# Randomized Progressive Hedging methods for Multi-stage Stochastic Programming

Gilles Bareilles, Y. Laguel, D. Grishchenko, F. Iutzeler, J. Malick

LJK, Univ. Grenoble Alpes

ROADEF 2020

# Multistage Stochastic Programming

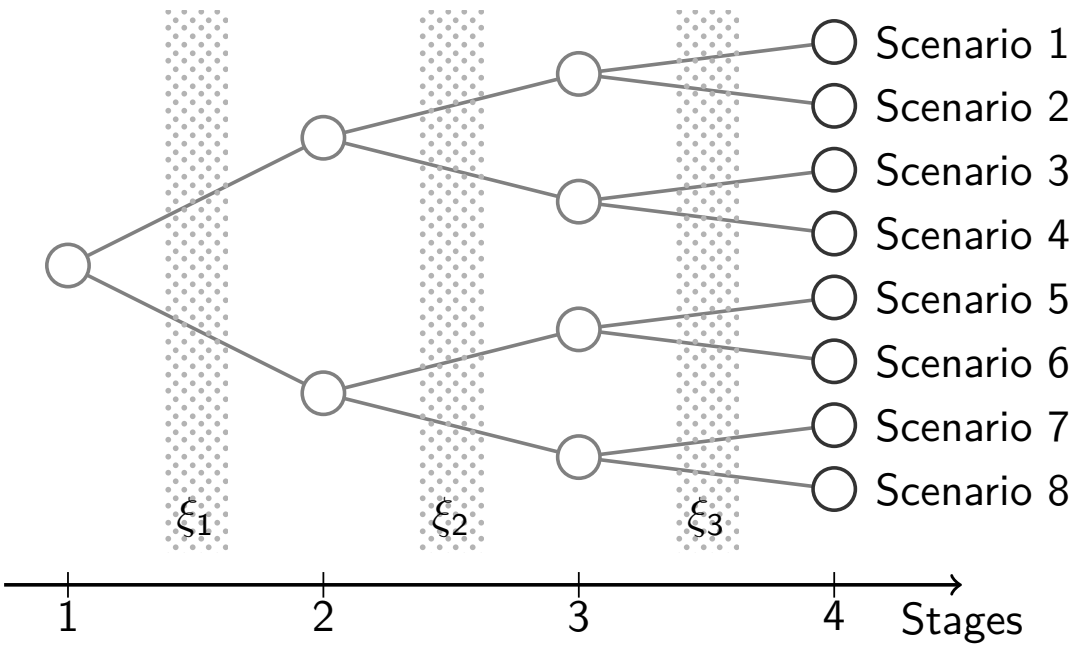
Make decisions for a system with ***randomness*** over ***several time steps***

***Randomness*** is formalized as $S$ scenarios with probability $p_1, \ldots, p_S$

For scenario $s$, best decision $x \in \mathbb{R}^n$: objective / constraint pair $(f^s, \mathcal{C}^s)$:

$$\arg\min_{x^s \in \mathcal{C}^s} f^s(x^s)$$

## Example




Manage dams over 4 days, uncertainty is rain

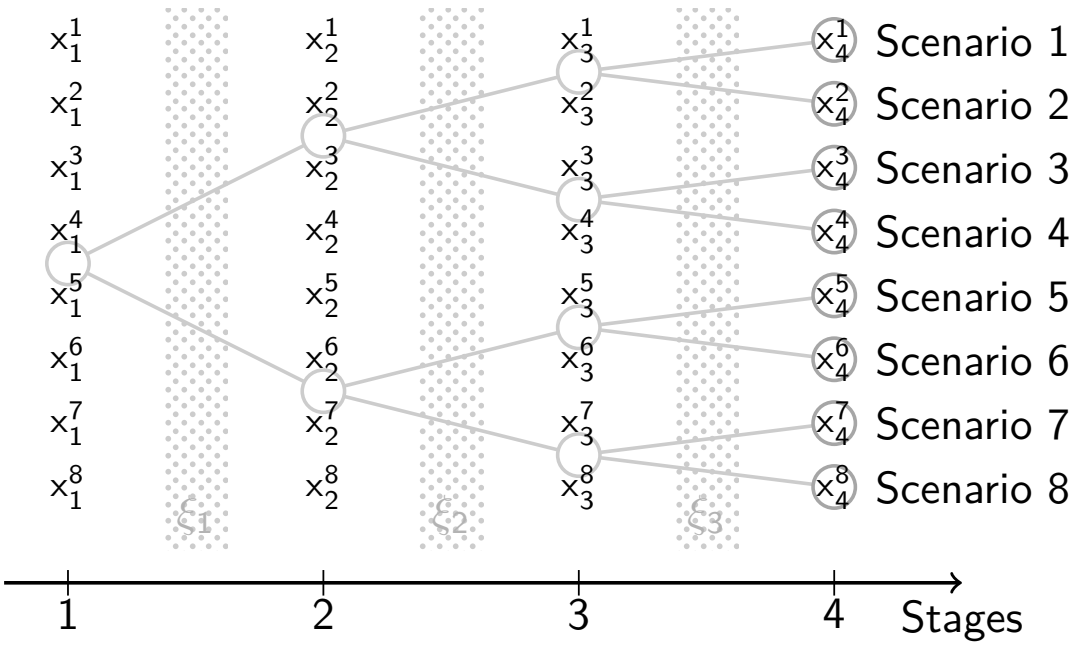
Scenario 1: 3 months of rain
Scenario 8: 3 months of drought

# Multistage Stochastic Programming

Decisions per scenario are collected in a matrix $x \in \mathbb{R}^{S \times n}$.

Additional global constraint $\mathcal{W}$: ***non anticipativity***. Decision at time $t$ should be identical for two scenario identical up to $t$, since we should not make decisions based on future randomness.

## Example



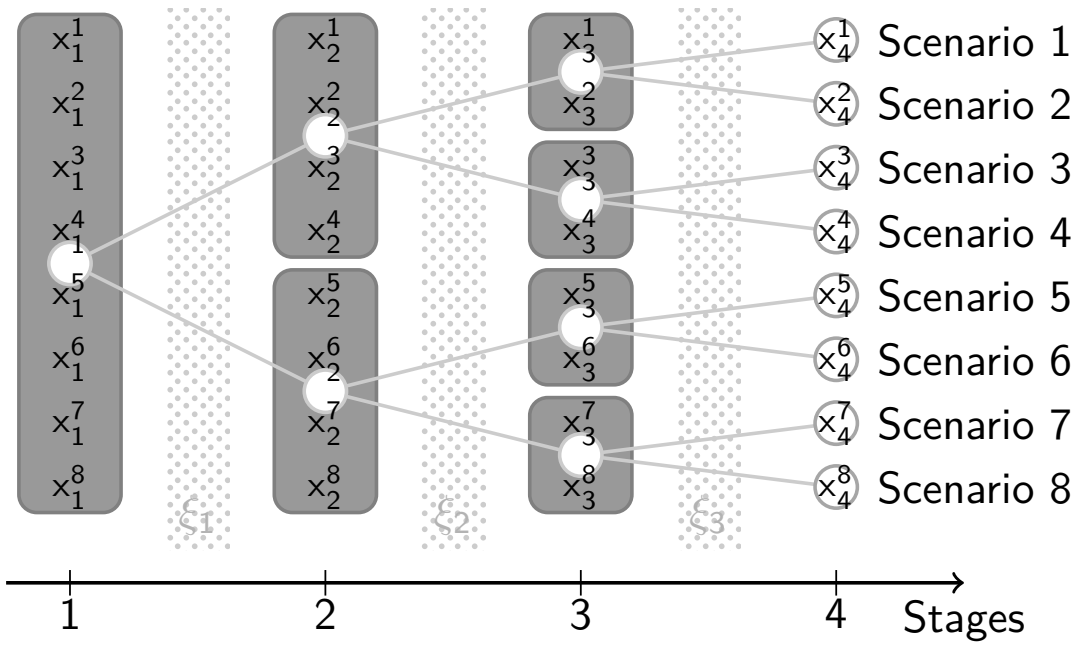

Matrix variable
$x \in \mathbb{R}^{S \times n}$

# Multistage Stochastic Programming

Decisions per scenario are collected in a matrix $x \in \mathbb{R}^{S \times n}$.

Additional global constraint $\mathcal{W}$: ***non anticipativity***. Decision at time $t$ should be identical for two scenario identical up to $t$, since we should not make decisions based on future randomness.

## Example




Matrix variable
$x \in \mathbb{R}^{S \times n}$

Coefficients equal per gray area.

# Energy management example

Manage several ***dams***, produce electricity to ***meet some demand***.

Optimal solution for *one scenario s*:

$$\underset{x=(q,y,e)\in\mathbb{R}^n}{\text{minimize}} \quad f^s(x) = \sum_{t=1}^{T} c_{H,t}^\top y_t + c_E e_t$$

$$\text{subject to} \quad \sum_{b=1}^{B} y_t^b + e_t \geq D \text{ for all } t \qquad \text{(demand is met at each stage)}$$

$$q_t^b = q_{t-1}^b - y_t^b + \xi_t^b \text{ for all } t \geq 2, b \qquad \text{(evolution of the amount of water)}$$

$$q_1^b = W_1^b - y_1^b \text{ for all } b \qquad \text{(init. amount of water per dam)}$$

$$q_t^b \leq W^b \text{ for all } t, b \qquad \text{(max. amount of water per dam)}$$

where, at stage $t$:

- $q_t^b \in \mathbb{R}_+$: quantity of water of dam $b$ at stage $t$;
- $y_t^b \in \mathbb{R}_+$: quantity of water converted in electricity;
- $e_t \in \mathbb{R}_+$: electricity bought externally.

# Multistage Stochastic Programming

$$\begin{aligned} \underset{x \in \mathcal{W} \subset \mathbb{R}^{S \times n}}{\text{minimize}} \quad & \sum_{s=1}^{S} p_s f^s(x^s) \\ \text{subject to} \quad & x^s \in \mathcal{C}^s,\ s = 1, \ldots, S \end{aligned} \tag{$\mathcal{P}$}$$

- The objective and constraints $(f^s, \mathcal{C}^s)$ are separable per scenario;
- Only constraint $\mathcal{W}$ ties scenario decisions together.

Progressive Hedging is a the classical algorithm for solving $(\mathcal{P})$;

it leverages the scenario separability of objective using duality.

- Rockafellar, Wets: Scenarios and policy aggregation in optimization under uncertainty (MOR 1991);
- Ruszczyński, Shapiro: Stochastic programming models, Handbooks in operations research and management science (2003).

# Progressive Hedging

**Initialize:** $x^0 \in \mathcal{W}, u^0 \in \mathcal{W}^\perp, \mu > 0$

**For** $k = 0, 1, \ldots$ **do:**

1) $y^{k+1,s} = \arg\min_{y \in \mathcal{C}^s} \left\{ f^s(y) + \frac{1}{2\mu} \left\| y - x^{k,s} + \mu u^{k,s} \right\|^2 \right\}$ for all $s$

2) $x^{k+1} = \mathbf{Proj}_{\mathcal{W}}(y^{k+1})$

3) $u^{k+1} = u^k + \frac{1}{\mu}(y^{k+1} - x^{k+1})$

**Return:** $x^{k+1}$

- 1) Decomposition: improve decision for each scenario independently;
- 2) Coordination: improved decisions are combined to form next feasible iterate.

# Progressive Hedging

**Initialize:** $x^0 \in \mathcal{W}, u^0 \in \mathcal{W}^\perp, \mu > 0$

**For** $k = 0, 1, \ldots$ **do:**

1) $y^{k+1,s} = \arg\min_{y \in \mathcal{C}^s} \left\{ f^s(y) + \frac{1}{2\mu} \left\| y - x^{k,s} + \mu u^{k,s} \right\|^2 \right\}$ for all $s$

2) $x^{k+1} = \mathbf{Proj}_{\mathcal{W}}(y^{k+1})$

3) $u^{k+1} = u^k + \frac{1}{\mu}(y^{k+1} - x^{k+1})$

**Return:** $x^{k+1}$

For ***large-scale*** stochastic programs, $S = 2^T \gg 1$:

1. each subproblem is costly to solve at least QP;
2. high number of scenarios make one iteration of progressive hedging very costly;

   $\rightarrow$ potential solution: parallelism on 1). Inefficient if high variability of subproblem resolution time.

*Can the global projection step be avoided, to make this method fully parallel?*

## Progressive Hedging – operator view

Denote $z^k = x^k + \mu u^k$ the primal-dual variable.

One iteration of Progressive Hedging is equivalent to

$$z^{k+1} = \frac{1}{2} \mathrm{O}_{\mu \mathrm{A}} \circ \mathrm{O}_{\mu \mathrm{B}}(z^k) + \frac{1}{2} z^k \qquad \text{(DR)}$$

...for some operators $\mathrm{O}_{\mu \mathrm{A}}$ and $\mathrm{O}_{\mu \mathrm{B}}$.

[Ruszczyński, Shapiro: Stochastic programming models (2003)]

This is Douglas-Rachford splitting...

## Progressive Hedging – operator view

Denote $z^k = x^k + \mu u^k$ the primal-dual variable.

One iteration of Progressive Hedging is equivalent to

$$z^{k+1} = \frac{1}{2} \mathrm{O}_{\mu \mathrm{A}} \circ \mathrm{O}_{\mu \mathrm{B}}(z^k) + \frac{1}{2} z^k \qquad \text{(DR)}$$

...for some operators $\mathrm{O}_{\mu \mathrm{A}}$ and $\mathrm{O}_{\mu \mathrm{B}}$.

[Ruszczyński, Shapiro: Stochastic programming models (2003)]

This is Douglas-Rachford splitting...We can ***randomize*** it:

Draw a scenario $s^k \in \{1, \ldots, S\}$ with probability $\mathbb{P}[s^k = s] = q_s$

$$\left| \begin{array}{ll} \mathrm{z}^{k+1,s^k} & = \frac{1}{2} \left[ \mathrm{O}_{\mu \mathrm{A}} \circ \mathrm{O}_{\mu \mathrm{B}}(\mathrm{z}^k) \right]^{s^k} + \frac{1}{2} \mathrm{z}^{k,s^k} \\ \mathrm{z}^{k+1,s} & = \mathrm{z}^{k,s} \qquad \text{for all } s \neq s^k \end{array} \right.$$

- Iutzeler, Bianchi, Ciblat, Hachem: Asynchronous distributed optimization using a randomized alternating direction method of multipliers (CDC 2013).

## Douglas Rachford randomized

What does $[\mathrm{O}_{\mu\mathrm{A}} \circ \mathrm{O}_{\mu\mathrm{B}}(z)]^{s^k}$ look like ?

1. $[\mathrm{O}_{\mu\mathrm{A}}(z)]^{s^k} = 2x^{s^k} - z^{s^k}$, where

$$x^{s^k} = \underset{y \in \mathcal{C}^s}{\arg\min} \left\{ f^{s^k}(y) + \frac{1}{2\mu} \left\| y - z^{s^k} \right\|^2 \right\}$$

$\Rightarrow [\mathrm{O}_{\mu\mathrm{A}}(z)]^{s^k}$ only involves $z^{s^k}$.

2. $[\mathrm{O}_{\mu\mathrm{B}}(z)]^{s^k} = 2x^{s^k} - z^{s^k}$, where $x^{s^k} = [\mathbf{Proj}_{\mathcal{W}}(z)]^{s^k}$

# Randomized Progressive Hedging

**Initialize:** $z^0 = x^0 \in \mathcal{W}, \mu > 0$

**For** $k = 0, 1, \ldots$ **do:**

$$\begin{cases} \text{Draw a scenario } s^k \in \{1, \ldots, S\} \text{ with probability } \mathbb{P}[s^k = s] = q_s \\ x^{k+1,s^k} = \left[\mathbf{Proj}_{\mathcal{W}}(z^k)\right]^{s^k} \\ y^{k+1,s^k} = \arg\min_{y \in \mathcal{C}^s} \left\{ f^{s^k}(y) + \frac{1}{2\mu} \left\| y - 2x^{k+1,s^k} + z^{k,s^k} \right\|^2 \right\} \\ \left| \begin{array}{l} z^{k+1,s^k} = z^{k,s^k} + y^{k+1,s^k} - x^{k+1,s^k} \\ z^{k+1,s} = z^{k,s} \text{ for all } s \neq s^k \end{array} \right. \end{cases}$$

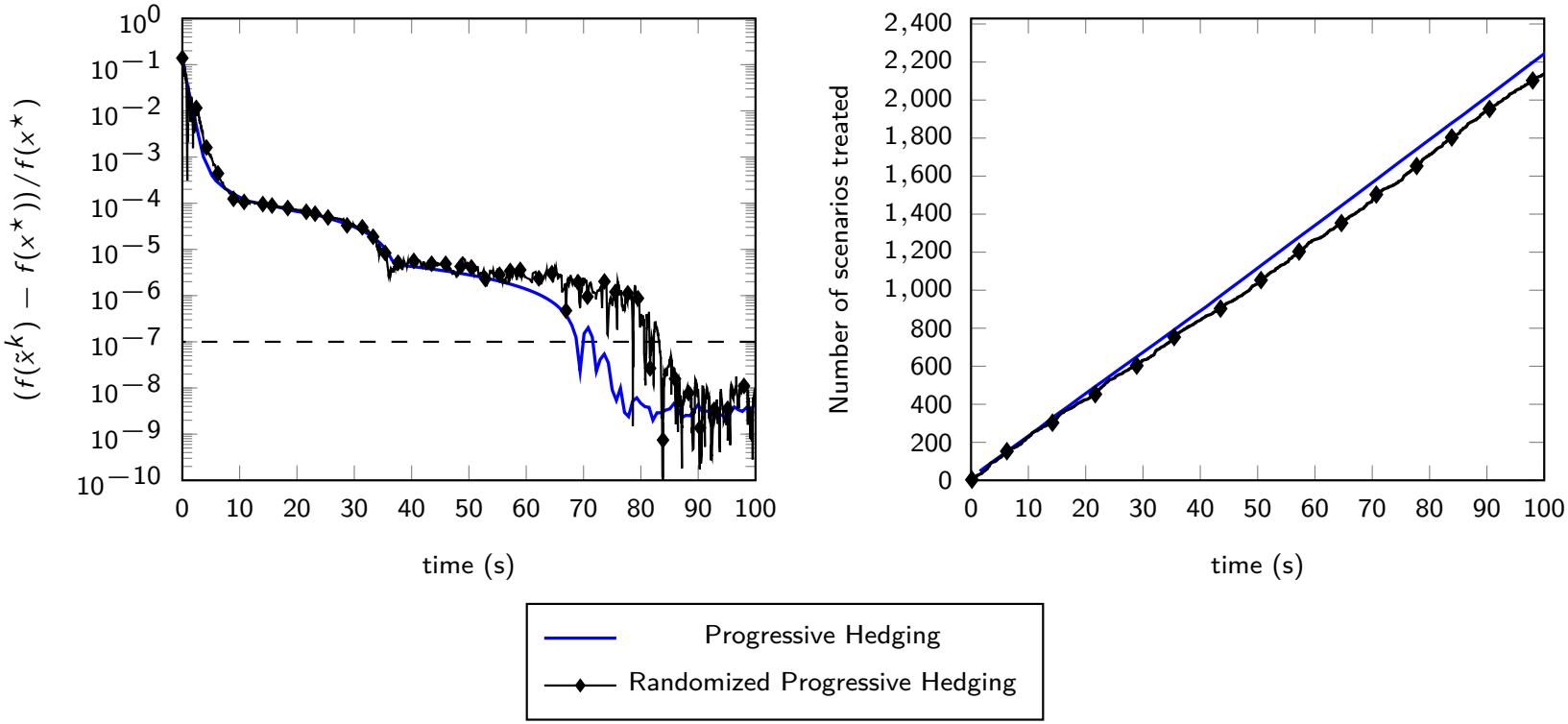
**Return:** $\tilde{x}^{k+1} = \mathbf{Proj}_{\mathcal{W}}(z^{k+1})$

- Projection updates only ***one line*** of matrix $x^{k+1}$
- Optimization subproblem concerns only ***one scenario*** $s^k$

- Natural single-thread algorithm; more interesting than PH due to incremental nature
- Scenario sampling can be done uniformly; according to scenario probabilities $p_s$

Halfway towards parallelism...

***Numerical experiments:*** problem with 5 stages – 16 scenarios.



# The Julia toolbox

## Distributed computing

Modern languages make *high-level distributed* programming possible:
algorithm implementation is independant of infrastructure, e.g. multicore, cluster, ssh connected agents, ...

```
using Distributed
addprocs(7); length(procs()) # one master + 7 workers

y_par = solve_randomized_par(pb)
```

## JuMP modeling language

Easy choice of solver for subproblem solve, default: Ipopt.

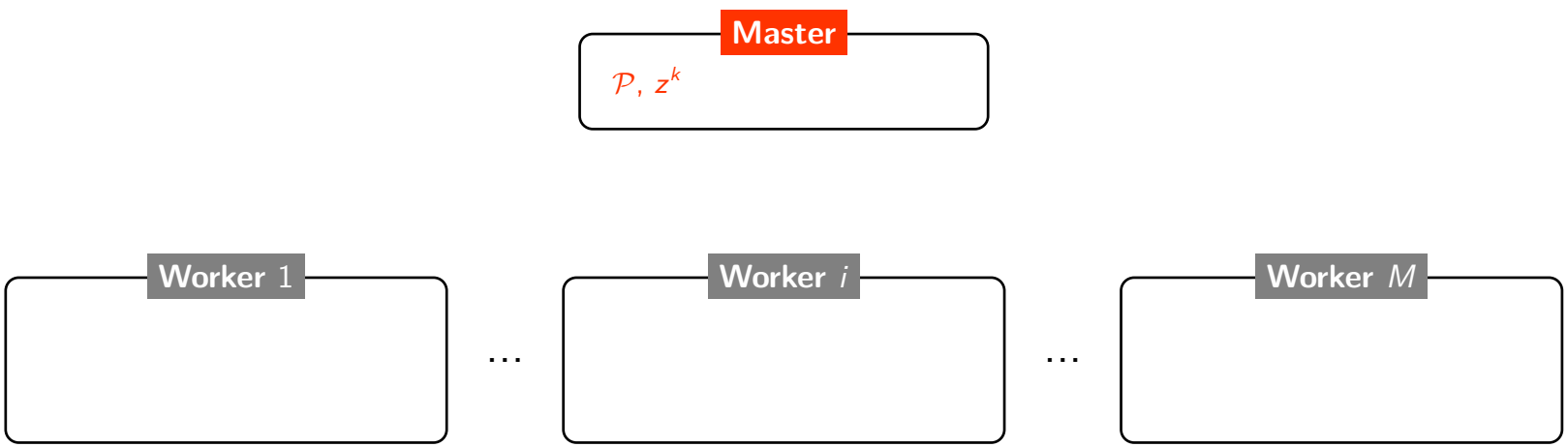
## The toolbox: RPH


`github.com/yassine-laguel/RandomizedProgressiveHedging.jl`

# Parallel Progressive Hedging

At each iteration,

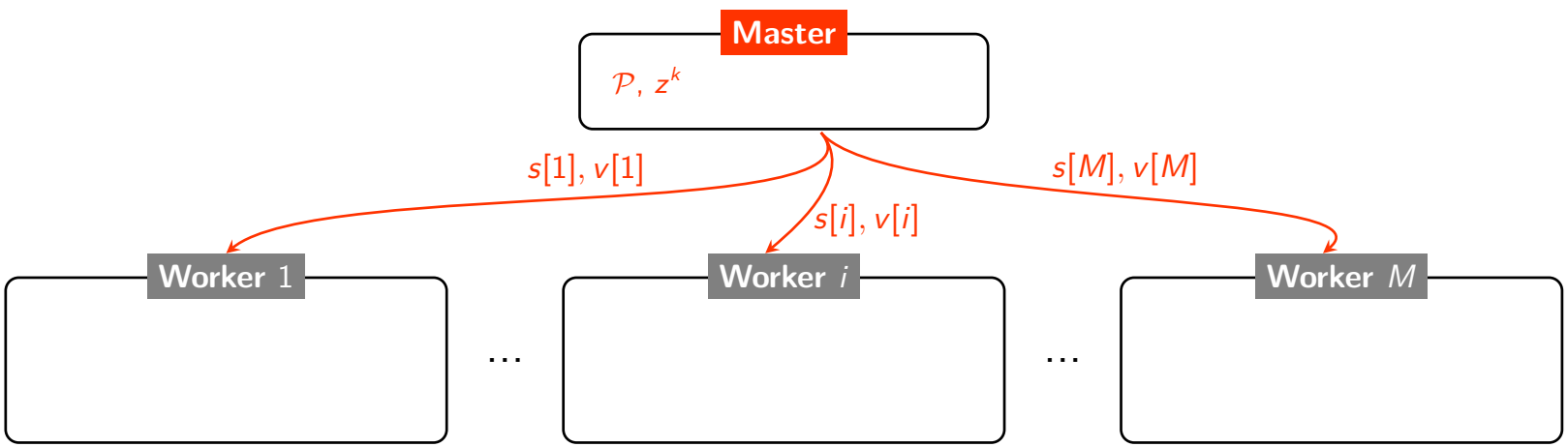
- all workers receive and solve one subproblem;
- their answer are incorporated in master variable.



# Parallel Progressive Hedging

At each iteration,

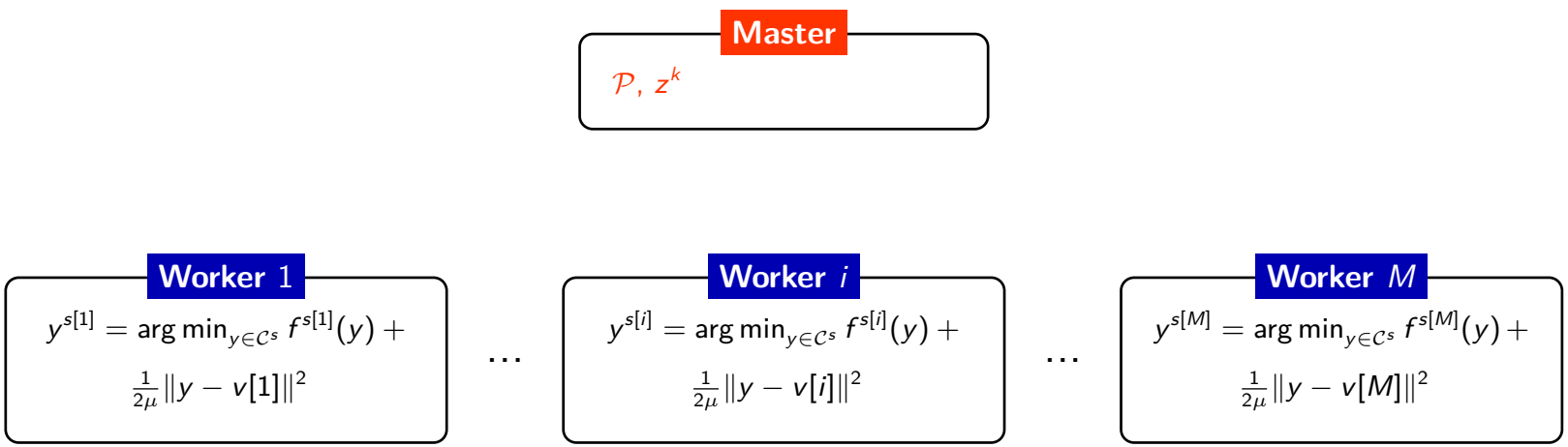
- all workers receive and solve one subproblem;
- their answer are incorporated in master variable.



# Parallel Progressive Hedging

At each iteration,

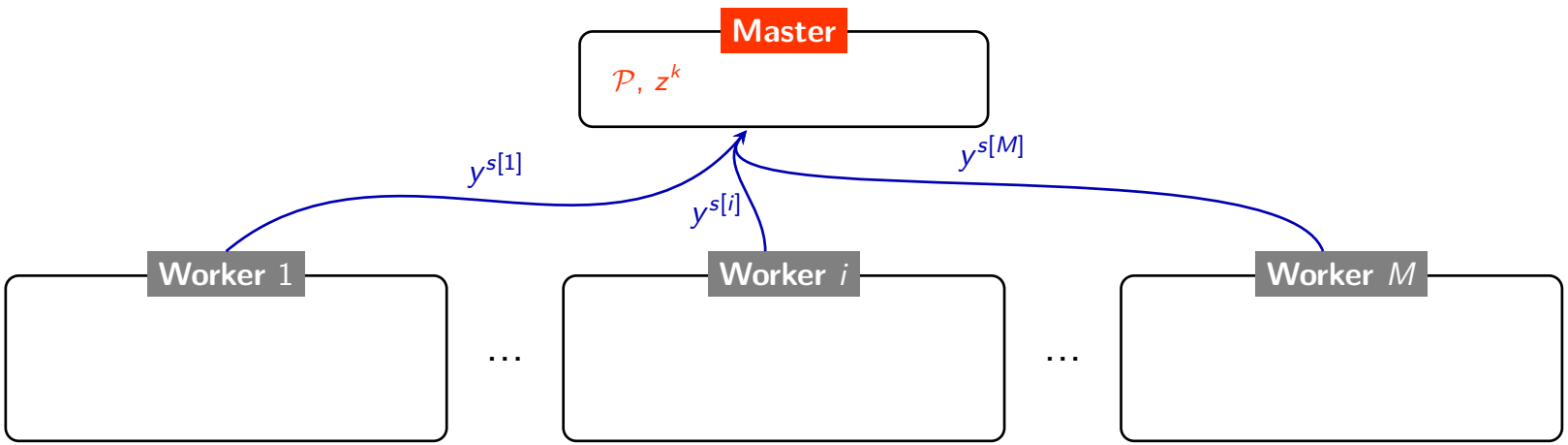
- all workers receive and solve one subproblem;
- their answer are incorporated in master variable.



# Parallel Progressive Hedging

At each iteration,

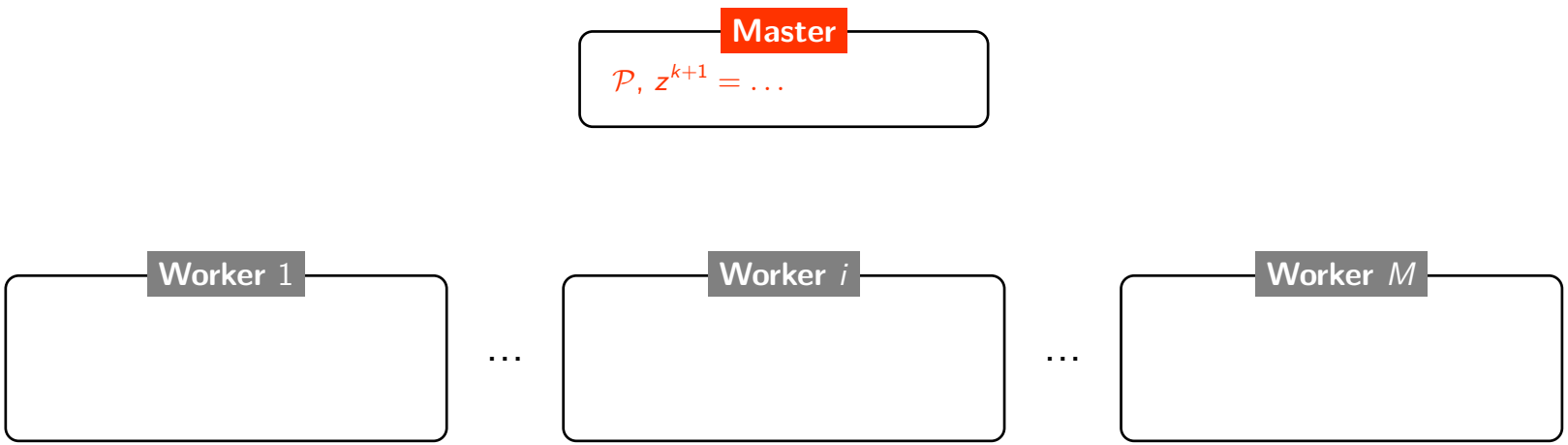
- all workers receive and solve one subproblem;
- their answer are incorporated in master variable.



# Parallel Progressive Hedging

At each iteration,

- all workers receive and solve one subproblem;
- their answer are incorporated in master variable.



# Parallel Progressive Hedging

**Initialize:** $x^0 = z^0 = v^0 \in \mathcal{W}, \mu > 0$

**For** $k = 0, 1, \ldots$ **do:**

Draw $M$ scenarios $(s[1], .., s[M]) \in \{1, \ldots, S\}^M$ with probability $\mathbb{P}[s[i] = s] = q_s$

Send a scenario/point pair $(s[i], v^{k+1,s[i]})$ to *each* worker $i = 1, .., M$

Receive $y^{s[i]}$ from *all* workers $i = 1, .., M$

$$\begin{cases} z^{k+1,s} = z^{k,s} + y^s - x^s & \text{for all } s \in (s[1], .., s[M]) \\ z^{k+1,s} = z^{k,s} & \text{for all } s \notin (s[1], .., s[M]) \end{cases}$$

$$x^{k+1,s} = \left[\mathbf{Proj}_{\mathcal{W}}(z^{k+1})\right]^s \quad \text{for all } s \notin (s[1], .., s[M])$$

$$v^{k+1} = 2x^{k+1} - z^{k+1}$$

**Return:** $\tilde{x}^{k+1} = \mathbf{Proj}_{\mathcal{W}}(z^{k+1})$

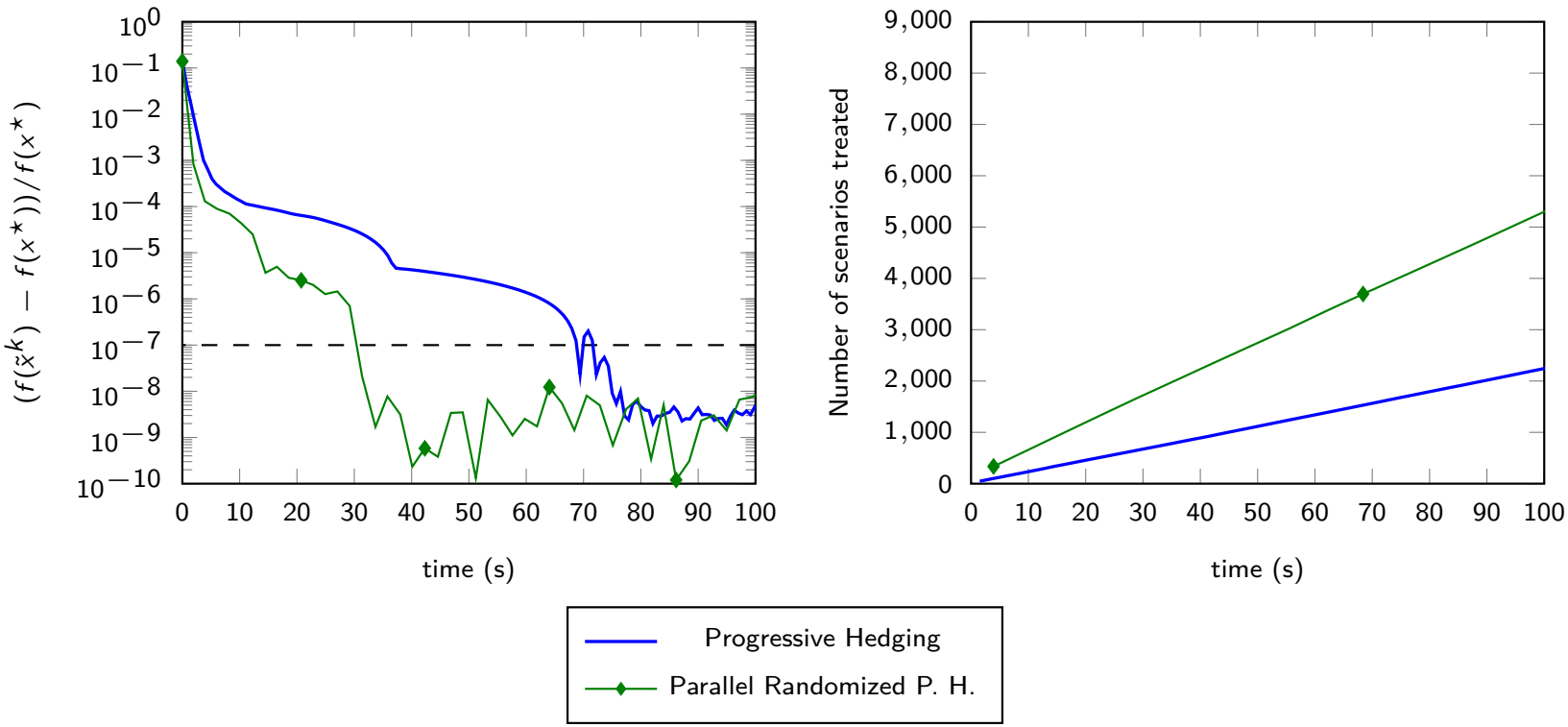
**As soon as a scenario/point pair is received:**

$$\begin{cases} \text{Receive scenario/point pair } (s[i], v[i]) \\ y^{s[i]} = \arg\min_{y \in \mathcal{C}^s} \left\{ f^{s[i]}(y) + \frac{1}{2\mu} \|y - v[i]\|^2 \right\} \\ \text{Send } y^{s[i]} \text{ to the Master} \end{cases}$$

- Heavy lifting done by workers;
- Master left with cheap operations.

# Parallel Progressive Hedging – experiments

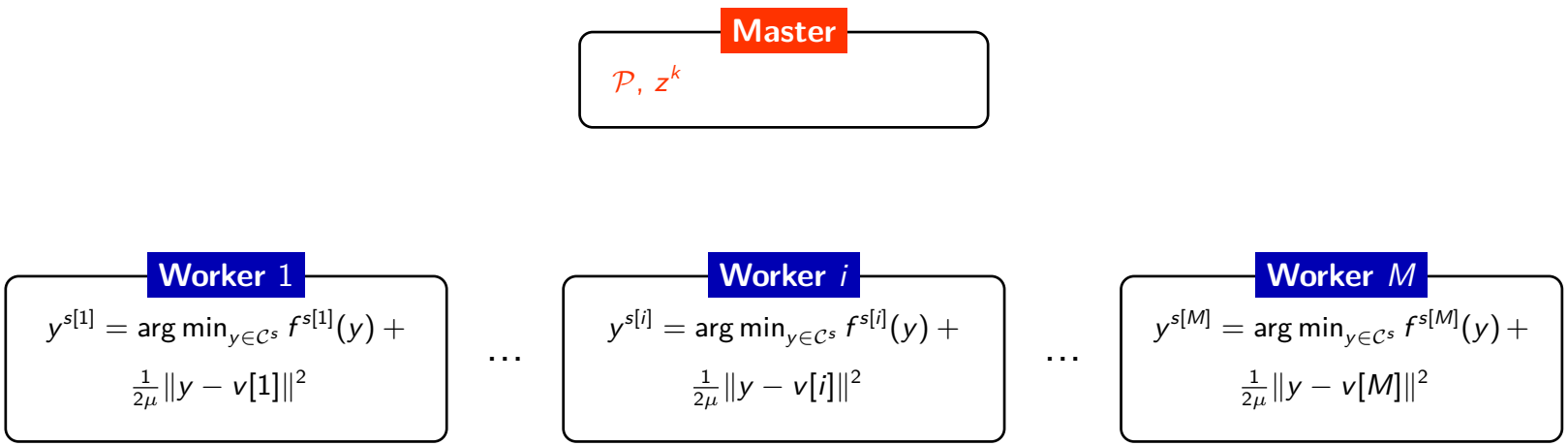
Problem with 5 stages – 16 scenarios, solved with 1 master, 7 workers. To simulate problem/infrastructure variability, 3 scenarios are delayed by 0.1s. Average resolution time is 0.02s.




- resolution takes less time;
- speed-up in rate of scenario treated: $\times$ 2.5.

# Asynchronous Progressive Hedging

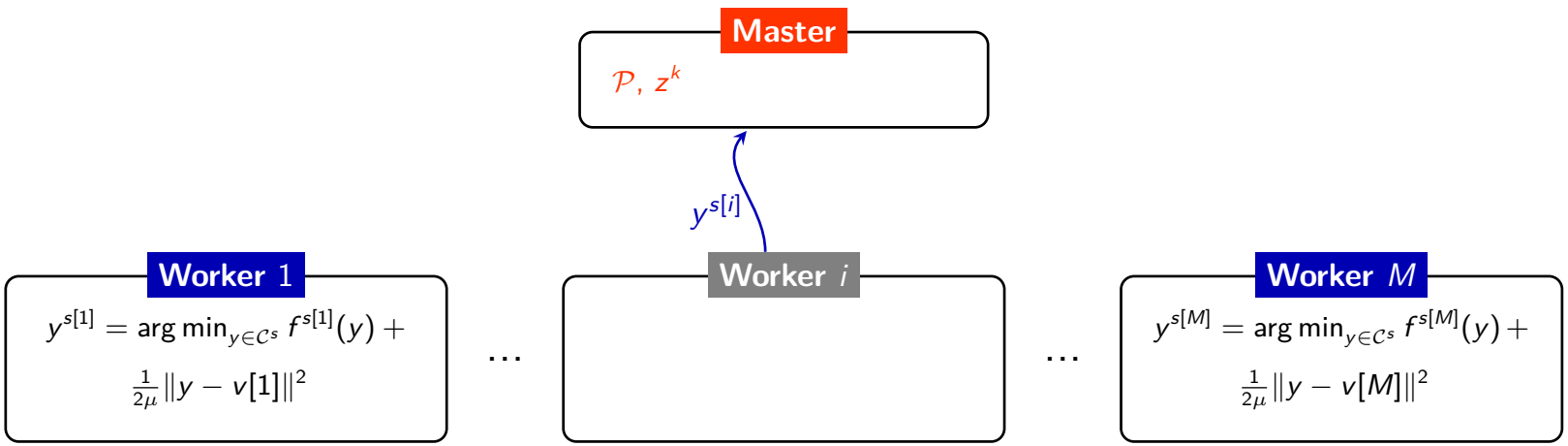
At given time, almost ***all workers work*** on (outdated) data.
Worker returns a step valid for a past version of master variable...Theory advices to do the step carefully, i.e. with a smaller steplength.




- Hannah and Yin: More iterations per second, same quality - Why asynchronous algorithms may drastically outperform traditional ones (2017)

# Asynchronous Progressive Hedging

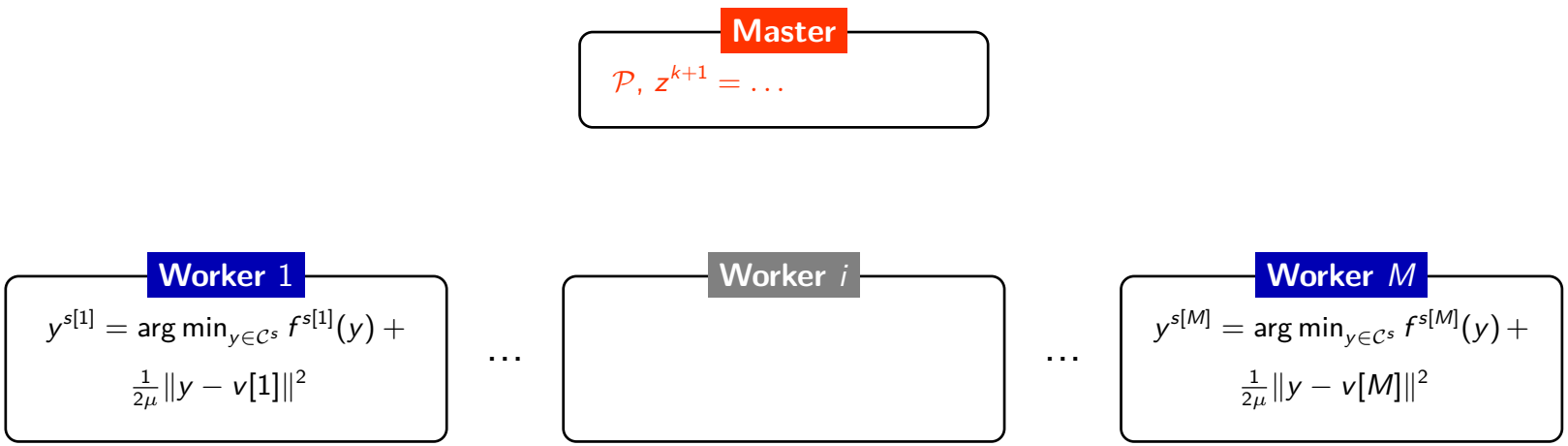
At given time, almost ***all workers work*** on (outdated) data.
Worker returns a step valid for a past version of master variable...Theory advices to do the step carefully, i.e. with a smaller steplength.




- Hannah and Yin: More iterations per second, same quality - Why asynchronous algorithms may drastically outperform traditional ones (2017)

# Asynchronous Progressive Hedging

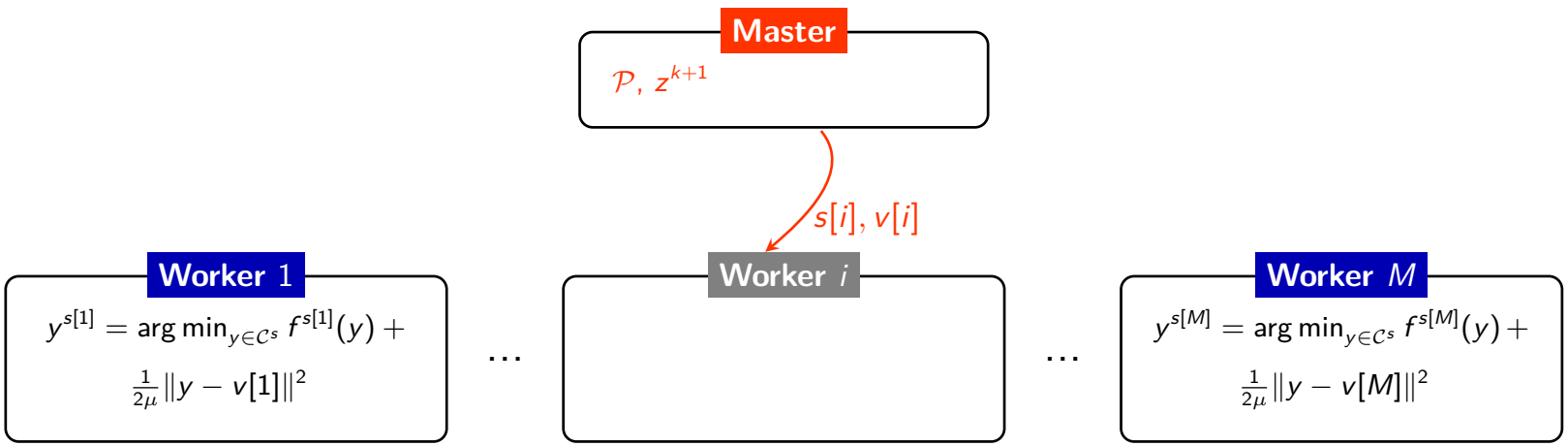
At given time, almost ***all workers work*** on (outdated) data.
Worker returns a step valid for a past version of master variable...Theory advices to do the step carefully, i.e. with a smaller steplength.




- Hannah and Yin: More iterations per second, same quality - Why asynchronous algorithms may drastically outperform traditional ones (2017)

# Asynchronous Progressive Hedging

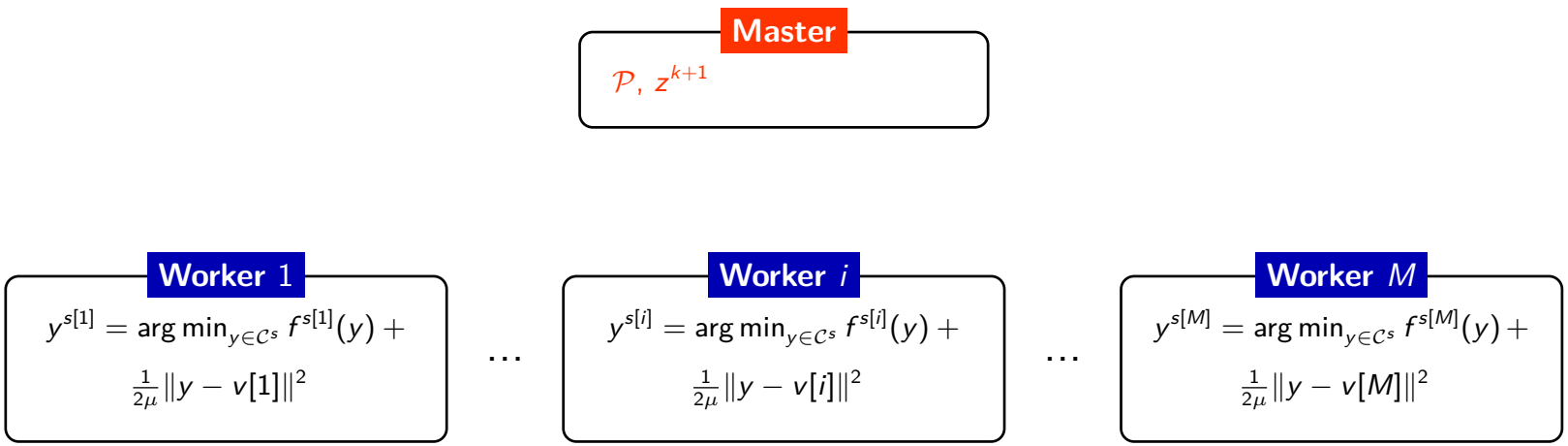
At given time, almost ***all workers work*** on (outdated) data.
Worker returns a step valid for a past version of master variable...Theory advices to do the step carefully, i.e. with a smaller steplength.




- Hannah and Yin: More iterations per second, same quality - Why asynchronous algorithms may drastically outperform traditional ones (2017)

# Asynchronous Progressive Hedging

At given time, almost ***all workers work*** on (outdated) data.
Worker returns a step valid for a past version of master variable...Theory advices to do the step carefully, i.e. with a smaller steplength.




- Hannah and Yin: More iterations per second, same quality - Why asynchronous algorithms may drastically outperform traditional ones (2017)

# Asynchronous Progressive Hedging

**Initialize:** $x^0 = z^0 \in \mathbb{R}^{S\times n}, \mu > 0, k = 0,$

$x[j], v[j] \in \mathbb{R}^n$ and $s[j] = j$ for every worker $j$

Send the scenario/point pair $(s[j], v[j])$ to every worker $j$

**As soon as a worker finishes its computation:**

$$\begin{cases} \text{Receive } y^{s[i]} \text{ from a worker, say } i \\ \left| \begin{array}{l} z^{k+1,s[i]} = z^{k,s[i]} + \frac{2\eta^k}{Sq_{s[i]}}\left(y^{s[i]} - x[i]\right) \\ z^{k+1,s} = z^{k,s} \text{ for all } s \neq s[i] \end{array} \right. \\ \text{Draw a new scenario for } i : s[i] \in \{1,\dots,S\} \text{ with probability } \mathbb{P}[s[i] = s] = q_s \\ x[i] = \left[\mathbf{Proj}_{\mathcal{W}}(z^{k+1})\right]^{s[i]} \\ v[i] = 2x[i] - z^{k+1,s[i]} \\ \text{Send the scenario/point pair } (s[i], \hat{v}[i]) \text{ to worker } i \end{cases}$$

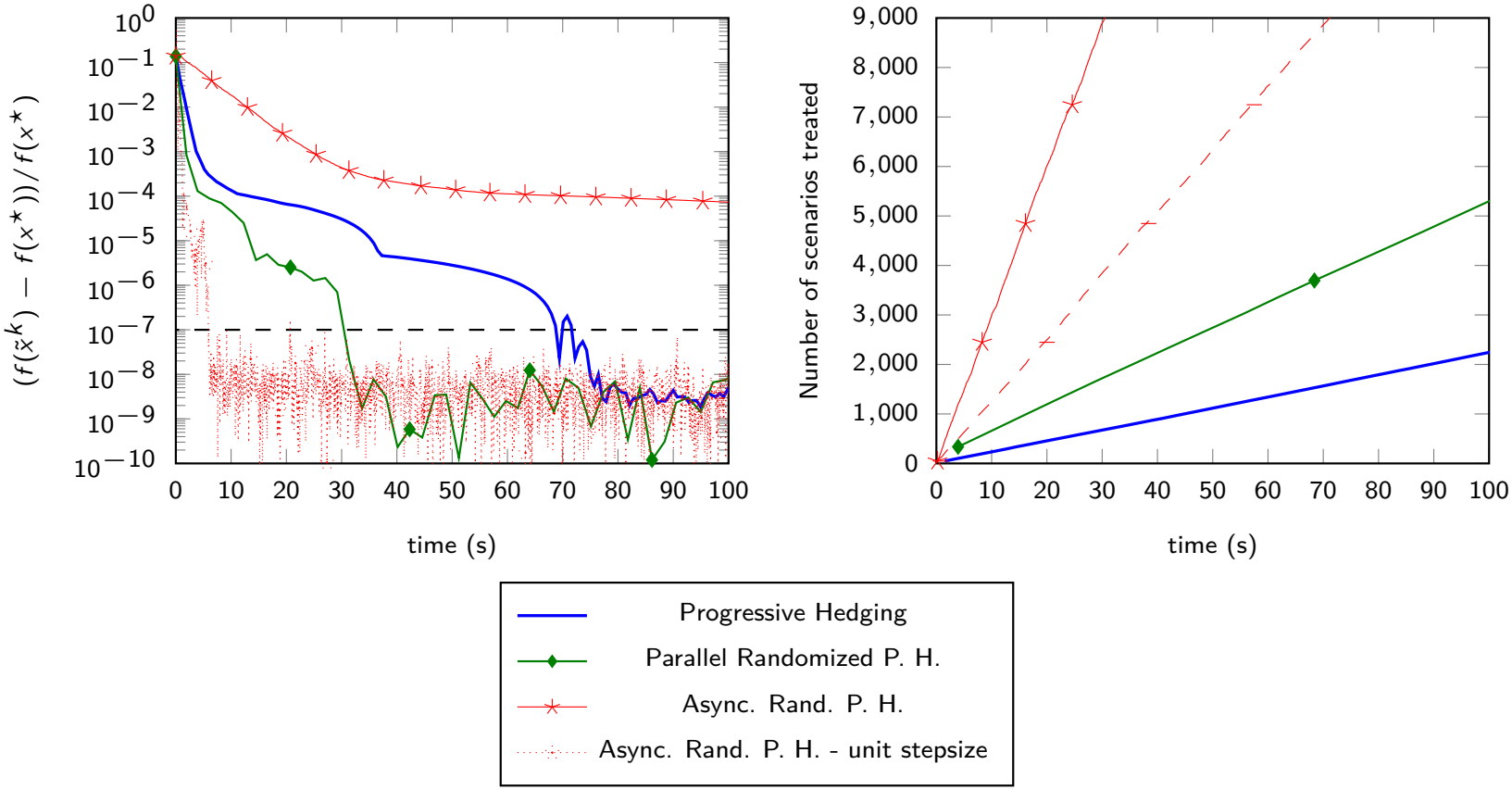
**Return:** $\tilde{x}^{k+1} = \mathbf{Proj}_{\mathcal{W}}(z^{k+1})$

**As soon as a scenario/point pair is received:**

$$\begin{cases} \text{Receive scenario/point pair } (s[i], v[i]) \\ y^{s[i]} = \arg\min_{y\in\mathcal{C}^s} \left\{ f^{s[i]}(y) + \frac{1}{2\mu}\|y - v[i]\|^2 \right\} \\ \text{Send } y^{s[i]} \text{ to the Master} \end{cases}$$

# Asynchronous Progressive Hedging – Numerical experiments

Same problem / computing structure as parallel experiments.




- Theoretical stepsize is overly pessimistic; 1 works well on a lot of cases;
- Speed-up in rate of scenario treated: $\times 5.8$, $\times 12.6$.

# Conclusion and perspectives

## Summary

- randomized PH alleviates the constraint of solving all scenarios before updating
- for fairly homogeneous problems, parallel PH provides significant speed-up
- for inhomogeneous problems or computing systems, asynchronous works well

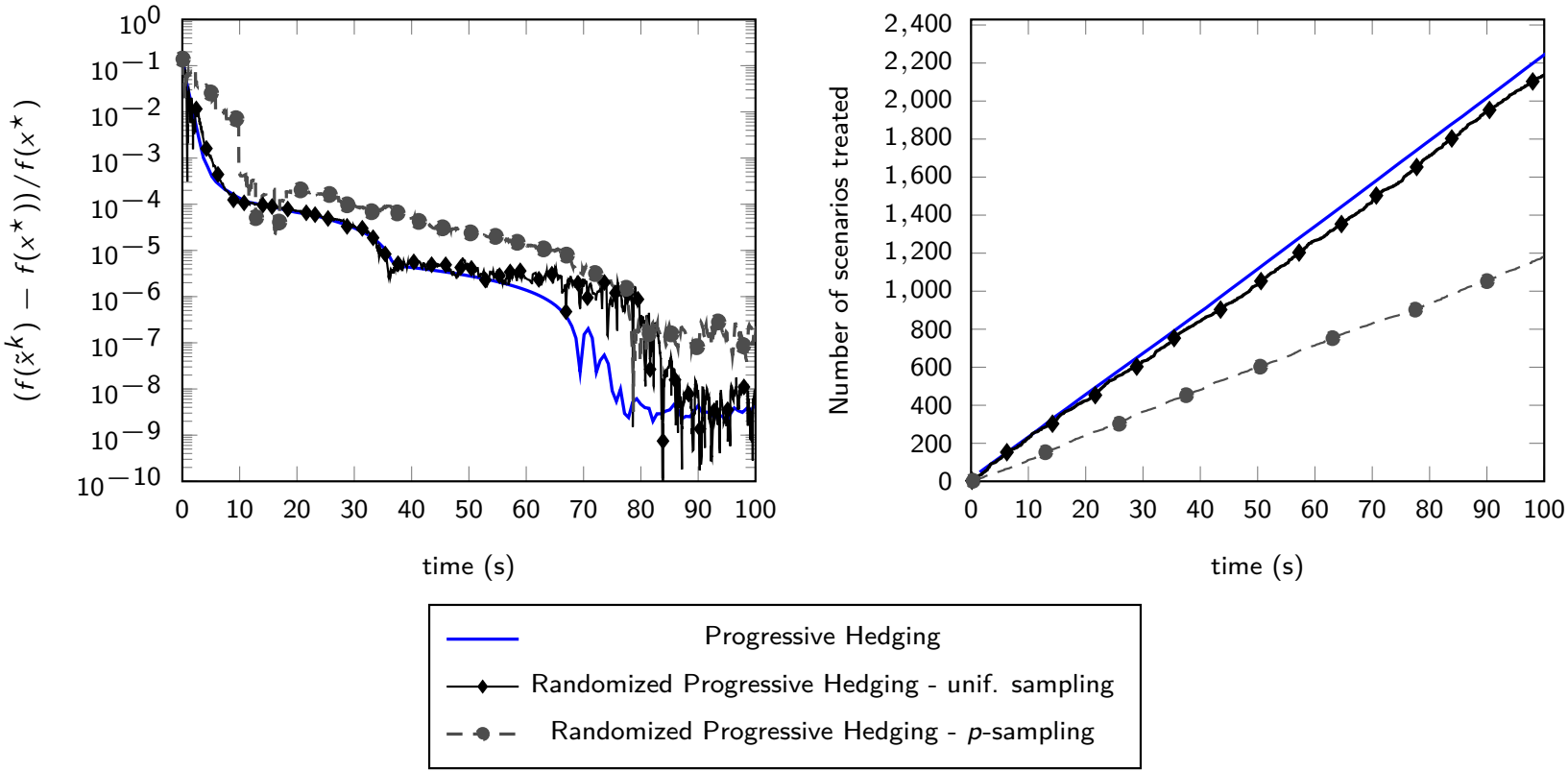
## Perspectives

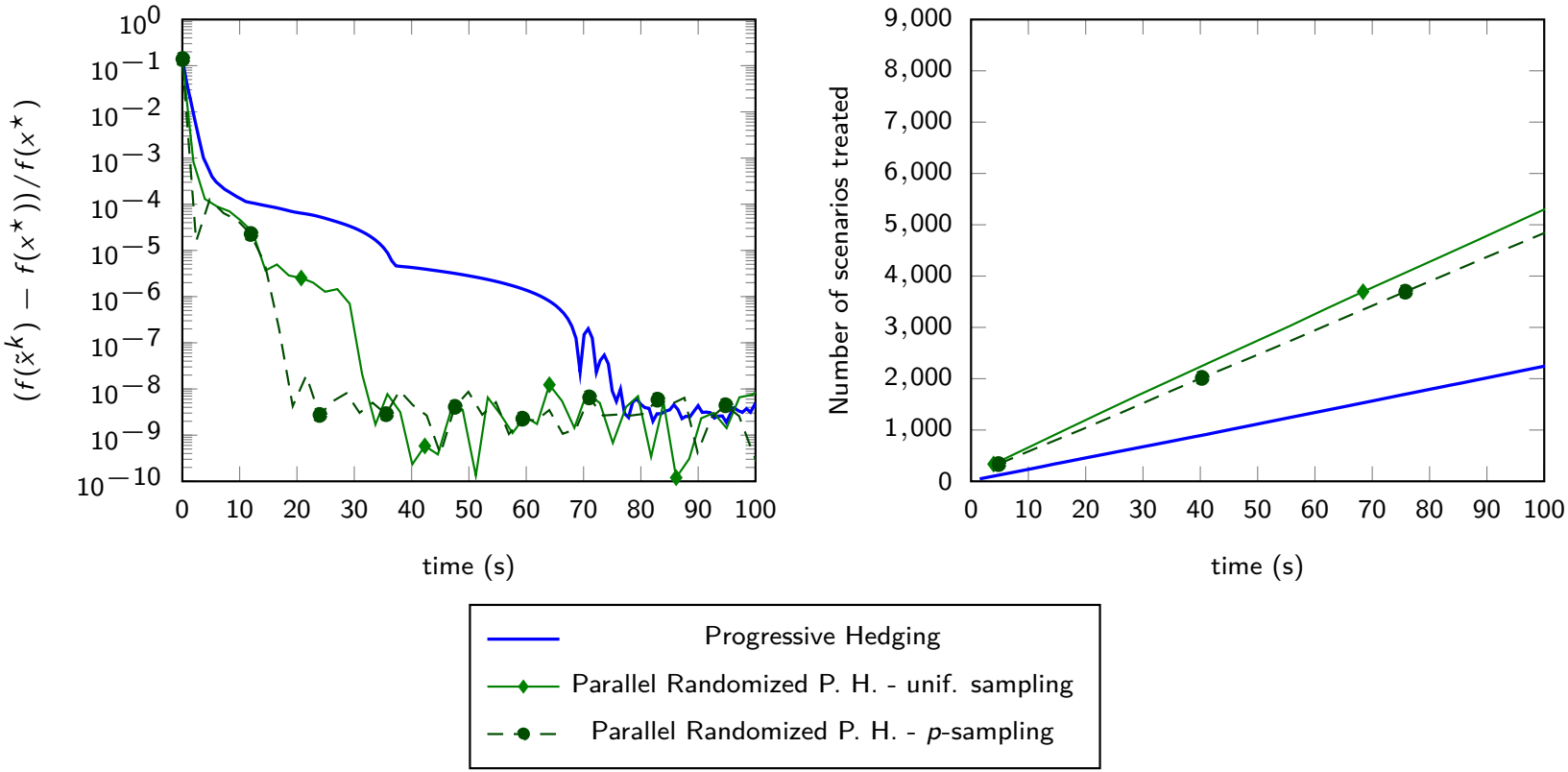
PH is used as heuristic for mixed integer problems, this direction is left as future work.

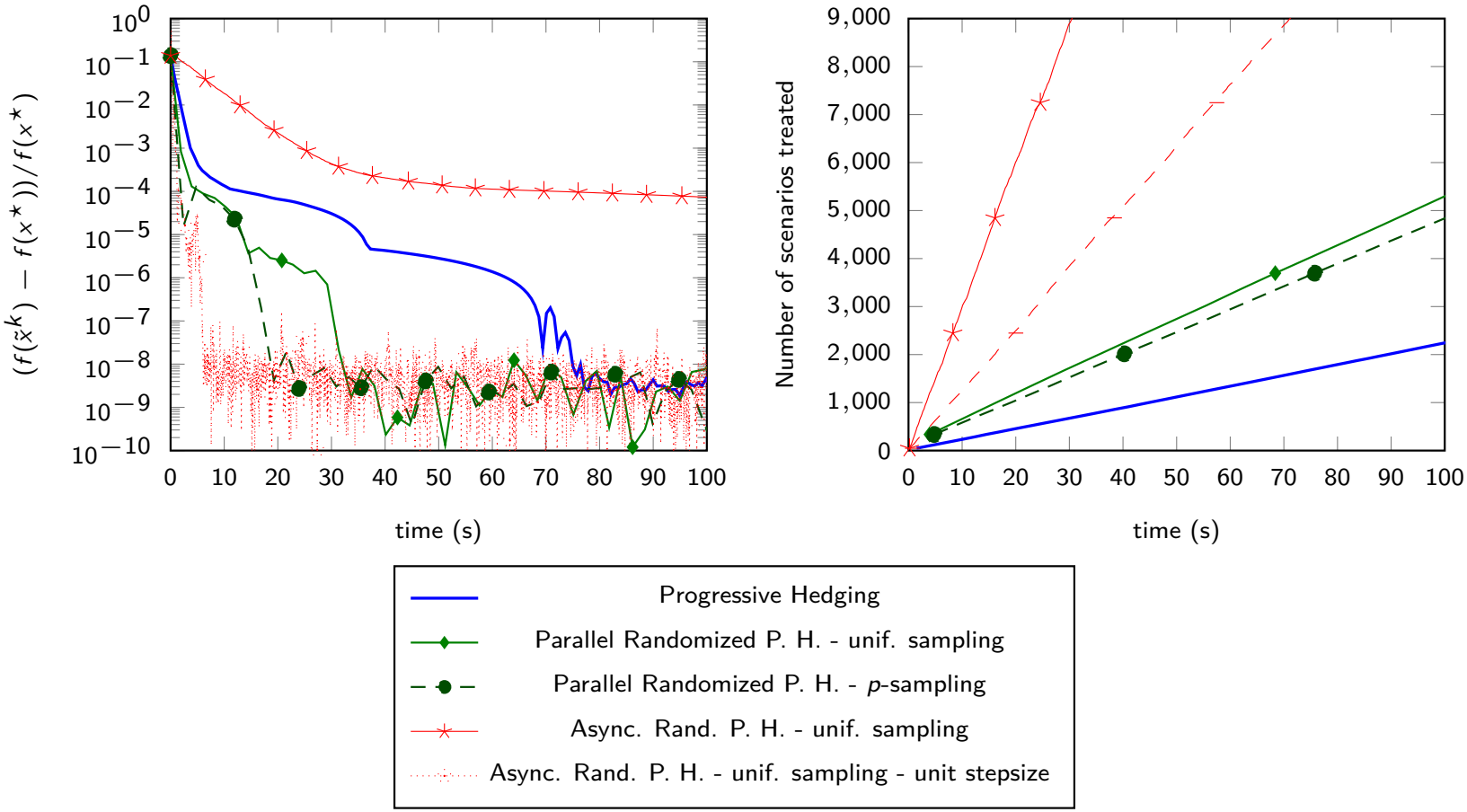
`github.com/yassine-laguel/RandomizedProgressiveHedging.jl`

Thank you!

# Numerical experiments, again


$(f(x^k) - f(x^\star))/f(x^\star)$
$10^{0}$
$10^{-1}$
$10^{-2}$
$10^{-3}$
$10^{-4}$
$10^{-5}$
$10^{-6}$
$10^{-7}$
$10^{-8}$
$10^{-9}$
$10^{-10}$
0 10 20 30 40 50 60 70 80 90 100
time (s)
Number of scenarios treated
2,400
2,200
2,000
1,800
1,600
1,400
1,200
1,000
800
600
400
200
0
0 10 20 30 40 50 60 70 80 90 100
time (s)
Progressive Hedging
Randomized Progressive Hedging - unif. sampling
Randomized Progressive Hedging - $p$-sampling


$(f(\bar{x}^k) - f(x^\star))/f(x^\star)$
$10^0$
$10^{-1}$
$10^{-2}$
$10^{-3}$
$10^{-4}$
$10^{-5}$
$10^{-6}$
$10^{-7}$
$10^{-8}$
$10^{-9}$
$10^{-10}$
0 10 20 30 40 50 60 70 80 90 100
time (s)
Number of scenarios treated
9,000
8,000
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0
0 10 20 30 40 50 60 70 80 90 100
time (s)
Progressive Hedging
Parallel Randomized P. H. - unif. sampling
Parallel Randomized P. H. - $p$-sampling


$(f(x^k) - f(x^\star))/f(x^\star)$
$10^{0}$
$10^{-1}$
$10^{-2}$
$10^{-3}$
$10^{-4}$
$10^{-5}$
$10^{-6}$
$10^{-7}$
$10^{-8}$
$10^{-9}$
$10^{-10}$
0 10 20 30 40 50 60 70 80 90 100
time (s)
Number of scenarios treated
9,000
8,000
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0
0 10 20 30 40 50 60 70 80 90 100
time (s)
Progressive Hedging
Parallel Randomized P. H. - unif. sampling
Parallel Randomized P. H. - $p$-sampling
Async. Rand. P. H. - unif. sampling
Async. Rand. P. H. - unif. sampling - unit stepsize