Hedging Against Uncertainty: A Modeling Language and Solver Library

You Plan

Stuff Happens

You Adjust

More Stuff Happens

PySP: Stochastic Programming in Python

Multi-Stage Planning for Uncertain Environments
- Explicitly capture recourse
- Uncertainty modeling framework
- Integrated solver strategies

What We Do:
- Mixed decision variables
  - Continuous
  - Integer/Binary
- General multi-stage
- Stochastic programming
  - Expected value
  - Conditional Value-at-Risk
  - Scenario selection
  - Cost confidence intervals

How We Do It:
- Deterministic equivalent
- Scenario-based decomposition
  - Progressive Hedging
  - Customizable accelerators
- Algebraic modeling via Pyomo
- SMP and cluster parallelism
- Integrated high-level language support
- Multi-platform, unrestricted license
- Open source, actively supported by Sandia
- Co-Managed by Sandia and COIN-OR

TO LEARN MORE VISIT > https://software.sandia.gov/trac/coopr/wiki/PySP

Sandia National Laboratories is a multi-program laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin company, for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.
Progressive Hedging: Basic Pseudo-Code

1. $k := 0$

2. For all $s \in S$, $x_s^{(k)} := \arg\min_x (c \cdot x + f_s \cdot y_s) : (x, y_s) \in Q_s$

3. $\bar{x}^k := (\sum_{s \in S} p_s d_s x_s^{(k)}) / \sum_{s \in S} p_s d_s$

4. For all $s \in S$, $w_s^{(k)} := \rho(x_s^{(k)} - \bar{x}^{(k)})$

5. $k := k + 1$

6. For all $s \in S$, $x_s^{(k)} := \arg\min_x (c \cdot x + w_s^{(k-1)} x + \rho/2 \|x - \bar{x}^{(k-1)}\|^2 + f_s \cdot y_s) : (x, y_s) \in Q_s$

7. $\bar{x}^{(k)} := (\sum_{s \in S} p_s d_s x_s^{(k)}) / \sum_{s \in S} p_s d_s$

8. For all $s \in S$, $w_s^{(k)} := w_s^{(k-1)} + \rho \left(x_s^{(k)} - \bar{x}^{(k)}\right)$

9. $g^{(k)} := \frac{(1-\alpha)|S|}{\sum_{s \in S} p_s d_s} \sum_{s \in S} \|x_s^{(k)} - \bar{x}^{(k)}\|$

10. If $g^{(k)} < \epsilon$, then go to step 5. Otherwise, terminate.

Rockafellar and Wets (1991)
Parallelization and Scenario-Based Decomposition

• Progressive Hedging is “trivially” parallelizable
  – Each batch of sub-problem solves is independent
  – So what’s the big deal?

• Maintaining parallel efficiency is a major issue in any practical implementation

• Key problem drivers
  – High variability in sub-problem solve times
  – Sub-problem solves too fast => communication dominates

• Key solution strategies
  – Relaxing barrier synchronization after each batch of sub-problem solves
  – Scenario “bundling” to increase sub-problem difficulty and to accelerate PH convergence
Asynchronous Sub-Problem Solves in PH

- In the case of mixed-integer optimization problems, variability of sub-problem solve times can be considerable
  - Observations “in the wild” vary over 4 or more orders of magnitude

- The presence of such dramatic variability clearly destroys any potential benefit of parallelism in PH

- Our solution
  - Relax the barrier synchronization, allow for asynchronous solves
  - Retains PH convergence properties, as long as sub-problem solves for each scenario periodically report back

- Challenges and Results
  - Significant interference with mixed-integer acceleration mechanisms
  - Slows PH convergence, but (empirically) only by a constant factor
Scenario Bundling in PH

• General idea
  – Cluster scenarios using some similarity (or dis-similarity) metric
  – Forming miniature “extensive forms”

• Benefits
  – Increases sub-problem solve times, dropping comm:compute ratio
  – (Often) dramatic accelerations in PH convergence

• Research questions
  – Do we bundle based on maximal similarity or maximal differences?
  – How to handle bundling in multi-stage scenario trees?

• Preliminary results
  – Even pairing of scenarios randomly yields very large reductions in the number of PH iterations required for convergence
Driver Applications for Asynch PH and Bundling

- Stochastic Unit Commitment
  - Two and multi-stage stochastic mixed-integer

- Transmission and Generation Expansion
  - Two and multi-stage stochastic mixed-integer

- Parameter estimation
  - Childhood disease models (SIR)
  - Two and multi-stage stochastic non-linear

- Network design
  - Academic, but very difficult (two-stage mixed-integer stochastic programs)

- Forestry management
  - Multi-stage mixed-integer; determining harvest schedule
Software and Contact Information

• All of these techniques are available in our PySP open-source software package
  – https://software.sandia.gov/trac/coopr/wiki/PySP
  – Distributed by Sandia and COIN-OR
  – Jointly developed and maintained by Sandia and UC Davis

• Asynchronous PH and bundling interfaces are currently supported in PySP
  – Alpha, but functional
  – The rest of PySP is rather stable

• Feel free to contact us!
  – Jean-Paul Watson (jwatson@sandia.gov)
  – David L. Woodruff (dlwoodruff@ucdavis.edu)