A Projective Operator Splitting Approach to Stochastic Programming

Jonathan Eckstein
Rutgers University, New Jersey, USA

Portions of the work here are joint with

Benar F. Svaiter – IMPA, Rio de Janeiro
Patrick Combettes – North Carolina State University
Jean-Paul Watson, Lawrence Livermore National Lab
David L. Woodruff, University of California, Davis

RUTGERS

RUTGERS
Rutgers Business School
Newark and New Brunswick

Ronald E. Bruck

- I didn’t know Ronald Bruck
- But I cited 5 of his pioneering papers in my dissertation
    - Evaluating resolvents of monotone set-valued operators (1973)
    - Extragradient methods for set-valued monotone operators (1974)
    - Steepest-descent paths for nonsmooth functions (1975)
    - Forward-backward splitting with set-valued monotone operators 1975, 1977
- I was still in high school when most of these were published

Firm Nonexpansiveness

- Firm nonexpansiveness (½-averagedness) of operators is a key tool in proving convergence all proximal algorithms
  - The proximal point algorithm / Krasnoselski-Mann iteration
  - Douglas-Rachford splitting
  - ADMM
  - Etc.
- If J is the algorithmic map, firm nonexpansiveness means

$$\left(\forall x, x'\right) \quad \left\|J(x) - J(x')\right\|^2 \leq \left\|x - x'\right\|^2 - \left\|(\mathrm{Id} - J)(x) - (\mathrm{Id} - J)(x')\right\|^2$$

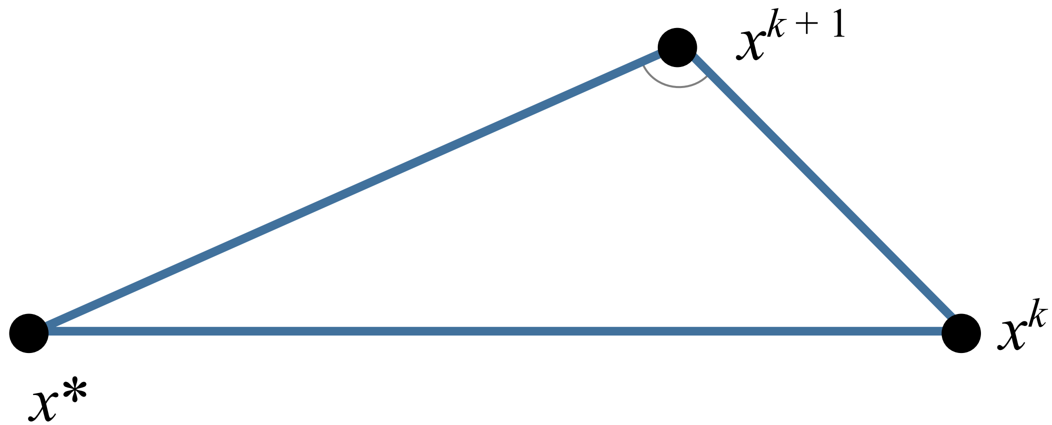
- If we let $x^{k+1} = J(x^k)$ and x^* is any solution / fixed point of J,

$$\left\|x^{k+1} - x^*\right\|^2 \leq \left\|x - x^*\right\|^2 - \left\|x^k - x^{k+1}\right\|^2$$

A Picture

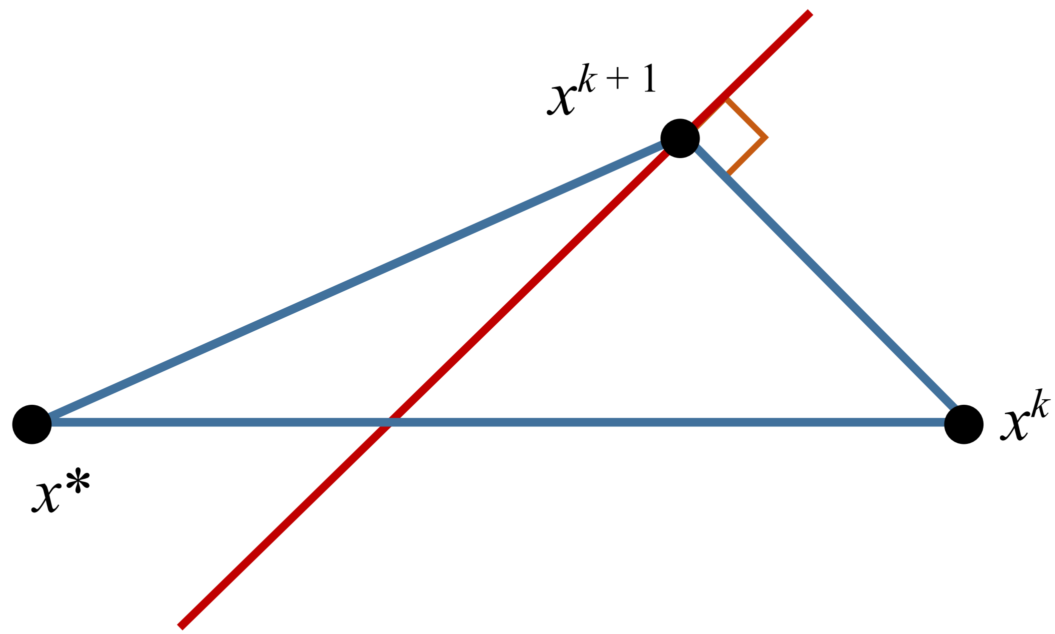
- Rewrite as

$$\left\|x^{k+1} - x^*\right\|^2 + \left\|x^k - x^{k+1}\right\|^2 \leq \left\|x - x^*\right\|^2$$



- The angle between $x^k - x^{k+1}$ and $x^* - x^{k+1}$ is at least 90º

Firm Nonexpansiveness $\Rightarrow$ Projection



- This means that x^{k+1} is the projection of x^k onto the halfspace

$$H_k = \left\{ x \in \mathcal{H} \;\middle|\; \left\langle x - x^{k+1}, x^k - x^{k+1} \right\rangle \leq 0 \right\} ,$$

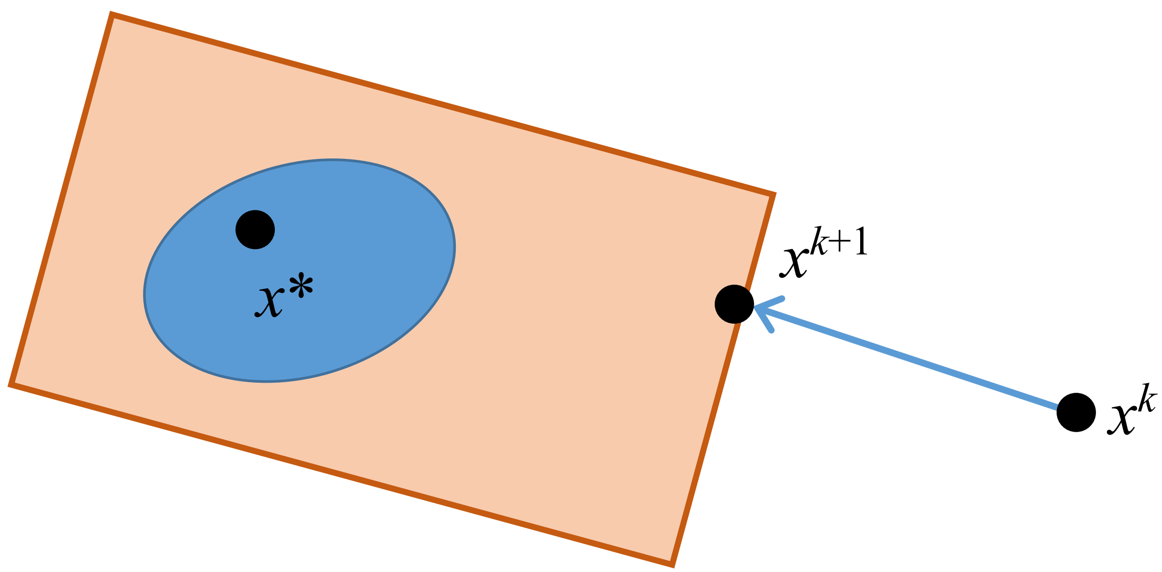
which must contain x^*

Firm Nonexpansiveness = Projection

- Consider any algorithm of the form

$$x^{k+1} = \text{proj}_{S_k}(x^k) \ ,$$

where S_k is a closed convex set containing all possible solutions x^*



- Any such process has exactly the same property,

$$\left\| x^{k+1} - x^* \right\|^2 \leq \left\| x - x^* \right\|^2 - \left\| x^k - x^{k+1} \right\|^2$$

Firm Nonexpansiveness = Projection

- Firmly nonexpansive maps can always be interpreted as projection
- Any projection algorithm looks firmly nonexpansive

- This insight can be used to construct and modify a wide range of algorithms

- With a little care, the same insight can be extended to any process with property, for some $\beta > 0$,

$$\left\| x^{k+1} - x^* \right\|^2 \le \left\| x - x^* \right\|^2 - \beta \left\| x^k - x^{k+1} \right\|^2$$

  (covers α-averaged operators for $\alpha \ne \frac{1}{2}$)

- Equivalent to over- or under-relaxed projection onto a separating hyperplane

General Problem Setting

Consider monotone inclusion problems of the form

$$0 \in \sum_{i=1}^{n} G_i^* T_i(G_i x)$$

where

- $\mathcal{H}_0, \ldots, \mathcal{H}_n$ are real Hilbert spaces
- $T_i : \mathcal{H}_i \rightrightarrows \mathcal{H}_i$ are maximal monotone operators, $i = 1, \ldots, n$
- $G_i : \mathcal{H}_0 \rightrightarrows \mathcal{H}_i$ are bounded linear maps, $i = 1, \ldots, n$

Generalizes

$$\min_{x \in \mathcal{H}_0} \left\{ \sum_{i=1}^{n} f_i(G_i x) \right\}$$

The Primal-Dual Solution Set (Kuhn-Tucker Set)

$$\mathcal{S} = \left\{ (z, w_1, \ldots, w_n) \;\middle|\; (\forall i = 1, \ldots n)\; w_i \in T_i(G_i z),\; \sum_{i=1}^{n} G_i^* w_i = 0 \right\}$$

Or, if we assume that $\mathcal{H}_n = \mathcal{H}_0, G_n = \mathrm{Id}$,

$$\mathcal{S} = \left\{ (z, w_1, \ldots, w_{n-1}) \;\middle|\; (\forall i = 1, \ldots n-1)\; w_i \in T_i(G_i z),\; -\sum_{i=1}^{n-1} G_i^* w_i \in T_n(z) \right\}$$

- This is the set of points satisfying the optimality conditions
- Standing assumption: $\mathcal{S}$ is nonempty
- Essentially in E & Svaiter 2009:

$\mathcal{S}$ is a closed convex set

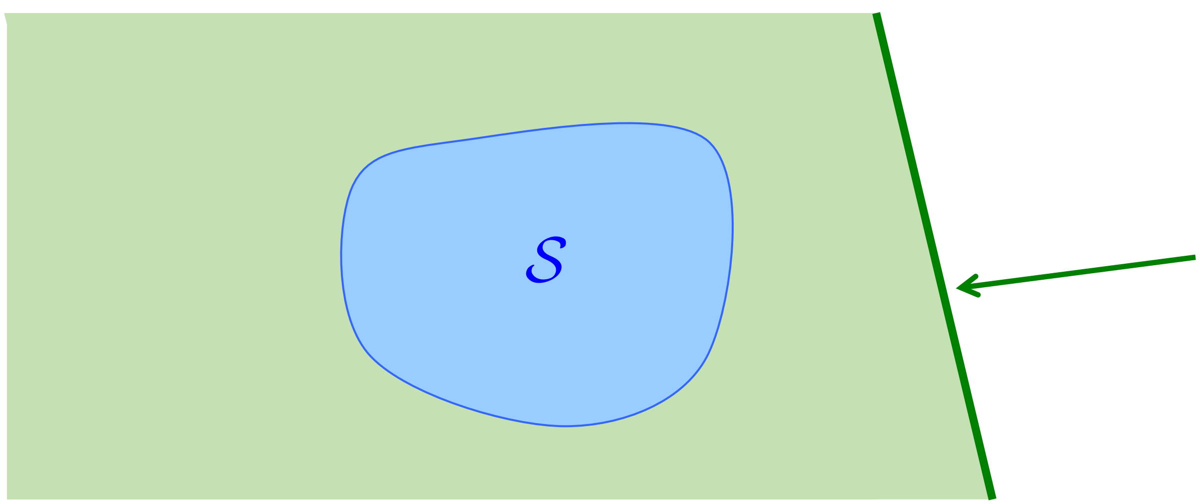
- In the $\mathcal{H}_n = \mathcal{H}_0, G_n = \mathrm{Id}$ case, streamline notation:

For $\boldsymbol{w} \in \mathcal{H}_1 \times \cdots \times \mathcal{H}_{n-1}$, let $w_n \triangleq -\sum_{i=1}^{n-1} G_i^* w_i$

Valid Inequalities for $\mathcal{S}$

- Take some $x_i, y_i \in \mathcal{H}_i$ such that $y_i \in T_i(x_i)$ for $i = 1, \ldots, n$
- If $(z, \boldsymbol{w}) \in \mathcal{S}$, then $w_i \in T_i(G_i z)$ for $i = 1, \ldots, n$
- Monotonicity implies that $\left\langle x_i - G_i z, y_i - w_i \right\rangle \geq 0$ for $i = 1, \ldots, n$
- Negate and add up:

$$\varphi(z, \boldsymbol{w}) = \sum_{i=1}^{n} \left\langle G_i z - x_i , y_i - w_i \right\rangle \leq 0 \qquad \forall (z, \boldsymbol{w}) \in \mathcal{S}$$



$$H = \left\{ p \,\middle|\, \varphi(p) = 0 \right\}$$

$$\varphi(p) \leq 0 \quad \forall p \in \mathcal{S}$$

Confirming that φ is Affine

The quadratic terms in $\varphi(z, \boldsymbol{w})$ take the form

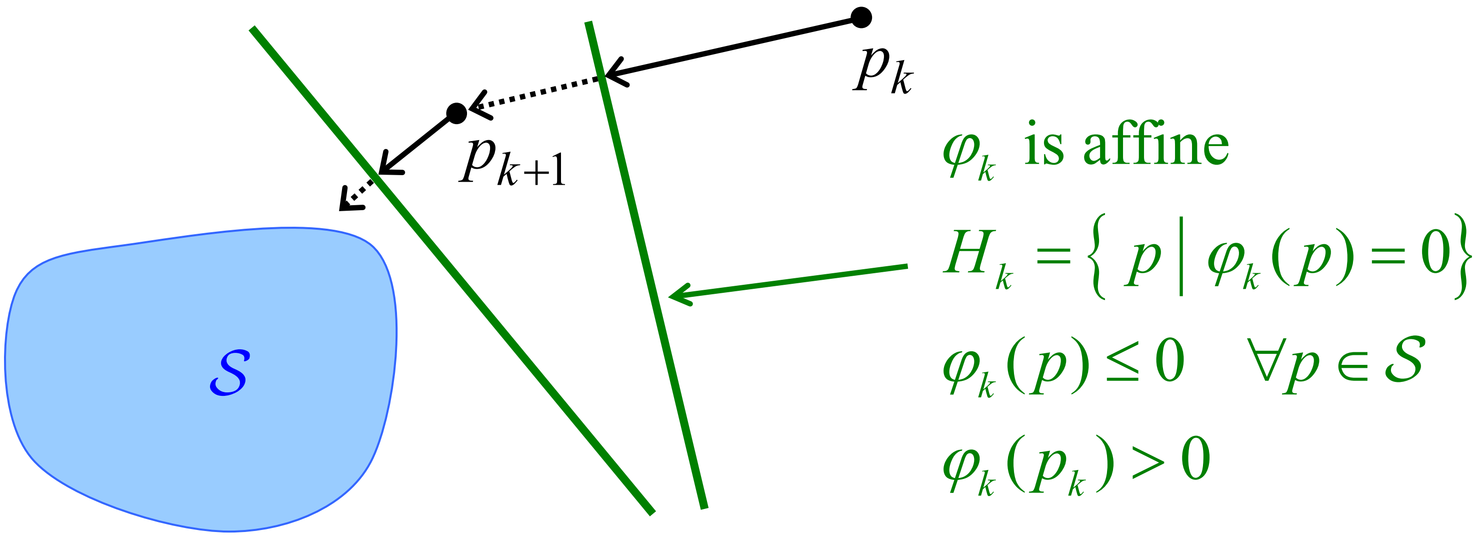
$$\sum_{i=1}^{n} \langle G_i z, -w_i \rangle = \sum_{i=1}^{n} \langle z, -G_i^{\mathsf{T}} w_i \rangle = \left\langle z, -\sum_{i=1}^{n} G_i^{\mathsf{T}} w_i \right\rangle = \langle z, -0 \rangle = 0$$

- Also true in the $\mathcal{H}_n = \mathcal{H}_0, G_n = \mathrm{Id}$ case where we drop the n^{th} index
  - Slightly different proof, same basic idea

Generic Projection Method for a Closed Convex Set $\mathcal{S}$ in a Hilbert Space $\mathcal{H}$

Apply the following general template:

- Given $p^k \in \mathcal{H}$, choose some affine function φ_k with $\varphi_k(p) \le 0 \;\forall p \in \mathcal{S}$
- Project p^k onto $H_k = \left\{ p \,\middle|\, \varphi_k(p) = 0 \right\}$, possibly with an over-relaxation factor $\lambda_k \in [\varepsilon, 2-\varepsilon]$, giving p_{k+1}, and repeat…



In our case: we find φ_k by picking some $x_i^k, y_i^k \in \mathcal{H}_i : y_i^k \in T_i(x_i^k), i = 1, \ldots, n$ and using the construction above

Selecting the Right φ_k

- If we pick φ_k badly, we may "stall"
- Selecting φ_k involves picking some $x_i^k, y_i^k \in \mathcal{H}_i : y_i^k \in T_i(x_i^k)$, $i = 1, \ldots, n$
- One key property is

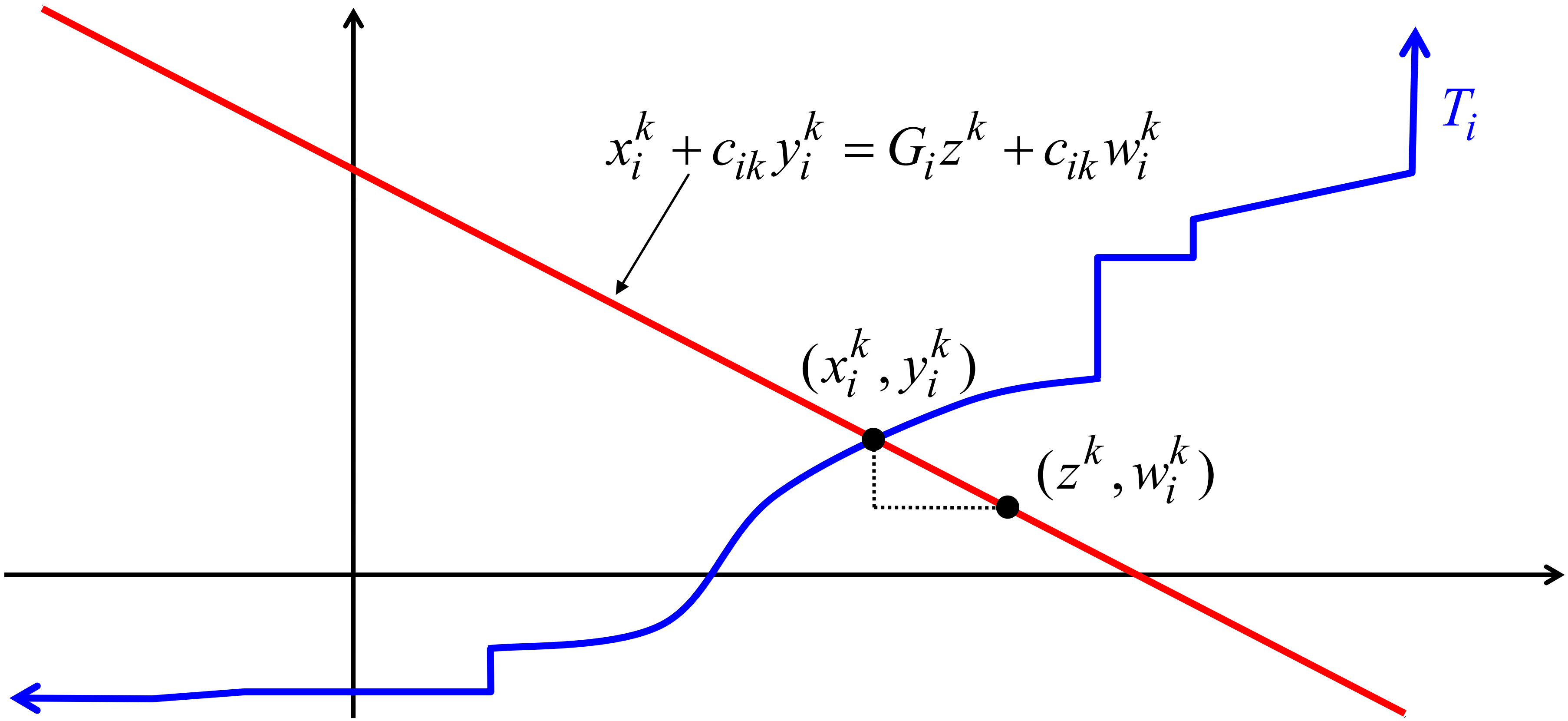
$$\varphi_k(z^k, \boldsymbol{w}^k) \triangleq \sum_{i=1}^{n} \left\langle G_i z^k - x_i^k, y_i^k - w_i^k \right\rangle \geq 0$$

with strict inequality if $(z^k, \boldsymbol{w}^k) \notin \mathcal{S}$

- The first suggestion is "prox" (E & Svaiter 2008 & 2009)

Prox Does the Job!

- We have an iterate $p^k = (z^k, \boldsymbol{w}^k) = (z^k, w_1^k, \ldots, w_n^k)$
- Take any $c_{ik} > 0$ and consider $(x_i^k, y_i^k) = \mathrm{Prox}_{c_{ik}T_i}(G_i z^k + c_{ik} w_i^k)$



- Then $x_i^k + c_{ik} y_i^k = G_i z^k + c_{ik} w_i^k \quad \Leftrightarrow \quad c_{ik}(y_i^k - w_i^k) = G_i z^k - x_i^k$
- Implying $\left\langle G_i z^k - x_i^k, y_i^k - w_i^k \right\rangle = c_{ik} \left\| G_i z^k - x_i^k \right\|^2 = c_{ik}^{-1} \left\| y_i^k - w_i^k \right\|^2 \geq 0$

Prox Finishes the Job

From

$$\left\langle G_i z^k - x_i^k, y_i^k - w_i^k \right\rangle = c_{ik} \left\| G_i z^k - x_i^k \right\|^2 = c_{ik}^{-1} \left\| y_i^k - w_i^k \right\|^2 \geq 0$$

we have that

$$\sum_{i=1}^{n} \left\langle G_i z^k - x_i^k, y_i^k - w_i^k \right\rangle \geq 0$$

and this inequality is strict unless $G_i z^k = x_i^k$ and $y_i^k = w_i^k$ for all i, which means that $(z^k, \boldsymbol{w}^k) \in \mathcal{S}$

The entire convergence proof follows from this same relationship.

Algorithm Including the Details

- Choose any $0 < \lambda_{\min} \leq \lambda_{\max} < 2$
- For $k = 1, 2, \ldots$

Process operators to find $x_i^k, y_i^k \in \mathbb{R}^{p_i} : \; y_i^k \in T_i(x_i^k), i = 1, \ldots, n$

$(u_1^k, \ldots, u_n^k) = \text{proj}_{\mathcal{G}}(x_1^k, \ldots, x_n^k)$, where $\mathcal{G} = \left\{ (w_1, \ldots, w_n) \;\middle|\; \sum_{i=1}^n G_i^\mathsf{T} w_i = 0 \right\}$

$$v^k = \sum_{i=1}^n G_i^\mathsf{T} y_i^k$$

$$\theta_k = \frac{\max\left\{ \sum_{i=1}^n \left\langle G_i z - x_i^k, y_i^k - w_i \right\rangle, 0 \right\}}{\left\| v^k \right\|^2 + \sum_{i=1}^n \left\| u_i^k \right\|^2}$$

Pick any $\lambda \in [\lambda_{\min}, \lambda_{\max}]$

$$z^{k+1} = z^k - \lambda_k \theta_k v^k$$

$$w_i^{k+1} = w_i^k - \lambda_k \theta_k u_i^k, \quad i = 1, \ldots, n$$

- Or, when $\mathcal{H}_n = \mathcal{H}_0, G_n = \text{Id}$, one can avoid the $\text{proj}_{\mathcal{G}}$ operation

Many Variations Possible in “Process Operators”

1. Inexact processing: the prox operations may be performed approximately using a relative error criterion
   - E & Svaiter 2009

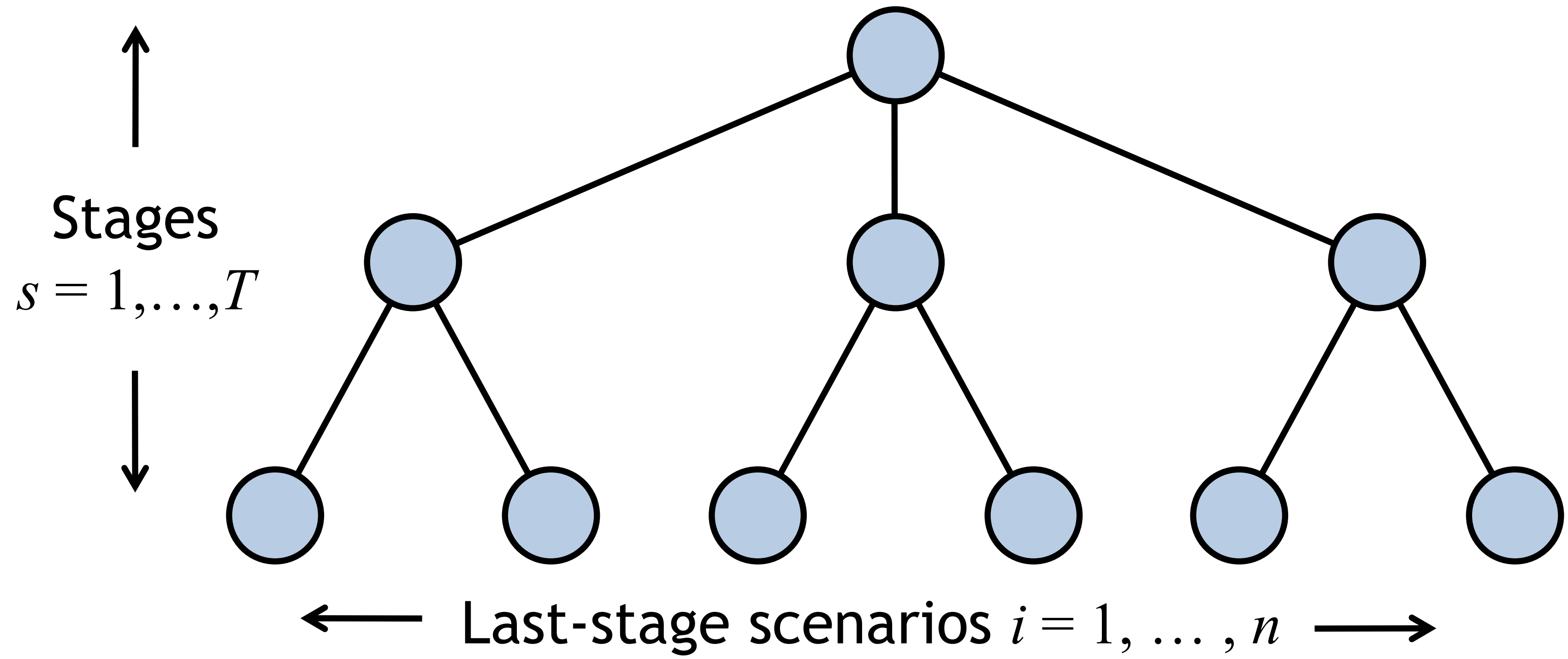
2. Block asynchrony: you do not have to process every operator at every iteration; you may process some subset and let $(x_i^k, y_i^k) = (x_i^{k-1}, y_i^{k-1})$ for the rest, so long as you process each operator at least once every M iterations
   - Combettes & E 2018, E 2017

3. Lag asynchrony: you may process operators using (boundedly) old information $(z^{d(i,k)}, \boldsymbol{w}^{d(i,k)})$, where $k \geq d(i,k) \geq k - K$
   - Combettes & E 2018, E 2017

4. Non-prox steps: For Lipschitz continuous gradients, procedures using one or two gradient steps may be substituted for the prox operations
   - Johnstone and E 2022, 2021
     also see Tranh-Dinh and Vũ 2015

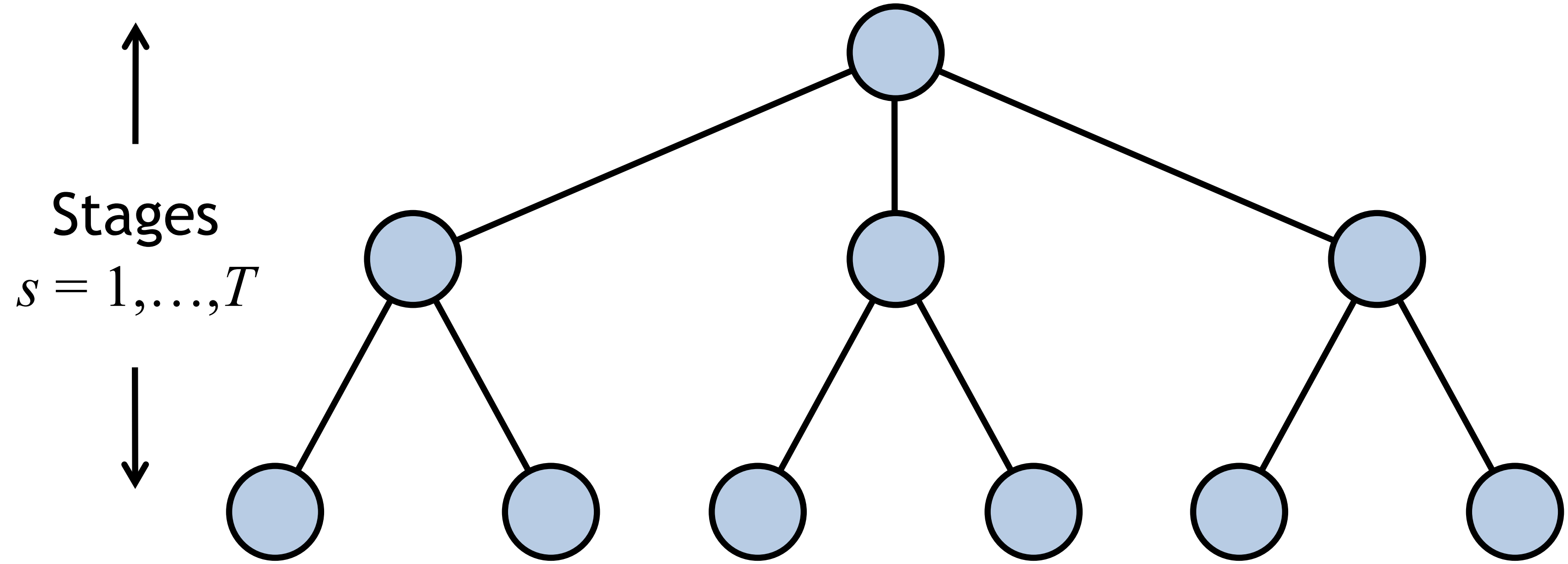
+ “mix and match”

An Application:
Uncertainty Model for Decision Making: A Scenario Tree



- π_i is the probability of last-stage scenario $i = 1, \ldots, n$
- Will use “scenario” as a shorthand for “last-stage scenario”
- Typically a discrete-time and sampled approximation of some infinite or much larger model

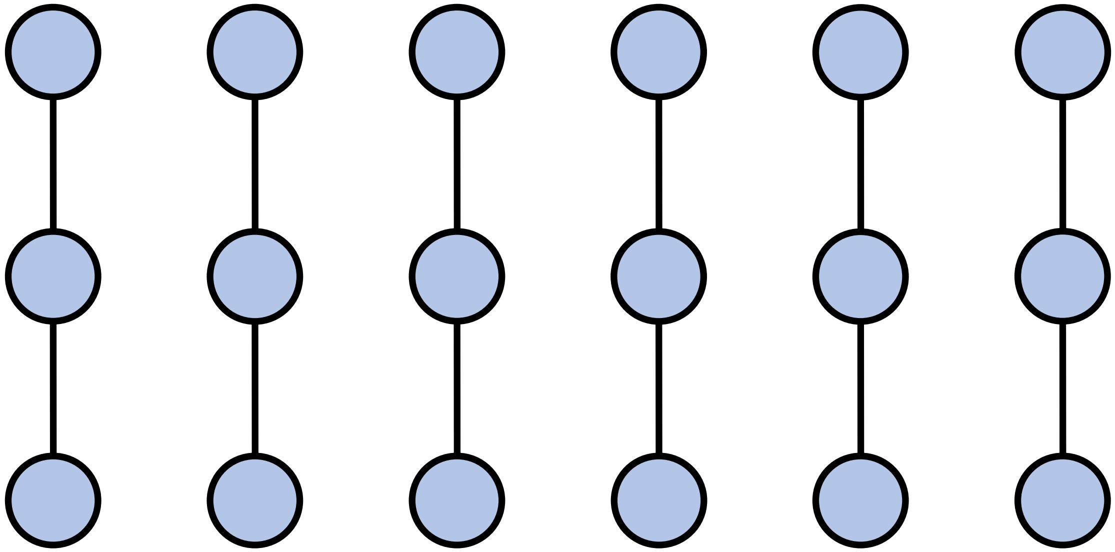
Stochastic Programming



- System walks randomly from the root to some leaf
- At each node there are decision variables, for example
  - How much of an investment to buy or sell
  - How much to run a power generator, etc...
- ... and constraints that depend on earlier decisions
- Model alternates decisions and uncertainty resolution

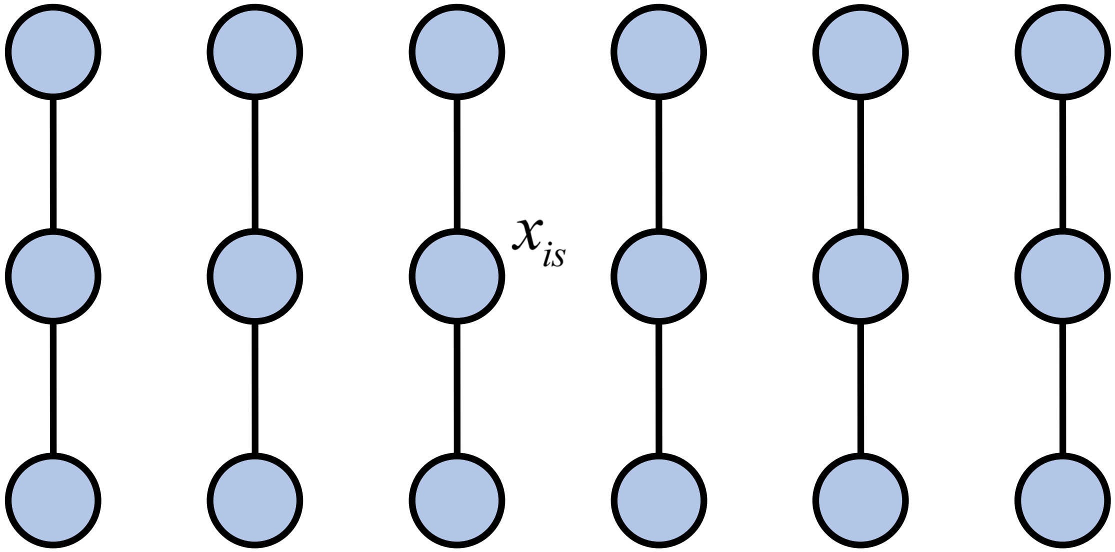
Notation

- Replicate decision variables: n copies at every stage



Notation

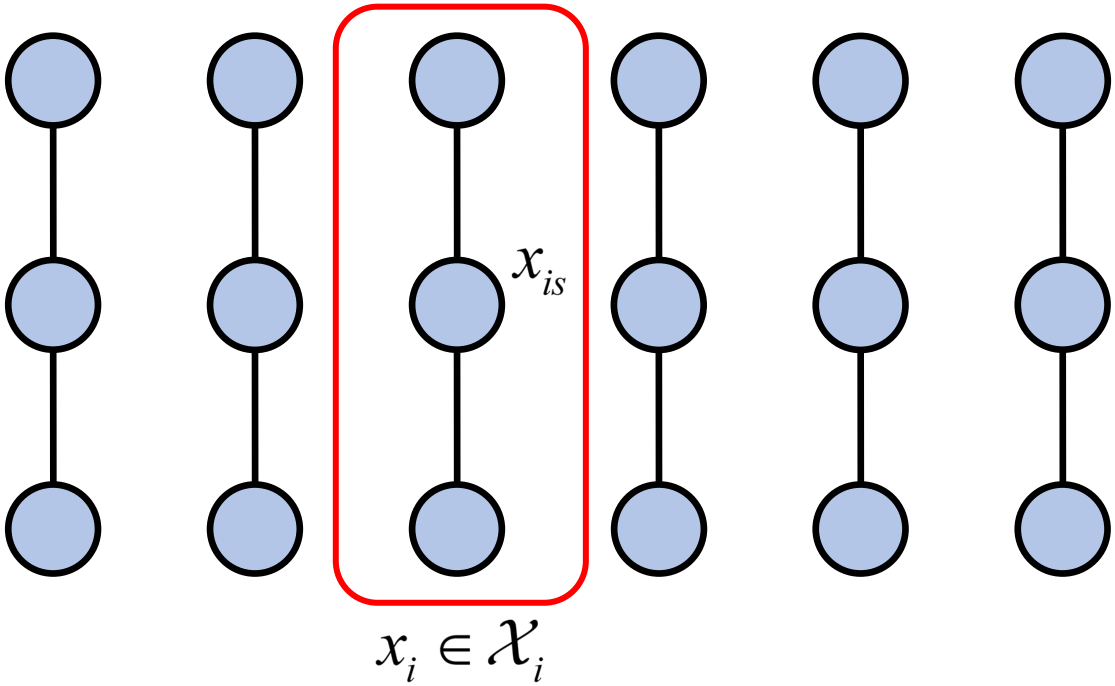
- Replicate decision variables: n copies at every stage



- x_{is} is the vector of decision variables for scenario i at stage s

Notation

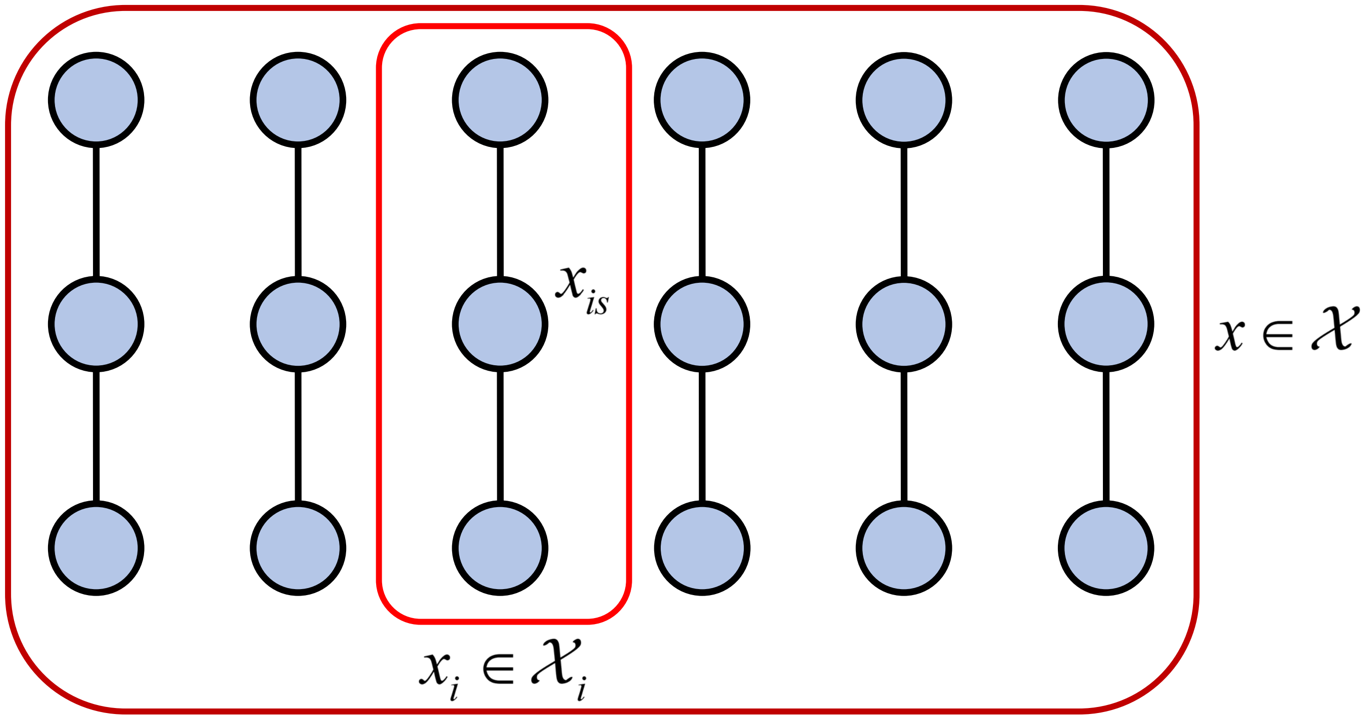
- Replicate decision variables: n copies at every stage



- x_{is} is the vector of decision variables for scenario i at stage s
- $\mathcal{X}_i$ is the space of all variables pertaining to scenario i ; elements are $x_i = (x_{i1}, \ldots, x_{iT})$

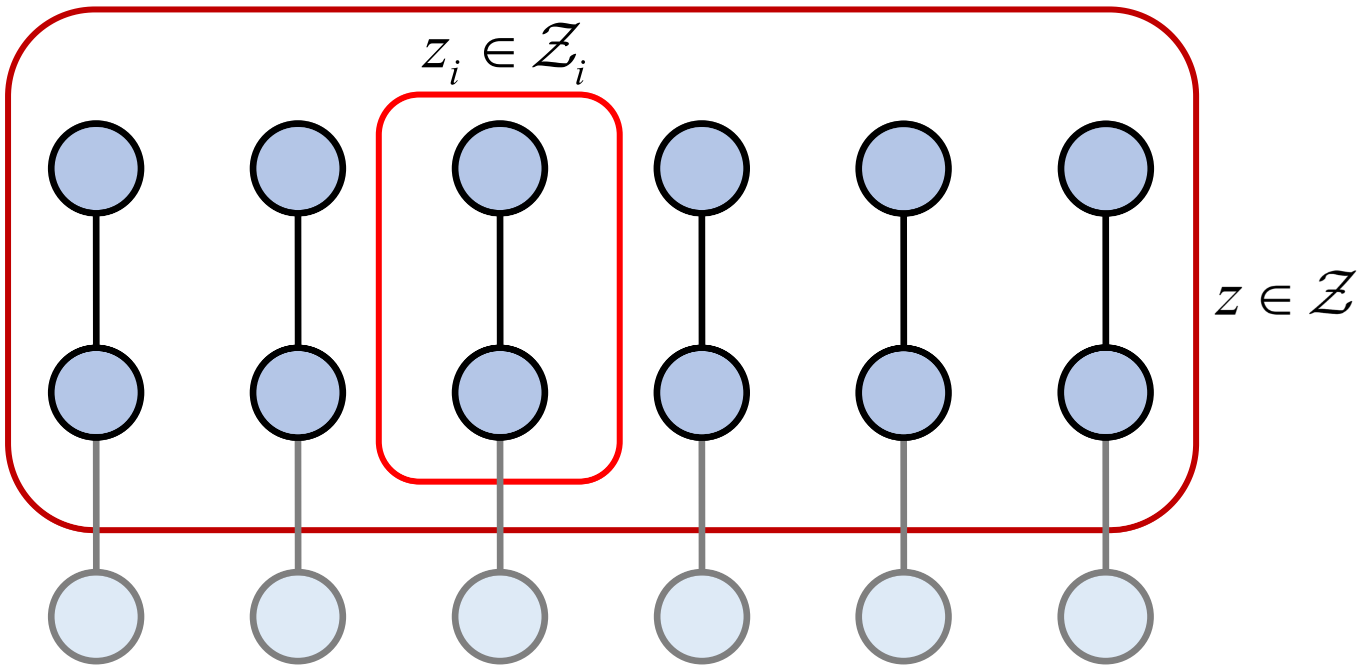
Notation

- Replicate decision variables: n copies at every stage



- x_{is} is the vector of decision variables for scenario i at stage s
- $\mathcal{X}_i$ is the space of all variables for scenario i; elements are
  $x_i = (x_{i1}, \ldots, x_{iT})$
- $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$ is space of all decision variables; elements are
  $x = (x_1, \ldots, x_n) = \big((x_{11}, \ldots, x_{1T}), \ldots, (x_{n1}, \ldots, x_{nT})\big)$

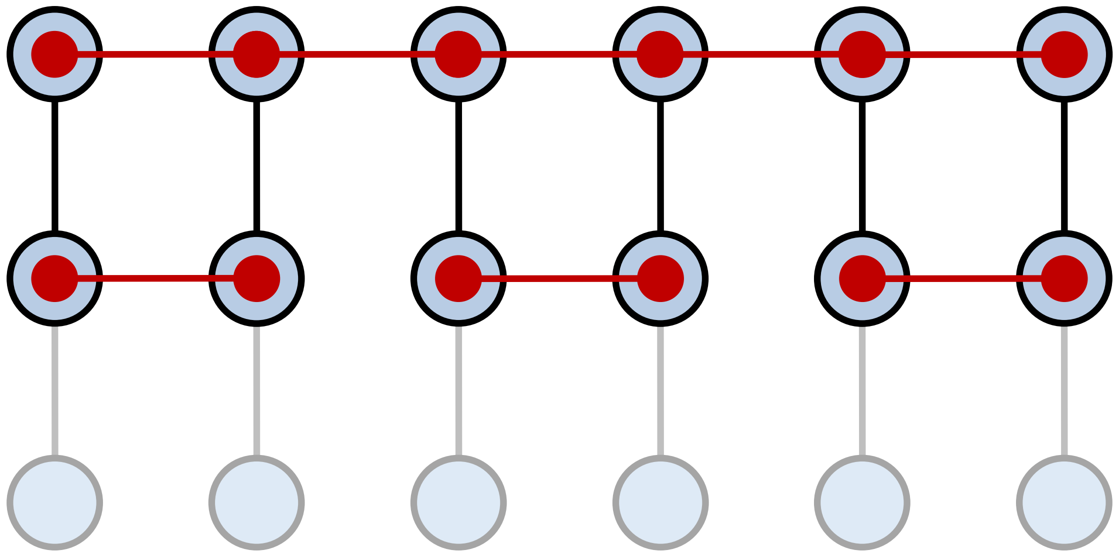
Notation

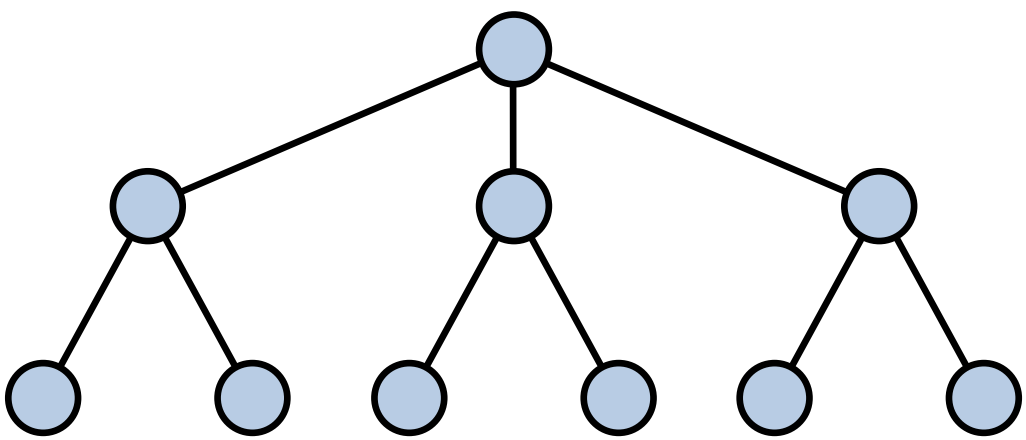


- $\mathcal{Z}_i$ is $\mathcal{X}_i$ without the last stage; elements $z_i = (z_{i1}, \ldots, z_{i,T-1})$
- $\mathcal{Z} = \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_n$ is the space of all variables except the last stage: elements $z = (z_1, \ldots, z_n) = \big((z_{11}, \ldots, z_{1,T-1}), \ldots, (z_{n1}, \ldots, z_{n,T-1})\big)$

Nonanticipativity Subspace

- $\mathcal{N} \subset \mathcal{Z}$ is the subspace of $\mathcal{Z}$ meeting the *nonanticipativity constraints* that $z_{is} = z_{js}$ whenever scenarios i and j are indistinguishable at stage s





Projecting onto the Nonanticipativity Space

- Following Rockafeller and Wets (1991), we use the following probability-weighted inner product on $\mathcal{Z}$:

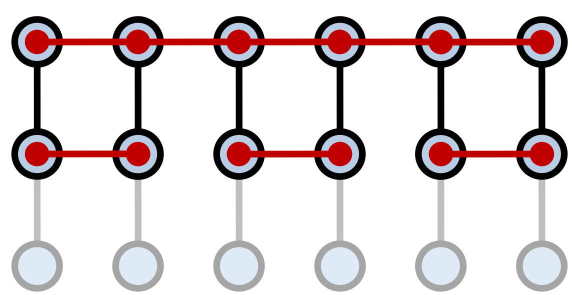
$$\left\langle (z_1,\ldots,z_n),(q_1,\ldots,q_n) \right\rangle = \sum_{i=1}^{n} \pi_i \left\langle z_i, q_i \right\rangle$$

- With this inner product, the projection map $\mathrm{proj}_{\mathcal{N}} : \mathcal{Z} \to \mathcal{N}$ is given by

$$\mathrm{proj}_{\mathcal{N}}(q) = z, \quad \text{where}$$

$$z_{is}^{k+1} = \frac{1}{\left(\sum_{j \in S(i,s)} \pi_j\right)} \sum_{j \in S(i,s)} \pi_j q_{js}^{k+1} \qquad i = 1,\ldots,n, \;\; s = 1,\ldots,T-1$$

and $S(i,s)$ is the set of scenarios indistinguishable from scenario i at time s.



Applying the ADMM: Progressive Hedging (PH)

- Applying the ADMM to this problem (details omitted) produces

$$
\begin{aligned}
x_i^{k+1} &= \arg\min_{x_i \in \mathcal{X}_i} \left\{ f_i(x_i) + \left\langle M_i x_i, w_i^k \right\rangle + \tfrac{\rho}{2} \left\| M_i x_i - z_i^k \right\|^2 \right\} \quad i = 1, \ldots, n \\
z^{k+1} &= \operatorname{proj}_{\mathcal{N}}\left(M x^{k+1} \right) \\
w^{k+1} &= w^k + \rho(M x^{k+1} - z^{k+1})
\end{aligned}
$$

- Here, $f_i : X_i \to \mathbb{R} \cup \{+\infty\}$ represents the objective and all constraints if it were somehow known in advance that leaf scenario i will occur
- M_i is the matrix that drops the last-stage variables from x_i
- M is the matrix that drops all last-stage variables from x
- All steps of this algorithm can be parallelized (not just the first one)

Projective Splitting Instead: Subproblem Processing

Subproblem: (may operate many copies in parallel)

Let $0 < \rho_{\min} \leq \rho_{\max} < \infty$ be fixed

Parameters for subproblem i:

- $z_i = (z_{i1}, \ldots, z_{i,T-1})$: scenario i "target" values (no last stage)
- w_i : multipliers (same dimensions as z_i)

Arguments: $z_i, w_i \in \mathcal{Z}_i$

Select some $\rho \in [\rho_{\min}, \rho_{\max}]$

Let $x_i \in \operatorname{Arg\,min}_{x_i} \left\{ f_i(x_i) + \langle M_i x_i, z_i \rangle + \frac{\rho}{2} \| M_i x_i - z_i \|^2 \right\}$

and $y_i = w_i + \rho(M_i x_i - z_i)$

Return $\tilde{x}_i \doteq M_i x_i, \; y_i$

Looks like PH subproblem + part of multiplier update

Projective-Splitting-Based Algorithm (with Block Asynchrony)

repeat

Pick some set $I_k \subseteq \{1,\ldots,n\}$ of scenarios to process

for $i \in I_k$, process scenario i as above: $z_i, w_i \mapsto \tilde{x}_i, y_i$

for $i \in \{1,\ldots,n\} \setminus I_k$, keep the previous $\tilde{x}_i, y_i$

$u \leftarrow \tilde{x} - \mathrm{proj}_{\mathcal{N}}(\tilde{x})$

$v \leftarrow \mathrm{proj}_{\mathcal{N}}(y)$

$\tau \leftarrow \|u\|^2 + \gamma\|v\|^2 = \sum_{i=1}^{n} \pi_i \|u_i\|^2 + \gamma \sum_{i=1}^{n} \pi_i \|v_i\|^2$

$\phi \leftarrow \langle z - \tilde{x}, w - y \rangle = \sum_{i=1}^{n} \pi_i (z_i - \tilde{x}_i)^{\mathsf{T}} (w_i - y_i)$

if $\phi > 0$ **then**

Choose some $\nu \in [\nu_{\min}, \nu_{\max}]$

$z \leftarrow z + (\nu\phi / \tau\gamma)v$

$w \leftarrow w + (\nu\phi / \tau)u$

until “termination detected”

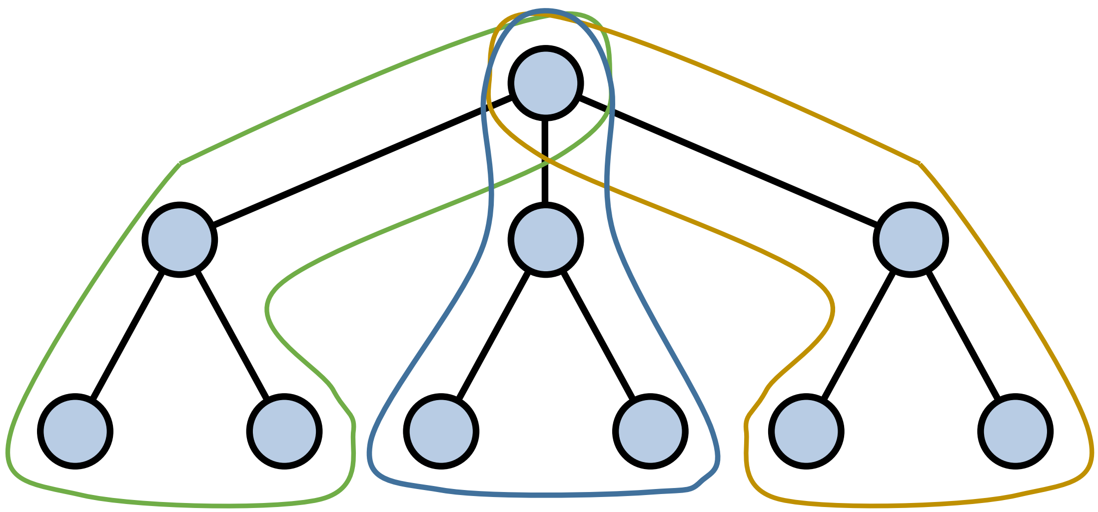
Called “APH”

- Coordination process is a bit more complicated than PH, but uses similar operations and can also be parallelized

“AirCond” Example Problem

- Single-product manufacturing/inventory problem
  - Has quadratic costs (for back-ordering)
  - Generated problem with 5 stages and 1,000,000 leaf scenarios

- Grouped model scenarios into “bundles”, which the algorithms treat as if they are scenarios, like



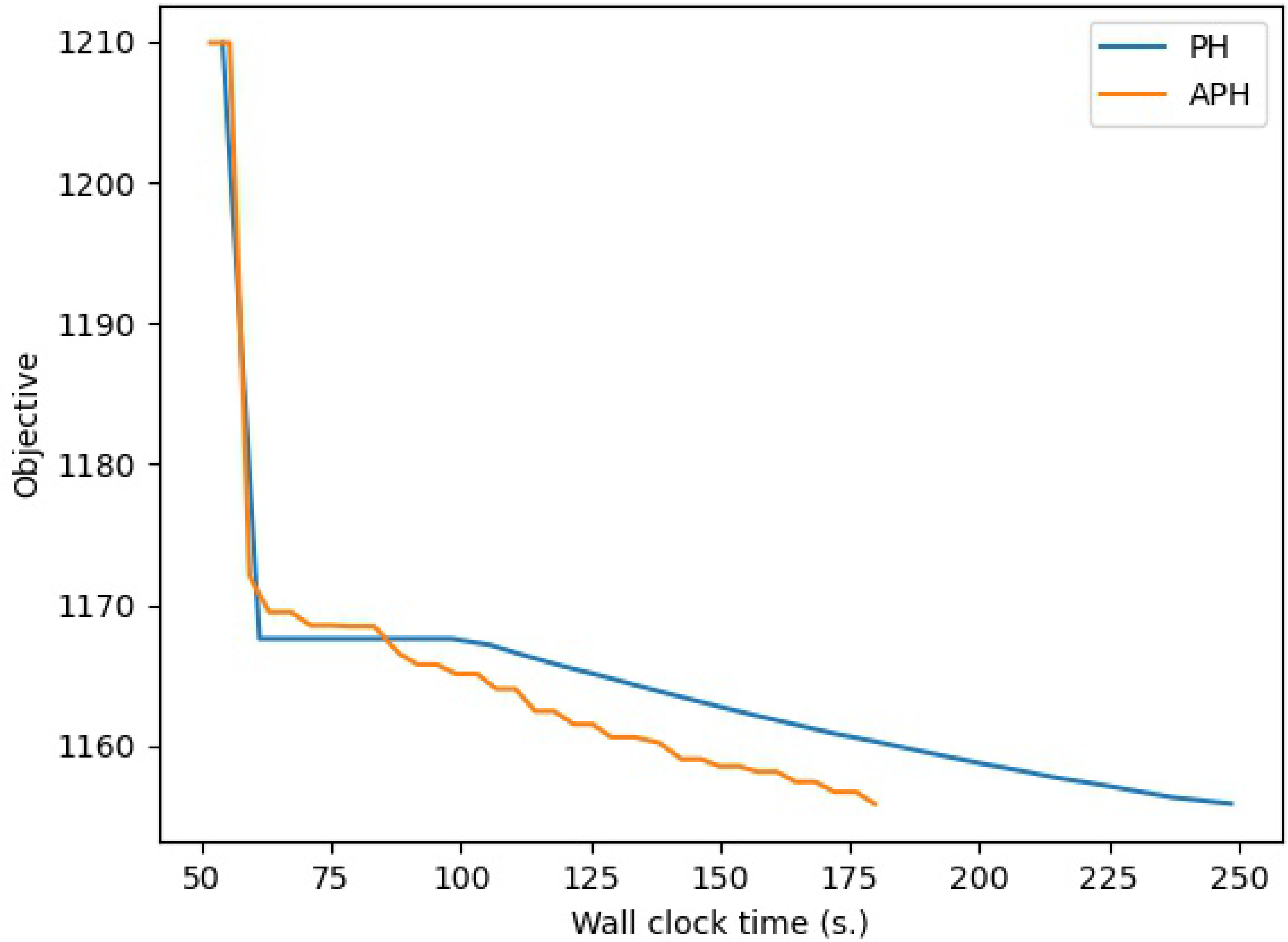
but with 1,000 scenarios per bundle

"Bundles per Rank"

- Here, a "rank" is pair of 2 CPU cores
- Can solve one subproblem efficiently

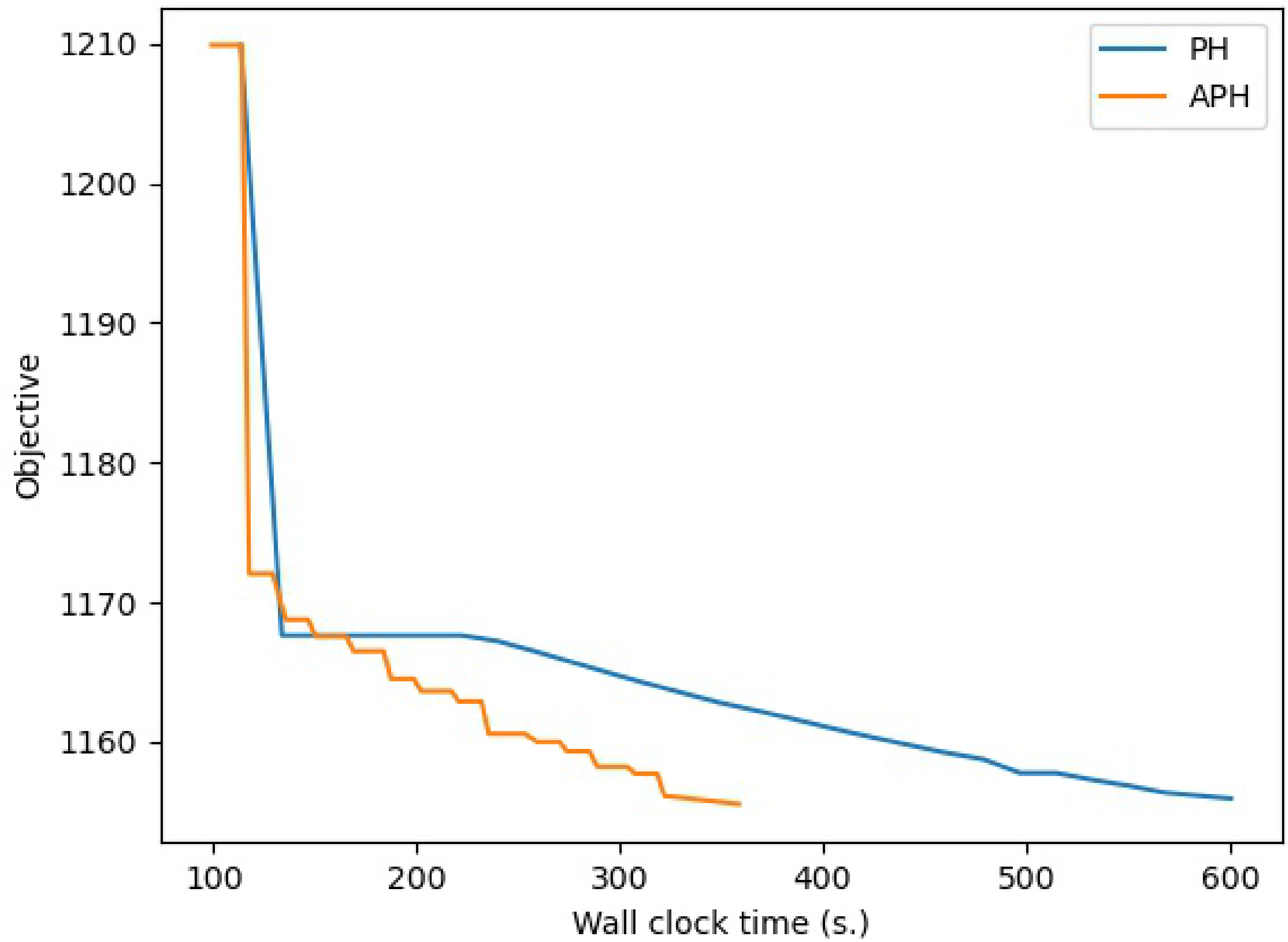
- 1 "bundle per rank":
    - Each rank has one bundle
    - For both PH and projective splitting, each rank solves a supproblem for this bundle at each iteration
- 5 "bundles per rank":
    - Each rank has 5 bundles
    - In PH, all have to be solved at each iteration
    - In projective splitting, each iteration picks one subproblem to solve (in a "greedy" way; details omitted)
- 10 "bundles per rank" – similar, but 10 instead of 5

Aircond: 1,000,000 Scenarios, 1 Bundle per Rank



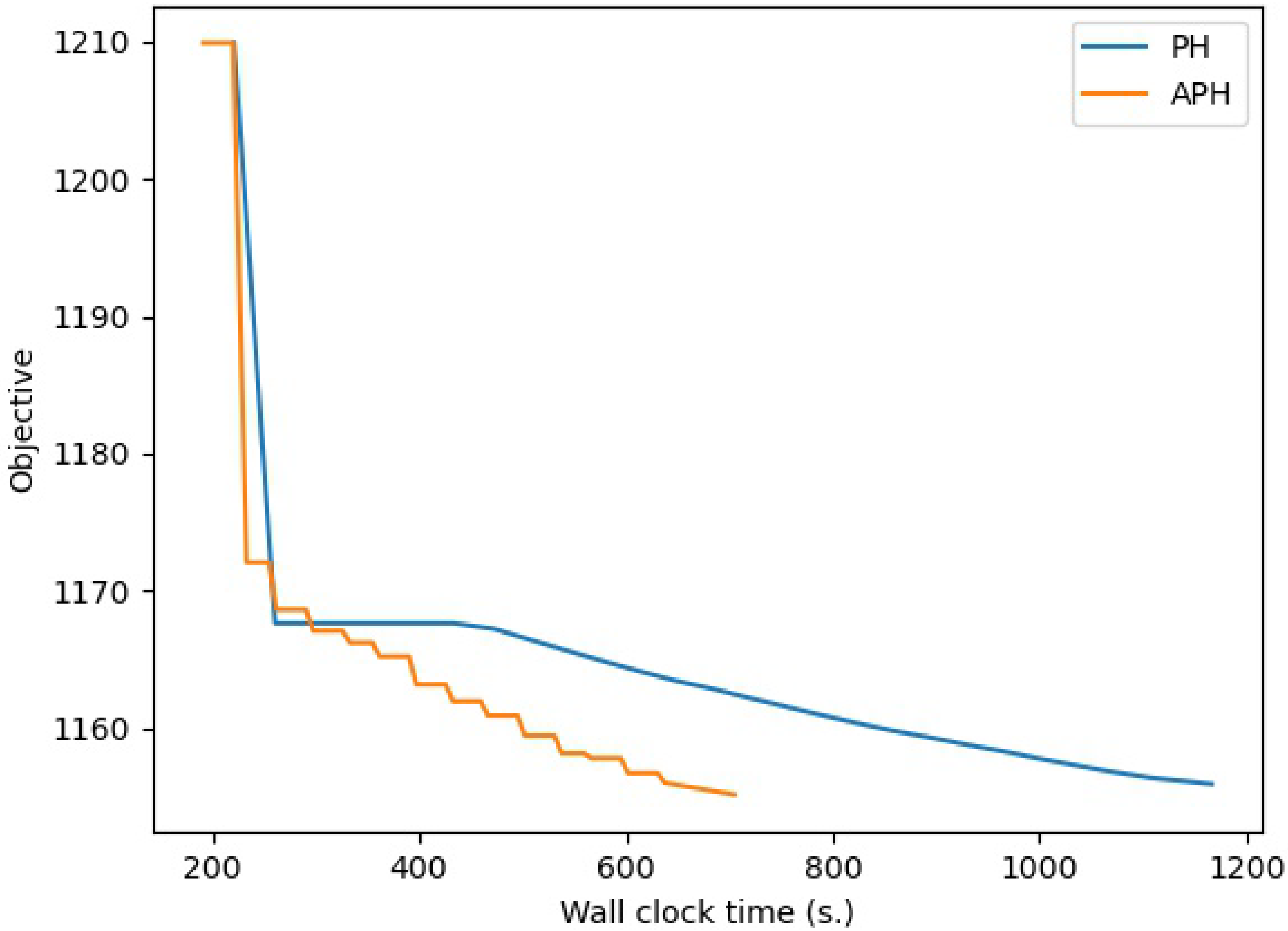
- Splitting algorithm: 2,000 CPU cores
- Total with upper and lower bound computation: 6,000 CPU cores

Aircond: 1,000,000 Scenarios, 5 Bundles per Rank



- Splitting algorithm: 400 CPU cores
- Total with bounders: 1,200 cores

Aircond: 1,000,000 Scenarios, 10 Bundles per Rank



- Splitting algorithm: 200 CPU cores
- Total with bounders: 600 CPU cores

Thanks for your attention!