`mpi-sppy`: Optimization Under Uncertainty for Pyomo

Bernard Knueven
David Mildebrath, Christopher Muir, John Siirola,
David Woodruff, Jean-Paul Watson

INFORMS Annual Meeting 2020

November 13, 2020

Introduction

`mpi-sppy` provides support for *scenario-based* optimization under uncertainty with support for

- massive parallelism
- convergence based on multiple upper and lower bounds.

`mpi-sppy`:

- “mpi” – we utilize MPI functions through `mpi4py`
- “sp” – Stochastic Programming
- “py” – Implemented in Python

Basic requirements:

- Deterministic-equivalent Pyomo model
- Function to create a scenario instance of said Pyomo model
- See David Woodruff’s talk in TD34 for a how-to
- `mpi4py` with an MPI implementation to utilize most functionality

Available: http://github.com/Pyomo/mpi-sppy

Overview

1 Architecture

2 Algorithms & Cylinders

3 Case Study

4 Conclusion

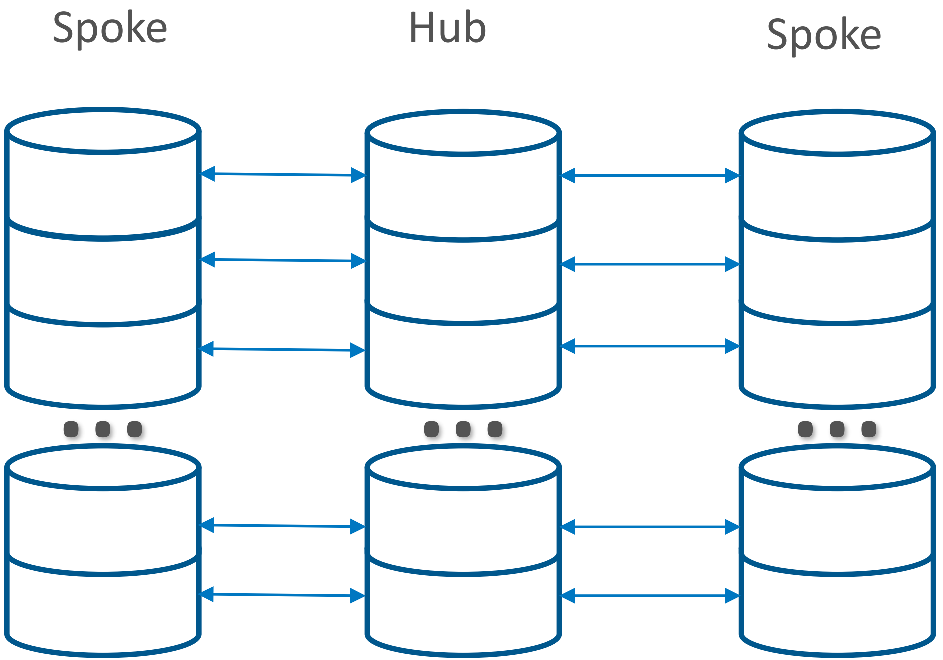
Architecture

The `mpi-sppy` architecture is divided into *cylinders* of compute units

– Typically synchronous communication within a cylinder
– Asynchronous communication between cylinders

The “Hub” cylinder carries out some iterative algorithm, and the “Spoke” cylinder(s) help the hub

– Bound computation
– Cutting planes



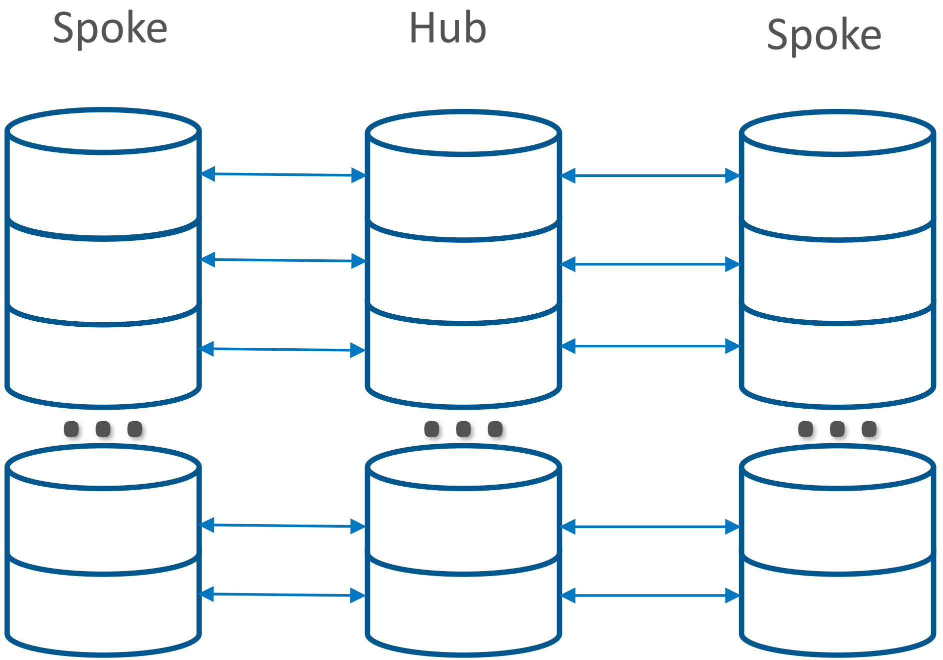
Architecture

One-to-one correspondence between a hub rank and its associated spoke rank(s), collectively called *strata*

- Within a strata, the hub and spoke ranks process the same scenarios

Two types of convergence:

- Traditional termination or convergence of Hub algorithm
- Inner and outer bounds as computed by Spokes is sufficiently small



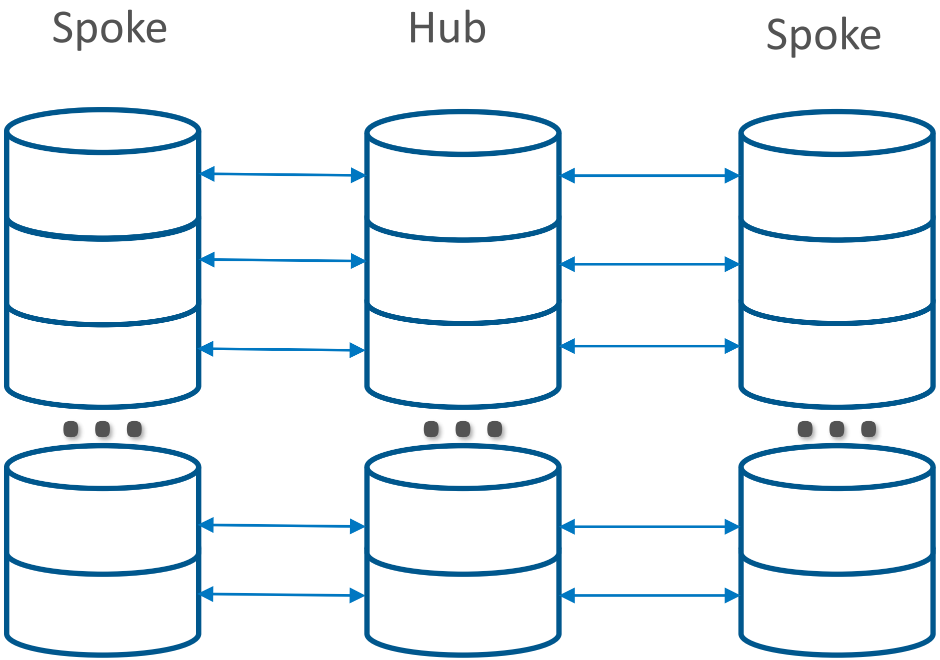
Architecture

Intra-cylinder communication is done through MPI reductions

- `mpi-sspy` utilizes the combined functionality of `mpi4py` and `numpy` such that the reductions occur on C arrays for speed and efficiency

Intra-strata (inter-cylinder) communication utilizes MPI Window objects for one-sided communication

- Passing happens using C arrays
- Generally non-blocking
- Spokes can read new information from hub when ready
- Hub acts on new information from spokes when ready



Algorithms & Cylinders

Hub Algorithms

- Progressive Hedging
- Asynchronous Projective Hedging
- L-Shaped Method[1]

Spoke Algorithms

- Frank-Wolfe Progressive Hedging[1] (dual bound)

Spoke Helpers

- Lagrangian (dual bound)
  - Uses subgradiants on non-anticipatory constraints computed by PH
- Lagranger (dual bound)
  - Computes subgradiants separately from PH
- Xhatters (primal bound)
  - Use non-anticipative decisions from Hub algorithms
  - Xhat-Specific, Xhat-Shuffle[1], Xhat L-Shaped[1]
- Slam Heuristics[1] (primal bound, PH)
  - Slam non-anticipative decisions to max/min of scenario solutions
- Cross-scenario Cuts[1] (PH)

[1] Two-stage problems only

Case Study

Stochastic Unit Commitment

- Schedule thermal generators (on/off) to meet uncertain load and supply from wind generators.
- Two-stage:
  - Stage 1: determine on/off status of thermal generators
  - Stage 2: dispatch thermal/wind generators for realized load and wind availability

Case Study

Stochastic Unit Commitment

- Thermal fleet based on WECC-240
  - 85 thermal generators
  - 48-hour time-horizon
- 1,000 aggregated wind scenarios based on CAISO data
  - Created using `mape_maker` (https://github.com/mape-maker/mape-maker)
  - Wind as percentage of load: 0-46%; maximal single-period difference: 45%
- Deterministic equivalent problem formulated using EGRET's unit commitment models (https://github.com/grid-parity-exchange/Egret)
  - 61833 constraints, 54805 variables (20533 binary), 226235 non-zeros
  - 4080 binary first-stage (non-anticipative) variables

Case Study

Stochastic Unit Commitment

- Full scenario decomposition using PH as Hub algorithm
  - 1 subproblem : 1 scenario
  - PH “Fixer” extension (fixed “converged” non-anticipative variables)
  - custom rho setter
- XhatShuffleLooper Spoke: discover incumbent solutions
- Lagrangian Spoke: Dual bounds from PH-calculated subgradients (Gade et al. 2016)
- FW-PH Spoke: Dual bounds using the method from Boland et al. (2018)
- 1000 subproblems with 4 cylinders: utilize up to 4000 MPI ranks

Case Study

Stochastic Unit Commitment

- Tested on NREL's HPC platform Eagle
- 36 cores per node; 223 nodes
- 4000 MPI ranks; 1000 per cylinder
- Subproblem solver Xpress (limited to 2 threads)
  - Using 8000 cores of the 8028 available
- 100 PH iterations (fixed by negative convergence criterion)

Case Study

```
[    0.00] Initializing mpi-sppy
[   36.07] Start SPBase.__init__
[   39.94] Start PHBase.__init__
[   40.02] Starting spcomm.main()
[   40.34] Creating solvers
[  177.43] Entering solve loop in PHBase.Iter0
[  187.54] Iter.            Best Bound  Best Incumbent      Rel. Gap        Abs. Gap
[  187.54]      1  L        47031.9895             inf           inf             inf
[  190.90]      2    X      47031.9895   51662636.7443   109745.7396   51615604.7547
[  193.53]      3    X      47031.9895      47300.5886        0.5711        268.5991
[  196.79]      4    X      47031.9895      47117.3894        0.1816         85.3999
[  199.46]      5    X      47031.9895      47111.5210        0.1691         79.5315
[  202.05]      6    X      47031.9895      47094.9459        0.1339         62.9564
[  204.85]      7           47031.9895      47094.9459        0.1339         62.9564
[  208.67]      8    X      47031.9895      47093.4180        0.1306         61.4285
[  210.47]      9           47031.9895      47093.4180        0.1306         61.4285
[  212.66]     10           47031.9895      47093.4180        0.1306         61.4285
[  214.00]     11           47031.9895      47093.4180        0.1306         61.4285
[  215.19]     12           47031.9895      47093.4180        0.1306         61.4285
                                 • • •
[  271.36]     96           47031.9895      47091.1755        0.1258         59.1860
[  271.93]     97           47031.9895      47091.1755        0.1258         59.1860
[  272.49]     98           47031.9895      47091.1755        0.1258         59.1860
[  273.06]     99           47031.9895      47091.1755        0.1258         59.1860
[  273.62]    100           47031.9895      47091.1755        0.1258         59.1860
[  274.10] Reached user-specified limit=100 on number of PH iterations
                                 • • •
[  274.23] Hub algorithm complete, waiting for termination barrier
[  310.03]
[  310.03] Statistics at termination
[  310.03] Iter.            Best Bound  Best Incumbent      Rel. Gap        Abs. Gap
[  310.03]    100           47031.9895      47091.1755        0.1258         59.1860
[  310.13] Windows freed
```

Warm-up to warm-down computation time: 97 sec.

Post warm-up time: 133 sec.

Total time: 310 sec.

Final gap: 0.1258%

Case Study

Stochastic Unit Commitment

- Same computation on LLNL's Quartz cluster
- 36 cores per node; 256 nodes
- 4000 MPI ranks; 1000 per cylinder
- Subproblem solver Gurobi (limited to 2 threads)
  - Using 8000 cores of the 9216 available
- 100 PH iterations (fixed by negative convergence criterion)

$\lvert S \rvert$	Time to PH iter. 0 solve loop (s)	Time to completion (s)	Δ	PH LB	PH UB
1000	69.49	182.92	113.43	47032.130	47097.578

Case Study

1000- vs. 3-scenario instance on Eagle

|S| (Iter)	1000 (0)	3 (0)	1000 (25)	3 (25)	1000 (50)	3 (50)	1000 (75)	3 (75)
Total iteration time	10.11 s	2.89 s	0.74 s	0.39 s	0.58 s	0.39 s	0.56 s	0.45 s
Pyomo & solver time	8.74 s	1.58 s	0.54 s	0.27 s	0.39 s	0.26 s	0.39 s	0.29 s
Difference	1.37 s	1.31 s	0.20 s	0.12 s	0.19 s	0.13 s	0.17 s	0.16 s

Very low additional inter-iteration overhead scaling from 3 scenarios on a single node to 1,000 scenarios on 200+ nodes

Conclusion

Available: http://github.com/Pyomo/mpi-sppy

Several examples (farmer, SSLP, unit commitment, network design, others) and documentation available:

– See David Woodruff's talk (TD34)

– UC driver used in the computation section can optionally use most of the functionality of mpi-sppy with progressive hedging; only ~400 lines of (unoptimized) Python over deterministic model.

– Easy to get started with existing two-stage PySP model

With enough compute power, `mpi-sppy` enables the solution of very large-scale stochastic optimization problems

Q&A

www.nrel.gov

NREL/PR-2C00-78043

This work was authored in part by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided in part by the U.S. Department of Energy Advanced Research Projects Agency - Energy. A portion of this research was performed using computational resources sponsored by the Department of Energy's Office of Energy Efficiency and Renewable Energy and located at the National Renewable Energy Laboratory. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government.