A Scalable Decomposition Approach for Stochastic Mixed-Integer Programs

Kibaek Kim

Jointly work with Brian Dandurand (ANL), Cosmin Petra (LLNL), and Victor Zavala (UW-Madison)

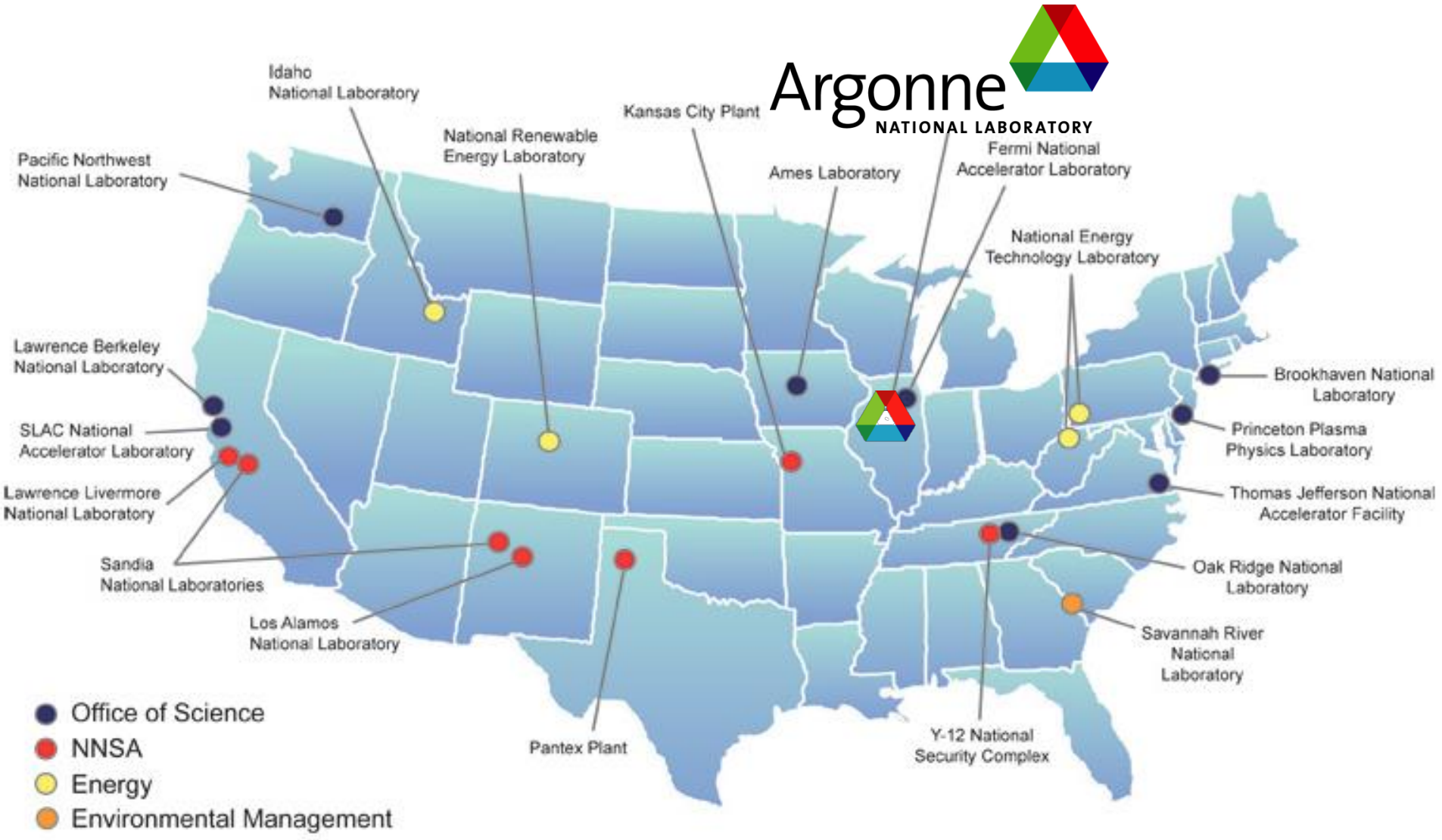
Mathematics and Computer Science Division

Argonne National Laboratory

UG Workshop, Berlin, Germany

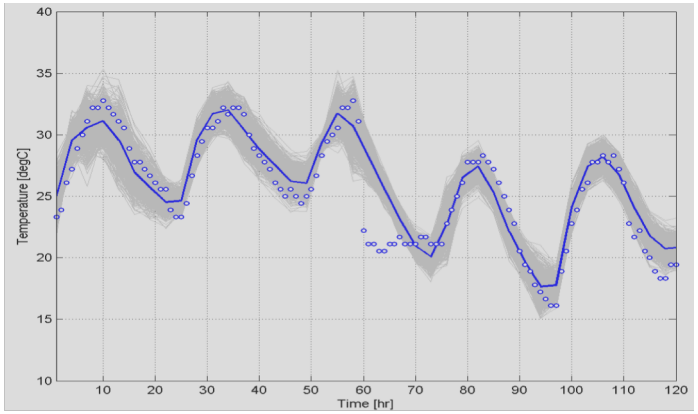
January 15, 2019

Argonne: Vital part of DOE National Laboratory System



Today's Talk is about …

- Stochastic Programming
  - Integer variables
  - Parallel algorithms
  - Software implementation
- Dual Decomposition
  - New parallel algorithms to accelerate solutions
  - Asynchronous parallel computation
  - New branch-and-bound for global optimality
- Numerical Results
  - Open-source software package
  - SIPLIB test instances
  - Test problems for stochastic unit commitment
  - Argonne's high performance computing clusters



$$\begin{array}{llll} Ax_0 & & & = b_0 \\ T_1x_0 & +W_1x_1 & & = b_1 \\ \vdots & & \ddots & \vdots \\ T_Nx_0 & & +W_Nx_N & = b_N \end{array}$$

$$\begin{array}{llll} y_0A & +y_1T_1 & \cdots & +y_NT_N = \pi_0 \\ & y_1W_1 & & = \pi_1 \\ & & \ddots & \vdots \\ & & +y_NW_N & = \pi_N \end{array}$$

Parallel Decomposition Methods for Stochastic MIP

Kim, Kibaek and Victor M. Zavala. "Algorithmic innovations and software for the dual decomposition method applied to stochastic mixed-integer programs." *Mathematical Programming Computation* (2017): 1-42.

Stochastic Mixed Integer Programming (SMIP)

General formulation of SMIP:

$$\begin{aligned} \min \quad & c^T x + \mathbb{E}[Q(x,\omega)] \\ \text{s.t.} \quad & Ax \geq b, \\ & x \in \mathbb{R}_+^{n_1-p_1} \times \mathbb{Z}_+^{p_1} \end{aligned}$$

Make *here-and-now* decision $\mathbf{x}$

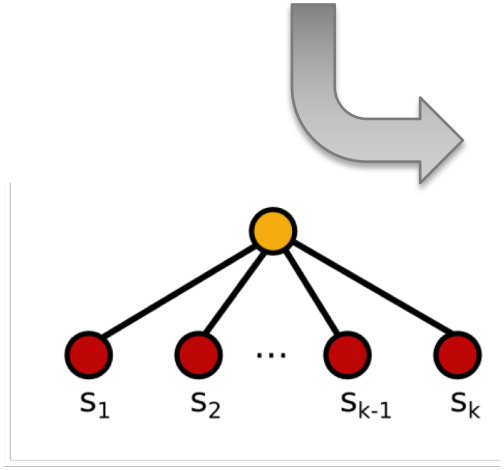
- Operational decisions
- Logical decisions
- Countable items

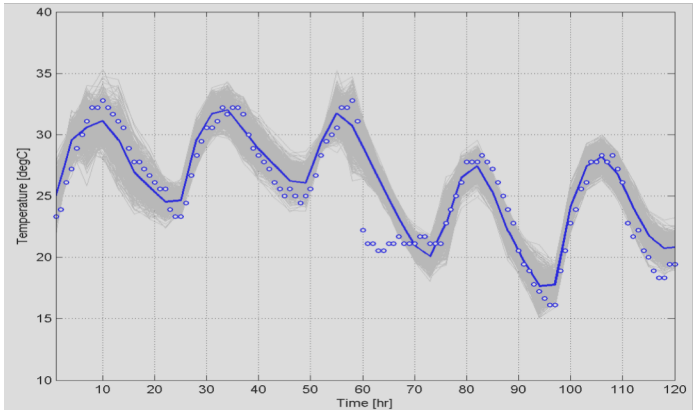
Recourse function of an integer program:

$$\begin{aligned} Q(x,\omega) = \min \quad & q(\omega)^T y \\ \text{s.t.} \quad & W(\omega)y \geq h(\omega) - T(\omega)x, \\ & y \in \mathbb{R}_+^{n_2-p_2} \times \mathbb{Z}_+^{p_2} \end{aligned}$$

Make *wait-and-see* decision $\mathbf{y}$ for given first-stage decision $\mathbf{x}$ and event ω

- Recourse action to event realization (e.g. re-scheduling, system restoration)
- Time-dependent decisions



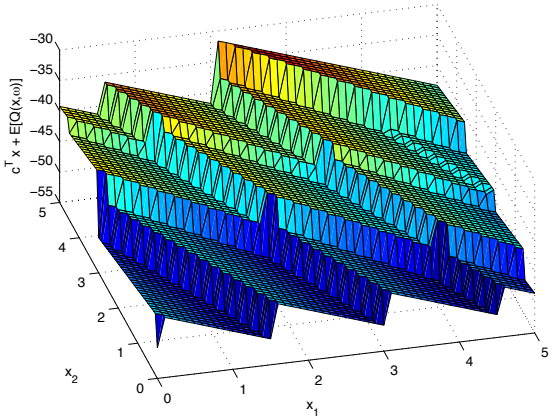


Observe *stochastic event* ω

- System Failure
- Demand and supply
- Cost and price
- Weather

Dual Decomposition for SMIP

$$
\begin{array}{lllllll}
\min & c^T x(\omega_1) & +q(\omega_1)^T y(\omega_1) & +\ldots & +c^T x(\omega_1) & +q(\omega_N)^T y(\omega_N) & \\
\text{s.t.} & Ax(\omega_1) & & & & & \geq b, \\
& T(\omega_1)x(\omega_1) & +W(\omega_1)y(\omega_1) & & & & \geq h(\omega_1) \\
& \vdots & & \ddots & & & \vdots \\
\text{Nonanticipaticity constraints} & & & & Ax(\omega_N) & & \geq b \\
& & & & T(\omega_N)x(\omega_N) & +W(\omega_N)y(\omega_N) & \geq h(\omega_N) \\
& x(\omega_1) & & \ldots & = x(\omega_N) & & \\
& x(\omega_1), & & \ldots & x(\omega_N) & & \in \mathbb{R}_+^{n_1-p_1} \times \mathbb{Z}_+^{p_1} \\
& & y(\omega_1), & \ldots & & y(\omega_N) & \in \mathbb{R}_+^{n_2-p_2} \times \mathbb{Z}_+^{p_2}
\end{array}
$$



- ***Becomes More Challenging!***
  - The recourse function is nonconvex and discontinuous.
  - Benders Decomposition cannot be used.
- **Dual Decomposition**
  - Lagrangian relaxation of the nonanticipativity constraints

$$
\begin{aligned}
z = \min_{x_0, x_j, y_j} \quad & \sum_{j=1}^{N} p_j \left(c^T x_j + q_j^T y_j\right) \\
\text{s.t.} \quad & (x_j, y_j) \in G_j, \quad j = 1, \ldots, N, \\
\text{Nonanticipaticity constraints} \quad & \sum_{j=1}^{N} H_j x_j = 0 \quad (\lambda)
\end{aligned}
$$

$$
\Rightarrow \quad D(\lambda) := \min_{x_j, y_j} \sum_{j=1}^{N} [p_j (c^T x_j + q_j^T y_j) - \lambda^T H_j x_j]
$$
$$
\text{s.t. } (x_j, y_j) \in G_j, \ j = 1, \ldots, N
$$

  - Seek for the best lower bound by solving

$$
z \geq z_{LD} := \max_{\lambda} \sum_{j=1}^{N} D_j(\lambda)
$$

Decomposed in each scenario j

Dual-Search Method 1: Subgradient Method

- ***WIDELY USED*** in non-differentiable optimization
- The dual variable is updated as

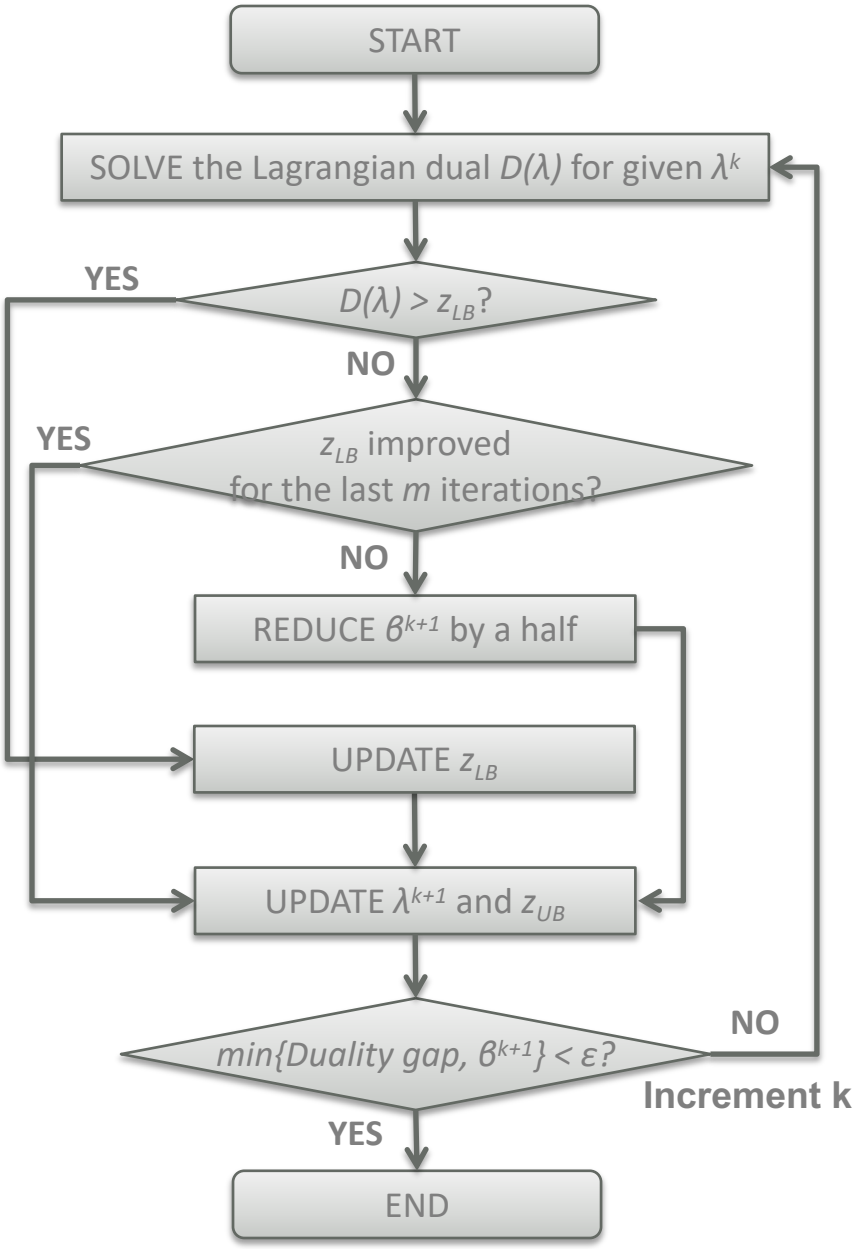
$$\lambda^{k+1} = \lambda^k - \alpha^k \sum_{s \in \mathcal{S}} H_s x_s^k$$

Subgradient of $D(\lambda)$ at λ^k

where the step size rule [1] is given by

$$\alpha_k := \beta_k \frac{z_{\mathrm{UB}} - D(\lambda^k)}{\| \sum_{s \in \mathcal{S}} H_s x_s^k \|_2^2}$$

- No way of proving optimality



1. Fisher, Marshall L. "The Lagrangian relaxation method for solving integer programming problems." *Management science* 50.12_supplement (2004): 1861-1871.

Dual-Search Method 2: Cutting Plane Method

- Outer approximation to solve

$$\max_{\lambda} \sum_{s \in \mathcal{S}} D_s(\lambda)$$

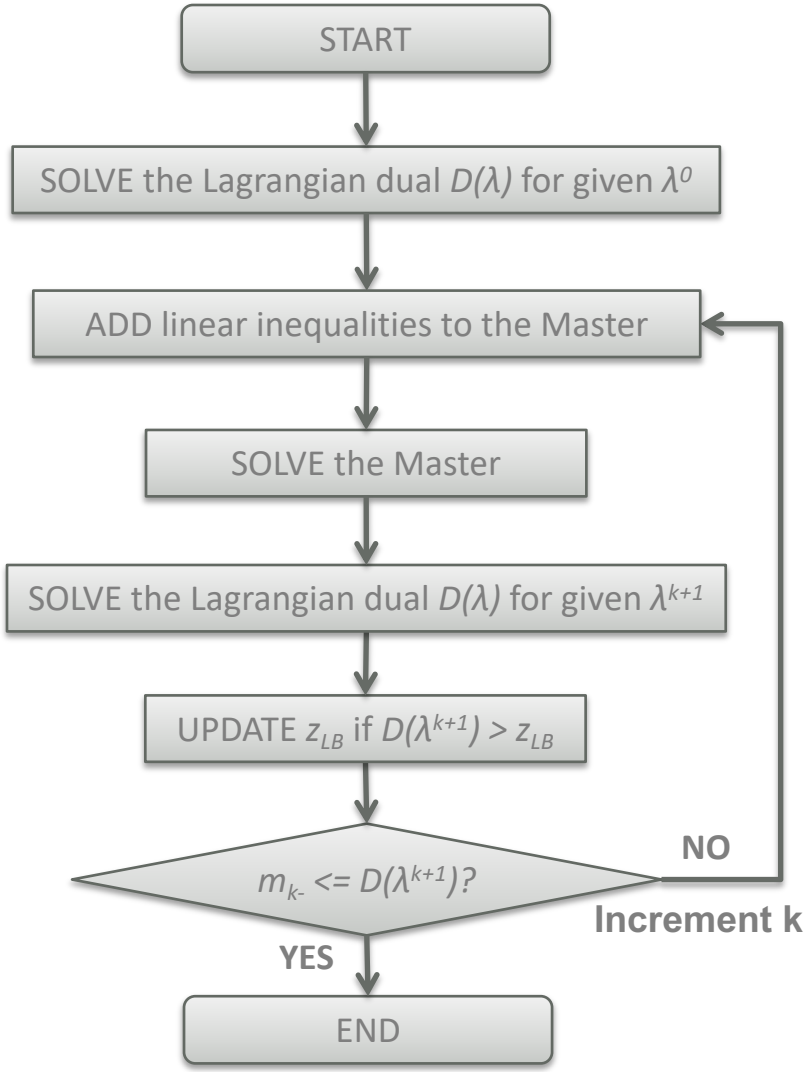
- *Piecewise linear concave* in λ [1]

$$D_s(\lambda) = \min_{x_s, y_s} \quad p_s \left(c^T x_s + q_s^T y_s\right) + \lambda^T \left(H_s x_s\right)$$
$$\text{s.t.} \quad (x_s, y_s) \in \text{conv.hull}(G_s)$$

- The Master problem is given by

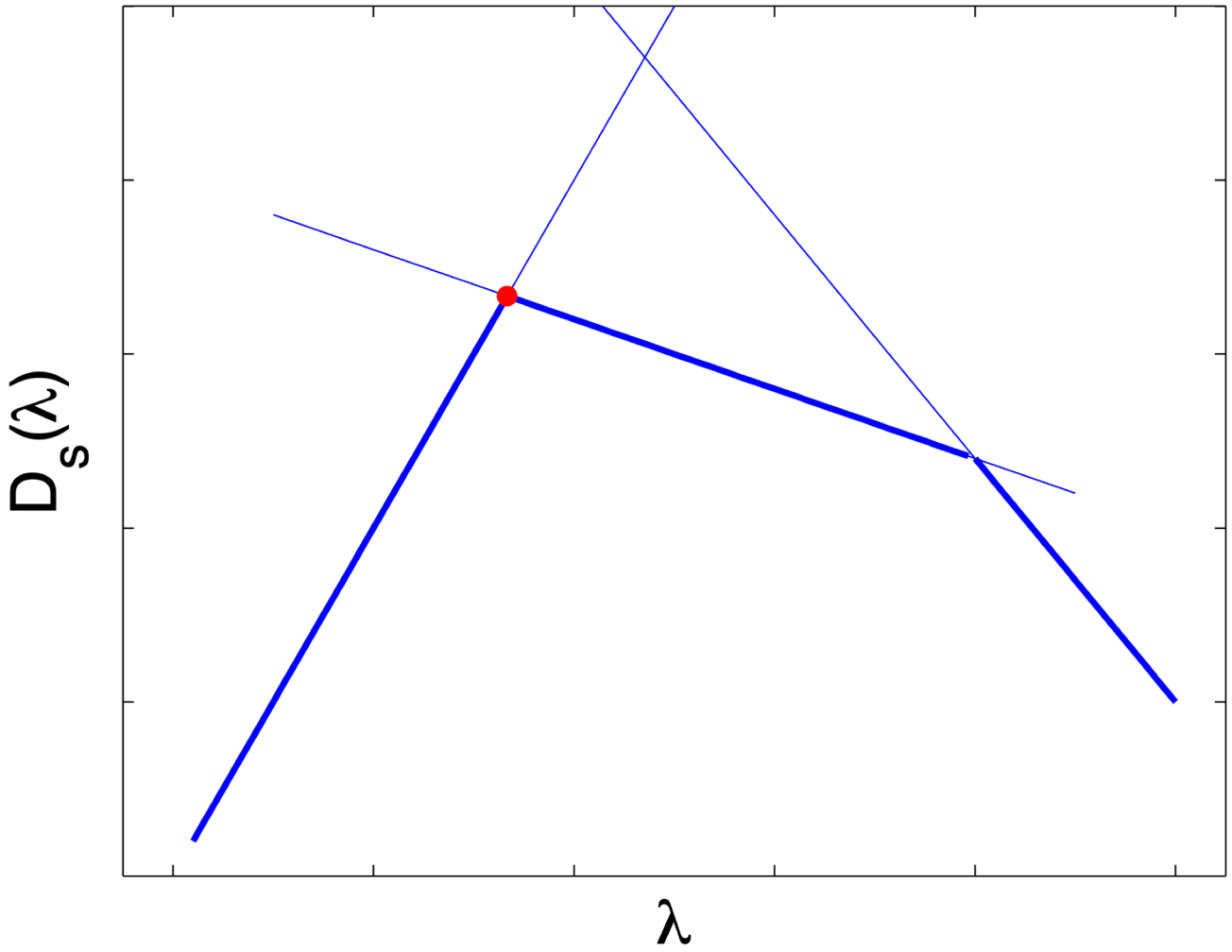
$$m_k = \max_{\theta_s, \lambda} \quad \sum_{s \in \mathcal{S}} \theta_s$$
$$\text{s.t.} \quad \theta_s \leq D_s(\lambda^l) + (H_s x_s^l)^T (\lambda - \lambda^l),$$
$$s \in \mathcal{S},\ l = 0, 1, \ldots, k$$

- *Finite* convergence with *Optimality*



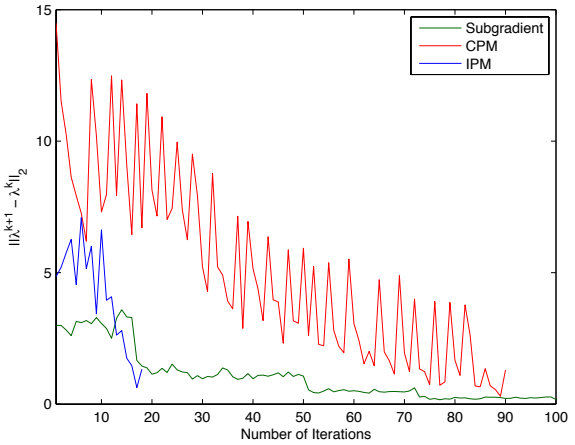
1. Fisher, Marshall L. "The Lagrangian relaxation method for solving integer programming problems." *Management science* 50.12_supplement (2004): 1861-1871.

Dual-Search Method 2: Cutting Plane Method



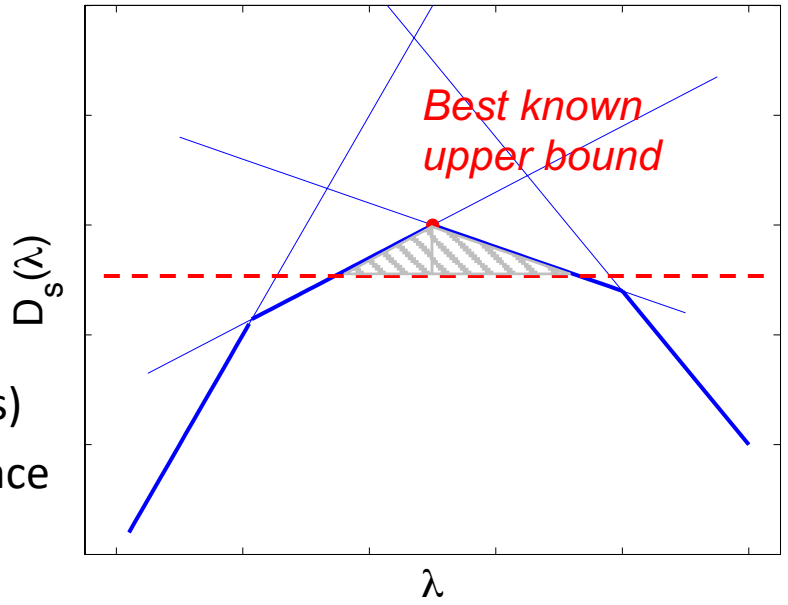
New Dual Decomposition Method

- The cutting-plane solutions of the Master problem can
  - oscillate significantly;
  - suffer from degeneracy.



- **Interior-Point Cutting-Plane Method**
  - Solve the master by using interior-point method to find *feasible solutions* (vs. *extreme* optimal points)
  - Also use the best known upper bound for early termination of IPM iterations

- **Adding a set of valid inequalities:**
  - Benders-type feasibility cuts
  - Benders-type optimality cuts
  - Other cuts can also be added (e.g., cover cuts)
  - Further cut away the suboptimal solution space

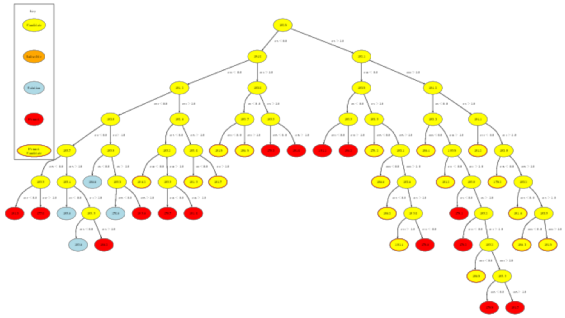


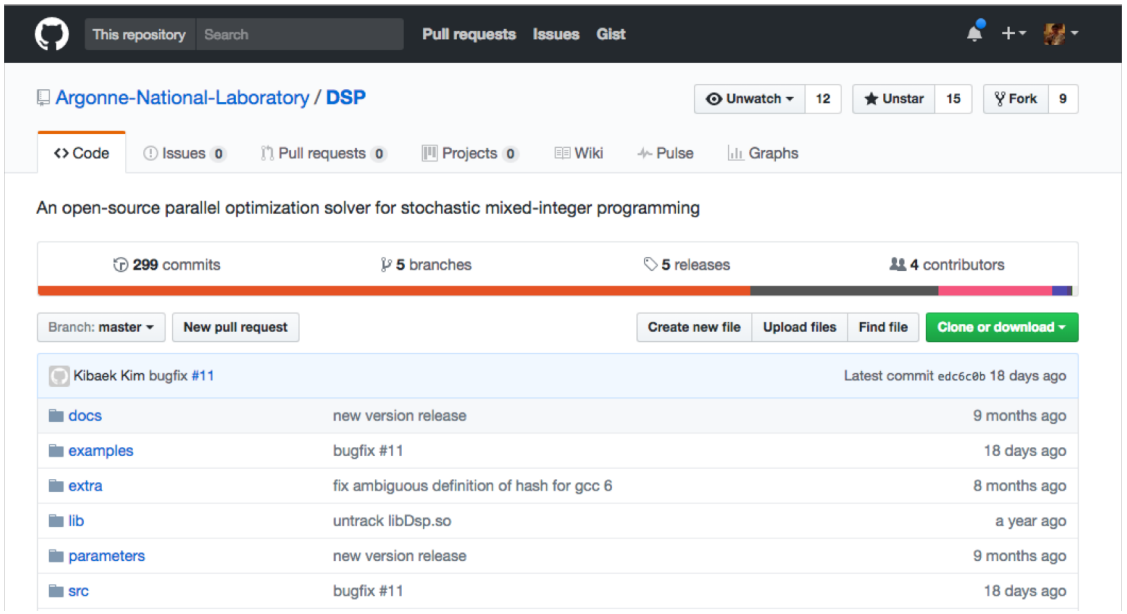
DSP: Scalable Decomposition Solver

- ***Decomposition*** methods for ***Structured Programming***
  - Exploiting block-angular structures
  - **Dantzig-Wolfe decomposition**
    + (Parallel) Branch-and-Bound
  - **Benders decomposition**
  - **Dual decompositio*Parallel***
- Parallel computing via MPI library

$$\begin{array}{llll} Ax_0 & & & = b_0 \\ T_1x_0 & +W_1x_1 & & = b_1 \\ \vdots & & \ddots & \vdots \\ T_Nx_0 & & +W_Nx_N & = b_N \end{array}$$

$$\begin{array}{llll} y_0A & +y_1T_1 \cdots & +y_NT_N & = \pi_0 \\ & y_1W_1 & & = \pi_1 \\ & & \ddots & \vdots \\ & & +y_NW_N & = \pi_N \end{array}$$





NEOS Contact Help Sign In Sign Up

NEOS Interfaces to DSP

Sample Submissions
WWW Form - Email - XML-RPC

DSP

The NEOS Server offers DSP for the solution of stochastic mixed-integer programming problems in SMPS format.

DSP was developed and is maintained by Kibaek Kim and Victor Zavala of the Mathematics and Computer Science Division of Argonne National Laboratory. Additional resources are available at the DSP website. Documentation for DSP is available online.

References:

- Kibaek Kim and Victor M. Zavala. Algorithmic Innovations and Software for the Dual Decomposition Method applied to Stochastic Mixed-Integer Programs". Optimization Online, 2015.
- John R Birge, Michael AH Dempster, Horand I Gassmann, Eldon A Gunn, Alan J King, and Stein W Wallace. A standard input format for multiperiod stochastic linear programs. IIASA Laxenburg Austria, 1987.

Using the NEOS Server for DSP/SMPS

DSP can also read a model provided in SMPS files (see above reference). In this format, a model is defined by three files: core, time, and stochastic with file extensions of .cor, .tim, and .sto, respectively. The *core file* defines the deterministic version of the model with a single reference scenario; the *time file* indicates a row and a column that split the deterministic data and stochastic data in the constraint matrix; the *stochastic file* defines random data.

Users may also optionally provide a parameter file for DSP (default settings). Setting the MIP or QP solvers in this file will override your choices in the form below.

Web Submission Form

DSP Algorithm

Select an algorithm.

- ○ Deterministic equivalent form
- ○ Benders decomposition
- ○ Dual decomposition

DSP reads models from Julia

$$\min \left\{ -1.5\,x_1 - 4\,x_2 + \sum_{s=1}^{3} p_s Q(x_1, x_2, \xi_1^s, \xi_2^s) : x_1, x_2 \in \{0, \ldots, 5\} \right\},$$

where

$$\begin{aligned} Q(x_1, x_2, \xi_1^s, \xi_2^s) = \min_{y_1, y_2, y_3, y_4} \quad & -16y_1 + 19y_2 + 23y_3 + 28y_4 \\ \text{s.t.} \quad & 2y_1 + 3y_2 + 4y_3 + 5y_4 \le \xi_1^s - x_1 \\ & 6y_1 + y_2 + 3y_3 + 2y_4 \le \xi_2^s - x_2 \\ & y_1, y_2, y_3, y_4 \in \{0, 1\} \end{aligned}$$

and $(\xi_1^s, \xi_2^s) \in \{(7,7), (11,11), (13,13)\}$ with probability $1/3$.

Only 15 lines of Julia script!

```
1  using Dsp, MPI # Load packages
2  MPI.Init() # Initialize MPI
3  m = Model(3) # Create a Model object with three scenarios
4  xi = [[7,7] [11,11] [13,13]] # random parameter
5  @variable(m, 0 <= x[i=1:2] <= 5, Int)
6  @objective(m, Min, -1.5*x[1]-4*x[2])
7  for s in 1:3
8    q = Model(m, s, 1/3);
9    @variable(q, y[j=1:4], Bin)
10   @objective(q, Min, -16*y[1]+19*y[2]+23*y[3]+28*y[4])
11   @constraint(q, 2*y[1]+3*y[2]+4*y[3]+5*y[4]<=xi[1,s]-x[1])
12   @constraint(q, 6*y[1]+1*y[2]+3*y[3]+2*y[4]<=xi[2,s]-x[2])
13 end
14 solve(m, solve_type=:Dual, param="myparams.txt")
15 MPI.Finalize() # Finalize MPI
```

SIPLIB Test Instances

- **SIPLIB:** publically available test library for stochastic integer programming
  - http://www2.isye.gatech.edu/~sahmed/siplib/
  - ***DCAP***: *complete recourse,* first-stage **mixed**-integer and second-stage **pure** integer
    - Number of scenarios: 200, 300 and 500
  - ***SSLP***: *complete recourse,* first-stage **pure** integer and second-stage **mixed**-integer
    - Number of scenarios: 5, 10, 15, 50 and 100

Name	# Rows	# Columns	# Integers
dcap233_200	3006	5412	5406
dcap233_300	4506	8112	8106
dcap233_500	7506	13512	13506
dcap243_200	3606	7212	7206
dcap243_300	5406	10812	10806
dcap243_500	9006	18012	18006
dcap332_200	2406	4812	4806
dcap332_300	3606	7212	7206
dcap332_500	6006	12012	12006
dcap342_200	2806	6412	6406
dcap342_300	4206	9612	9606
dcap342_500	7006	16012	16006

Name	# Rows	# Columns	# Integers
sslp_5_25_50	1501	6505	6255
sslp_5_25_100	3001	13005	12505
sslp_15_45_5	301	3465	3390
sslp_15_45_10	601	6915	6765
sslp_15_45_15	901	10365	10135
sslp_10_50_50	3001	25510	25010
sslp_10_50_100	6001	51010	50010

Computational Results

Instance	Scen (r)	Method	Iter	UB	LB	Gap (%)	Time (s)
dcap233	200	DDSub	10000	1835.34	1799.08	1.97	3853
		DDCP	85	1835.34	1833.38	0.11	935
		DSP	36	1835.34	1833.36	0.11	584
	300	DDSub	1503	1645.22	1633.88	0.68	5564
		DDCP	93	1645.22	1642.73	0.15	1705
		DSP	43	1645.22	1642.74	0.15	1145
	500	DDSub	2628	1737.94	1729.49	0.48	16791
		DDCP	117	1738.47	1736.66	0.10	9451
		DSP	44	1737.94	1736.66	0.10	2155
dcap243	200	DDSub	1995	2322.50	2311.37	0.47	3157
		DDCP	52	2322.50	2321.18	0.05	843
		DSP	37	2322.50	2321.18	0.05	694
	300	DDSub	10000	2559.48	2546.91	0.49	8869
		DDCP	53	2559.92	2556.66	0.12	1313
		DSP	36	2559.92	2556.67	0.12	1167
	500	DDSub	126	2167.97	2110.37	2.65	1992
		DDCP	64	2168.38	2165.47	0.13	2648
		DSP	39	2168.38	2165.46	0.13	2290
dcap332	200	DDSub	126	1068.24	983.23	7.95	443
		DDCP	85	1065.87	1059.08	0.64	708
		DSP	43	1063.52	1059.08	0.41	512
	300	DDSub	126	1260.42	1165.89	7.52	733
		DDCP	80	1257.01	1250.90	0.49	1298
		DSP	48	1257.01	1250.91	0.49	986
	500	DDSub	126	1596.49	1541.24	3.46	1333
		DDCP	76	1593.00	1587.06	0.37	2000
		DSP	43	1592.02	1587.06	0.31	1496
dcap342	200	DDSub	126	1620.19	1582.10	2.35	505
		DDCP	76	1620.76	1618.07	0.16	734
		DSP	37	1620.18	1618.07	0.13	577
	300	DDSub	126	2069.00	2020.11	2.36	881
		DDCP	73	2067.76	2065.43	0.11	1271
		DSP	37	2067.96	2065.42	0.12	930
	500	DDSub	126	1906.18	1861.25	2.36	1716
		DDCP	87	1905.38	1902.98	0.12	2400
		DSP	44	1905.36	1902.97	0.12	1788

- Smaller gaps in shorter time
- Less iterations
- Time reduced by a factor of 8
- Solution times: 9 ~ 37 mins.
- Still positive gaps

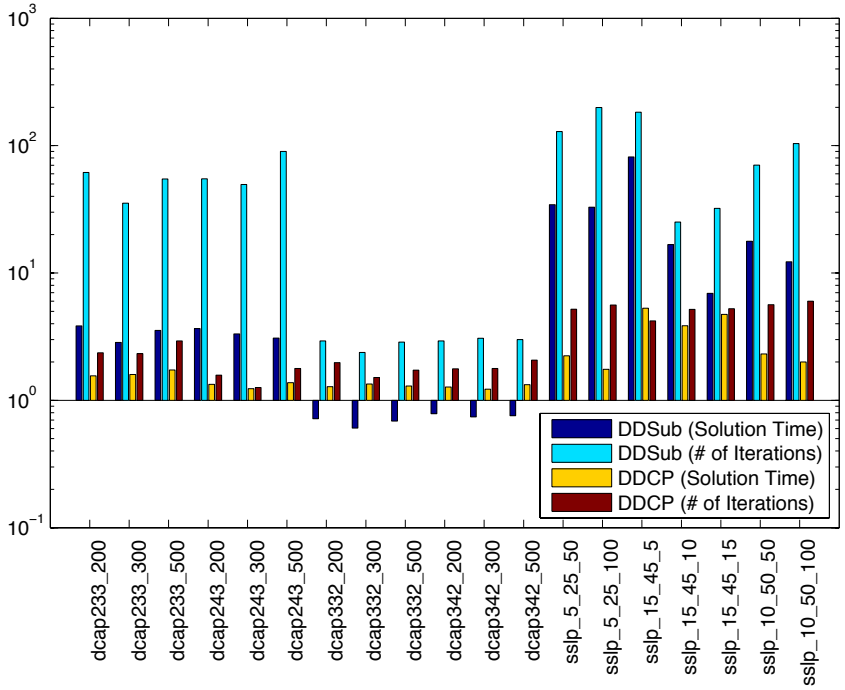
Computational Results

Instance	Scen (r)	Method	Iter	UB	LB	Gap (%)	Time (s)
sslp_5_25	50	DDSub	645	-121.60	-123.76	1.77	244
		DDCP	26	-121.60	-121.60	0.00	14
		DSP	5	-121.60	-121.60	0.00	6
	100	DDSub	995	-127.37	-128.94	1.23	643
		DDCP	28	-127.37	-127.37	0.00	32
		DSP	5	-127.37	-127.37	0.00	19
sslp_10_50	50	DDSub	774	-364.64	-369.21	1.25	3105
		DDCP	62	-364.64	-364.64	0.00	402
		DSP	11	-364.64	-364.64	0.00	174
	100	DDSub	1224	-354.19	-364.64	2.95	9591
		DDCP	72	-354.19	-354.19	0.00	1545
		DSP	12	-354.19	-354.19	0.00	777
	500	DDSub	344	-349.13	-356.35	2.06	$>$ 21600
		DDCP	107	-349.13	-349.13	0.00	10771
		DSP	17	-349.13	-349.13	0.00	1614
	1000	DDSub	177	-351.71	-358.88	2.03	$>$ 21600
		DDCP	66	-351.71	-351.99	0.07	$>$ 21600
		DSP	11	-351.71	-351.71	0.00	3251
	2000	DDSub	115	-347.26	-354.03	1.94	$>$ 21600
		DDCP	54	-347.26	-349.60	0.67	$>$ 21600
		DSP	10	-347.26	-347.26	0.00	4762
sslp_15_45	5	DDSub	914	-262.40	-262.42	0.01	2490
		DDCP	21	-262.40	-262.40	0.00	164
		DSP	5	-262.40	-262.40	0.00	32
	10	DDSub	427	-260.50	-268.15	2.94	8867
		DDCP	88	-260.50	-260.50	0.00	1988
		DSP	17	-260.50	-260.50	0.00	515
	15	DDSub	541	-253.60	-265.30	4.61	$>$ 21600
		DDCP	89	-253.60	-253.60	0.00	14917
		DSP	17	-253.60	-253.60	0.00	3092

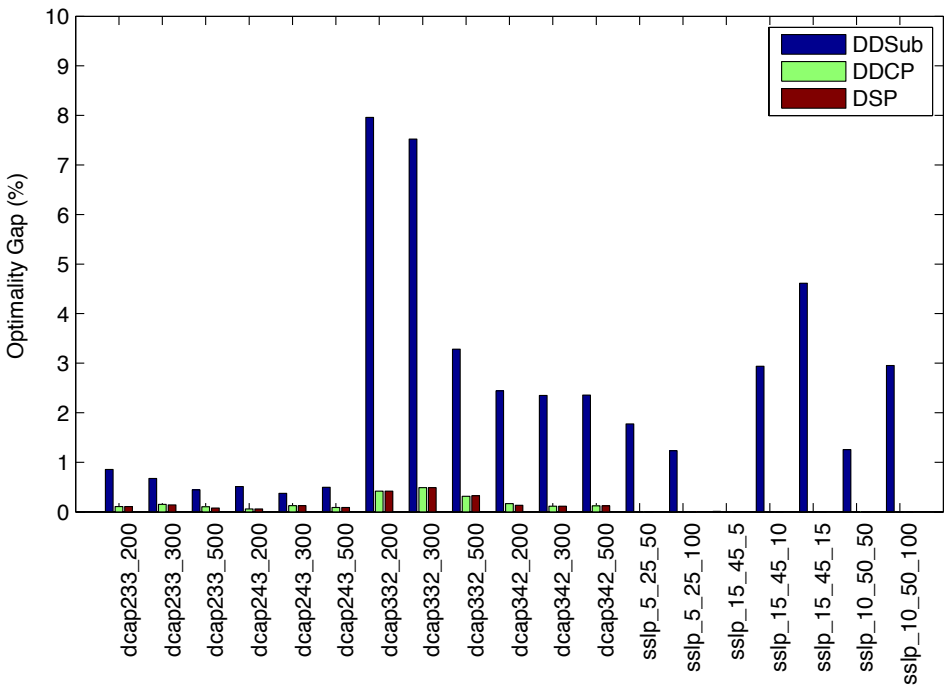
DSP vs. Existing Methods

- **Benchmark with**
  - DSP
  - Subgradient method (DDSub)
  - Standard cutting-plane method (DDCP)

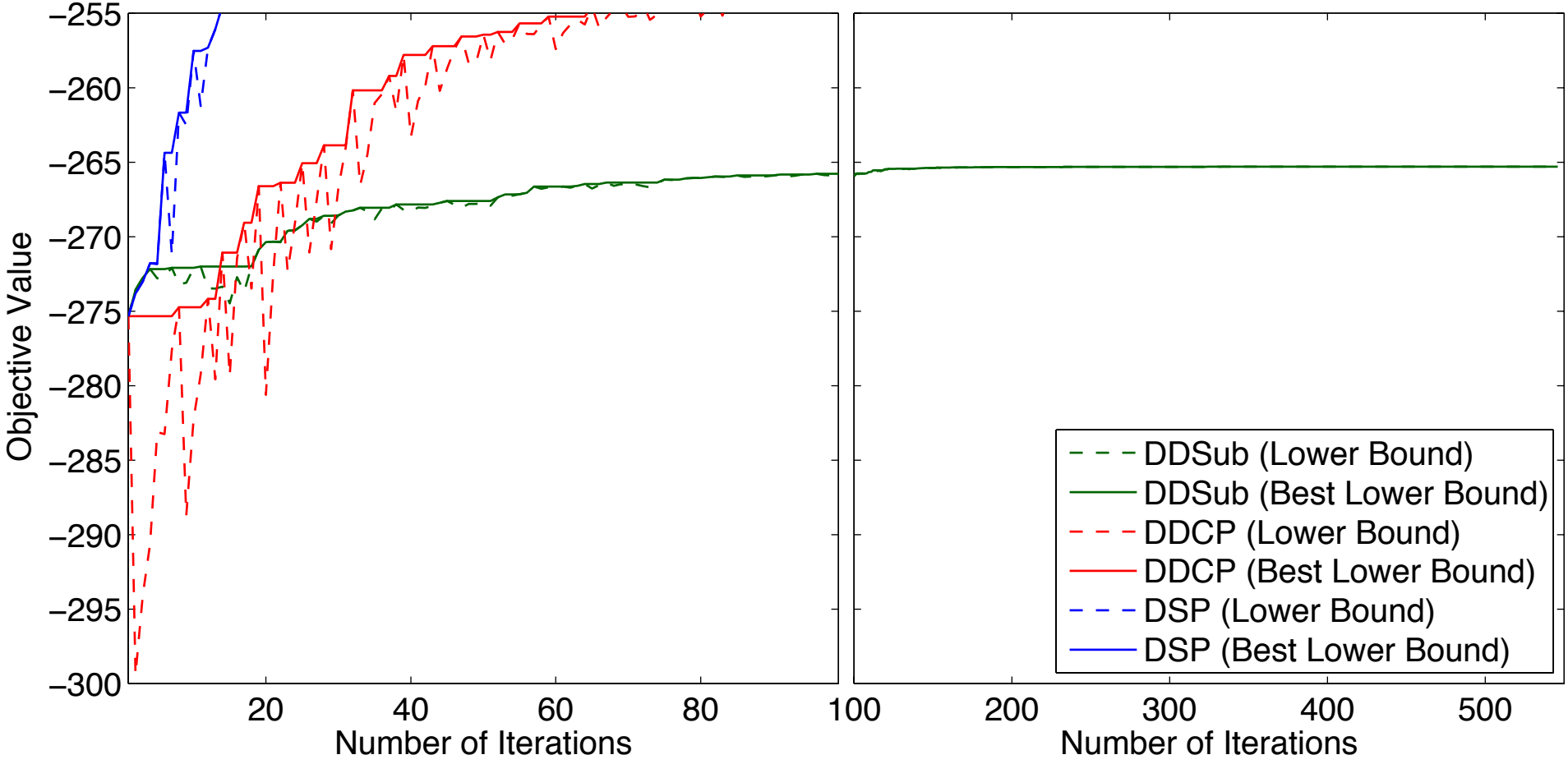
Number of Iterations and Solution Time



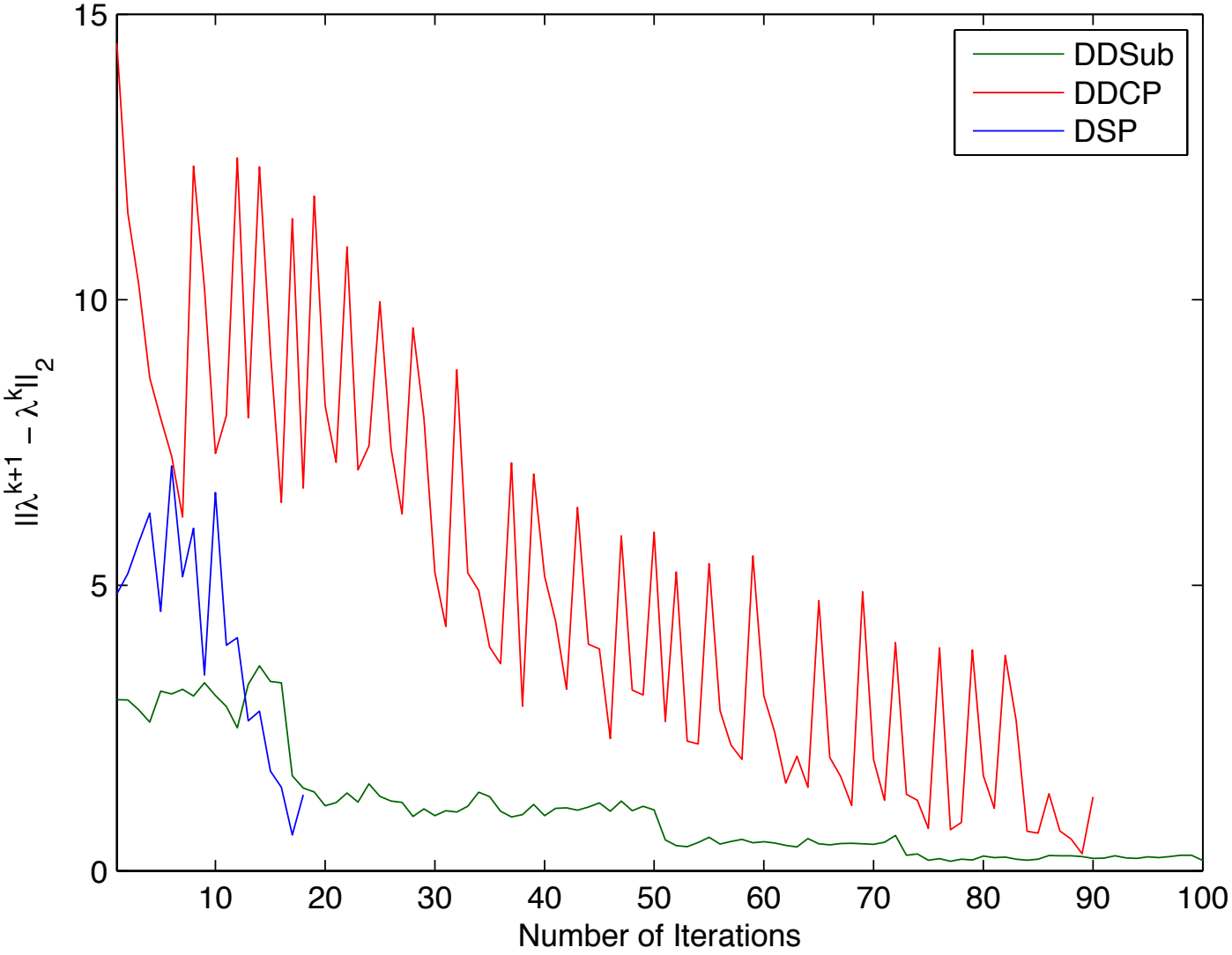
Optimality Gap



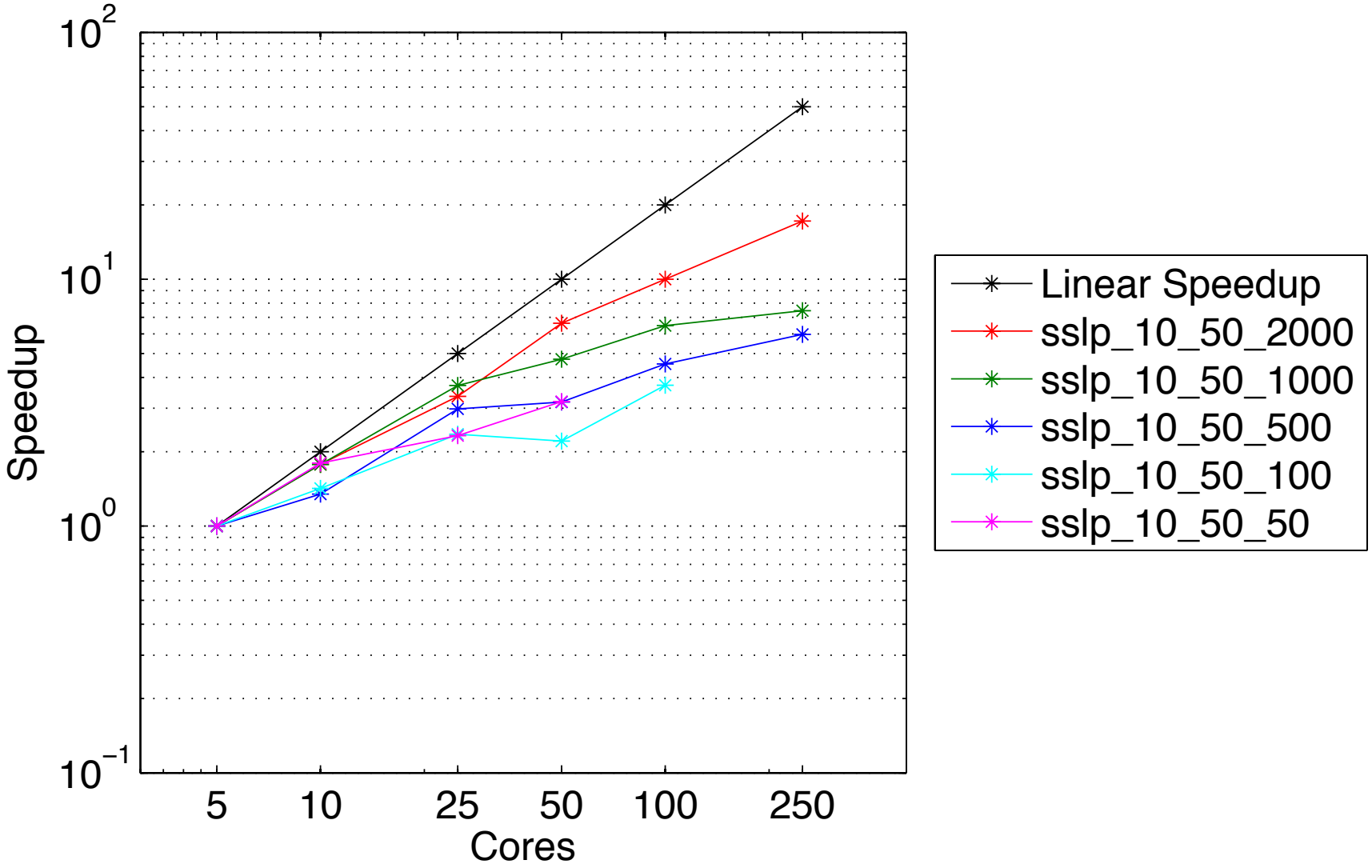
DSP vs. Existing Methods - Convergence



Oscillating Dual Variable Values



Strong Scaling Results of DSP

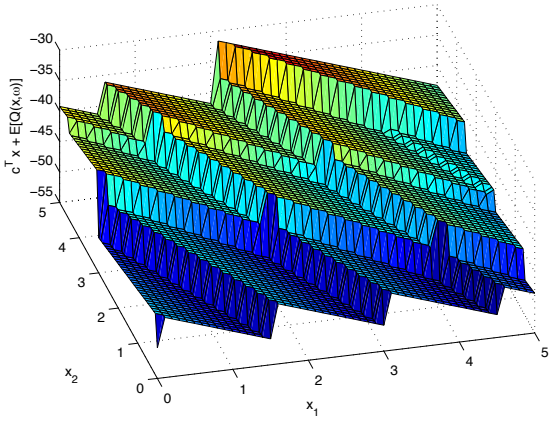


An Asynchronous Variant

Kim, Kibaek, Cosmin G. Petra, and Victor M. Zavala. "An Asynchronous Bundle-Trust-Region Method for Dual Decomposition of Stochastic Mixed-Integer Programming." *SIAM Journal on Optimization* (accepted).

Dual Decomposition for SMIP

$$
\begin{array}{lllllll}
\min & c^T x(\omega_1) & +q(\omega_1)^T y(\omega_1) & +\ldots & +c^T x(\omega_1) & +q(\omega_N)^T y(\omega_N) & \\
\text{s.t.} & Ax(\omega_1) & & & & & \geq b, \\
& T(\omega_1)x(\omega_1) & +W(\omega_1)y(\omega_1) & & & & \geq h(\omega_1) \\
& \vdots & & \ddots & & & \vdots \\
\text{Nonanticipaticity constraints} & & & & Ax(\omega_N) & & \geq b \\
& & & & T(\omega_N)x(\omega_N) & +W(\omega_N)y(\omega_N) & \geq h(\omega_N) \\
& x(\omega_1) & & \ldots & = x(\omega_N) & & \\
& x(\omega_1), & & \ldots & x(\omega_N) & & \in \mathbb{R}_+^{n_1-p_1} \times \mathbb{Z}_+^{p_1} \\
& & y(\omega_1), & \ldots & & y(\omega_N) & \in \mathbb{R}_+^{n_2-p_2} \times \mathbb{Z}_+^{p_2}
\end{array}
$$



- ***Becomes More Challenging!***
  - The recourse function is nonconvex and discontinuous.
  - Benders Decomposition cannot be used.
- **Dual Decomposition**
  - Lagrangian relaxation of the nonanticipativity constraints

$$
\begin{aligned}
z = \min_{x_0, x_j, y_j} \quad & \sum_{j=1}^{N} p_j \left(c^T x_j + q_j^T y_j\right) \\
\text{s.t.} \quad & (x_j, y_j) \in G_j, \quad j = 1, \ldots, N, \\
\text{Nonanticipaticity constraints} \quad & \sum_{j=1}^{N} H_j x_j = 0 \quad (\lambda)
\end{aligned}
$$

$$
D(\lambda) := \min_{x_j, y_j} \sum_{j=1}^{N} [p_j (c^T x_j + q_j^T y_j) + \lambda^T H_j x_j]
\quad \text{s.t. } (x_j, y_j) \in G_j,\ j = 1, \ldots, N
$$

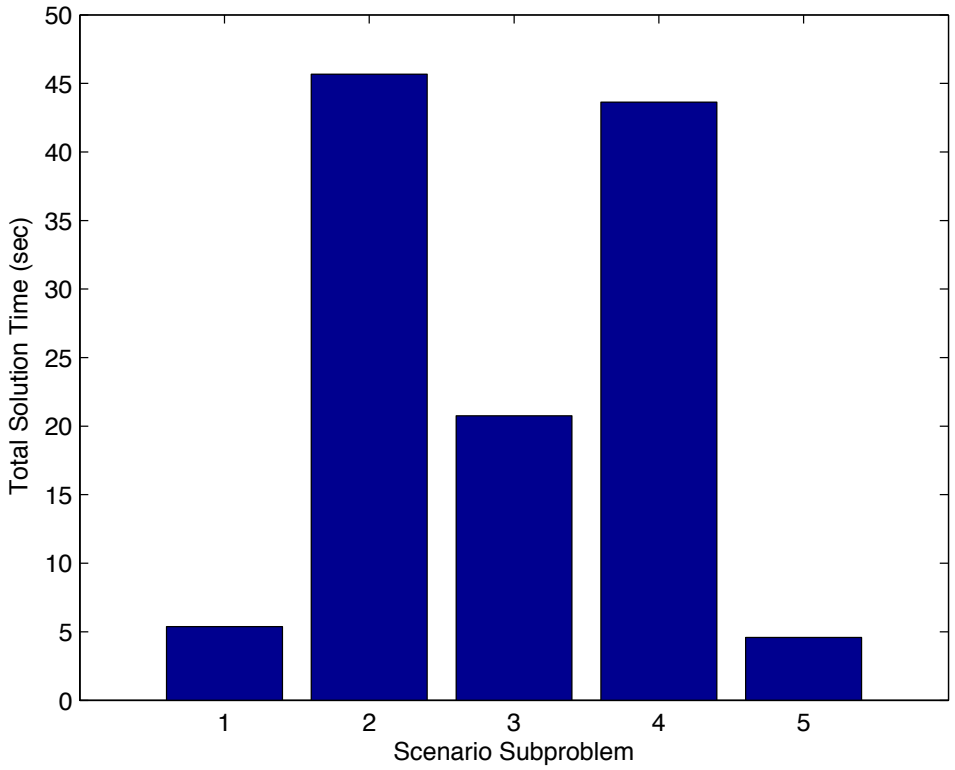
  - Seek for the best lower bound by solving

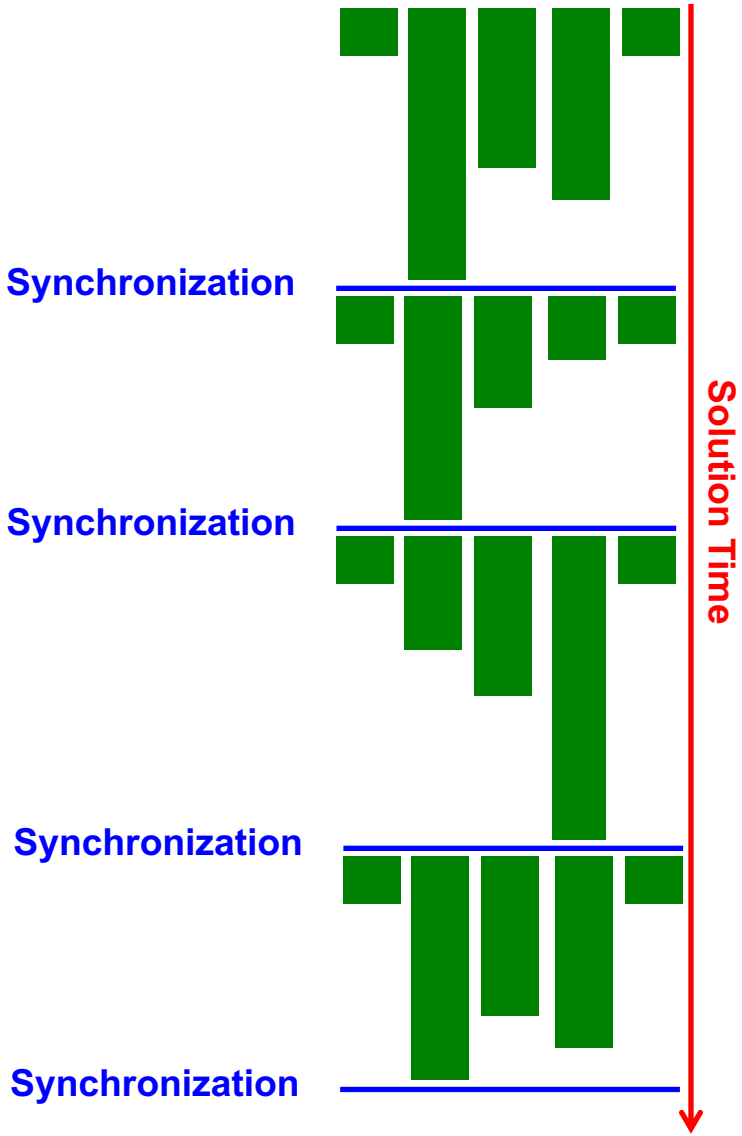
$$
z \geq z_{LD} := \max_{\lambda} \sum_{j=1}^{N} D_j(\lambda)
$$

Decomposed in each scenario j

Synchronous Bundle Method

- Processors have to wait for the slowest one.
- Causing *Load-Imbalancing*
- Resulting in *low parallel efficiency*
- Potentially increasing *solution time*





Asynchronous Bundle Method

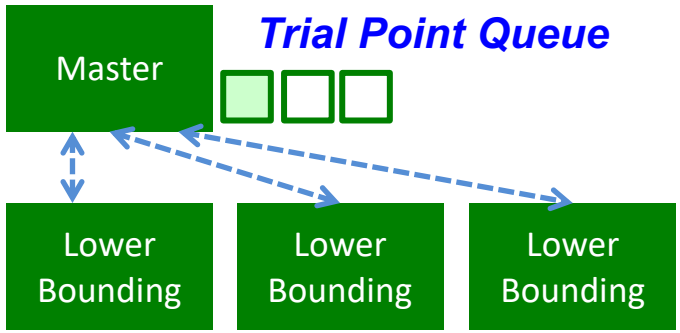
- Inexact Approximation of the Lagrangian Dual

$$\tilde{m}_{k,l}(\lambda) := \max \sum_{j \in J} \theta_j$$
$$\text{s.t. } \theta_j \leq D_j(\lambda^i) + (H_j x_j^i)^T(\lambda - \lambda^i),\ i \in \mathcal{B}^{k,l},\ j \in J^i$$

Update the Bundle with a subset of scenarios

- *Ingredients are not really new!*
  - Incremental Subgradient [*Kiwiel 2004, Bertsekas 2001, 2011, 2015*]
  - Incremental-Like Bundle [*Emiel and Sagastizabal 2010*]
  - Asynchronous Subgradient [*Nedic et al. 2001*]
  - Asynchronous Bundle [*Fischer and Helmber 2014*]
  - Asynchronous Benders [*Linderoth and Wright 2003*]
- Trial point queue (for synchronization points)
  - Necessary to update the trust region
  - Size of the queue
  - FIFO vs. LIFO
- Minimum number of worker processors to wait for updating the bundle
- Static vs. dynamic scenario allocations

Algorithm should be carefully designed!



Asynchronous Bundle Method

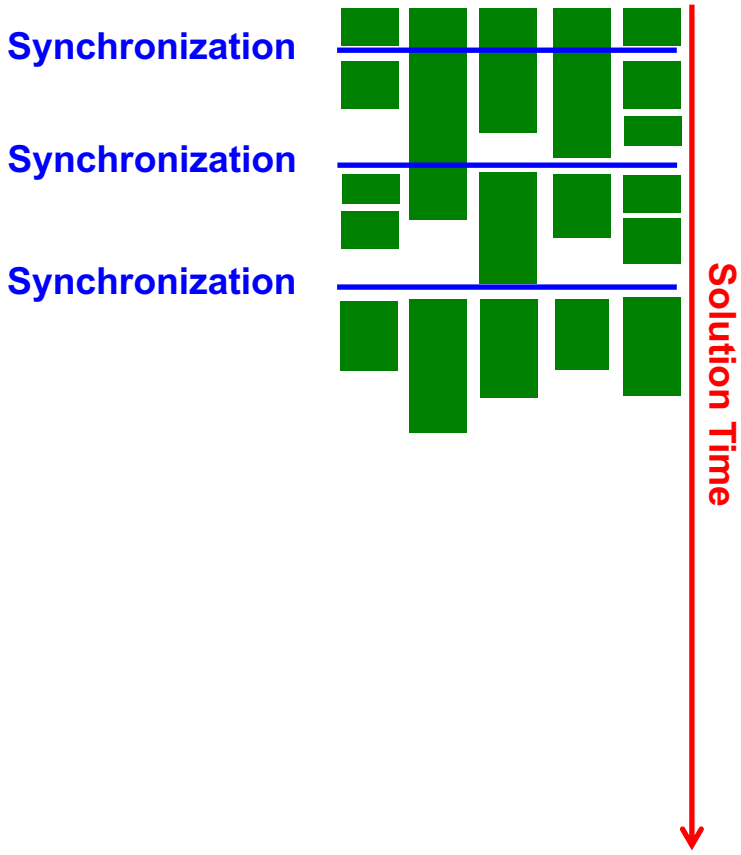
$$\max_{\theta_j,\lambda} \quad \theta_1+\theta_2+\theta_3$$

s.t.

$$\theta_1 \le \bar{D}_1^1 + (H_1\bar{x}_1^1)^T(\lambda-\lambda^1)$$
$$\theta_3 \le \bar{D}_3^1 + (H_3\bar{x}_3^1)^T(\lambda-\lambda^1)$$

$$\theta_1 \le \bar{D}_1^2 + (H_1\bar{x}_1^2)^T(\lambda-\lambda^2)$$
$$\theta_2 \le \bar{D}_1^1 + (H_1\bar{x}_1^1)^T(\lambda-\lambda^1)$$
$$\theta_3 \le \bar{D}_3^2 + (H_3\bar{x}_3^2)^T(\lambda-\lambda^2)$$

$$\theta_2 \le \bar{D}_1^2 + (H_1\bar{x}_1^2)^T(\lambda-\lambda^2)$$
$$\theta_3 \le \bar{D}_3^3 + (H_3\bar{x}_3^3)^T(\lambda-\lambda^3)$$

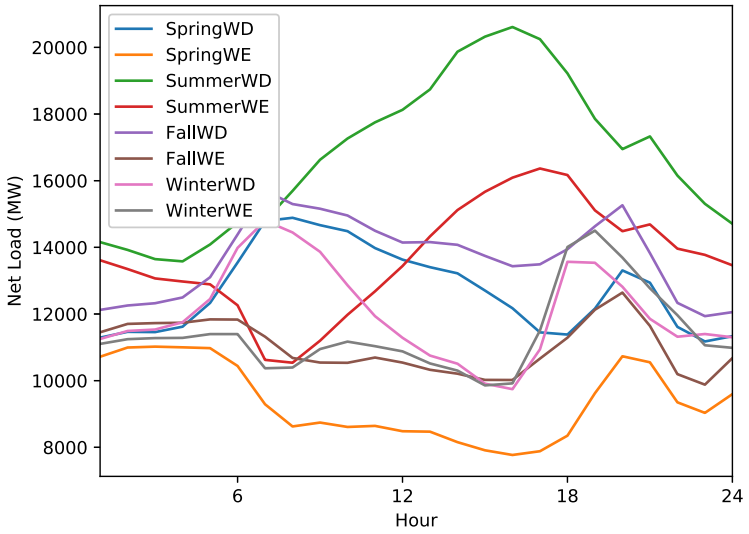


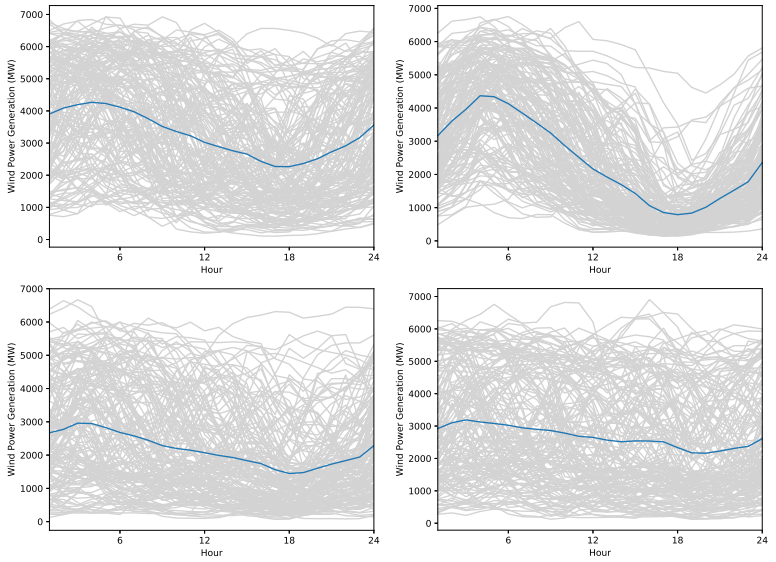
Computational Experiment Settings

- DSP: an open-source software framework for parallel decomposition methods
  - Written in C++ with MPI library
  - Uses CPLEX for solving the master (by barrier method) and sub-MIP problems with default parameter settings
- All computations were performed on the Blues cluster.
  - 630-node computing cluster at Argonne National Laboratory
  - Intel Sandy Bridge Xeon E5-2670 2.6GHz
  - 16 cores per compute node
  - 64 GB of memory on each node

Problem Instances - Stochastic Unit Commitment

- Scheduling power generators and dispatching power from generators to loads
  - Over electric grid network
  - For 24-hour time horizon
  - Under supply uncertainty (wind power generation)
- A test system data for the Western Electricity Coordinating Council
  - 8 load profiles
  - 10 sets of wind power scenarios
  - Created 80 problem instances
- Each SMIP has 340,244 variables and 352,840 constraints.





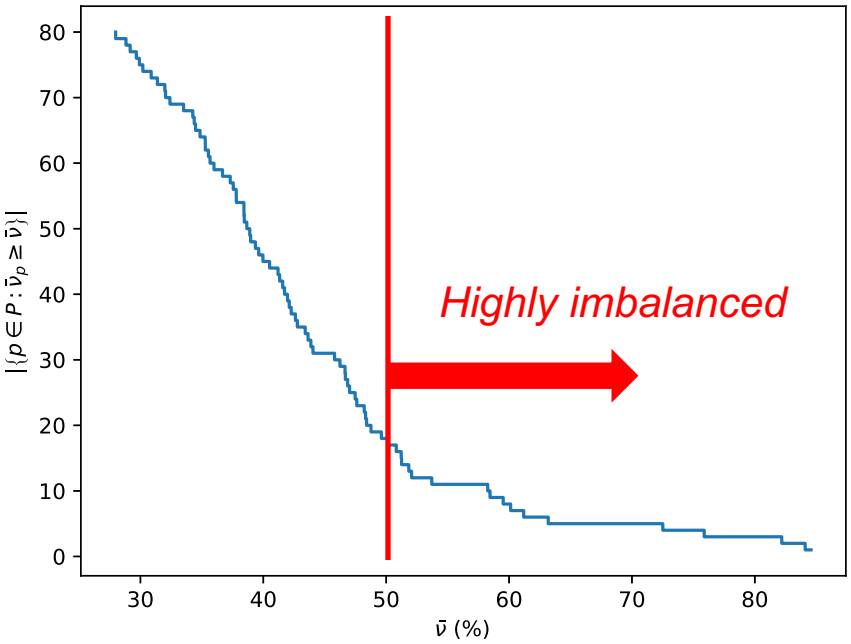
Load Imbalance

- Percent imbalance metric:

$$\nu_{pk} := \left(\frac{t_{pk}^{max}}{\bar{t}_{pk}} - 1 \right) \times 100\%,$$

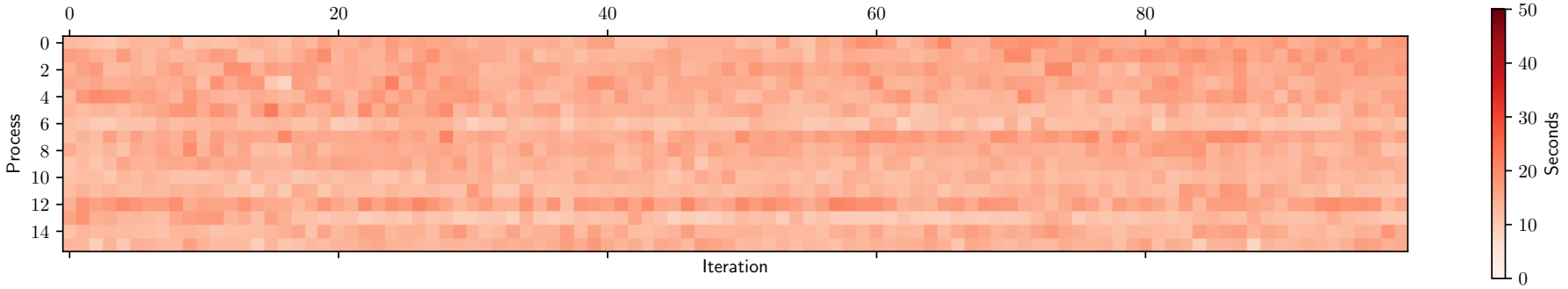
  - How much the maximum time is deviated from the mean time.

- Distribution of the average percent imbalance metrics resulting from the synchronous BTR
  - 27% ~ 84%
  - 18 highly imbalanced instances (> 50%)

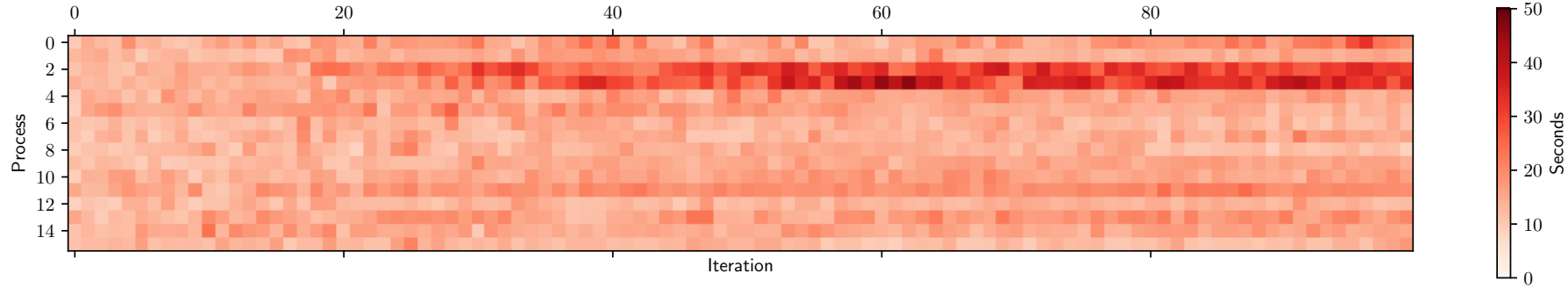


Load Imbalance - What you really see…

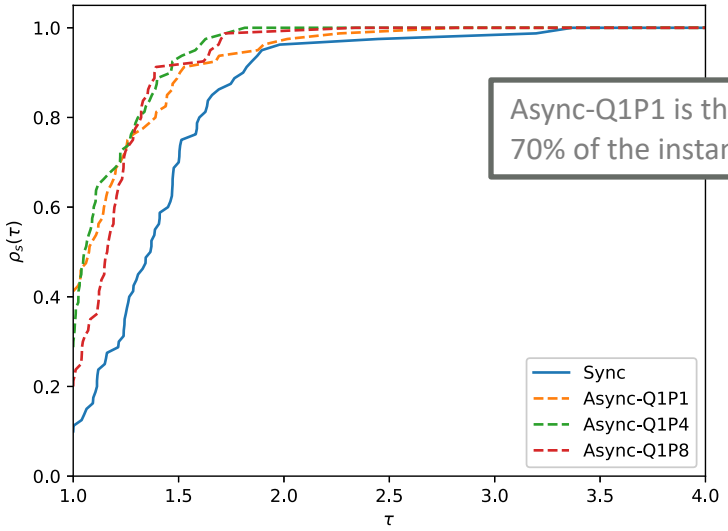
Low Imbalanced Instance

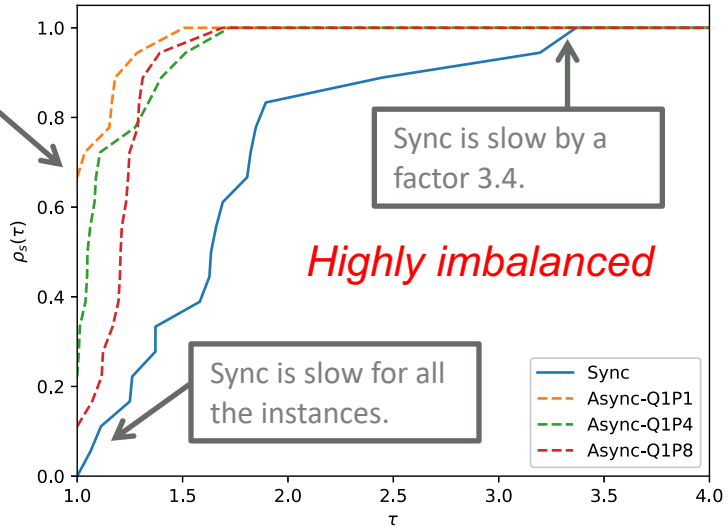


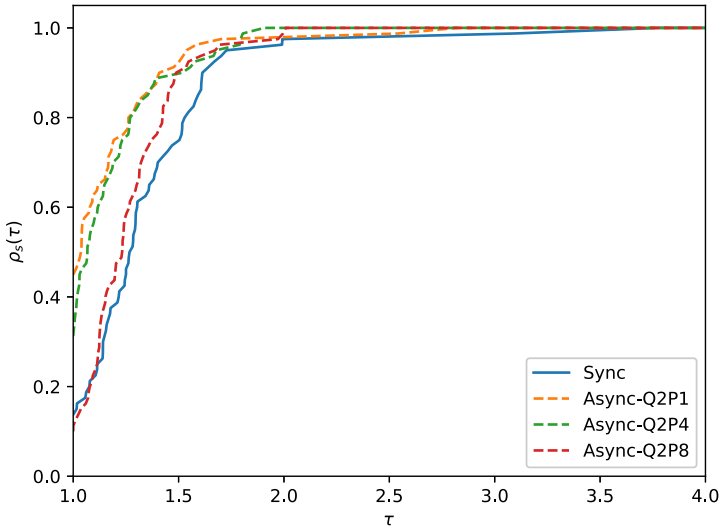
Highly Imbalanced Instance

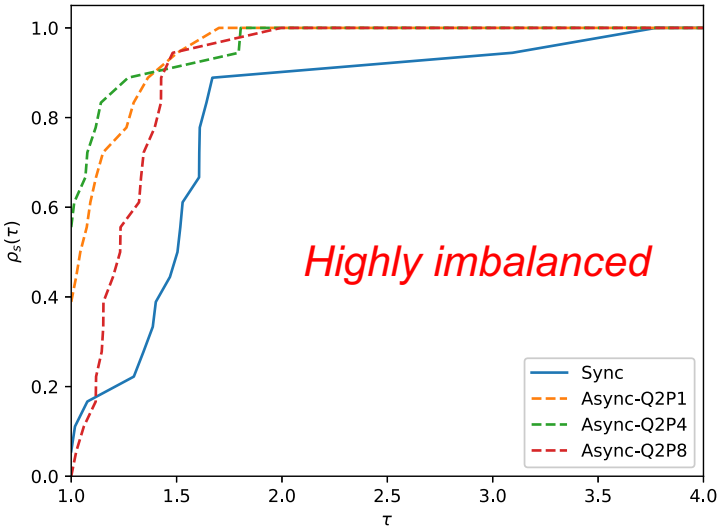


Asynchronous Computing Results



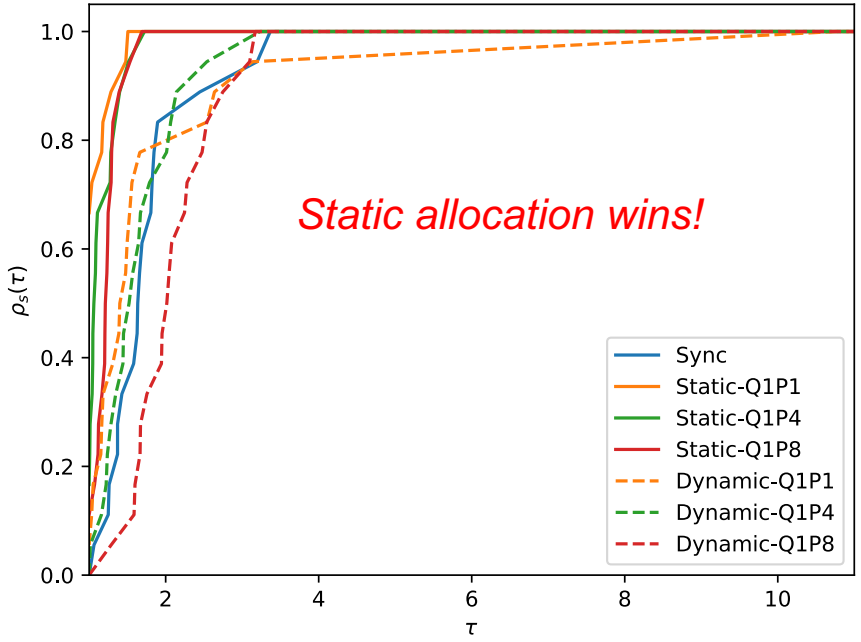


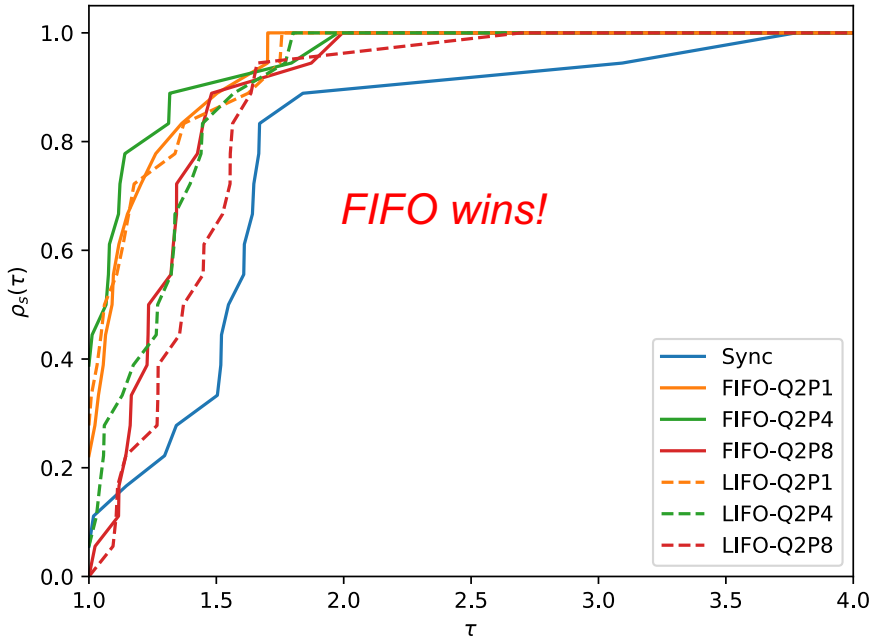




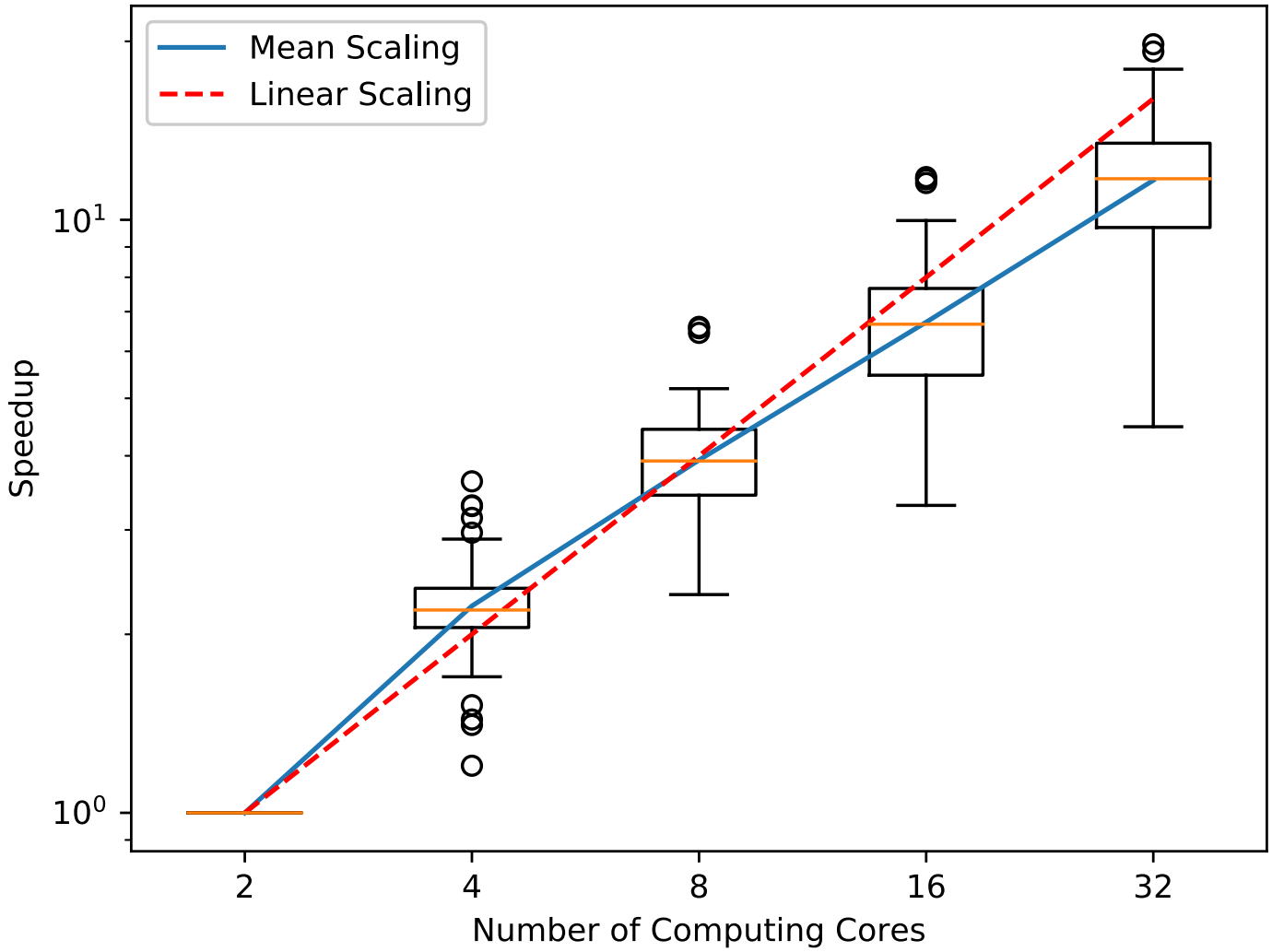
Computational Experiments - Other Variants

- Static vs. Dynamic allocation of the scenario subproblems
  - Dynamic allocation improves parallel efficiency in general.
  - However, the dynamic allocation loses the MIP warm-start.
- FIFO vs. LIFO for evaluating trial points
  - FIFO shows marginally better results.
- All these variations do not affect the convergence results of the method.





Strong Scaling of the Asynchronous Method



Scalable Branching on Dual Decomposition

Kim, Kibaek and Brian Dandurand. "Scalable Branching on Dual Decomposition of Stochastic Mixed-Integer Programming Problems." *Mathematical Programming Computation* (submittted).

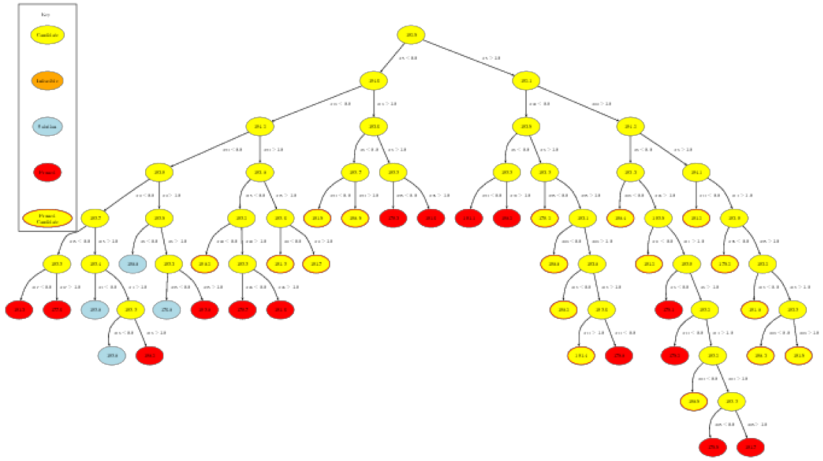
Scalable Branching Approaches

- Dual decomposition is a *bounding method*.

$$z \geq z_{LD} := \max_{\lambda} \sum_{j=1}^{N} D_j(\lambda)$$

- Branch-and-bound is the most natural approach to proving optimality for MIP.
  - We already have a very tight bound from DD.
  - Let's branch!

- Branching rules to consider:
  - at which solution point (or value)
  - in which direction



Branch-and-Price Method

DWD is the dual of the dual decomposition.

- A natural choice for branching is on fractional integer variables in the original space.

$$\max_{\theta_n,\lambda_n} \quad \sum_{n=1}^{N} \theta_n$$

$$\text{s.t.} \quad \sum_{n=1}^{N} \lambda_n = 0 \quad (x_0)$$

$$\theta_n \le D_n(\lambda_n^k) + (x_n^k)^T(\lambda_n - \lambda_n^k),$$
$$n = 1,\dots,N,\ k = 1,\dots,K \quad (\alpha_n^k)$$

Dualizing

$$\min_{\alpha_n^k,x_0} \quad \sum_{n=1}^{N}\sum_{k=1}^{K} \left[D_n(\lambda_n^k) - (x_j^k)^T\lambda_n^k\right]\alpha_n^k$$

$$\text{s.t.} \quad \sum_{k=1}^{K} x_n^k\alpha_n^k = x_0, \quad n = 1,\dots,N,$$

$$\sum_{k=1}^{K} \alpha_n^k = 1, \quad n = 1,\dots,N,$$

$$\alpha_n^k \ge 0, \quad n = 1,\dots,N,\ k = 1,\dots,K$$

1. Lulli, Guglielmo, and Suvrajeet Sen. "A branch-and-price algorithm for multistage stochastic integer programming with application to stochastic batch-sizing problems." *Management Science* 50.6 (2004): 786-796.

Branch-and-Price Method

- At an optimum (or feasible) of DWD,

$$\sum_{k=1}^{K} x_n^k \hat{\alpha}_n^k = \hat{x}_0, \quad \sum_{k=1}^{K} y_n^k \hat{\alpha}_n^k = \hat{y}_n, \quad n = 1, \ldots, N,$$

Fractional? Fractional?

- Branching is of the form:

$$\sum_{k=1}^{K} x_{nj}^k \alpha_n^k \leq \lfloor \hat{x}_{0j} \rfloor \quad \bigvee \quad \sum_{k=1}^{K} x_{nj}^k \alpha_n^k \geq \lfloor \hat{x}_{0j} \rfloor + 1$$

$$\sum_{k=1}^{K} y_{nj}^k \alpha_n^k \leq \lfloor \hat{y}_{nj} \rfloor \quad \bigvee \quad \sum_{k=1}^{K} y_{nj}^k \alpha_n^k \geq \lfloor \hat{y}_{nj} \rfloor + 1$$

The second-stage integer variables need branched!

Caroe-and-Schultz Branching

- Interesting idea to avoid branching on the second-stage integer variables
  - At the last iteration *K*, they used the average for the variable value to branch:

$$\tilde{x}_0 = \sum_{n=1}^{N} p_n x_n^K,$$

  - For fractional integer variables,

$$x_{nj} \leq \lfloor \tilde{x}_{0j} \rfloor \quad \bigvee \quad x_{nj} \geq \lfloor \tilde{x}_{0j} \rfloor + 1$$

  - For continuous variables,

$$x_{nj} \leq \tilde{x}_{0j} - \epsilon \quad \bigvee \quad x_{nj} \geq \tilde{x}_{0j} + \epsilon$$

Great, the branchings are applied to ALL scenarios!

Can we do better?

Observations

1. Carøe-and-Schultz (CS) branching uses an arbitrary point (i.e., average solution) to branch.
2. DWD solution point provides the Lagrangian dual bound.

- Let $\hat{x}_0$ be the DW solution point (vs. the average point $\tilde{x}_0$ by CS) at some node.
- Assume that $\hat{x}_0 \neq \tilde{x}_0$.
- When branching on continuous variables at the CS point, one of the child nodes would still contain the DW solution point.
- Best lower bound in the tree would not be improved in the CS branching.

New Branching Approach

We use the DWD solution point to branch:

$$\hat{x}_0 = \sum_{k=1}^{K} x_n^k \hat{\alpha}_n^k \quad \left(\text{v.s.} \quad \tilde{x}_0 = \sum_{n=1}^{N} p_n x_n^K \right) \tag{1}$$

- For fractional integer variables,

$$x_{nj} \leq \lfloor \hat{x}_{0j} \rfloor \quad \bigvee \quad x_{nj} \geq \lfloor \hat{x}_{0j} \rfloor + 1 \tag{2}$$

- For continuous variables,

$$x_{nj} \leq \hat{x}_{0j} - \epsilon \quad \bigvee \quad x_{nj} \geq \hat{x}_{0j} + \epsilon \tag{3}$$

Finite termination has been proved for zero epsilon!

Some Details of Implementation

- Algorithmic Details:
  - Upper bounds are evaluated at a given point from each method, after rounding any fractional integer variable values.
  - No warm- or hot-start for node subproblem solutions
  - Gap tolerance: 0.01%
  - 2-hour wall clock time limit

- Computation Settings:
  - Implemented in DSP with Coin-ALPS
  - The master was solved by CPLEX-12.7.0.
  - Subproblems were solved in parallel using MPICH.
  - Ran on Argonne's Bebop cluster (Intel Broadwell 36 cores)

SIPLIB Test Instances

- **SIPLIB:** publically available test library for stochastic integer programming
  - http://www2.isye.gatech.edu/~sahmed/siplib/
  - ***DCAP***: *complete recourse,* first-stage **mixed**-integer and second-stage **pure** integer
    - Number of scenarios: 200, 300 and 500
  - ***SSLP***: *complete recourse,* first-stage **pure** integer and second-stage **mixed**-integer
    - Number of scenarios: 5, 10, 15, 50 and 100

Name	# Rows	# Columns	# Integers
dcap233_200	3006	5412	5406
dcap233_300	4506	8112	8106
dcap233_500	7506	13512	13506
dcap243_200	3606	7212	7206
dcap243_300	5406	10812	10806
dcap243_500	9006	18012	18006
dcap332_200	2406	4812	4806
dcap332_300	3606	7212	7206
dcap332_500	6006	12012	12006
dcap342_200	2806	6412	6406
dcap342_300	4206	9612	9606
dcap342_500	7006	16012	16006

Name	# Rows	# Columns	# Integers
sslp_5_25_50	1501	6505	6255
sslp_5_25_100	3001	13005	12505
sslp_15_45_5	301	3465	3390
sslp_15_45_10	601	6915	6765
sslp_15_45_15	901	10365	10135
sslp_10_50_50	3001	25510	25010
sslp_10_50_100	6001	51010	50010

Numerical Results - DCAP (Easy)

Instance	Method	UB	LB	Gap (%)	Nodes Solved	Nodes Left	Time
dcap233_200	BNP	1842.09	1833.43	0.47	1630	1629	TO
	CS	1834.71	1834.53	OPT	29	0	99
	CS+DW	1834.58	1834.54	OPT	31	0	93
dcap233_300	BNP	1671.87	1642.76	1.74	377	376	TO
	CS	1644.35	1644.19	OPT	57	0	751
	CS+DW	1644.25	1644.19	OPT	25	0	253
dcap233_500	BNP	1775.76	1736.69	2.20	165	164	TO
	CS	1737.61	1737.50	OPT	51	0	589
	CS+DW	1737.52	1737.52	OPT	29	0	446
dcap243_200	BNP	2336.11	2321.44	0.62	711	684	TO
	CS	2322.67	2322.48	OPT	137	0	1155
	CS+DW	2322.50	2322.47	OPT	33	0	178
dcap243_300	BNP	2574.20	2556.59	0.68	520	519	TO
	CS	2559.41	2559.12	OPT	29	0	195
	CS+DW	2559.19	2559.19	OPT	31	0	196
dcap243_500	BNP	2193.46	2165.40	1.27	241	240	TO
	CS	2167.41	2167.21	OPT	33	0	394
	CS+DW	2167.39	2167.31	OPT	43	0	660

- BNP cannot solve any instance in 2-hour time limit.
- New approach tends to solve fewer node subproblems and thus faster than CS.

Numerical Results - DCAP (Hard)

Instance	Method	UB	LB	Gap (%)	Nodes Solved	Nodes Left	Time
dcap332_200	BNP	1098.54	1059.09	3.59	626	627	TO
	CS	1060.75	1060.66	OPT	395	0	1661
	CS+DW	1060.70	1060.59	OPT	89	0	349
dcap332_300	BNP	1313.85	1250.84	4.79	277	274	TO
	CS	1252.81	1252.68	OPT	395	0	4481
	CS+DW	1252.76	1252.63	OPT	89	0	832
dcap332_500	BNP	1695.66	1586.94	6.41	135	134	TO
	CS	1589.27	1588.32	0.05	232	213	TO
	CS+DW	1588.91	1588.66	0.01	223	166	TO
dcap342_200	BNP	1691.89	1618.09	4.36	524	523	TO
	CS	1619.64	1619.48	OPT	443	0	2496
	CS+DW	1619.56	1619.47	OPT	177	0	793
dcap342_300	BNP	2136.25	2065.51	3.31	301	300	TO
	CS	2067.61	2067.41	OPT	151	0	2437
	CS+DW	2067.52	2067.43	OPT	125	0	773
dcap342_500	BNP	2028.81	1902.89	6.20	145	144	TO
	CS	1904.92	1904.61	0.01	99	72	TO
	CS+DW	1904.73	1904.53	OPT	95	0	1699

- CS also failed to solve some instances in 2-hour time limit.
- Clear to see that the new approach outperformed CS

Numerical Results - SSLP

Instance	Method	UB	LB	Gap (%)	Nodes Solved	Nodes Left	Time
sslp_5_25_50	BNP	-121.6	-121.6	OPT	1	0	1
	CS	-121.6	-121.6	OPT	1	0	1
	CS+DW	-121.6	-121.6	OPT	1	0	1
sslp_5_25_100	BNP	-127.37	-127.37	OPT	1	0	1
	CS	-127.37	-127.37	OPT	1	0	1
	CS+DW	-127.37	-127.37	OPT	1	0	1
sslp_10_50_50	BNP	-364.64	-364.64	OPT	1	0	31
	CS	-364.64	-364.64	OPT	1	0	31
	CS+DW	-364.64	-364.64	OPT	1	0	30
sslp_10_50_100	BNP	-354.19	-354.19	OPT	1	0	36
	CS	-354.19	-354.19	OPT	1	0	37
	CS+DW	-354.19	-354.19	OPT	1	0	36
sslp_10_50_500	BNP	-349.136	-349.136	OPT	1	0	792
	CS	-349.136	-349.136	OPT	1	0	786
	CS+DW	-349.136	-349.136	OPT	1	0	788
sslp_10_50_1000	BNP	-351.711	-351.711	OPT	1	0	1964
	CS	-351.711	-351.711	OPT	1	0	2275
	CS+DW	-351.711	-351.711	OPT	1	0	2544
sslp_10_50_2000	BNP	-347.263	-347.263	OPT	1	0	2024
	CS	-347.263	-347.263	OPT	1	0	2006
	CS+DW	-347.263	-347.263	OPT	1	0	2008
sslp_15_45_5	BNP	-262.4	-262.4	OPT	1	0	4
	CS	-262.4	-262.4	OPT	1	0	4
	CS+DW	-262.4	-262.4	OPT	1	0	4
sslp_15_45_10	BNP	-260.5	-260.5	OPT	1	0	101
	CS	-260.5	-260.5	OPT	1	0	101
	CS+DW	-260.5	-260.5	OPT	1	0	101
sslp_15_45_15	BNP	-253.601	-253.601	OPT	1	0	267
	CS	-253.601	-253.601	OPT	1	0	267
	CS+DW	-253.601	-253.601	OPT	1	0	267

Summary and Future Research Questions

- Dual decomposition is the Lagrangian relaxation applied to SMIP
  - Not just that.
- Allowing to exploit special structures embedded in SMIP
  - Benders-type cuts
  - Inexact interior point methods and other bundle methods
  - Any better dual search methods, customized to this setting?
- Parallel computing
  - Asynchronous communication significantly reduces the solution time.
- Hope for global optimality
  - Specialized scalable branching method!
  - Other branching ideas? Generalized disjunctions, strong branching, etc.
- Other questions
  - Scenario partitioning/grouping
  - Primal heuristics
  - Extended to multi-stage stochastic programs

Questions?

- Contact:
  - Kibaek Kim, kimk@anl.gov

- Acknowledgements:
  - U.S. Department of Energy, Office of Science, ECRP
  - U.S. Department of Energy, Office of Science, MMICCS
  - U.S. Department of Energy, Office of Electricity, GMLC