MWLshaped: M
M
M
M
WWWW

A Cutting Plane Code for Stochastic Programming on Metacomputers

Jeff Linderoth

Steve Wright

MCS Division

Argonne National Labs

Philadelphia INFORMS, 11/8/99

Outline

- Two-Stage Linear Recourse Problem
- Cutting Plane Methods
- Metacomputers
- MW
- MWLShaped
  - Matching the Cutting Plane Algorithm to the Metacomputer
- Computational Results and Future Directions

Two-Stage Linear Recourse Program

$$
\begin{array}{ll}
\min & c^T x + \sum_{k=1}^{K} p_k q_k^T y_k, \\
\text{s.t.} & Ax = b, \\
& T_k x + W y_k = h_k, \quad k = 1, 2, \ldots, K, \\
& x \geq 0, \quad y_k \geq 0, \quad k = 1, 2, \ldots, K.
\end{array}
$$

- K: A number of "scenarios"
- x: First-stage variables
- y_k: Second-stage variables
- p_k: Scenario probabilities

Reformulating

- We can also express this problem as:

$$\min_x c^T x + \mathcal{Q}(x), \quad \text{s.t. } Ax = b, x \geq 0,$$

where $\mathcal{Q}(x)$ is defined as

$$\begin{aligned} \mathcal{Q}(x) = \min_{y_1, y_2, \ldots, y_K} \textstyle\sum_{k=1}^K p_k q_k^T y_k, \\ \text{s.t.} \quad W y_k = h_k - T_k x, \quad k = 1, 2, \ldots, K. \\ y_k \geq 0, \quad k = 1, 2, \ldots, K. \end{aligned}$$

- Important Facts:
  - ◇ $\mathcal{Q}(x)$ is a piecewise linear convex function of x.
  - ◇ Evaluation of $\mathcal{Q}(x)$ is a separable problem. (We can solve for $k = 1, 2, \ldots, K$ independently).

L-shaped (Benders) decomposition

- Represent $\mathcal{Q}$ by an artificial variable θ and find supporting planes for θ: $\theta \geq \beta_\ell - \alpha_\ell^T x, \ \ell = 1, 2, \ldots, L$.
  - The inequalities are obtained as a byproduct of evaluating $\mathcal{Q}(x)$.

L-Shaped Method

1. Solve a **master problem** with the current θ-approximation to $\mathcal{Q}(x)$ for $(\hat{x}, \hat{\theta})$.
2. Solve the **subproblems** for $y_1, y_2, \ldots, y_K$ (evaluating $\mathcal{Q}(\hat{x})$) and add supporting plane for θ approximation at $\hat{x}$.
3. Goto 1.

Quasi-Multi-Cuts

- "Classical" L-Shaped $\Rightarrow \mathcal{Q} \sim \theta$

$$\theta \geq \sum_{k=1}^{K} p_k \pi_{k\nu}^T h_k - \sum_{k=1}^{K} p_k \pi_{k\nu}^T T_k x$$

- "Multicut" $\Rightarrow \mathcal{Q} \sim \sum_{k=1}^{K} \theta_k$

$$\theta_k \geq \pi_{k\nu}^T h_k - \pi_{k\nu}^T T_k x$$

- We can actually partition the scenarios any way we like: "Quasi"-Multicut $\Rightarrow \mathcal{Q} \sim \sum_{j=0}^{n} \theta_{[\kappa_j \ldots \kappa_{j+1}]}$.

$$\theta_{[\kappa_j \ldots \kappa_{j+1}]} \geq \sum_{k=\kappa_j}^{\kappa_{j+1}} p_k \pi_{k\nu}^T h_k - \sum_{k=\kappa_j}^{\kappa_{j+1}} p_k \pi_{k\nu}^T T_k x$$

Metacomputers are...

A General Metacomputer	Our Metacomputer
Dynamic	*Very* Dynamic
Heterogenous	3 Architectures, 2 Operating Systems Speeds From 30 - 453 MIPS Two LP Solvers – CPLEX and SOPLEX
"Communicationally Challenged"	All machines live in Wisconsin (for now)
Large Scale	Over 350 machines in the pool, but we have to share them.

MW

- An object oriented framework to develop master-worker applications.
- To use MW, the user must implement 8 functions in three classes:
  - MWMaster
  - MWTask
    - Holds both the work to be done and the result of the task
  - MWWorker
- Our cutting plane algorithm is built on top of MW

MW – What's in it for me?

- MW will hunt for "idle" nodes, and start the appropriate executables on them. Free computing cycles!!!
  - Currently, processor hunting is done by Condor, but stay tuned
- MW does the message passing for you
  - You must tell it which items to pack and unpack
  - Can use PVM or shared files for message passing
- MW recognizes when processors fail and taskes care of rescheduling failed tasks
- MW will "checkpoint" and restart your job if the master process fails
- Lots, lots more!!!

MWLShaped – LShapedMaster

- `get_userinfo()`
  - ◇ Reads initial data, solves the expected value problem to get a good guess for x.
- `pack_worker_init_data()`
  - ◇ Initial data are W and "compact" forms of p_k, q_k, T_k, h_k.
- `setup_initial_tasks()`
  - ◇ Solves first stage I LP.
  - ◇ Creates a batch of tasks for the first iteration.
- `act_on_completed_task()`
  - ◇ If all tasks for this iteration are complete, add the cuts, solve the linear program and add a new batch of tasks for the next iteration.

MWLshaped – LShapedTask

- Work
  - ◇ Starting scenario index (k_b), and ending scenario index (k_e).
  - ◇ Stage I solution ($\hat{x}$).
  - ◇ Estimated expected stage II objective function value ($\hat{\theta}_{[k_b,k_e]}$).
- Result
  - ◇ A cut, if the estimate of ($\theta_{[k_b,k_e]}$) is not good enough.
- We form one cut per cluster of scenarios.

⋆ The computing environment has influenced our choice of algorithm!!

MWLShaped – LShapedWorker

- `unpack_init_data()`
  - Unpacks the problems initial data and loads the "base" stage II LP to the solver.
- `execute_task()`
  - Solve LPs for all scenarios $k_b \leq k \leq k_e$, and creates a cut if possible.

Synchronicity BAD!

- Different processors act at different speeds – many wait idle for the "slowpoke"
  - ◇ "Balance" $|k_e - k_b|$ with the speed of the machines as they arrive
    - – Involves adding both columns and rows to the master LP
- All workers wait for the master LP
  - ◇ Don't have the master wait for all tasks!!
    - \+ If $\alpha\%$ of the tasks for an iteration have reported with cuts, start a new iteration

⋆ Again – the computing environment has influenced our algorithmic choices!

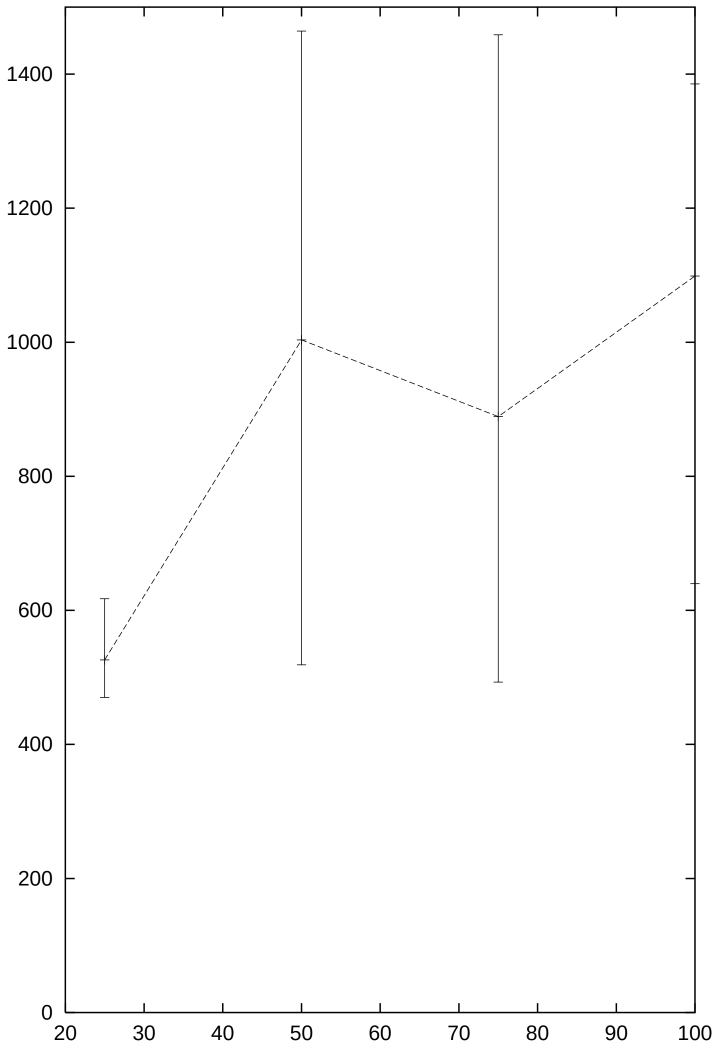
Computational Results

- There are not too many large, difficult instances publicly available! `:-(`
  - ◇ We'd love to solve some for you if you have them.
- "Traditional" efficiency measures don't really apply
  - ◇ Each time an instance is run, the results are (very) different.
- The main focus is on solving the problems – not necessarily on how efficiently we can do it.

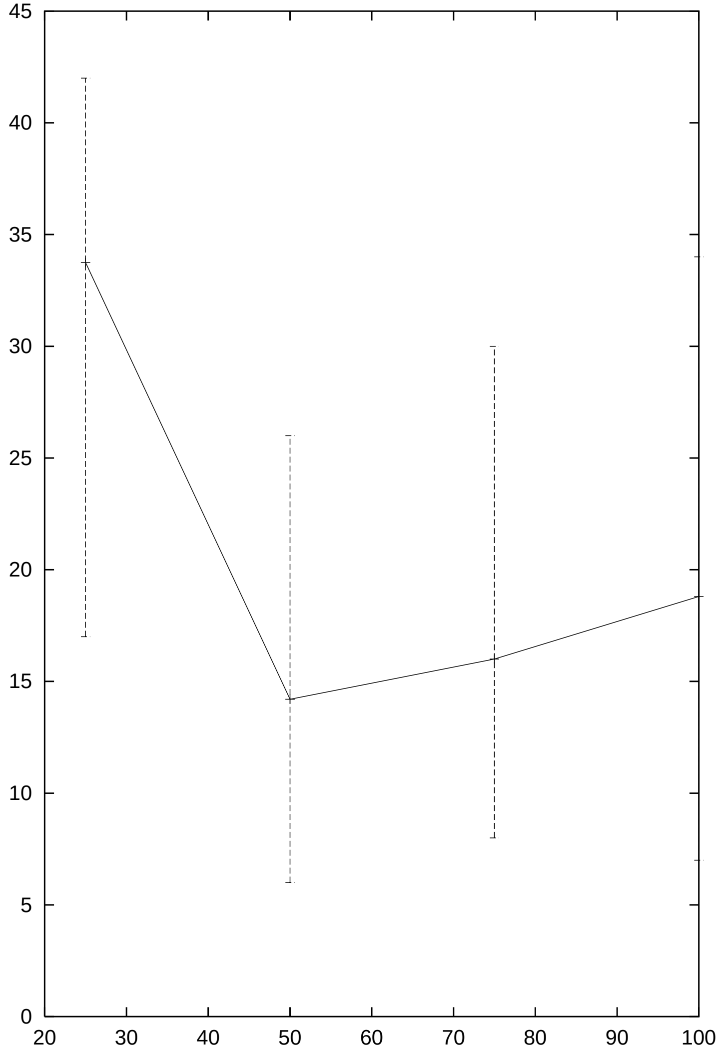
Instances and Parameters

Name	K	# rows	# Columns
storm	1000	525,185	1,259,120
ssn	10000	1,750,000	7,060,090

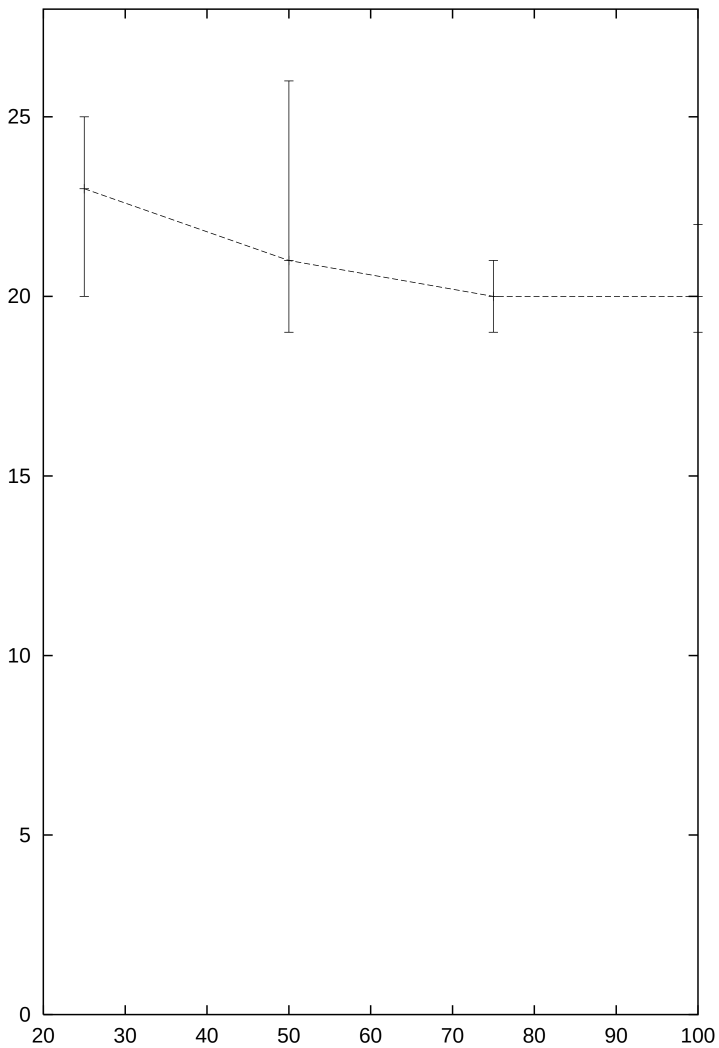
- Instances run 5 times to reduce variance
- Master LP (usually) solved with CPLEX, and subproblem LPs solved with SOPLEX.
- "Chuck size" – $|k_e - k_b| = 0.02K$
- Synchronicity coefficient – $\alpha \in \{0, 25, 0.5, 0.75, 1\}$



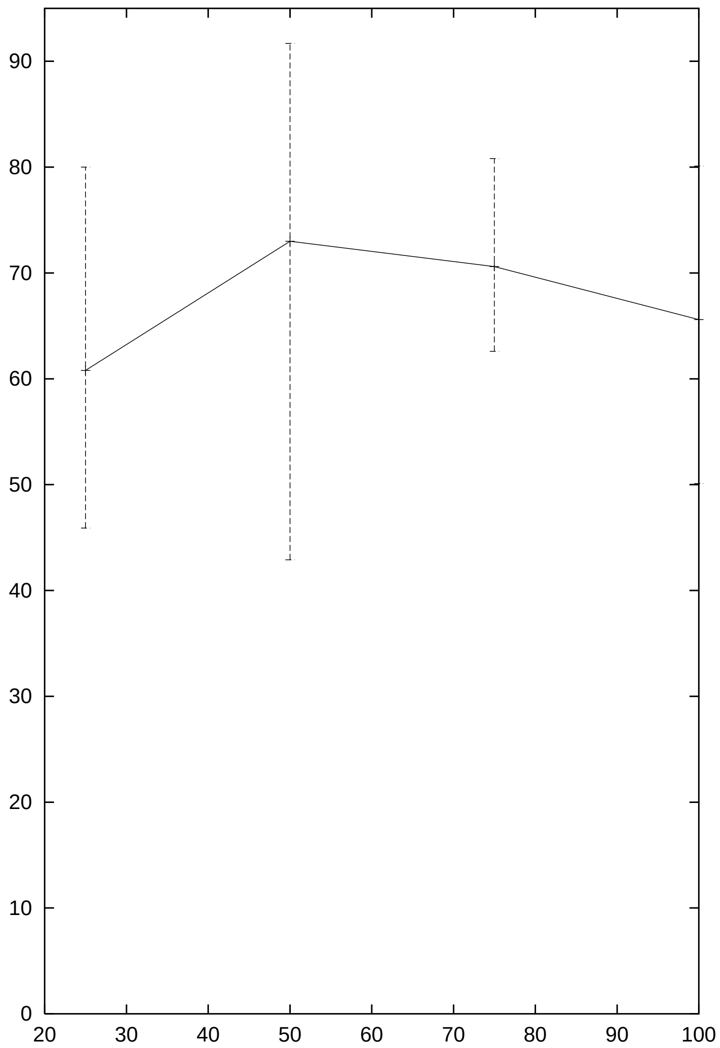
α versus Wall Time (sec.)



Number of Workers



α versus Master Iterations

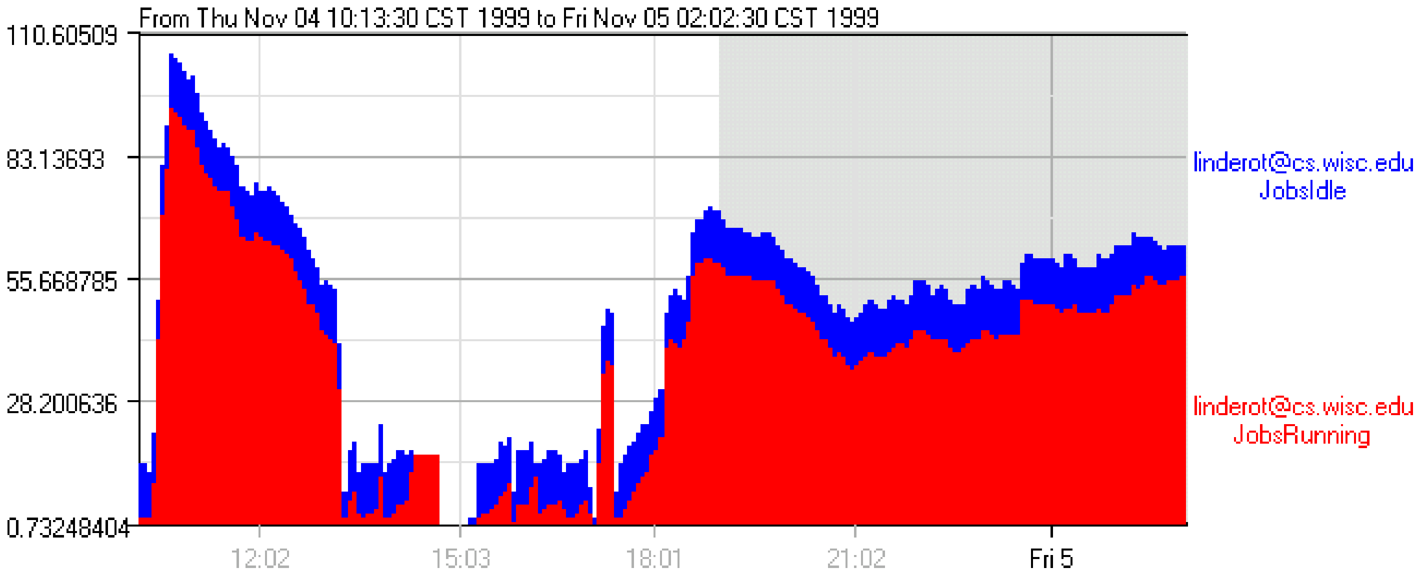


α versus "Efficiency"

The Big Kahuna!

- We can do it – ssn-10000 solves in less than 8 hours!
  - ◇ Over 200 master iterations required
  - ◇ 60 Different Workers
  - ◇ "Efficiency" of 73%

UW-Madison Comp Sci Condor Pool User Statistics for Day



Future Directions

- We want YOU to give us ideas!
  - Better Master Problem Solver?
  - Vary "chunk size" dynamically?
- We want YOU to give us instances
  - Convert multistage problems to two stage problems?
  - Sampling?

Conclusions

- There is much yet to do
- With MW, you can easily build Metacomputing, fault-tolerant, "opportunistic" applications
- On our extensive test suite, the asynchronous cutting plane algorithm performs well in solving stochastic programming problems.
- Metacomputing is Neat!