Advanced Features in Stochastic Programming Algorithms

IMA New Directions Short Course on Mathematical Optimization

Jeff Linderoth

Department of Industrial and Systems Engineering
Wisconsin Institutes of Discovery
University of Wisconsin-Madison

August 9, 2016
Decomposition Algorithms

Two Ways of Thinking

- Algorithms are “equivalent” regardless of how you think about them.
- But thinking in different ways gives different insights.
Decomposition Algorithms

Two Ways of Thinking

- Algorithms are “equivalent” regardless of how you think about them.
- But thinking in different ways gives different insights

Complementary Viewpoints

1. As a “large-scale” problem for which you will apply decomposition techniques
2. As a “oracle” convex optimization problem
Oracle-Based Methods
Two-Stage Stochastic LP with Recourse

- Our problem is

\[
\min_{x \in X} f(x) \overset{\text{def}}{=} c^T x + \mathbb{E}[Q(x, \omega)]
\]

where

\[
X = \{ x \in \mathbb{R}^n_+ \mid Ax = b \}
\]

\[
Q(x, \omega) = \min_{y \geq 0} \{ q(\omega)^T y \mid T(\omega)x + W(\omega)y = h(\omega) \}
\]

- In \(Q(x, \omega)\), as \(x\) changes, the right hand side of the linear program changes.

- So, we should care very much about the value function of a linear program with respect to changes in its right-hand-side: \( v : \mathbb{R}^m \to \bar{\mathbb{R}} \)

\[
v(z) = \min_{y \in \mathbb{R}^p_+} \{ q^T y \mid W y = z \}
\]
Nice Theorems

Nice Theorem 1

Assume that

- $\Pi = \{ \pi \in \mathbb{R}^m \mid W^T \pi \leq q \} \neq \emptyset$
- $\exists z_0 \in \mathbb{R}^m$ such that $\exists y_0 \geq 0$ with $Wy_0 = z_0$

then $v(z)$ is a

- proper, convex, polyhedral function
- $\partial v(z_0) = \arg \max \{ \pi^T z_0 \mid \pi \in \Pi \}$
Nice Theorems

Nice Theorem 2

Under similar conditions (on each scenario \( W_s, q_s \))

\[ f(x) = c^T x + \mathbb{E}[Q(x, \omega)] = c^T x + \phi(x) \]

is proper, convex, and polyhedral

- subgradients of \( f \) come from (transformed and aggregated) optimal dual solutions of the second stage subproblems:

\[
\partial f(x_0) = c + \sum_{s=1}^{S} p_s \left(-T_s^T \arg \max_{\pi \in \Pi_s} \{\pi^T (h_s - T_s x_0)\}\right)
\]
Easy Peasy?

\[(2SP) \quad \min_{x \in X \cap C} f(x)\]

- We know that \(f(x)\) is a "nice"\(^1\) function.
- It is also true that \(X \cap C\) is a "nice" polyhedral set, so it should be easy to solve (2SP).

What's the Problem?!

- \(f(x)\) is given implicitly: To evaluate \(f(x)\), we must solve \(S\) linear programs.

---

\(^1\)proper, convex, polyhedral
Overarching Theme

- We will approximate\(^2\) \(f\) by ever-improving functions of the form
  \[ f(x) \approx c^T x + m^k(x) \]
- Where \(m^k(x)\) is a model of our expected recourse function:
  \[ m^k(x) \approx \sum_{s=1}^{S} p_s Q(x, \omega_s) \overset{\text{def}}{=} \phi(x). \]
- We will also build ever-improving outer approximations of \(C\):
  \( (C^k \supseteq C) \).

\(^2\)often underapproximate
Overarching Theme

- We will approximate \( f \) by ever-improving functions of the form
  \[ f(x) \approx c^T x + m^k(x) \]
- Where \( m^k(x) \) is a model of our expected recourse function:
  \[ m^k(x) \approx \sum_{s=1}^{S} p_s Q(x, \omega_s) \overset{\text{def}}{=} \phi(x). \]
- We will also build ever-improving outer approximations of \( C \): \((C^k \supseteq C)\).

Since we know that \( Q(x^k, \omega_s) \) is convex, and
\[
\partial Q(x^k, \omega_s) = -T_s^T \arg \max_{\pi \in \Pi_s} \{ \pi^T (h_s - T_s x^k) \}
\]
we can underapproximate \( Q(x^k, \omega_s) \) using a (sub)-gradient inequality.
Building a model

- By definition of convexity, we get

\[ Q(x, \omega_s) \geq Q(x^k, \omega_s) + y^T(x - x^k) \quad \forall y \in \partial Q(x^k, \omega_s) \]

\[ \geq Q(x^k, \omega_s) + [-T_s^T \pi^k_s]^T(x - x^k) \]

for some \( \pi^k_s \in \arg\ max_{\pi \in \Pi_s} \{\pi^T(h_s - T_s x^k)\} \)

\[ = Q(x^k, \omega_s) + [\pi^k_s]^T T_s x^k - \pi^k_s T_s x \]

\[ = \beta^k_s + (\alpha^k_s)^T x \]

- We\(^3\) aggregate these together to build a model of \( \phi(x) \)

\[ \phi(x) = \sum_{s=1}^{S} p_s Q(X, \omega_s) \geq \sum_{s=1}^{S} p_s \beta^k_s + \sum_{s=1}^{S} p_s [\alpha^k_s]^T x \]

\[ = \bar{\beta}^k + [\bar{\alpha}^k]^T x \]

\[^3\text{sometimes}\]
Our Model

- Choose some different \( x_j \in X, \pi^j_s \in \arg \max_{\pi \in \Pi_s} \{ \pi^T(h_s - T_s x^k) \}, j = 1, \ldots k - 1 \)

\[
\beta^j_s = Q(x^j, \omega_s) + [\pi^j_s]^T T_s x^j \quad \bar{\beta}^j = \sum_{s=1}^{S} p_s \beta^j_s
\]

\[
\alpha^j_s = -\pi^k_s T_s \quad \bar{\alpha}^j = \sum_{s=1}^{S} p_s \alpha^j_s
\]

Our Model (to minimize)

\[
m^k(x) = \max_{j=1, \ldots, k-1} \{ \bar{\beta}^j + [\bar{\alpha}^j]^T x \}
\]

- We model the process of minimizing the maximum using an auxiliary variable \( \theta \):

\[
\theta \geq \bar{\beta}^j + [\bar{\alpha}^j]^T x \quad \forall j = 1, \ldots k - 1
\]
An Oracle-Based Method for Convex Minimization

- We assume $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ is a convex function given by an “oracle”: We can get values $f(x^k)$ and subgradients $s_k \in \partial f(x^k)$ for $x^k \in X$

1. Find $x^1 \in X$, $k \leftarrow 1$, $\theta^1 \leftarrow -\infty$, $\text{UB} \leftarrow \infty$, $I = \emptyset$
2. Subproblem: Compute $f(x^k)$, $s_k \in \partial f(x^k)$. $\text{UB} \leftarrow \min\{f(x^k), \text{UB}\}$
3. If $\theta^k = f(x^k)$. STOP, $x^k$ is optimal.
4. Else: $I = I \cup \{k\}$. Solve Master:

$$\min_{\theta, x \in X} \{\theta \mid \theta \geq f(x^i) + s^T_i (x - x^i) \ \forall i \in I\}.$$ 

Let solution be $x^{k+1}, \theta^k$. Go to 2.
Worth 1000 Words
Worth 1000 Words

\[ \phi(x) \]

\[ x \quad x^k \]
Worth 1000 Words
A Dumb Algorithm?

\[ x^{k+1} \in \arg\min_{x \in \mathbb{R}_+^n} \{ c^T x + m^k(x) \mid Ax = b \} \]

- What happens if you start the algorithm with an initial iterate that is the optimal solution \( x^* \)?
- Are you done?
A Dumb Algorithm?

\[ x^{k+1} \in \arg \min_{x \in \mathbb{R}^n_+} \{ c^T x + m^k(x) \mid Ax = b \} \]

- What happens if you start the algorithm with an initial iterate that is the optimal solution \( x^* \)?
- Are you done?

- Unfortunately, no.
- At the first iterations, we have a very poor model \( m^k(\cdot) \), so when we minimize this model, we may move very far away from \( x^* \).
- A variety of methods in stochastic programming use well-known methods from nonlinear programming/convex optimization to ensure that iterations are well-behaved.
Regularizing

- 🌟 Borrow the trust region concept from NLP 🌟 (Linderoth and Wright [2003])
  - At iteration $k$
    - Have an “incumbent” solution $x^k$
    - Impose constraints $\|x - x^k\|_\infty \leq \Delta_k$

- $\Delta_k$ large $\Rightarrow$ like LShaped
- $\Delta_k$ small $\Rightarrow$ “stay very close”.
- This is often called Regularizing the method.
Another (Good) Idea

- “Penalize” the length of the step you will take.
  - \( \min c^T x + \sum_{j \in C} \theta_j + 1/(2\rho)\|x - x^k\|^2 \)
  - \( \rho \) large ⇒ like LShaped
  - \( \rho \) small ⇒ “stay very close”.

- This is known as the regularized decomposition method.
- Pioneered in stochastic programming by Ruszczyński [1986].
Trust Region Effect: Step Length
Trust Region Effect: Function Values
Bundle-Trust

- These ideas are known in the nondifferentiable optimization community as “Bundle-Trust-Region” methods.
  - **Bundle** — Build up a bundle of subgradients to better approximate your function. (Get a better model $m(\cdot)$)
  - **Trust region** — Stay close (in a region you trust), until you build up a good enough bundle to model your function accurately

- Accept new iterate if it improves the objective by a “sufficient” amount. Potentially increase $\Delta_k$ or $\rho$. *(Serious Step)*
- Otherwise, improve the estimation of $\phi(x^k)$, resolve master problem, and potentially reduce $\Delta_k$ of $\rho$ *(Null Step)*
- These methods can be shown to converge, even if cuts are deleted from the master problem.
Vanilla Trust Region

- $f(x) = c^T x + \phi(x)$
- $\hat{f}^k(x) = c^T x + m^k(x)$

1. Let $x^1 \in X$, $\Delta > 0$, $\mu \in (0, 1)$, $k = 1$, $y^1 = x^1$
Vanilla Trust Region

- \( f(x) = c^T x + \phi(x) \)
- \( \hat{f}^k(x) = c^T x + m^k(x) \)

1. Let \( x^1 \in X, \Delta > 0, \mu \in (0, 1), k = 1, y^1 = x^1 \)
2. Compute \( f(y^1) \) and subgradient model update information: \( (\bar{\beta}, \bar{\alpha}_j) \) if LShaped.
Vanilla Trust Region

- \( f(x) = c^T x + \phi(x) \)
- \( \hat{f}^k(x) = c^T x + m^k(x) \)

1. Let \( x^1 \in X, \Delta > 0, \mu \in (0, 1), k = 1, y^1 = x^1 \)
2. Compute \( f(y^1) \) and subgradient model update information: \( (\bar{\beta}, \bar{\alpha}_j) \) if LShaped.
3. Master: Let \( y^{k+1} \in \arg \min \{ c^T x + m^k(x) \mid x \in (B(x^k, \Delta) \cap X) \} \)
4. Compute predicted decrease:
   \[
   \delta^k = f(x^k) - \hat{f}^k(y^{k+1})
   \]
5. If \( \delta^k \leq \epsilon \) Stop, \( y^{k+1} \) is optimal.
Vanilla Trust Region (2)

6. **Subproblems:** Compute $f(y^{k+1})$ and subgradient information. Update $m^k(x)$ with subgradient information from $y^{k+1}$.

- **If** $f(x^k) - f(y^{k+1}) \geq \mu \delta^k$, then **Serious Step:** $x^{k+1} \leftarrow y^{k+1}$
- **Else:** **Null Step:** $x^{k+1} \leftarrow x^k$
Vanilla Trust Region (2)

6 Subproblems: Compute $f(y_{k+1}^k)$ and subgradient information. Update $m_k(x)$ with subgradient information from $y_{k+1}^k$.

- If $f(x_k^k) - f(y_{k+1}^k) \geq \mu \delta_k$, then Serious Step: $x_{k+1}^k \leftarrow y_{k+1}^k$
- Else: Null Step: $x_{k+1}^k \leftarrow x_k^k$

It Works Theorem

If $\epsilon = 0$, then

$$\lim_{k \to \infty} f(x_k^k) = \min_{x \in X} f(x)$$
Level Method

Basic Idea

- Instead of restricting search to points in the neighborhood of the current iterate, you restrict the research to points whose objective lies in the neighborhood of the current iterate.
- Idea is from Lemaréchal et al. [1995]
Level Method

Basic Idea

- Instead of restricting search to points in the neighborhood of the current iterate, you restrict the research to points whose objective lies in the neighborhood of the current iterate.
- Idea is from Lemaréchal et al. [1995]

\[ m^k(x) = \max_{i=1,...,k} \{ f(x^i) + s_i^T(x - x^i) \} \]
Level Method

Basic Idea

- Instead of restricting search to points in the neighborhood of the current iterate, you restrict the research to points whose objective lies in the neighborhood of the current iterate.
- Idea is from Lemaréchal et al. [1995]

\[ m^k(x) = \max_{i=1,...,k} \{ f(x^i) + s^T_i(x - x^i) \} \]

1. Choose \( \lambda \in (0, 1) \), \( x^1 \in X \), \( k = 1 \)
2. Compute \( f(x^k) \), \( s^k \in \partial f(x^k) \), update \( m^k(x) \)
Level Method

Basic Idea

- Instead of restricting search to points in the neighborhood of the current iterate, you restrict the research to points whose objective lies in the neighborhood of the current iterate.
- Idea is from Lemaréchal et al. [1995]

\[ m^k(x) = \max_{i=1,...,k} \{ f(x^i) + s^T_i (x - x^i) \} \]

1. Choose \( \lambda \in (0,1) \), \( x^1 \in X \), \( k = 1 \)
2. Compute \( f(x^k), s^k \in \partial f(x^k) \), update \( m^k(x) \)
3. Minimize Model: \( \bar{z}^k = \min_{x \in X} m^k(x) \) Let \( \bar{z}^k = \min_{i=1,...,k} \{ f(x^i) \} \) be the best objective value you’ve seen so far
Level Method

Basic Idea

- Instead of restricting search to points in the neighborhood of the current iterate, you restrict the research to points whose objective lies in the neighborhood of the current iterate.
- Idea is from Lemaréchal et al. [1995]

\[
m^k(x) = \max_{i=1,...,k} \{ f(x^i) + s_i^T(x - x^i) \}
\]

1. Choose \( \lambda \in (0, 1) \), \( x^1 \in X \), \( k = 1 \)
2. Compute \( f(x^k) \), \( s^k \in \partial f(x^k) \), update \( m^k(x) \)
3. Minimize Model: \( \bar{z}^k = \min_{x \in X} m^k(x) \) Let \( \bar{z}^k = \min_{i=1,...,k} \{ f(x^i) \} \) be the best objective value you’ve seen so far
4. Project: \( \ell^k = \bar{z}^k + \lambda (\bar{z}^k - \bar{z}^k) \).
   \( x_{k+1} \in \arg\min_{x \in X} \{ \| x - x^k \|_2^2 \mid m^k(x) \leq \ell^k \} \). \( k \leftarrow k + 1 \). **Go to 2.**
Convergence Rate

- A function $f : X \rightarrow \mathbb{R}$ is **Lipschitz continuous** over its domain $X$ if
  $\exists L \in \mathbb{R}$ such that
  $$|f(y) - f(x)| \leq L\|y - x\| \ \forall x, y \in X.$$
Convergence Rate

- A function $f : X \to \mathbb{R}$ is **Lipschitz continuous** over its domain $X$ if $\exists L \in \mathbb{R}$ such that
  \[ |f(y) - f(x)| \leq L \| y - x \| \quad \forall x, y \in X. \]

- The **diameter** of a compact set $X$ is
  \[ \text{diam}(X) \overset{\text{def}}{=} \max_{x, y \in X} \| x - y \|. \]
Convergence Rate

Smart Guy Theorem

\[ \tilde{z}^k - \tilde{z}^k \leq \epsilon \quad \forall k \geq C(\lambda) \left( \frac{LD}{\epsilon} \right)^2, \]
Convergence Rate

Smart Guy Theorem

\[ z^k - \tilde{z}^k \leq \epsilon \quad \forall k \geq C(\lambda) \left( \frac{LD}{\epsilon} \right)^2, \]

- \( C(\lambda) = \frac{1}{\lambda(1-\lambda)^2(2-\lambda)} \)
- This rate is independent of the number of variables of the problem
- The minimum \( C(\lambda^*) = 4 \) when \( \lambda^* = 0.2929 \)
Papers with Computational Experience

- Some computational experience in Zverovich’s Ph.D. thesis: [Zverovich, 2011]
- Zverovich et al. [2012] have a nice, comprehensive comparison between
  - Solving extensive form using simplex method and barrier method
  - LShaped-method (aggregated forms)
  - Regularized Decomposition
  - Level method
  - Trust region method

Who’s the winner?
- Hard to pick. But I think level method wins, simplex on extensive form is slowest
Performance Profile [Zverovich et al., 2012]
A Dual Idea

Dual Decomposition

- Create copies of the first-stage decision variables for each scenario
A Dual Idea

**Dual Decomposition**

- Create *copies* of the first-stage decision variables for each scenario

\[
\begin{align*}
\text{minimize} & \quad \sum_{s \in S} p_s c^T x_s + q^T y_s \\
\text{subject to} & \quad A x_s = b \\
& \quad T_s x_s + W y_s = h_s \quad \forall s \in S \\
& \quad x_s \geq 0 \quad \forall s \in S \\
& \quad y_s \geq 0 \quad \forall s \in S \\
& \quad x_1 = x_2 = \ldots = x_{|S|}
\end{align*}
\]
Relax Nonanticipativity

- Constraints \(x_0 = x_1, x_0 = x_2, \ldots x_0 = x_s\) are called nonanticipativity constraints.
- We relax the nonanticipativity constraints, so the problem decomposes by scenario, and then we do Lagrangian Relaxation:

\[
\max_{\lambda_1, \ldots, \lambda_s} \sum_{s \in S} D_s(\lambda_s)
\]

where \(D_s(\lambda_s) = \min_{(x_s, y_s) \in F_s} \{p_s(c^T x_s + q^T y_s) + \lambda_s^T(x_s - x_0), \}
\]

and \(F_s = \{(x, y) \mid Ax = b, T_s x + W_s y = h_s, x \geq 0, y \geq 0\}\)
Relax Nonanticipativity

- Constraints $x_0 = x_1, x_0 = x_2, \ldots x_0 = x_s$ are called nonanticipativity constraints.
- We relax the nonanticipativity constraints, so the problem decomposes by scenario, and then we do Lagrangian Relaxation:

$$\max_{\lambda_1, \ldots, \lambda_s} \sum_{s \in S} D_s(\lambda_s)$$

where $D_s(\lambda_s) = \min_{(x_s, y_s) \in F_s} \left\{ p_s (c^T x_s + q^T y_s) + \lambda_s^T (x_s - x_0), \right\}$

and $F_s = \{(x, y) \mid Ax = b, T_s x + W_s y = h_s, x \geq 0, y \geq 0\}$

Even Fancier

- You can do Augmented Lagrangian or Progressive Hedging [Rockafellar and Wets, 1991] by adding a quadratic “proximal” term to the Lagrangian function
Bunching

- This idea is found in the works of Wets [1988] and Gassmann [1990].
- If $W_s = W, q_s = q, \forall s = 1, \ldots, S$, then to evaluate $\phi(x)$ we must solve $|S|$ linear programs that differ only in their right hand side.
Bunching

- This idea is found in the works of Wets [1988] and Gassmann [1990]
- If $W_s = W, q_s = q, \forall s = 1, \ldots, S$, then to evaluate $\phi(x)$ we must solve $|S|$ linear programs that differ only in their right hand side.
- Therefore, the dual LPs differ only the objective function:

$$
\pi^*_s \in \arg \max_{\pi} \{ \pi^T (h_s - T_s \hat{x}) : \pi^T W \leq q \}.
$$

Basic Idea

- $\pi^*_s$ is feasible for all scenarios, and we have a dual feasible basis matrix $W_B$
- For a new scenario $(h_r, T_r)$, with new objective $(h_r - T_r \hat{x})$, if all reduced costs are negative, then $\pi^*_s$ is also optimal for scenario $r$

- Use dual simplex to solve scenario linear programs evaluating $\phi(x)$
Other Ideas for 2SP

Interior Point Methods

\[ c^T x + p_1 q_1^T y_1 + p_2 q_2^T y_2 + \cdots + p_s q_s^T y_s \]
\[ Ax + T_1 x + W_1 y_1 + T_2 x + W_2 y_2 + \cdots + T_S x + W_S y_s = b \]
\[ T_1 x + W_1 y_1 + T_2 x + W_2 y_2 + \cdots + T_S x + W_S y_s = h_1 \]
\[ y_1 \in Y \quad y_2 \in Y \quad y_s \in Y \]

Since extensive form is highly structured, then matrices of kkt system that must be solved (via Newton-type methods) for interior point methods can also be exploited.
Small SP’s are Easy!

In my experience, using barrier/interior point method is faster than simplex/pivoting-based methods for solving extensive form LPs.

Jeff Linderoth (UW-Madison)
Small SP’s are Easy!

- In my experience, using barrier/interior point method is faster than simplex/pivoting-based methods for solving extensive form LPs.
Parallelizing

- In decomposition algorithms, the evaluation of $\phi(x)$ — solving the different LP’s, can be done independently.
  - If you have $K$ computers, send them each one of $|S|/K$ linear programs, and your evaluation of $\phi(x)$ will be completed $K$ times faster.
Parallelizing

- In decomposition algorithms, the evaluation of $\phi(x)$ — solving the different LP’s, can be done independently.
  - If you have $K$ computers, send them each one of $|S|/K$ linear programs, and your evaluation of $\phi(x)$ will be completed $K$ times faster.

Factors Affecting Efficiency

- Synchronization: Waiting for all parallel machines to complete
- Solving the master problem – worker machines waiting for master to complete
Worker Usage
Stamp Out Synchronicity!

- We start a new iteration only after all LPs have been evaluated
  - In cloud/heterogeneous computing environments, different processors act at different speeds, so many may wait idle for the “slowpoke”
  - Even worse, in many cloud environments, machines can be reclaimed before completing their tasks.

Distributed Computing Fact

Asynchronous methods are preferred for traditional parallel computing environments. They are nearly *required* for heterogenous and dynamic environments!
ATR – An Asynchronous Trust Region Method

- Keep a “basket” $\mathcal{B}$ of trial points for which we are evaluating the objective function
- Make decision on whether or accept new iterate $x^{k+1}$ after entire $f(x^k)$ is computed
- Convergence theory and cut deletion theory is similar to the synchronous algorithm
- Populate the basket quickly by initially solving the master problem after only $\alpha\%$ of the scenario LPs have been solved

- *Greatly* reduces the synchronicitiy requirements
- Might be doing some “unnecessary” work – the candidate points might be better if you waited for complete information from the preceding iterations
The World’s Largest LP

- Storm – A cargo flight scheduling problem (Mulvey and Ruszczyński)
- In 2000, we aimed to solve an instance with 10,000,000 scenarios
- \( x \in \mathbb{R}^{121}, y(\omega_s) \in \mathbb{R}^{1259} \)
- The deterministic equivalent is of size
  \[
  A \in \mathbb{R}^{985,032,889 \times 12,590,000,121}
  \]

- Cuts/iteration: 1024, \(|B| = 4\)
- Started from an \( N = 20000 \) solution, \( \Delta_0 = 1 \)
## The Super Storm Computer

<table>
<thead>
<tr>
<th>Number</th>
<th>Type</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>184</td>
<td>Intel/Linux</td>
<td>Argonne</td>
</tr>
<tr>
<td>254</td>
<td>Intel/Linux</td>
<td>New Mexico</td>
</tr>
<tr>
<td>36</td>
<td>Intel/Linux</td>
<td>NCSA</td>
</tr>
<tr>
<td>265</td>
<td>Intel/Linux</td>
<td>Wisconsin</td>
</tr>
<tr>
<td>88</td>
<td>Intel/Solaris</td>
<td>Wisconsin</td>
</tr>
<tr>
<td>239</td>
<td>Sun/Solaris</td>
<td>Wisconsin</td>
</tr>
<tr>
<td>124</td>
<td>Intel/Linux</td>
<td>Georgia Tech</td>
</tr>
<tr>
<td>90</td>
<td>Intel/Solaris</td>
<td>Georgia Tech</td>
</tr>
<tr>
<td>13</td>
<td>Sun/Solaris</td>
<td>Georgia Tech</td>
</tr>
<tr>
<td>9</td>
<td>Intel/Linux</td>
<td>Columbia U.</td>
</tr>
<tr>
<td>10</td>
<td>Sun/Solaris</td>
<td>Columbia U.</td>
</tr>
<tr>
<td>33</td>
<td>Intel/Linux</td>
<td>Italy (INFN)</td>
</tr>
<tr>
<td>1345</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TA-DA!!!!!

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wall clock time</td>
<td>31:53:37</td>
</tr>
<tr>
<td>CPU time</td>
<td>1.03 Years</td>
</tr>
<tr>
<td>Avg. # machines</td>
<td>433</td>
</tr>
<tr>
<td>Max # machines</td>
<td>556</td>
</tr>
<tr>
<td>Parallel Efficiency</td>
<td>67%</td>
</tr>
<tr>
<td>Master iterations</td>
<td>199</td>
</tr>
<tr>
<td>CPU Time solving the master problem</td>
<td>1:54:37</td>
</tr>
<tr>
<td>Maximum number of rows in master problem</td>
<td>39647</td>
</tr>
</tbody>
</table>
Number of Workers

![Graph showing the number of workers over time, with a peak at around 40000 seconds and a drop to 0 near 100000 seconds.](image-url)
Conclusions

- Regularization can be very important for solving two-stage stochastic programs
- Trust Region Method, Penalize Step Length, Level Method, . . .
- Dual Decomposition: Just copy the variables and penalize difference. (Lagrangian Relaxation)
- Decomposition allows for large-scale parallelism


Stochastic Decomposition

- A primary initial reference is Higle and Sen [1991]

0. Let $k = 1$, $x_k = 0$, $V = \emptyset$

1a. Draw random sample $\omega_k$, and solve...

$$
\pi_k = \arg\max_{\pi \in \mathbb{R}^m} \{ \pi^T (h(\bar{\omega}_k) - T((\bar{\omega}_k)x^k)) | W^T \pi \leq q \}
$$

1b. $V = V \cup \pi^k$. For $j = 1, 2, \ldots, k - 1$, solve

$$
\pi^j = \arg\max_{\pi \in V} \{ \pi^T (h(\bar{\omega}_j) - T((\bar{\omega}_j)x^k)) \}
$$
Stochastic Decomposition

2a. Create cut as...

\[ \theta \geq 1/k \sum_{j=1}^{k} \pi_j^T (h(\omega_j) - T(\omega_j) x_k) \]

- Call the cut \((\alpha_k + \beta_k^T x)\).

2b. For \(j = 1, 2, \ldots, k - 1\), Phase Out old cuts as

\[ \alpha_k + \beta_k^T x = \frac{k - 1}{k} (\alpha_{k-1} + \beta_{k-1}^T x) \]
Stochastic Decomposition

3. Solve Master Problem

\[(x_k, \theta_k) = \arg\min_{x \in X} c^T x + \theta\]

subject to

\[\theta \geq \alpha_k + \beta_k x \quad \forall k = 1, 2, \ldots\]

- Go to 1.
- There is some \textit{subsequence} of the \(x^k \rightarrow x^*\)
- Typically people use some sort of statistical based stopping criteria
Stochastic Approximation

- Goes back to (seminal) work of Robbins and Monro [1951].
- A class of (simple) iterative methods, where iterations take the form
  \[ x^{k+1} = x^k - \alpha_k \eta^k. \]
Stochastic Approximation

- Goes back to (seminal) work of Robbins and Monro [1951].
- A class of (simple) iterative methods, where iterations take the form
  \[ x^{k+1} = x^k - \alpha_k \eta^k. \]
- \(-\eta^k\) is a direction satisfying some property. (e.g. \(E[-\eta^k]\) is a true descent direction for \(f(x)\))
Stochastic Approximation

- Goes back to (seminal) work of Robbins and Monro [1951].
- A class of (simple) iterative methods, where iterations take the form

\[ x^{k+1} = x^k - \alpha_k \eta^k. \]

- \(-\eta^k\) is a direction satisfying some property. (e.g. \( \mathbb{E}[-\eta^k] \) is a true descent direction for \( f(x) \))
- \( \alpha_k \) chosen such that the sequence \( \{ \alpha_k \} \) converges to zero, but not too quickly:

\[ \sum_{k=1}^{\infty} \alpha_k = \infty, \sum_{k=0}^{\infty} \alpha_k^2 < \infty. \]
Stochastic Quasi-Gradient

- If $f(x)$ is convex, we can use a (negative) direction $\eta^k$ that satisfies:
  $$\mathbb{E}[\eta^k \mid x^0, x^1, \ldots, x^k] = \nabla f(x^k) + b^k,$$
  where $\{b^k\}$ is such that $\|b^k\| \to 0$.
- A primary reference is Ermoliev [1983].
- There is some numerical experience reported in Gavironski [1988].
Mirror Descent

- Pioneered in paper by Nemirovski et al. [2009]
- Instead of using iteration like
  \[ x^{k+1} = x^k - \alpha_k \eta^k. \]
  use
  \[ x^{k+1} = P_{x^k}(\beta \eta^k), \]
  where \( \eta^k \) is an unbiased estimator of \( \nabla(f(x^k)) \)
- \( P_x(\cdot) \) is the so-called prox-mapping:
  \[ P_x(y) = \arg \min_{z \in X} y^T (z - x) + V(x, z), \]
  where \( V(x, z) = \omega(z) - \omega(x) - \nabla \omega(x)^T (z - x) \), and \( \omega(x) \) is a smooth (strongly) convex function (like \( \| \cdot \|_2 \)).
- Some very nice computational results are analysis is given in Lan et al. [2011].