum_{i \in U} \hat{p}_{i,t}) + \frac{\rho}{2}\|\hat{p}_t - p_t^k - \frac{\mu_t^k}{\rho}\|_2^2 + \frac{K}{2}\|\hat{p}_t - \hat{p}_t^k\|_2^2,$$

$$p^{k+1} \in \underset{p_{i,t} \in \bar{X}_i}{\arg\min} \sum_{i \in U, t \in T} C_{i,t}(p_{i,t}) + \frac{\rho}{2}\|p_{i,t} - \hat{p}_{i,t}^k + \frac{\mu_{i,t}^k}{\rho}\|_2^2 + \frac{K}{2}\|p_{i,t} - p_{i,t}^k\|_2^2,$$

$$\mu^{k+1} = \mu^k + \tau\rho(p^{k+1} - \hat{p}^{k+1}),$$

Notations

- $\hat{X}_i$: a *split* of constraint X_i
- $\bar{X}_i$:*complementary split* of X_i

2.2 Current algorithm used

Phase 2 algorithm

Proximal Jacobian ADMM

$$\hat{p}^{k+1} \in \underset{\hat{p}_{i,t} \in \hat{X}_i}{\arg\min} \sum_{t \in T} \hat{C}_t(d_t - \textstyle\sum_{i \in U} \hat{p}_{i,t}) + \frac{\rho}{2}\|\hat{p}_t - p_t^k - \frac{\mu_t^k}{\rho}\|_2^2 + \frac{K}{2}\|\hat{p}_t - \hat{p}_t^k\|_2^2,$$

$$p^{k+1} \in \underset{p_{i,t} \in \bar{X}_i}{\arg\min} \sum_{i \in U, t \in T} C_{i,t}(p_{i,t}) + \frac{\rho}{2}\|p_{i,t} - \hat{p}_{i,t}^k + \frac{\mu_{i,t}^k}{\rho}\|_2^2 + \frac{K}{2}\|p_{i,t} - p_{i,t}^k\|_2^2,$$

$$\mu^{k+1} = \mu^k + \tau\rho(p^{k+1} - \hat{p}^{k+1}),$$

Price signal estimate

Notations

- $\hat{X}_i$: a *split* of constraint X_i
- $\bar{X}_i$:*complementary split* of X_i

2.2 Current algorithm used

Phase 2 algorithm

Proximal Jacobian ADMM

$$\hat{p}^{k+1} \in \underset{\hat{p}_{i,t} \in \hat{X}_i}{\arg\min} \sum_{t \in T} \hat{C}_t(d_t - \textstyle\sum_{i \in U} \hat{p}_{i,t}) + \frac{\rho}{2}\|\hat{p}_t - p_t^k - \frac{\mu_t^k}{\rho}\|_2^2 + \frac{K}{2}\|\hat{p}_t - \hat{p}_t^k\|_2^2,$$

$$p^{k+1} \in \underset{p_{i,t} \in \bar{X}_i}{\arg\min} \sum_{i \in U, t \in T} C_{i,t}(p_{i,t}) + \frac{\rho}{2}\|p_{i,t} - \hat{p}_{i,t}^k + \frac{\mu_{i,t}^k}{\rho}\|_2^2 + \frac{K}{2}\|p_{i,t} - p_{i,t}^k\|_2^2,$$

$$\mu^{k+1} = \mu^k + \tau\rho(p^{k+1} - \hat{p}^{k+1}),$$

Price signal estimate

Parameters

- **τ** : *damping term*
- **ρ** : penalization term
- K : proximal term

Notations

- $\hat{X}_i$: a *split* of constraint X_i
- $\bar{X}_i$:*complementary split* of X_i

2.2 Current algorithm used

Phase 2 algorithm

Proximal Jacobian ADMM

$$\left.\begin{aligned}\hat{p}^{k+1} &\in \underset{\hat{p}_{i,t}\in\hat{X}_i}{\arg\min}\; \hat{C}_t\big(d_t - \textstyle\sum_{i\in U}\hat{p}_{i,t}\big) + \frac{\rho}{2}\|\hat{p}_t - p_t^k - \frac{\mu_t^k}{\rho}\|_2^2 + \frac{K}{2}\|\hat{p}_t - \hat{p}_t^k\|_2^2,\\ p^{k+1} &\in \underset{p_{i,t}\in\bar{X}_i}{\arg\min}\; \sum_{t\in T} C_{i,t}(p_{i,t}) + \frac{\rho}{2}\|p_{i,t} - \hat{p}_{i,t}^k + \frac{\mu_{i,t}^k}{\rho}\|_2^2 + \frac{K}{2}\|p_{i,t} - p_{i,t}^k\|_2^2,\end{aligned}\right\}$$

Fully distributed version

$$\mu^{k+1} = \mu^k + \tau\rho(p^{k+1} - \hat{p}^{k+1}),$$

Price signal estimate

Parameters

- τ : *damping term*
- ρ : penalization term
- K : proximal term

Notations

- $\hat{X}_i$: a *split* of constraint X_i
- $\bar{X}_i$:*complementary split* of X_i

2.2 Current algorithm used

Phase 2 algorithm

Proximal Jacobian ADMM

$$\hat{p}^{k+1} \in \underset{\hat{p}_{d,t},\ \hat{p}_t \in \hat{X}_i}{\arg\min}\ \hat{C}_t(\hat{p}_{d,t}) + \frac{\rho}{2}\|\hat{p}_t - p_t^k - \frac{\mu_t^k}{\rho}\|_2^2 + \frac{K}{2}\|\hat{p}_t - \hat{p}_t^k\|_2^2,$$

$$p^{k+1} \in \underset{p_i \in \bar{X}_i}{\arg\min} \sum_{t\in T} C_{i,t}(p_{i,t}) + \frac{\rho}{2}\|p_{i,t} - \hat{p}_{i,t}^k + \frac{\mu_{i,t}^k}{\rho}\|_2^2 + \frac{K}{2}\|p_{i,t} - p_{i,t}^k\|_2^2,$$

Fully distributed version

$$\mu^{k+1} = \mu^k + \tau\rho(p^{k+1} - \hat{p}^{k+1}),$$

Price signal estimate

Parameters

- **τ** : *damping term*
- **ρ** : penalization term
- K : proximal term

Notations

- $\hat{X}_i$: a *split* of constraint X_i
- $\bar{X}_i$:*complementary split* of X_i

2.2 Current algorithm used

Phase 2 algorithm

Proximal Jacobian ADMM

$$\hat{p}^{k+1} \in \underset{\hat{p}_{d,t},\ \hat{p}_t \in \hat{X}_i}{\arg\min} \ \hat{C}_t(\hat{p}_{d,t}) + \frac{\rho}{2}\|\hat{p}_t - p_t^k - \frac{\mu_t^k}{\rho}\|_2^2 + \frac{K}{2}\|\hat{p}_t - \hat{p}_t^k\|_2^2,$$

$$p^{k+1} \in \underset{p_i \in \bar{X}_i}{\arg\min} \sum_{t\in T} C_{i,t}(p_{i,t}) + \frac{\rho}{2}\|p_{i,t} - \hat{p}_{i,t}^k + \frac{\mu_{i,t}^k}{\rho}\|_2^2 + \frac{K}{2}\|p_{i,t} - p_{i,t}^k\|_2^2,$$

Fully distributed version

$$\mu^{k+1} = \mu^k + \tau\rho(p^{k+1} - \hat{p}^{k+1}),$$

Price signal estimate

Parameters

- τ : *damping term*
- ρ : penalization term
- K : proximal term

Convergence conditions

- $0 < \tau \leq 1$
- $0 < \rho < K$

Notations

- $\hat{X}_i$: a *split* of constraint X_i
- $\bar{X}_i$:*complementary split* of X_i

2.2 Current algorithm used

Phase 2 algorithm

Proximal Jacobian ADMM

$$\hat{p}^{k+1} \in \underset{\hat{p}_{d,t},\ \hat{p}_t \in \hat{X}_i}{\arg\min} \quad \hat{C}_t(\hat{p}_{d,t}) + \frac{\rho}{2}\|\hat{p}_t - p_t^k - \frac{\mu_t^k}{\rho}\|_2^2 + \frac{K}{2}\|\hat{p}_t - \hat{p}_t^k\|_2^2,$$

$$p^{k+1} \in \underset{p_i \in \bar{X}_i}{\arg\min} \sum_{t \in T} C_{i,t}(p_{i,t}) + \frac{\rho}{2}\|p_{i,t} - \hat{p}_{i,t}^k + \frac{\mu_{i,t}^k}{\rho}\|_2^2 + \frac{K}{2}\|p_{i,t} - p_{i,t}^k\|_2^2,$$

Fully distributed version

$$\mu^{k+1} = \mu^k + \tau\rho(p^{k+1} - \hat{p}^{k+1}),$$

Price signal estimate

Parameterization used

- τ = 1 : no *damping*
- ρ = K/2 : penalization term
- K > 0 : proximal term

Convergence conditions

- $0 < \tau \leq 1$
- $0 < \rho < K$

Notations

- $\hat{X}_i$: a *split* of constraint X_i
- $\bar{X}_i$:*complementary split* of X_i

2.3 Some algorithmic improvements

Variations experimented

- Automatic calibration techniques
- Acceleration techniques

2.3 Some algorithmic improvements

Automatic calibration techniques

Current calibration

- $\boldsymbol{\rho}$ et K a strictly increasing interpolation;
- $\boldsymbol{\rho} = K / 2$;
- $\boldsymbol{\tau} = 1$;

2.3 Some algorithmic improvements

Automatic calibration techniques

Current calibration

- $\boldsymbol{\rho}$ et K a strictly increasing interpolation;
- $\boldsymbol{\rho}$ = K / 2;
- $\boldsymbol{\tau}$ = 1;

« *vanilla* » Variation:

- $\boldsymbol{\rho}$ = K / $\boldsymbol{\alpha}$ with $\boldsymbol{\alpha}$ strictly greater than 2;
- $\boldsymbol{\tau}$ = 0.95;
- K another interpolation with slower variation;

2.3 Some algorithmic improvements

Automatic calibration techniques

Vanilla « automatic » technic (inspired from Method of Multipliers based algorithms):

- $\boldsymbol{\rho}$ = K / $\boldsymbol{\alpha}$ with $\boldsymbol{\alpha}$ strictly greater than 2;
- $\boldsymbol{\tau}$ = 0.95;
- K another interpolation with slower variation;
- If $\| \mathbf{p}^{k+1} - \hat{\mathbf{p}}^{k+1} \| \geq 0.95 \| \mathbf{p}^{k} - \hat{\mathbf{p}}^{k} \|$ (at some frequency):
  - Incrase $\boldsymbol{\rho}$ (we don’t penalize enough the constraint)

References:

Birgin, E. G., & Martínez, J. M. (2014). *Practical augmented Lagrangian methods for constrained optimization*. Society for Industrial and Applied Mathematics.

2.3 Some algorithmic improvements

Automatic calibration techniques

More advanced technic (OSQP like):

- $\boldsymbol{\rho}$ = K / $\boldsymbol{\alpha}$ with $\boldsymbol{\alpha}$ strictly greater than 2;
- $\boldsymbol{\tau}$ = 0.95;
- K another interpolation with slower variation;
- If primal residual larger than dual one (at some frequency):
  - Incrase $\boldsymbol{\rho}$ (dual converges too quickly)
- If dual residual larger than primal one (at some frequency):
  - Decrease $\boldsymbol{\rho}$ (primal converges too quickly)

References:

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., & Boyd, S. (2020). OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, *12*(4), 637-672.

2.3 Some algorithmic improvements

Acceleration techniques

General principle

ADMM based algorithms can be accelerated

Tested technics:

- Over-Relaxation
- Anderson acceleration

2.3 Some algorithmic improvements

Acceleration techniques

General principle

ADMM based algorithms can be accelerated

Tested technics:

- Over-Relaxation
- Anderson acceleration

$$\hat{p}^{k+1/2} \in \operatorname*{arg\,min}_{\hat{p}_{d,t},\ \hat{p}_t \in \hat{X}_i} \hat{C}_t(\hat{p}_{d,t}) + \frac{\rho}{2}\|\hat{p}_t - p_t^k - \frac{\mu_t^k}{\rho}\|_2^2 + \frac{K}{2}\|\hat{p}_t - \hat{p}_t^k\|_2^2,$$

$$p^{k+1} \in \operatorname*{arg\,min}_{p_i \in \bar{X}_i} \sum_{t \in T} C_{i,t}(p_{i,t}) + \frac{\rho}{2}\|p_{i,t} - \hat{p}_{i,t}^k + \frac{\mu_{i,t}^k}{\rho}\|_2^2 + \frac{K}{2}\|p_{i,t} - p_{i,t}^k\|_2^2,$$

$$\mu^{k+1} = \mu^k + \tau\rho(p^{k+1} - \alpha\hat{p}^{k+1/2} - (1-\alpha)\hat{p}^k),$$

$$\hat{p}^{k+1} = \alpha\hat{p}^{k+1/2} + (1-\alpha)\hat{p}^k.$$

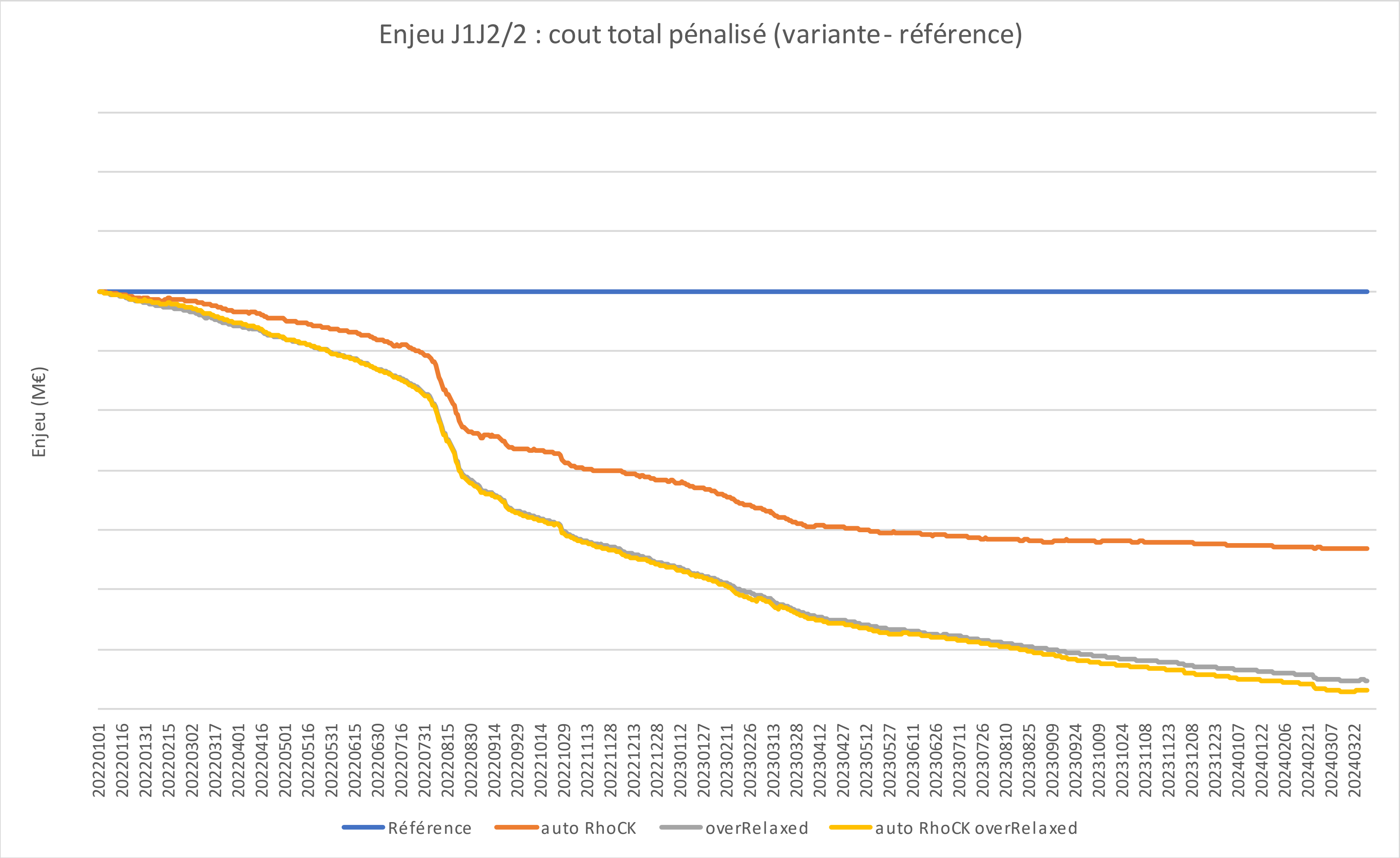
Remark:
ADMM converges for $0 < \alpha < 2$

- $\alpha > 1$: *over-relaxation*
- $\alpha < 1$: *under-relaxation*

It has been observed that for $\alpha > 1$ it goes quicker (especially for $\alpha=1.6$)

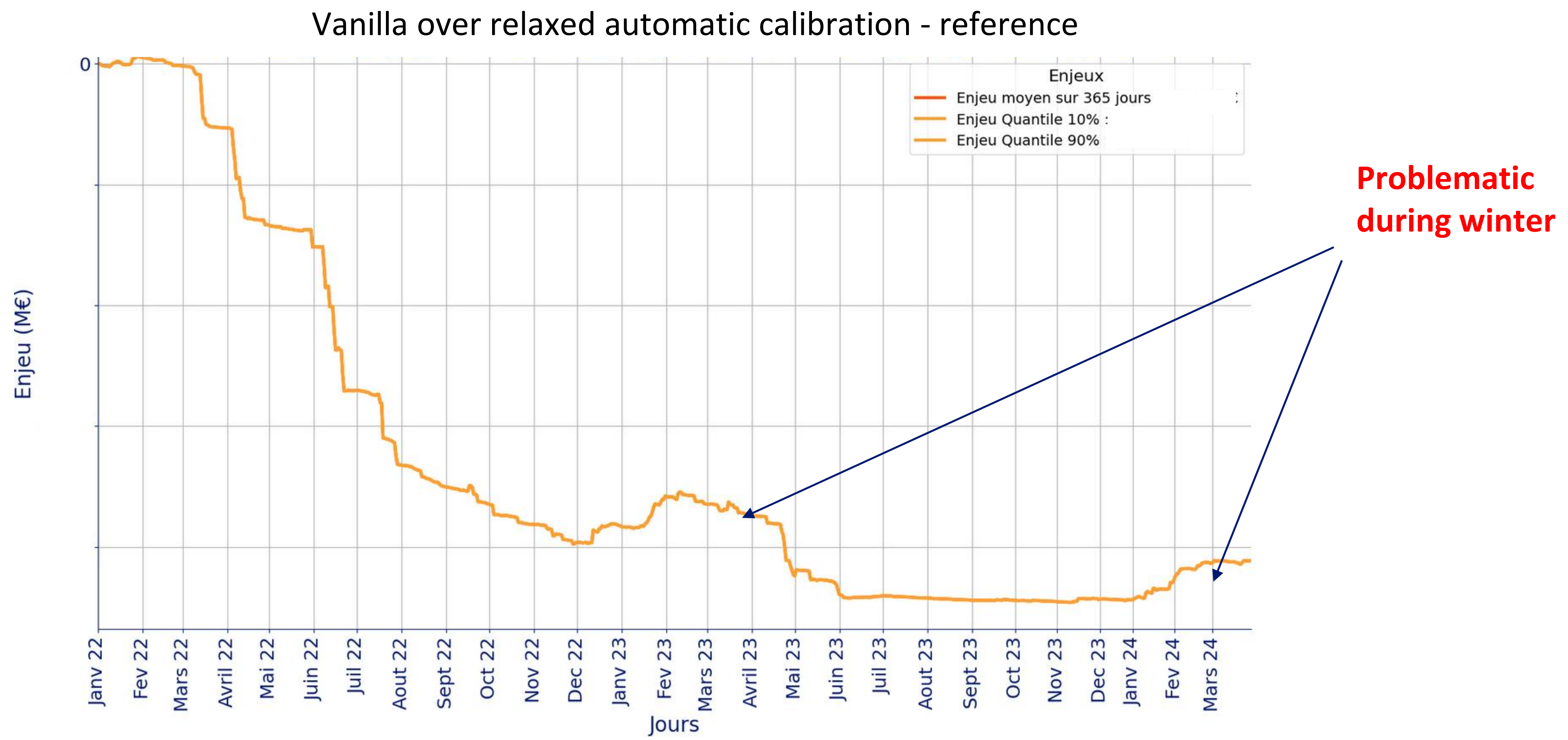
2.4 Experimental results

Time discretization: 30 minutes



2.4 Experimental results

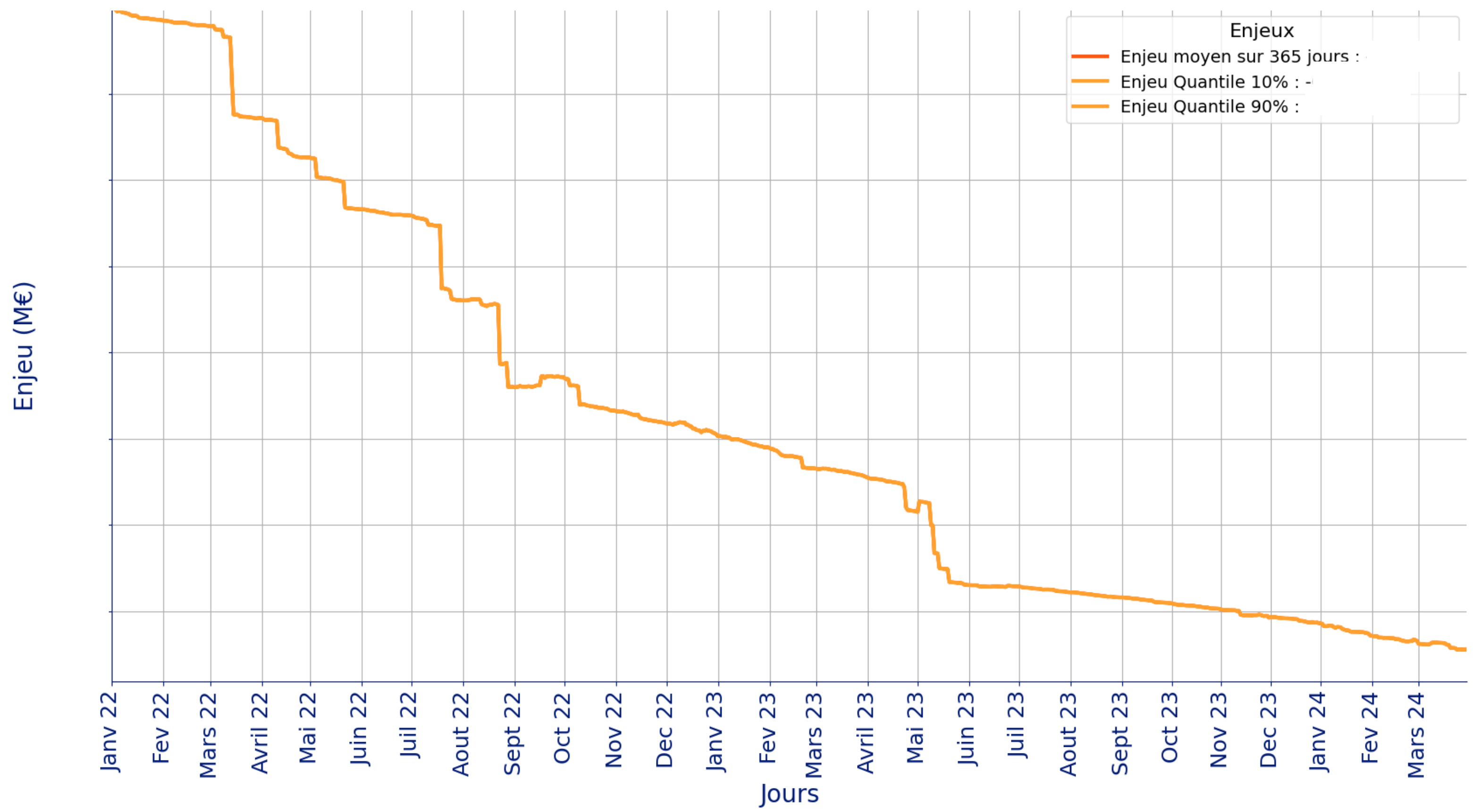
Time discretization: 15 minutes



2.4 Experimental results

Time discretization: 15 minutes

More advanced relaxed automatic calibration - reference



2.6 Conclusions

Key results

- ADMM based algorithms are very powerful;
- They are very sensitive to the step size calibration;
- New improvements integrated into production;

Merci