Computational Stochastic Programming

Jeff Linderoth

Dept. of ISyE
Dept. of CS
Univ. of Wisconsin-Madison
linderoth@wisc.edu

SPXIII Bergamo, Italy July 7, 2013

Mission Impossible!

- It is impossible to discuss all of "computational SP" in 90 minutes.
- I will focus on a few basic topics, and (try to) provide references for a few others.

Bias

Bias

- The bias of an estimator is the difference between this estimator's expected value $\mathbb{E}[\hat{\theta}]$ and the true value of the parameter θ.
- An estimator is unbiased if $\mathbb{E}[\hat{\theta} - \theta] = 0$

Bias

Bias

- The bias of an estimator is the difference between this estimator's expected value $\mathbb{E}[\hat{\theta}]$ and the true value of the parameter θ.
- An estimator is unbiased if $\mathbb{E}[\hat{\theta} - \theta] = 0$

I Am Biased!

- But this lecture is also extremely biased, in the English sense of the word.
- It will cover best things that I know most about
- There is *lots* of great work in computational SP that I (unfortunately) won't mention.

What I WILL cover

Stochastic LP w/Recourse (Primarily 2-Stage)

- Decomposition.
    - Benders Decomposition
    - Lagrangian Relaxation—Dual Decomposition
- Stochastic approximation
- Modern/Bundle-type methods.
    - Trust region methods
    - Regularized Decomposition
    - The level method
- Multistage Extensions

What I WILL cover

Stochastic LP w/Recourse (Primarily 2-Stage)

- Decomposition.
  - Benders Decomposition
  - Lagrangian Relaxation—Dual Decomposition
- Stochastic approximation
- Modern/Bundle-type methods.
  - Trust region methods
  - Regularized Decomposition
  - The level method
- Multistage Extensions

Software Tools

- SMPS format
- Some available software tools for modeling and solving
- Role of parallel computing

Other Things Covered

Sampling

- “Exterior” Sampling Methods – Sample Average Approximation
- “Interior” Sampling Methods
  - Stochastic Quasi-Gradient
  - Stochastic Decomposition
  - Mirror-Prox Methods

Other Things Covered

Sampling

- "Exterior" Sampling Methods – Sample Average Approximation
- "Interior" Sampling Methods
  - Stochastic Quasi-Gradient
  - Stochastic Decomposition
  - Mirror-Prox Methods

Stochastic Integer Programming

- Integer L-Shaped
- Dual Decomposition

Stochastic Programming

A Stochastic Program

$$\min_{x \in X} f(x) \stackrel{\text{def}}{=} \mathbb{E}_{\omega}[F(x, \omega)]$$

Stochastic Programming

A Stochastic Program

$$\min_{x \in X} f(x) \stackrel{\text{def}}{=} \mathbb{E}_\omega[F(x, \omega)]$$

2 Stage Stochastic LP w/Recourse

$$F(x, \omega) \stackrel{\text{def}}{=} c^T x + Q(x, \omega)$$

- $c^T x$: Pay me now
- $Q(x, \omega)$: Pay me later

The Recourse Problem

$$Q(x, \omega) \stackrel{\text{def}}{=} \min q(\omega)^T y$$

$$\begin{aligned} W(\omega)y &= h(\omega) - T(\omega)x \\ y &\geq 0 \end{aligned}$$

Stochastic Programming

A Stochastic Program

$$\min_{x \in X} f(x) \stackrel{\text{def}}{=} \mathbb{E}_\omega[F(x, \omega)]$$

2 Stage Stochastic LP w/Recourse

$$F(x, \omega) \stackrel{\text{def}}{=} c^T x + Q(x, \omega)$$

- $c^T x$: Pay me now
- $Q(x, \omega)$: Pay me later

The Recourse Problem

$$Q(x, \omega) \stackrel{\text{def}}{=} \min q(\omega)^T y$$

$$\begin{aligned} W(\omega) y &= h(\omega) - T(\omega) x \\ y &\geq 0 \end{aligned}$$

- $\mathbb{E}[F(x, \omega)] = c^T x + \mathbb{E}[Q(x, \omega)] \stackrel{\text{def}}{=} c^T x + \phi(x)$

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

Decomposition: A Popular Method

M

E

T

H O D

Large Scale = Extensive Form

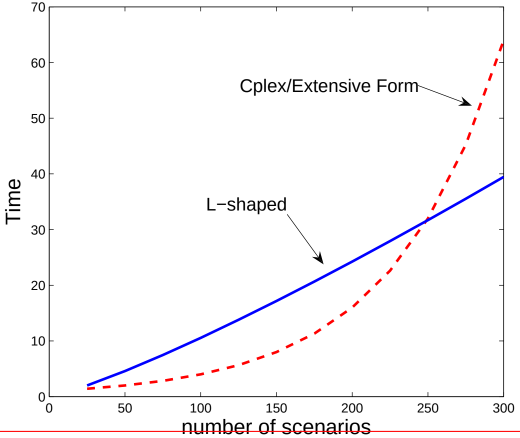
- This is sometimes called the deterministic equivalent, but I prefer the term extensive form
- Assume $\Omega = \{\omega_1, \omega_2, \ldots \omega_S\} \subseteq \mathbb{R}^r$, $P(\omega = \omega_s) = p_s, \forall s = 1, 2, \ldots, S$
- $T_s \stackrel{\text{def}}{=} T(\omega_s), h_s \stackrel{\text{def}}{=} h(\omega_s), q_s \stackrel{\text{def}}{=} q(\omega_s), W_S = W(\omega_s)$
- Then can the write extensive form as just a large LP:

Large Scale = Extensive Form

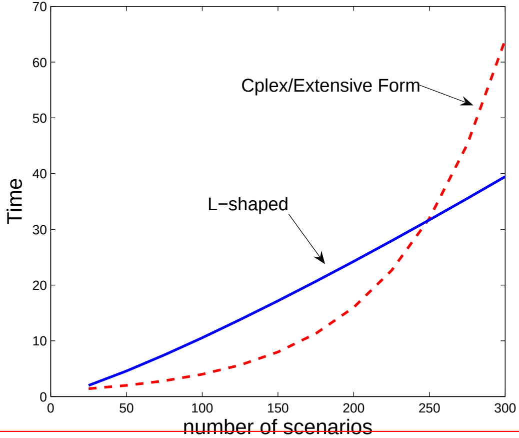
- This is sometimes called the deterministic equivalent, but I prefer the term extensive form
- Assume $\Omega = \{\omega_1, \omega_2, \ldots \omega_S\} \subseteq \mathbb{R}^r$, $P(\omega = \omega_s) = p_s, \forall s = 1, 2, \ldots, S$
- $T_s \stackrel{\text{def}}{=} T(\omega_s), h_s \stackrel{\text{def}}{=} h(\omega_s), q_s \stackrel{\text{def}}{=} q(\omega_s), W_S = W(\omega_s)$
- Then can the write extensive form as just a large LP:

$$
\begin{array}{ccccccccccc}
c^T x & + & p_1 q_1^T y_1 & + & p_2 q_2^T y_2 & + & \cdots & + & p_s q_s^T y_s & & \\
Ax & & & & & & & & & = & b \\
T_1 x & + & W_1 y_1 & & & & & & & = & h_1 \\
T_2 x & & & + & W_2 y_2 & & & & & = & h_2 \\
\vdots & & & + & & & \ddots & & & & \vdots \\
T_S x & & & & & & & + & W_S y_s & = & h_s \\
x \in X & & y_1 \in Y & & y_2 \in Y & & & & y_s \in Y & &
\end{array}
$$

Small SP's are Easy!



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.

The Upshot

- If it is too large to solve directly, then we must exploit the structure.
- If I fix the first stage variables x, then the problem decomposes by scenario

The Upshot

- If it is too large to solve directly, then we must exploit the structure.
- If I fix the first stage variables x, then the problem decomposes by scenario

$$
\begin{array}{ccccccccccc}
c^T x & + & p_1 q_1^T y_1 & + & p_2 q_2^T y_2 & + & \cdots & + & p_s q_s^T y_s & & \\
Ax & & & & & & & & & = & b \\
T_1 x & + & W_1 y_1 & & & & & & & = & h_1 \\
T_2 x & & & + & W_2 y_2 & & & & & = & h_2 \\
\vdots & & & + & & & \ddots & & & & \vdots \\
T_S x & & & & & & & + & W_S y_s & = & h_s \\
x \in X & & y_1 \in Y & & y_2 \in Y & & & & y_s \in Y & &
\end{array}
$$

The Upshot

- If it is too large to solve directly, then we must exploit the structure.
- If I fix the first stage variables x, then the problem decomposes by scenario

$$
\begin{array}{ccccccccccc}
c^T x & + & p_1 q_1^T y_1 & + & p_2 q_2^T y_2 & + & \cdots & + & p_s q_s^T y_s & & \\
Ax & & & & & & & & & = & b \\
T_1 x & + & W_1 y_1 & & & & & & & = & h_1 \\
T_2 x & & & + & W_2 y_2 & & & & & = & h_2 \\
\vdots & & & + & & & \ddots & & & & \vdots \\
T_S x & & & & & & & + & W_S y_s & = & h_s \\
x \in X & & y_1 \in Y & & y_2 \in Y & & & & y_s \in Y & &
\end{array}
$$

Key Idea

- Benders Decomposition: Characterize the solution of a scenario linear program as a function of first stage solution x

Benders of Extensive Form

- $z_{\mathrm{LP}}(x) = \sum_{s=1}^{S} z_{\mathrm{LP}}^{s}(x)$, where

$$z_{\mathrm{LP}}^{s}(x) = \inf\{q_s^{\mathsf{T}} y \mid W_s y = h_s - T_s x, y \geq 0\}$$

- The dual of the LP defining $z_{\mathrm{LP}}^{s}(x)$ is

$$\sup\{(h_s - T_s x)^{\mathsf{T}} \pi \mid W_s^{\mathsf{T}} \pi \leq q_s\}.$$

Benders of Extensive Form

- $z_{\text{LP}}(x) = \sum_{s=1}^{S} z^s_{\text{LP}}(x)$, where

$$z^s_{\text{LP}}(x) = \inf\{q_s^T y \mid W_s y = h_s - T_s x, y \geq 0\}$$

- The dual of the LP defining $z^s_{\text{LP}}(x)$ is

$$\sup\{(h_s - T_s x)^T \pi \mid W_s^T \pi \leq q_s\}.$$

- Set of dual feasible solutions for scenario s:

$$\Pi_s = \{\pi \in \mathbb{R}^m \mid W_s^T \pi \leq q_s\}$$

- Vertices of Π_s:

$$V(\Pi_s) = \{v_{1s}, v_{2s}, \ldots, v_{V_s,s}\}$$

- Extreme rays of Π_s:

$$R(\Pi_s) = \{r_{1s}, r_{2s}, \ldots, r_{R_s,s}\}$$

(In Case You Forgot...) Minkowski's Theorem

- There are two ways to describe polyhedra. As an intersection of halfspaces (by its *facets*), or by appropriate combinations of its *extreme points* and *extreme rays*

Minkowski-Weyl Theorem

Let $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ have extreme points $V(P) = \{v_1, v_2, \ldots v_V\}$ and extreme rays $R(P) = \{r_1, r_2, \ldots r_R\}$, then

$$P = \left\{x \in \mathbb{R}^n \mid x = \sum_{j=1}^{V} \lambda_j v_j + \sum_{j=1}^{R} \mu_j r_j \right.$$
$$\left. \sum_{j=1}^{V} \lambda_j = 1, \lambda_j \geq 0 \ \forall j = 1, \ldots V, \mu_j \geq 0 \ \forall j = 1, \ldots R \right\}$$

A Little LP Theory

- We assume that $z^s_{\mathrm{LP}}(x)$ is bounded below.

A Little LP Theory

- We assume that $z^s_{LP}(x)$ is bounded below.
- The LP is feasible $(z^s_{LP}(x) < +\infty)$ if there is not a direction of unboundedness in the dual LP.
    - $\nexists r \in R(\Pi_s)$ $(W_s^T r \leq 0)$ such that $(h_s - T_s x)^T r > 0$
- So x is a feasible solution $(z^s_{LP}(x) < +\infty)$ if $(h_s - T_s x)^T r \leq 0 \ \forall r = 1, \ldots, R_s$

A Little LP Theory

- We assume that $z^s_{LP}(x)$ is bounded below.
- The LP is feasible $(z^s_{LP}(x) < +\infty)$ if there is not a direction of unboundedness in the dual LP.
    - $\nexists r \in R(\Pi_s)$ $(W_s^T r \leq 0)$ such that $(h_s - T_s x)^T r > 0$
- So x is a feasible solution $(z^s_{LP}(x) < +\infty)$ if $(h_s - T_s x)^T r \leq 0 \ \forall r = 1, \ldots, R_s$
- If there is an optimal solution to an LP, then there is an optimal solution that occurs at an extreme point.
- By strong duality, the optimal solution to primal and dual LPs will have same objective value, so

$$z^s_{LP}(x) = \max_{j=1,2,\ldots V_S} \{(h_s - T_s x)^T v_{js} \mid r_{ks}^T (h_s - T_s x) \leq 0 \ \forall k = 1, 2, \ldots R_s\}$$

Developing Benders Decomposition

- Scenario s second stage feasibility set:

$$\begin{aligned} C_s &\stackrel{\text{def}}{=} \{x \mid \exists y \geq 0 \text{ with } W_s y = h_s - T_s x\} \\ &= \{x \mid h_s - T_s x \in \text{pos}(W_s)\} \end{aligned}$$

- First stage feasibility set $X \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n_+ \mid Ax = b\}$
- Second stage feasibility set: $C \stackrel{\text{def}}{=} \cap_{s=1}^{S} C_s$

$$\text{(2SP)} \qquad \min_{x \in X \cap C} f(x) \stackrel{\text{def}}{=} c^T x + \sum_{s=1}^{S} p_s Q(x, \omega_s)$$

Developing Benders Decomposition

- Scenario s second stage feasibility set:

$$\begin{aligned} C_s &\stackrel{\text{def}}{=} \{x \mid \exists y \geq 0 \text{ with } W_s y = h_s - T_s x\} \\ &= \{x \mid h_s - T_s x \in \text{pos}(W_s)\} \end{aligned}$$

- First stage feasibility set $X \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n_+ \mid Ax = b\}$
- Second stage feasibility set: $C \stackrel{\text{def}}{=} \cap_{s=1}^{S} C_s$

$$\text{(2SP)} \qquad \min_{x \in X \cap C} f(x) \stackrel{\text{def}}{=} c^T x + \sum_{s=1}^{S} p_s Q(x, \omega_s)$$

- $x \in C_s \Leftrightarrow (h_s - T_s x)^T r_{js} \leq 0 \ \forall j = 1, \ldots R_s$
- $\theta_s \geq Q(x, \omega_s) \Leftrightarrow \theta_s \geq (h_s - T_s x)^T v_{js} \ \forall j = 1, \ldots V_s$

Benders-LShaped

- Use these results
- Introduce "auxiliary" variables θ_s to represent the value of $Q(x, \omega_s)$
- N.B. I am changing notation just a little bit

Benders-LShaped

- Use these results
- Introduce "auxiliary" variables θ_s to represent the value of $Q(x, \omega_s)$
- N.B. I am changing notation just a little bit

Unaggregated: Full Multicut

$$
\begin{aligned}
\min\ c^T x + \sum_{s \in S} p_s \theta_s & \\
r^T T_s x \geq r^T h_s & \ \forall s \in S, \forall r \in R(\Pi_s) \\
\theta_s + v^T T_s x \geq v^T h_s & \ \forall s \in S, \forall v \in V(\Pi_s) \\
Ax = b & \\
x \geq 0 &
\end{aligned}
$$

LShaped Method.

- Aggregate inequalities and remove variables θ_s for each scenario.
- Instead introduce variable: $\Theta \geq \sum_{s \in S} p_s \theta_s \geq p_s(v_s^T h_s - v_s^T T_s x)$ (choosing any $v_s \in V(\Pi_s)$).

Fully-Aggregated: LShaped

$$
\begin{aligned}
\min\ & c^T x + \Theta \\
& r^T T_s x \geq r^T h_s \ \forall s \in S, \forall r \in R(\Pi_s) \\
\Theta + \sum_{s \in S} p_s v^T T_s x & \geq \sum_{s \in S} p_s v^T h_s \ \forall v \in V(\Pi_s) \\
Ax &= b \\
x &\geq 0
\end{aligned}
$$

LShaped Method.

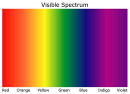
- Aggregate inequalities and remove variables θ_s for each scenario.
- Instead introduce variable: $\Theta \geq \sum_{s \in S} p_s \theta_s \geq p_s(v_s^T h_s - v_s^T T_s x)$ (choosing any $v_s \in V(\Pi_s)$).

Fully-Aggregated: LShaped

$$\begin{aligned} \min\ c^T x + \Theta & \\ r^T T_s x \geq r^T h_s & \ \forall s \in S, \forall r \in R(\Pi_s) \\ \Theta + \sum_{s \in S} p_s v^T T_s x \geq \sum_{s \in S} p_s v^T h_s & \ \forall v \in V(\Pi_s) \\ Ax = b & \\ x \geq 0 & \end{aligned}$$

- N.B. Different aggregations are possible

A Whole Spectrum

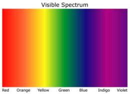


- Complete Aggregation (Traditional LShaped): One variable for $\phi(x)$
- Complete Multicut: $|S|$ variables for $\phi(x)$
- We can do anything in between...
- Partition the scenarios into C "clusters" $\mathcal{S}_1, \mathcal{S}_2, \ldots \mathcal{S}_C$.

$$\phi_{[\mathcal{S}_k]}(x) = \sum_{s \in S_k} p_s Q(x, \omega_s)$$

- $\Theta_{[\mathcal{S}_k]} \geq \sum_{s \in S_k} p_s Q(x, \omega_s)$

A Whole Spectrum



- Complete Aggregation (Traditional LShaped): One variable for $\phi(x)$
- Complete Multicut: $|S|$ variables for $\phi(x)$
- We can do anything in between...
- Partition the scenarios into C "clusters" $\mathcal{S}_1, \mathcal{S}_2, \ldots \mathcal{S}_C$.

$$\phi_{[\mathcal{S}_k]}(x) = \sum_{s \in S_k} p_s Q(x, \omega_s)$$

- $\Theta_{[\mathcal{S}_k]} \geq \sum_{s \in S_k} p_s Q(x, \omega_s)$

References

- Original multicut paper: [Birge and Louveaux, 1988]
- You need not stay with one fixed aggregation:
  - Recent paper by Trukhanov et al. [2010]
  - Ph.D. thesis by Janjarassuk [2009].

Benders Decomposition

- Regardless of aggregation, linear program is likely to have exponentially many constraints.
- Benders Decomosition is a cutting plane method for solving one of the linear programs.
- I will describe for (full) multicut, but other algorithms are really just aggregated version of this one

Benders Decomposition

- Regardless of aggregation, linear program is likely to have exponentially many constraints.
- Benders Decomosition is a cutting plane method for solving one of the linear programs.
- I will describe for (full) multicut, but other algorithms are really just aggregated version of this one

Basic Questions

- For a given $x^k, \theta_1^k, \ldots, \theta_s^k$, we must check for each scenario $s \in S$
  1. If there $\exists r \in R(\Pi_s)$ such that $rT_s x <^T h_s$
  2. If there $\exists v \in V(\Pi_s)$ such that $\theta_s^k + v^T T_s x^k < v^T h_s$

Find a Ray?

Phase 1 LP:

$$\begin{aligned} U_s(x^k) = \min \quad & 1^T u + 1^T v \\ W_s y + u - v &= h_s - T_s x^k \\ y, u, v &\geq 0 \end{aligned}$$

Its Dual

$$\begin{aligned} \max \quad & \pi^T (h_s - T_s x) \\ & W_s^T \pi \leq 0 \\ & -1 \leq \pi \leq 1 \end{aligned}$$

- If $U_s(x^k) > 0$, then by strong duality it has an optimal dual solution π^k such that $[\pi^k]^T (h_s - T_s x^k) > 0$, so if we add the inequality

  $$(h_s - T_s x)^T \pi^k \leq 0$$

  this will exclude the solution x^k
- π^k from Phase-1 LP is the dual extreme ray!

Find a Vertex

- Question: Does $\exists v \in V(\Pi_s)$ such that $\theta_s^k + v T_s x^k < v^T h_s$?
- So we should
$$\max_{v \in V(\Pi_s)} \{(h_s - T_s x^k) v\}$$
- Note this is the same as solving
$$\sup\{(h_s - T_s x)^T \pi \mid W_s^T \pi \leq q_s\}.$$
- And by duality, this is also the same as solving
$$z_{LP}^s(x) = \inf\{q_s^T y \mid W_s y = h_s - T_s x, y \geq 0\},$$
and looking at the (optimal) dual variables for the constraints $W_s y = h_s - T_s x$.

Master Problem

Cutting Plane Algorithm Will Identify

- $\mathcal{R}_s \subseteq R(\Pi_s)$ subset of extreme rays of dual feasible set Π_s
- $\mathcal{V}_s \subseteq V(\Pi_s)$ subset of extreme points of dual feasible set Π_s

Master Problem

Cutting Plane Algorithm Will Identify

- $\mathcal{R}_s \subseteq R(\Pi_s)$ subset of extreme rays of dual feasible set Π_s
- $\mathcal{V}_s \subseteq V(\Pi_s)$ subset of extreme points of dual feasible set Π_s

Full LP

$$
\begin{aligned}
\min\ c^T x + \sum_{s \in S} p_s \theta_s & \\
r^T T_s x \geq r^T h_s & \quad \forall s \in S, \forall r \in R(\Pi_s) \\
\theta_s + v^T T_s x \geq v^T h_s & \quad \forall s \in S, \forall v \in V(\Pi_s) \\
Ax = b & \\
x \geq 0 &
\end{aligned}
$$

Master Problem

$$
\begin{aligned}
\min\ c^T x + \sum_{s \in S} p_s \theta_s & \\
r^T T_s x \geq r^T h_s & \quad \forall s \in S, \forall r \in \mathcal{R}_s \\
\theta_s + v^T T_s x \geq v^T h_s & \quad \forall s \in S, \forall v \in \mathcal{V}_s \\
Ax = b & \\
x \geq 0 &
\end{aligned}
$$

Benders/Cutting Plane Method

1. $k = 1, \mathcal{R}_s = \mathcal{V}_s = \emptyset \ \forall s \in S$, $\text{LB} = -\infty$, $\text{UB} = \infty$, $x^1 \in X$

Benders/Cutting Plane Method

1. $k = 1, \mathcal{R}_s = \mathcal{V}_s = \emptyset \ \forall s \in S$, $\text{LB} = -\infty$, $\text{UB} = \infty$, $x^1 \in X$
2. $\textsc{Done} = \texttt{true}$. **For each** $s \in S$

Benders/Cutting Plane Method

1. $k = 1, \mathcal{R}_s = \mathcal{V}_s = \emptyset \ \forall s \in S$, $\textsc{lb} = -\infty$, $\textsc{ub} = \infty$, $x^1 \in X$
2. $\textsc{Done} =$ `true`. **For each** $s \in S$
    - Solve Phase 1 LP to get $U_s(x^k)$. If $U_s(x^k) > 0 \Rightarrow Q(x^k, \omega_s) = \infty$. Let π_s^k be optimal dual solution to Phase 1 LP. $\mathcal{R}_s \leftarrow \mathcal{R}_s \cup \{\pi_s^k\}$, $\textsc{Done} =$ `false`. **Go to 5.**

Benders/Cutting Plane Method

1. $k = 1, \mathcal{R}_s = \mathcal{V}_s = \emptyset \ \forall s \in S$, $\text{LB} = -\infty$, $\text{UB} = \infty$, $x^1 \in X$
2. $\textsc{Done} = \texttt{true}$. **For each** $s \in S$
    - Solve Phase 1 LP to get $U_s(x^k)$. If $U_s(x^k) > 0 \Rightarrow Q(x^k, \omega_s) = \infty$. Let π_s^k be optimal dual solution to Phase 1 LP. $\mathcal{R}_s \leftarrow \mathcal{R}_s \cup \{\pi_s^k\}$, $\textsc{Done} = \texttt{false}$. **Go to 5.**
    - Solve Phase-2 LP for $Q(x^k, \omega_s)$, let π_s^k be its optimal dual multiplier. **If** $Q(x^k, \omega_s) > \theta_s^k$, then $\mathcal{V}_s \leftarrow \mathcal{V}_s \cup \{\pi_s^k\}$, $\textsc{Done} = \texttt{false}$.

Benders/Cutting Plane Method

1. $k = 1, \mathcal{R}_s = \mathcal{V}_s = \emptyset \ \forall s \in S$, $\text{LB} = -\infty$, $\text{UB} = \infty$, $x^1 \in X$
2. $\textsc{Done} = \texttt{true}$. **For each** $s \in S$
   - Solve Phase 1 LP to get $U_s(x^k)$. If $U_s(x^k) > 0 \Rightarrow Q(x^k, \omega_s) = \infty$. Let π_s^k be optimal dual solution to Phase 1 LP. $\mathcal{R}_s \leftarrow \mathcal{R}_s \cup \{\pi_s^k\}$, $\textsc{Done} = \texttt{false}$. **Go to 5.**
   - Solve Phase-2 LP for $Q(x^k, \omega_s)$, let π_s^k be its optimal dual multiplier. **If** $Q(x^k, \omega_s) > \theta_s^k$, then $\mathcal{V}_s \leftarrow \mathcal{V}_s \cup \{\pi_s^k\}$, $\textsc{Done} = \texttt{false}$.
3. $\text{UB} = c^T x^k + \sum_{s \in S} p_s Q(x^k, \omega_s)$

Benders/Cutting Plane Method

1. $k = 1, \mathcal{R}_s = \mathcal{V}_s = \emptyset \ \forall s \in S$, $\text{LB} = -\infty$, $\text{UB} = \infty$, $x^1 \in X$
2. $\textsc{Done} = \texttt{true}$. **For each** $s \in S$
   - Solve Phase 1 LP to get $U_s(x^k)$. If $U_s(x^k) > 0 \Rightarrow Q(x^k, \omega_s) = \infty$. Let π_s^k be optimal dual solution to Phase 1 LP. $\mathcal{R}_s \leftarrow \mathcal{R}_s \cup \{\pi_s^k\}$, $\textsc{Done} = \texttt{false}$. **Go to 5.**
   - Solve Phase-2 LP for $Q(x^k, \omega_s)$, let π_s^k be its optimal dual multiplier. **If** $Q(x^k, \omega_s) > \theta_s^k$, then $\mathcal{V}_s \leftarrow \mathcal{V}_s \cup \{\pi_s^k\}$, $\textsc{Done} = \texttt{false}$.
3. $\text{UB} = c^T x^k + \sum_{s \in S} p_s Q(x^k, \omega_s)$
4. **If** $\text{UB} - \text{LB} \leq \epsilon$ **or** $\textsc{Done} = \texttt{true}$ **then** Stop. x^k is an optimal solution.

Benders/Cutting Plane Method

1. $k = 1, \mathcal{R}_s = \mathcal{V}_s = \emptyset \ \forall s \in S$, $\text{LB} = -\infty$, $\text{UB} = \infty$, $x^1 \in X$
2. $\textsc{Done} = \texttt{true}$. **For each** $s \in S$
   - Solve Phase 1 LP to get $U_s(x^k)$. If $U_s(x^k) > 0 \Rightarrow Q(x^k, \omega_s) = \infty$. Let π_s^k be optimal dual solution to Phase 1 LP. $\mathcal{R}_s \leftarrow \mathcal{R}_s \cup \{\pi_s^k\}$, $\textsc{Done} = \texttt{false}$. **Go to 5.**
   - Solve Phase-2 LP for $Q(x^k, \omega_s)$, let π_s^k be its optimal dual multiplier. **If** $Q(x^k, \omega_s) > \theta_s^k$, then $\mathcal{V}_s \leftarrow \mathcal{V}_s \cup \{\pi_s^k\}$, $\textsc{Done} = \texttt{false}$.
3. $\text{UB} = c^T x^k + \sum_{s \in S} p_s Q(x^k, \omega_s)$
4. **If** $\text{UB} - \text{LB} \leq \epsilon$ **or** $\textsc{Done} = \texttt{true}$ **then** Stop. x^k is an optimal solution.
5. Solve **Master problem.** Let lb be its optimal solution value, and let $k \leftarrow k + 1$. Let x^k be the optimal solution to the master problem. **Go to** 2.

A First Example

$$\min x_1 + x_2$$

subject to

$$\begin{aligned} \omega_1 x_1 + x_2 &\geq 7 \\ \omega_2 x_1 + x_2 &\geq 4 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

- $\omega = (\omega_1, \omega_2) \in \Omega = \{(1, 1/3), (5/2, 2/3), (4, 1)\}$
- Each outcome has $p_s = \frac{1}{3}$

A First Example

$$\min x_1 + x_2$$

subject to

$$\begin{aligned} \omega_1 x_1 + x_2 &\geq 7 \\ \omega_2 x_1 + x_2 &\geq 4 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

- $\omega = (\omega_1, \omega_2) \in \Omega = \{(1, 1/3), (5/2, 2/3), (4, 1)\}$
- Each outcome has $p_s = \frac{1}{3}$

Huh?

- This problem doesn't make sense!

Recourse Formulation

$$
\begin{aligned}
\min x_1 + x_s + \lambda \sum_{s \in S} p_s (y_{1s} + y_{2s}) & \\
\omega_{1s} x_1 + x_2 + y_{1s} \geq 7 & \quad \forall s = 1, 2, 3 \\
\omega_{2s} x_1 + x_2 + y_{2s} \geq 4 & \quad \forall s = 1, 2, 3 \\
x_1, x_2, y_{1s}, y_{2s} \geq 0 & \quad \forall s = 1, 2, 3
\end{aligned}
$$

Recourse Formulation

$$\min x_1 + x_s + \lambda \sum_{s \in S} p_s(y_{1s} + y_{2s})$$

$$\begin{aligned} \omega_{1s}x_1 + x_2 + y_{1s} &\geq 7 \quad \forall s = 1,2,3 \\ \omega_{2s}x_1 + x_2 + y_{2s} &\geq 4 \quad \forall s = 1,2,3 \\ x_1, x_2, y_{1s}, y_{2s} &\geq 0 \quad \forall s = 1,2,3 \end{aligned}$$

Stop!

Gammer Time?

Oracle-Based Methods

Two-Stage Stochastic LP with Recourse

- Our problem is

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

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: $v : \mathbb{R}^m \to \bar{\mathbb{R}}$

$$v(z) = \min_{y \in \mathbb{R}^p_+} \{q^T y \mid Wy = z\}$$

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 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 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:

Easy Peasy?

$$\text{(2SP)} \qquad \min_{x \in X \cap C} f(x)$$

- 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:

$$m^k(x) \approx \sum_{s=1}^{S} p_s Q(x, \omega_s) \stackrel{\text{def}}{=} \phi(x).$$

- We will also build ever-improving outerapproximations of C: $(C^k \supseteq C)$.

[2] often underapproximate

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:

$$m^k(x) \approx \sum_{s=1}^{S} p_s Q(x, \omega_s) \stackrel{\text{def}}{=} \phi(x).$$

- We will also build ever-improving outerapproximations 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

[2] often underapproximate

Building a model

- By definition of convexity, we get

$$
\begin{aligned}
Q(x, \omega_s) &\geq Q(x^k, \omega_s) + y^T(x - x^k) \ \forall y \in \ \partial Q(x^k, \omega_s) \\
&\geq Q(x^k, \omega_s) + [-T_s^T \pi_s^k]^T (x - x^k) \\
&\qquad\qquad \text{for some } \pi_s^k \in \arg\max_{\pi \in \Pi_s} \{\pi^T (h_s - T_s x^k)\} \\
&= Q(x^k, \omega_s) + [\pi_s^k]^T T_s x^k - \pi_s^k T_s x \\
&= \beta_s^k + (\alpha_s^k)^T x
\end{aligned}
$$

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

$$
\begin{aligned}
\phi(x) = \sum_{s=1}^{S} p_s Q(X, \omega_s) &\geq \sum_{s=1}^{S} p_s \beta_s^k + \sum_{s=1}^{S} p_s [\alpha_s^k]^T x \\
&= \bar{\beta}^k + [\bar{\alpha}^k]^T x
\end{aligned}
$$

[3] sometimes

Our Model

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

$$\beta_s^j = Q(x^j, \omega_s) + [\pi_s^j]^T T_s x^j \qquad \bar{\beta}^j = \sum_{s=1}^{S} p_s \beta_s^j$$

$$\alpha_s^j = \qquad -\pi_s^k T_s \qquad \bar{\alpha}^j = \sum_{s=1}^{S} p_s \alpha_s^j$$

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 θ:

$$\theta \geq \bar{\beta}^j + [\bar{\alpha}^j]^T x \quad \forall j = 1, \ldots k-1$$

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 $x^k \in X$

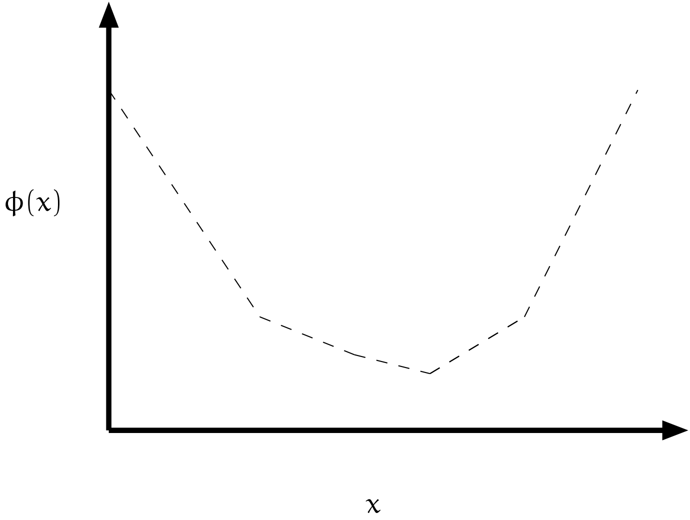
---

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:

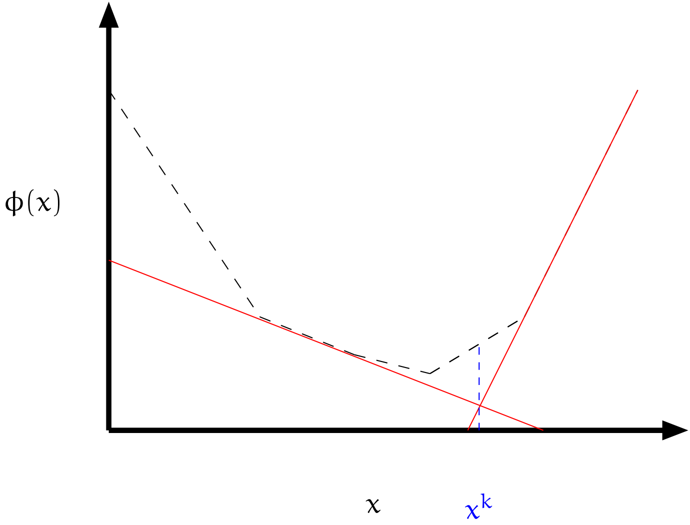
$$\min_{\theta, x \in X} \{\theta \mid \theta \geq f(x^i) + s_i^T (x - x^i) \ \forall i \in I\}.$$

Let solution be x^{k+1}, θ^k. Go to 2.

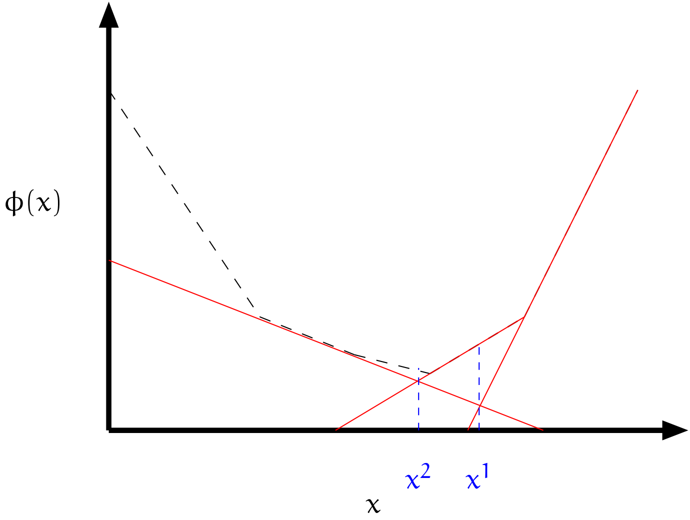
Worth 1000 Words



Worth 1000 Words



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 $\|x - x^k\|_\infty \leq \Delta_k$
- Δ_k large $\Rightarrow$ like LShaped
- Δ_k small $\Rightarrow$ "stay very close".
- This is often called *Regularizing* the method.

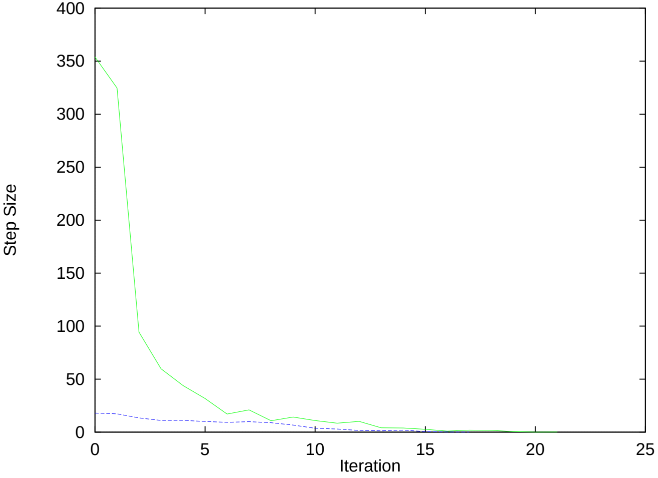
Regularizing

- Borrow the trust region concept from NLP (Linderoth and Wright [2003])
  - At iteration k
    - Have an "incumbent" solution x^k
    - Impose constraints $\|x - x^k\|_\infty \leq \Delta_k$
- Δ_k large $\Rightarrow$ like LShaped
- Δ_k small $\Rightarrow$ "stay very close".
- This is often called *Regularizing* the method.

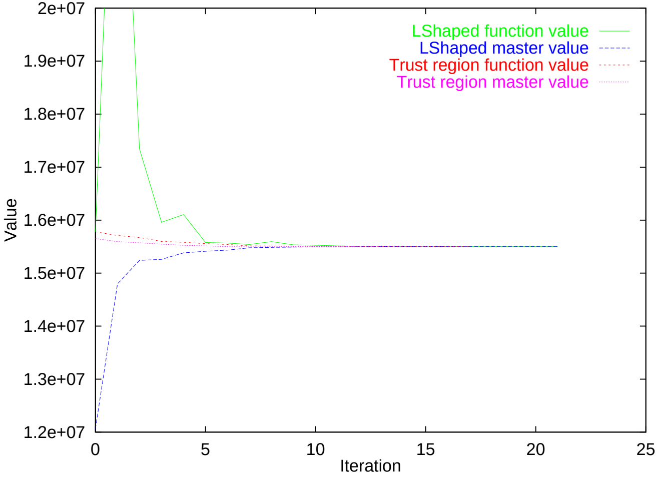
Another (Good) Idea

- "Penalize" the length of the step you will take.
  - $\min c^T x + \sum_{j \in C} \theta_j + 1/(2\rho)\|x - x^k\|^2$
  - ρ large $\Rightarrow$ like LShaped
  - ρ small $\Rightarrow$ "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 Δ_k or ρ. (*Serious Step*)
- Otherwise, improve the estimation of $\phi(x^k)$, resolve master problem, and potentially reduce Δ_k of ρ (*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 $x^1 \in X, \Delta > 0, \mu \in (0,1) \mathbb{K} = \emptyset k = 1, y^1 = x^1$

Vanilla Trust Region

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

1. Let $x^1 \in X, \Delta > 0, \mu \in (0,1) \mathbb{K} = \emptyset k = 1, y^1 = x^1$
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 $x^1 \in X, \Delta > 0, \mu \in (0,1) \mathbb{K} = \emptyset k = 1, y^1 = x^1$
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

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

1. Let $x^1 \in X, \Delta > 0, \mu \in (0,1) \mathbb{K} = \emptyset k = 1, y^1 = x^1$
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.
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:
7. $x^{k+1} \leftarrow x^k$

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]

- $m^k(x) = \max_{i=1,\ldots,k}\{f(x^i) + s_i^T(x - x^i)\}$

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]

- $m^k(x) = \max_{i=1,\ldots,k}\{f(x^i) + s_i^T(x - x^i)\}$

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 $m^k(x)$

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]

- $m^k(x) = \max_{i=1,\dots,k}\{f(x^i) + s_i^T(x - x^i)\}$

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: $\underline{z}^k = \min_{x \in X} m^k(x)$ Let $\bar{z}^k = \min_{i=1,\dots,k}\{f(x^i)\}$ 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]

- $m^k(x) = \max_{i=1,\ldots,k}\{f(x^i) + s_i^T(x - x^i)\}$

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: $\underline{z}^k = \min_{x \in X} m^k(x)$ Let $\bar{z}^k = \min_{i=1,\ldots,k}\{f(x^i)\}$ be the best objective value you've seen so far
4. Project: Let $\ell^k = \underline{z}^k + \lambda(\bar{z}^k - \underline{z}^k)$.
$x_{k+1} \in \arg\min_{x \in X}\{\|x - x^k\|^2 \mid m^k(x) \leq \ell^k\}. k \leftarrow k + 1.$ **Go to** 2.

Convergence Rate

- A function $f : X \to \mathbb{R}$ is Lipschitz continuous over its domain X if $\exists L \in \mathbb{R}$ such that

$$|f(y) - f(x)| \leq L\|y - x\| \ \forall x, y \in X.$$

Convergence Rate

- A function $f : X \to \mathbb{R}$ is Lipschitz continuous over its domain X if $\exists L \in \mathbb{R}$ such that

$$|f(y) - f(x)| \leq L\|y - x\| \;\forall x, y \in X.$$

- The diameter of a compact set X is

$$\mathrm{diam}(X) \stackrel{\mathrm{def}}{=} \max_{x,y \in X} \|x - y\|$$

Convergence Rate

- A function $f : X \to \mathbb{R}$ is Lipschitz continuous over its domain X if $\exists L \in \mathbb{R}$ such that

$$|f(y) - f(x)| \leq L\|y - x\| \ \forall x, y \in X.$$

- The diameter of a compact set X is

$$\operatorname{diam}(X) \stackrel{\text{def}}{=} \max_{x,y \in X} \|x - y\|$$

Smart Guy Theorem

$$\bar{z}^k - \underline{z}^k \leq \epsilon \quad \forall k \geq C(\lambda)\left(\frac{LD}{\epsilon}\right)^2,$$

Convergence Rate

- A function $f : X \to \mathbb{R}$ is Lipschitz continuous over its domain X if $\exists L \in \mathbb{R}$ such that

$$|f(y) - f(x)| \leq L\|y - x\| \ \forall x, y \in X.$$

- The diameter of a compact set X is

$$\mathrm{diam}(X) \stackrel{\mathrm{def}}{=} \max_{x,y \in X} \|x - y\|$$

Smart Guy Theorem

$$\bar{z}^k - \underline{z}^k \leq \epsilon \quad \forall k \geq C(\lambda)\left(\frac{LD}{\epsilon}\right)^2,$$

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

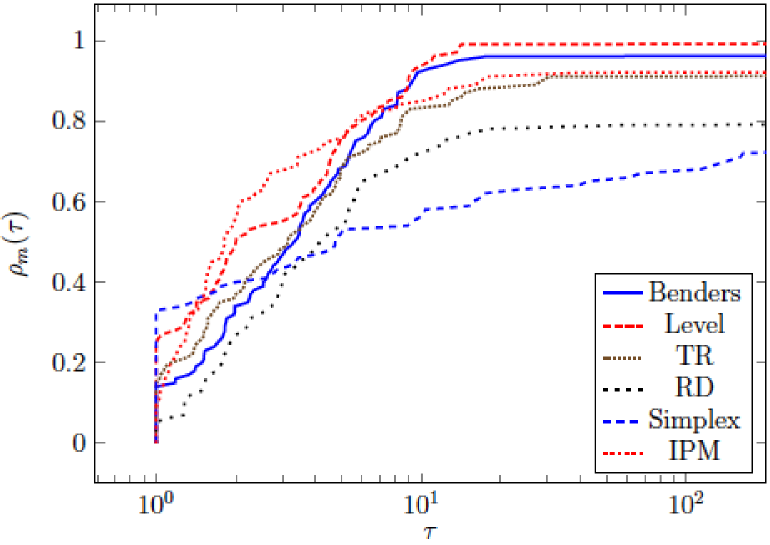
Papers with Computational Experience

- 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

minimize

$$\sum_{s \in S} p_s c^T x_s + q^T y_s$$

subject to

$$\begin{aligned}
A x_s &= b \\
T_s x_s + W y_s &= h_s \qquad \forall s \in S \\
x_s &\geq 0 \qquad \forall s \in S \\
y_s &\geq 0 \qquad \forall s \in S \\
x_1 = x_2 = &\ \ldots = x_{|S|}
\end{aligned}$$

Relax Nonanticipativity

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

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

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

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

Relax Nonanticipativity

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

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

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

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

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

- π_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 π_s^* is also *optimal* for scenario r

---

- Use dual simplex to solve scenario linear programs evaluating $\phi(x)$

Interior Point methods

$$
\begin{array}{ccccccccccccc}
c^T x & + & p_1 q_1^T y_1 & + & p_2 q_2^T y_2 & + & \cdots & + & p_s q_s^T y_s & & \\
Ax & & & & & & & & & = & b \\
T_1 x & + & W_1 y_1 & & & & & & & = & h_1 \\
T_2 x & & & + & W_2 y_2 & & & & & = & h_2 \\
\vdots & & & + & & & \ddots & & & & \vdots \\
T_S x & & & & & & & + & W_S y_s & = & h_s \\
x \in X & & y_1 \in Y & & y_2 \in Y & & & & y_s \in Y & &
\end{array}
$$

- 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.

Interior Point methods

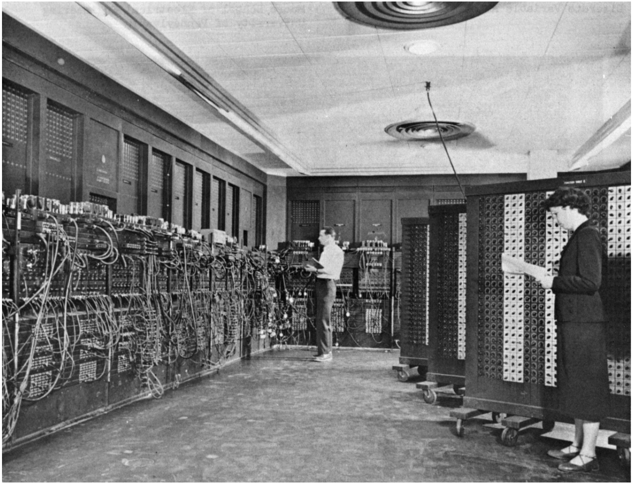
$$
\begin{array}{ccccccccccc}
c^T x & + & p_1 q_1^T y_1 & + & p_2 q_2^T y_2 & + & \cdots & + & p_s q_s^T y_s & & \\
Ax & & & & & & & & & = & b \\
T_1 x & + & W_1 y_1 & & & & & & & = & h_1 \\
T_2 x & & & + & W_2 y_2 & & & & & = & h_2 \\
\vdots & & & + & & & \ddots & & & & \vdots \\
T_S x & & & & & & & + & W_S y_s & = & h_s \\
x \in X & & y_1 \in Y & & y_2 \in Y & & & & y_s \in Y & &
\end{array}
$$

- 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.

He's The Expert!

- Definitely stick around for Jacek Gondzio's final plenary "Recent computational advances in solving very large stochastic programs", 5PM on Friday.

Computing



SMPS Format

- How do we specify a stochastic programming instance to the solver?
- We could form the extensive form ourselves, but...
  - For *really* big problems, forming the extensive form is out of the questions.
  - We need to just specify the random parts of the model.
- We can do this using SMPS format
  - There *is* some recent research work in developed stochastic programming support in an AML.

Modeling Language Support

- AMPL: (www.ampl/com)
  - Talk by Gautum Mitra: Formulation and solver support for optimisation under uncertainty, Thursday afternoon, Room 3.
  - SML: Colombo et al. [2009]. Adds keywords extending AMPL that encode problem structure.
- GAMS: (www.gams.com). Uses GAMS EMP (Extended Math Programming) framework. Manual at http://www.gams.com/dd/docs/solvers/empsp.pdf
- LINDO: (www.lindo.com). Has support for built-in sampling[4] procedures.
- MPL: (www.maximalsoftware.com). Has built-in decomposition solvers. Some introductory slides at http://www.slideshare.net/bjarnimax/seminar-stochastic

[4] We'll talk about sampling shortly

SMPS Components

- Core file
  - Like MPS file for “base” instance
- Time file
  - Specifies the time dependence structure
- Stoch file
  - Specifies the randomness

SMPS Components

- Core file
  - Like MPS file for "base" instance
- Time file
  - Specifies the time dependence structure
- Stoch file
  - Specifies the randomness

SMPS References

- Birge et al. [1987], Gassmann and Schweitzer [2001]

SMPS Core File

- Like an MPS file specifying a "base" scenario
- *Must* permute the rows and columns so that the time indexing is sequential. (Columns for stage j listed before columns for stage $j + 1$).

SMPS Core File

- Like an MPS file specifying a "base" scenario
- *Must* permute the rows and columns so that the time indexing is sequential. (Columns for stage j listed before columns for stage $j+1$).

$$\min x_1 + x_s + \lambda \sum_{s \in S} p_s (y_{1s} + y_{2s})$$

$$\begin{aligned} \omega_{1s} x_1 + x_2 + y_{1s} &\geq 7 \quad \forall s = 1,2,3 \\ \omega_{2s} x_1 + x_2 + y_{2s} &\geq 4 \quad \forall s = 1,2,3 \\ x_1, x_2, y_{1s}, y_{2s} &\geq 0 \quad \forall s = 1,2,3 \end{aligned}$$

little.cor

```
NAME          little
ROWS
 G  R0001
 G  R0002
 N  R0003
COLUMNS
    C0001     R0001     2.8276271688
    C0001     R0002     0.4599153687
    C0001     R0003     1
    C0002     R0001     1
    C0002     R0002     1
    C0002     R0003     1
    C0003     R0001     1
    C0003     R0003     5
    C0004     R0002     1
    C0004     R0003     5
RHS
    B         R0001     7
    B         R0002     4
ENDATA
```

little.tim

- Specify which row/column starts each time period.
- Must be sequential!

```
*23456789 123456789 123456789 123456789 123456789
TIME          little
PERIODS       IMPLICIT
    C0001     R0001              T1
    C0003     R0001              T2
ENDATA
```

Stoch File

- BLOCKS
  - Specify a "block" of parameters that changes together
- INDEP
  - Specify that all the parameters you are specifying are all independent random variables
- SCENARIO
  - Specify a "base" scenario
  - Specify what things change and when...

litle.sto

```
*23456789 123456789
STOCH          little
*23456789 123456789 123456789 123456789 123456789 123456789
BLOCKS         DISCRETE
 BL BLOCK1     T2           0.3333333
    C0001      R0001       1.0
    C0001      R0002       0.3333333
 BL BLOCK1     T2           0.3333333
    C0001      R0001       2.5
    C0001      R0002       0.6666666
 BL BLOCK1     T2           0.3333333
    C0001      R0001       4.0
    C0001      R0002       1.0
ENDATA
```

```
*23456789 123456789
STOCH          little
*23456789 123456789 123456789 123456789 123456789 123456789
INDEP          DISCRETE
    C0001      R0001       1.0                         0.5
    C0001      R0001       4.0                         0.5
    C0001      R0002       0.333                       0.5
    C0001      R0002       1.0                         0.5
ENDATA
```

Some Utility Libraries

- If you need to read and write SMPS files and manipulate and query the instance as part of build an algorithm in a programming language, you can try the following libraries

Some Utility Libraries

- If you need to read and write SMPS files and manipulate and query the instance as part of build an algorithm in a programming language, you can try the following libraries

---

- PySP: `https://software.sandia.gov/trac/coopr/wiki/PySP` [Watson et al., 2012]
  - Based on Pyomo [Hart et al., 2011]
  - Also allows to build models
  - Some algorithmic support, especially for progressive hedging type algorithms
  - Watson and Woodruff [2011]
- Coin-SMI: `http://www.coin-or.org/projects/Smi.xml`
  - From Coin-OR collection of open source code.
- SUTIL: `http://coral.ie.lehigh.edu/~sutil/`
  - A little bit dated, but being refectored now
  - Has implemented methods for sampling from distribution specified in SMPS files

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.

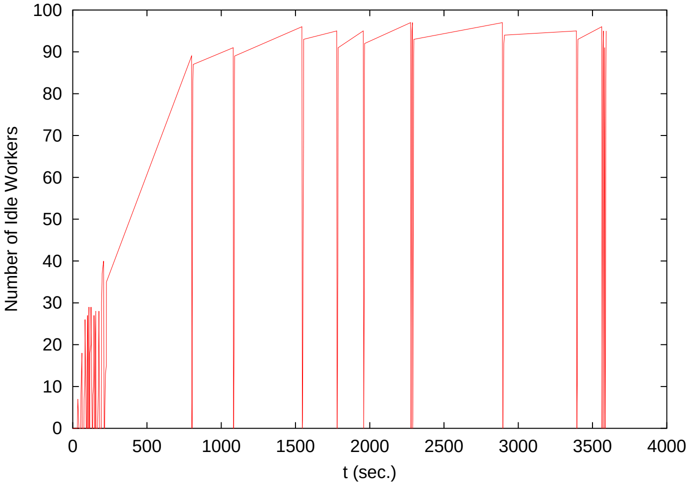
Parallelizing

- 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

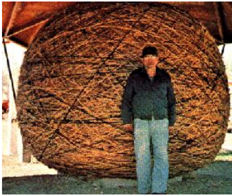
Asynchronous methods are preferred for traditional parallel computing environments. They are nearly *required* for heterogenous and dynamic environments!

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 synchronicity requirements
- Might be doing some "unnecessary" work – the candidiate points might be better if you waited for complete information from the preceeding 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, # Chunks 1024, $|\mathcal{B}| = 4$
- Started from an $N = 20000$ solution, $\Delta_0 = 1$

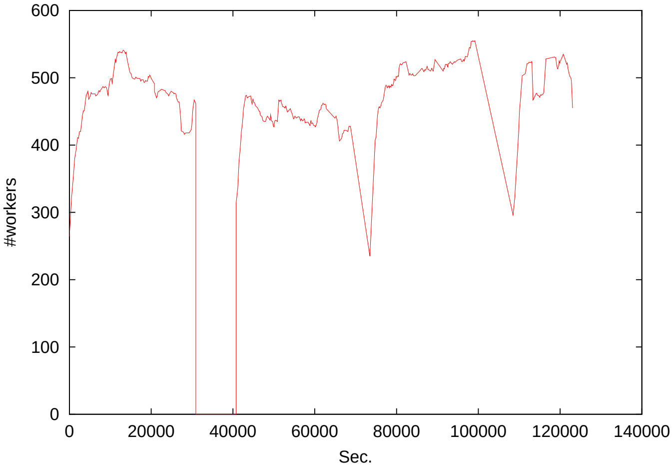
The Super Storm Computer

Number	Type	Location
184	Intel/Linux	Argonne
254	Intel/Linux	New Mexico
36	Intel/Linux	NCSA
265	Intel/Linux	Wisconsin
88	Intel/Solaris	Wisconsin
239	Sun/Solaris	Wisconsin
124	Intel/Linux	Georgia Tech
90	Intel/Solaris	Georgia Tech
13	Sun/Solaris	Georgia Tech
9	Intel/Linux	Columbia U.
10	Sun/Solaris	Columbia U.
33	Intel/Linux	Italy (INFN)
1345		

TA-DA!!!!!

Wall clock time	31:53:37
CPU time	1.03 Years
Avg. # machines	433
Max # machines	556
Parallel Efficiency	67%
Master iterations	199
CPU Time solving the master problem	1:54:37
Maximum number of rows in master problem	39647

Number of Workers



Monte Carlo Methods

The Ugly Truth

- Imagine the following (real) problem. A Telecom company wants to expand its network in a way in which to meet an unknown (random) demand.
- There are 86 unknown demands. Each demand is independent and may take on one of five values.
- $S = |\Omega| = \Pi_{k=1}^{86}(5) = 5^{86} = 4.77 \times 10^{72}$
  - The number of subatomic particles in the universe.
- How do we solve a problem that has more variables and more constraints than the number of subatomic particles in the universe?

The Ugly Truth

- Imagine the following (real) problem. A Telecom company wants to expand its network in a way in which to meet an unknown (random) demand.
- There are 86 unknown demands. Each demand is independent and may take on one of five values.
- $S = |\Omega| = \Pi_{k=1}^{86}(5) = 5^{86} = 4.77 \times 10^{72}$
  - The number of subatomic particles in the universe.
- How do we solve a problem that has more variables and more constraints than the number of subatomic particles in the universe?

- The answer is we can't!
- We solve an approximating problem obtained through sampling.

Monte Carlo Methods

$$(*) \qquad \min_{x \in X}\{f(x) \equiv \mathbb{E}_P F(x,\omega) \equiv \int_\Omega F(x,\omega)dP(\omega)\}$$

- Most of the theory presented holds for (*)—A very general SP problem
- Naturally it holds for our favorite SP problem:
  - $X \stackrel{\text{def}}{=} \{x \mid Ax = b, \ x \geq 0\}$
  - $f(x) \equiv c^T x + \mathbb{E}\{Q(x,\omega)\}$
  - $Q(x,\omega) \equiv \min_{y \geq 0}\{q(\omega)^T y \mid Wy = h(\omega) - T(\omega)x\}$

Monte Carlo Methods

$$(*) \qquad \min_{x \in X} \{ f(x) \equiv \mathbb{E}_P F(x, \omega) \equiv \int_\Omega F(x, \omega) dP(\omega) \}$$

- Most of the theory presented holds for (*)—A very general SP problem
- Naturally it holds for our favorite SP problem:
  - $X \stackrel{\text{def}}{=} \{x \mid Ax = b, \ x \geq 0\}$
  - $f(x) \equiv c^T x + \mathbb{E}\{Q(x, \omega)\}$
  - $Q(x, \omega) \equiv \min_{y \geq 0} \{q(\omega)^T y \mid Wy = h(\omega) - T(\omega)x\}$

The Dirty Secret

- Evaluating $f(x)$ is completely intractable!
- $\int \int \int \cdots \int \int \int \int \cdots$

Sampling Methods

"Interior" Sampling Methods—Sample while solving

- LShaped Method [Dantzig and Infanger, 1991]
- Stochastic Decomposition [Higle and Sen, 1991]
- Stochastic Approximation Methods
  - Stochastic Quasi-Gradient [Ermoliev, 1983]
  - Mirror-Descent Stochastic Approximation [Nemirovski et al., 2009]

Sampling Methods

"Interior" Sampling Methods—Sample while solving

- LShaped Method [Dantzig and Infanger, 1991]
- Stochastic Decomposition [Higle and Sen, 1991]
- Stochastic Approximation Methods
    - Stochastic Quasi-Gradient [Ermoliev, 1983]
    - Mirror-Descent Stochastic Approximation [Nemirovski et al., 2009]

"Exterior" sampling methods—Sample. Then Solve.

- Sample Average Approximation
- Key—Obtain (Statistical) bounds on solution quality

Sample Average Approximation

- Instead of solving (*), we solve an approximating problem.
- Let $\xi^1, \ldots, \xi^N$ be N realizations of the random variable ξ:

$$\min_{x \in S} \{ f_N(x) \equiv N^{-1} \sum_{j=1}^{N} F(x, \xi^j) \}.$$

- $f_N(x)$ is just the *sample average* function
- For any x, we consider $f_N(x)$ a random variable, as it depends on the random sample
- Since ξ^j drawn from P, $f_N(x)$ is an unbiased estimator of $f(x)$
  - $\mathbb{E}[f_N(x)] = f(x)$

Lower Bounds

$$v^* = \min_{x \in S} \{ f(x) \equiv \mathbb{E}_P F(x, \omega) \equiv \int_\Omega F(x, \omega) p(\omega) d\omega \}$$

For some sample $\xi^1, \ldots, \xi^N$, let

$$v_N = \min_{x \in S} \{ f_N(x) \equiv N^{-1} \sum_{j=1}^{N} F(x, \xi^j) \}.$$

Thm:

$$\mathbb{E}[v_N] \leq v^*$$

Proof

$$
\begin{aligned}
\min_{x\in X} N^{-1}\sum_{j=1}^{N} F(x,\xi_j) &\le N^{-1}\sum_{j=1}^{N} F(x,\xi_j) \quad \forall x\in X \qquad \Leftrightarrow \\
\mathbb{E}\left[\min_{x\in X} N^{-1}\sum_{j=1}^{N} F(x,\xi_j)\right] &\le \mathbb{E}\left[N^{-1}\sum_{j=1}^{N} F(x,\xi_j)\right] \quad \forall x\in X \Leftrightarrow \\
\mathbb{E}\left[v_N\right] &\le \mathbb{E}\left[N^{-1}\sum_{j=1}^{N} F(x,\xi_j)\right] \quad \forall x\in X \\
&\le \min_{x\in X}\mathbb{E}\left[N^{-1}\sum_{j=1}^{N} F(x,\xi_j)\right] = v^*
\end{aligned}
$$

Next?

- Now we need to somehow estimate $\mathbb{E}[v_n]$
- Idea: Generate M independent samples, $\xi^{1,j}, \ldots, \xi^{N,j}$, $j = 1, \ldots, M$, each of size N, and solve the corresponding SAA problems

$$\min_{x \in X} \left\{ f_N^j(x) := N^{-1} \sum_{i=1}^{N} F(x, \xi^{i,j}) \right\}, \tag{1}$$

- for each $j = 1, \ldots, M$. Let v_N^j be the optimal value of problem (1), and compute

$$L_{N,M} \equiv \frac{1}{M} \sum_{j=1}^{M} v_N^j$$

Lower Bounds

- The estimate $L_{N,M}$ is an unbiased estimate of $\mathbb{E}[v_N]$.
- By our last theorem, it provides a statistical lower bound for the true optimal value v^*.
- When the M batches $\xi^{1,j}, \xi^{2,j}, \ldots, \xi^{N,j}$, $j = 1, \ldots, M$, are i.i.d. (although the elements *within* each batch do not need to be i.i.d.) have by the Central Limit Theorem that

$$\sqrt{M}\,[L_{N,M} - \mathbb{E}(v_N)] \rightarrow \mathcal{N}(0, \sigma_L^2)$$

Confidence Intervals

- I can estimate the variance of my estimate $L_{M,N}$ as

$$s_L^2(M) \equiv \frac{1}{M-1} \sum_{j=1}^{M} \left(v_N^j - L_{N,M} \right)^2 .$$

Defining z_α to satisfy $P\{N(0,1) \leq z_\alpha\} = 1 - \alpha$, and replacing σ_L by $s_L(M)$, we can obtain an approximate $(1-\alpha)$-confidence interval for $\mathbb{E}[v_N]$ to be

$$\left[L_{N,M} - \frac{z_\alpha s_L(M)}{\sqrt{M}}, L_{N,M} + \frac{z_\alpha s_L(M)}{\sqrt{M}} \right]$$

Upper Bounds

$$v^* = \min_{x \in X} \{f(x) \stackrel{\text{def}}{=} \mathbb{E}_P F(x;\omega) \stackrel{\text{def}}{=} \int_\Omega F(x;\omega)p(\omega)d\omega\}$$

- Quick, Someone prove...

$$f(\hat{x}) \geq v^* \quad \forall x \in X$$

- How can we estimate $f(\hat{x})$?

Estimating $f(\hat{x})$

- Generate T independent batches of samples of size $\bar{N}$, denoted by $\xi^{1,j}, \xi^{2,j}, \ldots, \xi^{\bar{N},j}$, $j = 1, 2, \ldots, T$, where each batch has the unbiased property, namely

$$\mathbb{E}\left[f_{\bar{N}}^{j}(x) := \bar{N}^{-1}\sum_{i=1}^{\bar{N}} F(x, \xi^{i,j})\right] = f(x), \quad \text{for all } x \in X.$$

We can then use the average value defined by

$$U_{\bar{N},T}(\hat{x}) := T^{-1}\sum_{j=1}^{T} f_{\bar{N}}^{j}(\hat{x})$$

as an estimate of $f(\hat{x})$.

More Confidence Intervals

By applying the Central Limit Theorem again, we have that

$$\sqrt{T}\left[U_{\bar{N},T}(\hat{x}) - f(\hat{x})\right] \Rightarrow N(0, \sigma_U^2(\hat{x})), \text{ as } T \to \infty,$$

where $\sigma_U^2(\hat{x}) := \text{Var}\left[f_{\bar{N}}(\hat{x})\right]$. We can estimate $\sigma_U^2(\hat{x})$ by the sample variance estimator $s_U^2(\hat{x}; T)$ defined by

$$s_U^2(\hat{x}; T) := \frac{1}{T-1} \sum_{j=1}^{T} \left[f_{\bar{N}}^j(\hat{x}) - U_{\bar{N},T}(\hat{x})\right]^2.$$

By replacing $\sigma_U^2(\hat{x})$ with $s_U^2(\hat{x}; T)$, we can proceed as above to obtain a $(1-\alpha)$-confidence interval for $f(\hat{x})$:

$$\left[U_{\bar{N},T}(\hat{x}) - \frac{z_\alpha s_U(\hat{x}; T)}{\sqrt{T}}, U_{\bar{N},T}(\hat{x}) + \frac{z_\alpha s_U(\hat{x}; T)}{\sqrt{T}}\right].$$

Putting it all together

- v_N is the optimal solution value for the sample average function:
  - $v_N \equiv \min_{x \in S} \left\{ f_N(x) := N^{-1} \sum_{j=1}^{N} F(x, \omega^j) \right\}$
- Estimate $\mathbb{E}(v_N)$ as $\widehat{\mathbb{E}(v_N)} = L_{N,M} = M^{-1} \sum_{j=1}^{M} v_N^j$
  - Solve M stochastic LP's, each of sampled size N.
- $f_N(x)$ is the sample average function
  - Draw $\omega^1, \dots \omega^N$ from P
  - $f_N(x) \equiv N^{-1} \sum_{j=1}^{N} F(x, \omega^j)$
  - For Stochastic LP w/recourse $\Rightarrow$ solve N LP's.

Recapping Theorems

$$\text{Thm.} \qquad \mathbb{E}(v_N) \le v^* \le f(x) \quad \forall x$$

$$\text{Thm.} \qquad \widehat{f}_{N'}(\hat{x}) - \mathbb{E}(v_N) \to f(\hat{x}) - v^*, \text{ as } N, M, N' \to \infty$$

- We are mostly interested in estimating the quality of a given solution $\hat{x}$. This is $f(\hat{x}) - v^*$.
- $\widehat{f}_{N'}(\hat{x})$ computed by solving N' independent LPs.
- $\widehat{\mathbb{E}(v_N)}$ computed by solving M independent stochastic LPs.
- Independent $\Rightarrow$ no synchronization $\Rightarrow$ easy to do in parallel
- Independent $\Rightarrow$ can construct confidence intervals around the estimates

An experiment

- M times – Solve a stochastic sampled approximation of size N. (Thus obtaining an estimate of $\mathbb{E}(v_N)$).
- For each of the M solutions $x^1, \ldots x^M$, estimate $f(\hat{x})$ by solving N' LP's.
- Test Instances

Name	Application	$\lvert\Omega\rvert$	(m_1, n_1)	(m_2, n_2)
LandS	HydroPower Planning	10^6	(2,4)	(7,12)
gbd	Fleet Routing	6.46×10^5	(?,?)	(?,?)
storm	Cargo Flight Scheduling	6×10^{81}	(185, 121)	(?,1291)
20term	Vehicle Assignment	1.1×10^{12}	(1,5)	(71,102)
ssn	Telecom. Network Design	10^{70}	(1,89)	(175,706)

Convergence of Optimal Solution Value

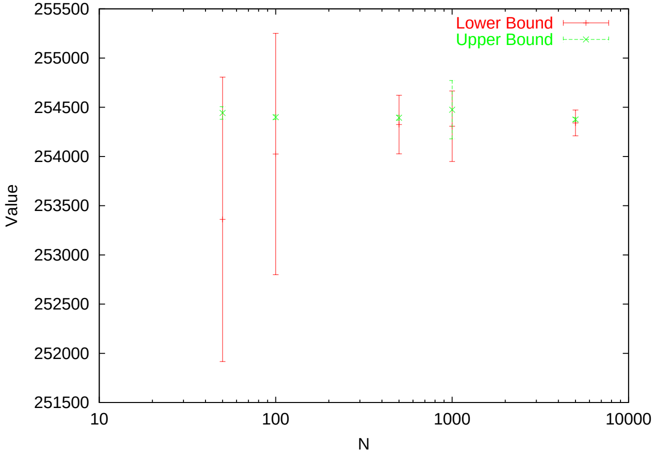
- $9 \leq M \leq 12$, $N' = 10^6$
- Monte Carlo Sampling

Instance	$N = 50$		$N = 100$		$N = 500$		$N = 1000$		$N = 5000$	
20term	253361	254442	254025	254399	254324	254394	254307	254475	254341	254376
gbd	1678.6	1660.0	1595.2	1659.1	1649.7	1655.7	1653.5	1655.5	1653.1	1655.4
LandS	227.19	226.18	226.39	226.13	226.02	226.08	225.96	226.04	225.72	226.11
storm	1550627	1550321	1548255	1550255	1549814	1550228	1550087	1550236	1549812	1550239
ssn	4.108	14.704	7.657	12.570	8.543	10.705	9.311	10.285	9.982	10.079

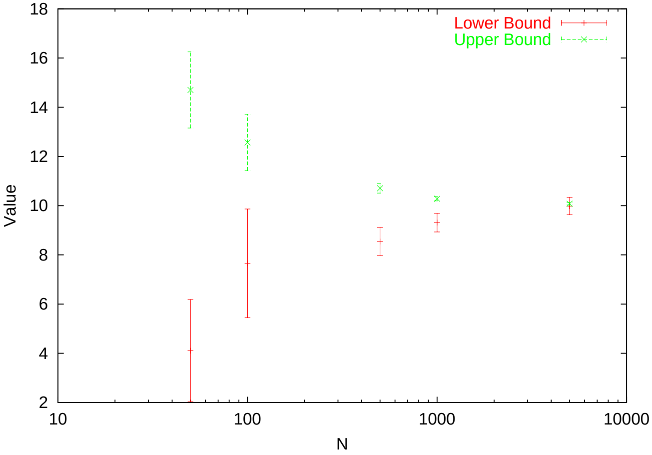
- Latin Hypercube Sampling

Instance	$N = 50$		$N = 100$		$N = 500$		$N = 1000$		$N = 5000$	
20term	254308	254368	254387	254344	254296	254318	254294	254318	254299	254313
gbd	1644.2	1655.9	1655.6	1655.6	1655.6	1655.6	1655.6	1655.6	1655.6	1655.6
LandS	222.59	222.68	225.57	225.64	225.65	225.63	225.64	225.63	225.62	225.63
storm	1549768	1549879	1549925	1549875	1549866	1549873	1549859	1549874	1549865	1549873
ssn	10.100	12.046	8.904	11.126	9.866	10.175	9.834	10.030	9.842	9.925

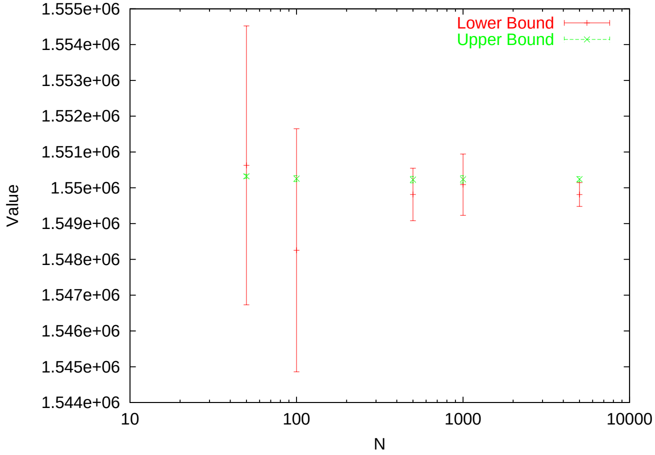
20term Convergence. Monte Carlo Sampling



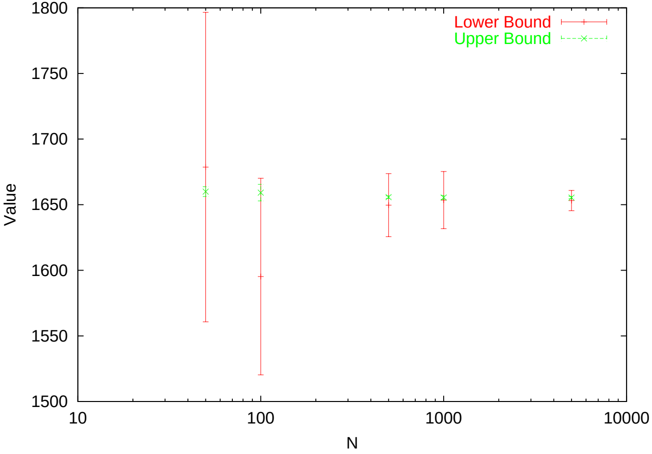
ssn Convergence. Monte Carlo Sampling



storm Convergence. Monte Carlo Sampling



gbd Convergence. Monte Carlo Sampling



Bibliography

J. R. Birge and F. V. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operations Research*, 34:384–392, 1988.

J. R. Birge, M. A. H. Dempster, H. I. Gassmann, E. A. Gunn, and A. J. King. A standard input format for multiperiod stochastic linear programs. *COAL Newsletter*, 17:1–19, 1987.

J. R. Birge, C. J. Donohue, D. F. Holmes, and O. G. Svintsiski. A parallel implementation of the nested decomposition algorithm for multistage stochastic linear programs. *Mathematical Programming*, 75:327–352, 1996.

M. Colombo, A. Grothey, J. Hogg, K. Woodsend, and J. Gondzio. A structure-conveying modelling language for mathematical and stochastic programming. *Mathematical Programming Computation*, 1(4):223–247, 2009.

G. Dantzig and G. Infanger. Large-scale stochastic linear programs — Importance sampling and Bender's decomposition. In C. Brezinski and U. Kulisch, editors, *Computational and Applied Mathematics I (Dublin, 1991)*, pages 111–120. North-Holland, Amsterdam, 1991.

Y. Ermoliev. Stochastic quasi-gradient methods and their application to systems optimization. *Stochastics*, 4:1–37, 1983.

H. I. Gassmann. MSLiP: A computer code for the multistage stochastic linear programming problem. *Mathematical Programming*, 47:427–423, 1990.

H.I. Gassmann and E. Schweitzer. A comprehensive input format for stochastic linear programs. *Annals of Operations Research*, 104:89–125, 2001.

A. Gavironski. Implementation of stochastic quasigradient methods. In *Numerical Techniques for Stochastic Optimization*. Springer-Verlag, 1988.

W. E. Hart, J.-P. Watson, and D. L. Woodruff. Pyomo: Modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, 3(3):219–260, 2011.

J. L. Higle and S. Sen. Stochastic decomposition: An algorithm for two stage linear programs with recourse. *Mathematics of Operations Research*, 16(3):650–669, 1991.

U. Janjarassuk. *Using Computational Grids for Effective Solution of Stochastic Programs*. PhD thesis, Department of Industrial and Systems Engineering, Lehigh University, 2009.

G. Lan, A. Nemirovski, and A. Shapiro. Validation analysis of mirror descent stochastic approximation method. *Mathematical Programming*, pages 1–34, 2011.

C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. New variants of bundle methods. *Mathematical Programming*, 69:111–147, 1995.

J. T. Linderoth and S. J. Wright. Implementing a decomposition algorithm for stochastic programming on a computational grid. *Computational Optimization and Applications*, 24:207–250, 2003. Special Issue on Stochastic Programming.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19:1574–1609, 2009.

H. Robbins and S. Monro. On a stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

R.T. Rockafellar and Roger J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1):119–147, 1991.

A. Ruszczyński. A regularized decomposition for minimizing a sum of polyhedral functions. *Mathematical Programming*, 35:309–333, 1986.

A. Ruszczyński. Parallel decomposition of multistage stochastic programming problems. *Mathematical Programming*, 58:201–228, 1993.

S. Trukhanov, L. Ntaimo, and A. Schaefer. Adaptive multicut aggregation for two-stage stochastic linear programs with recourse. *European Journal of Operational Research*, 206:395–406, 2010.

J.-P. Watson and D. L. Woodruff. Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science*, 8(4):355–370, 2011.

J.-P. Watson, D. L. Woodruff, and W. E. Hart. Pysp: Modeling and solving stochastic programs in Python. *Mathematical Programming Computation*, 4(2):109–149, 2012.

R. J. B. Wets. Large-scale linear programming techniques in stochastic programming. In *Numerical Techniques for Stochastic Optimization*. Springer-Verlag, 1988.

V. Zverovich. *Modelling and Solution Methods for Stochastic Optimization*. PhD thesis, Brunel university, 2011.

V. Zverovich, C. I. Fábián, E. F. D. Ellison, and G. Mitra. A computational study of a solver system for processing two-stage stochastic LPs with enhanced Benders decomposition. *Mathematical Programming Computation*, 4(3):21–238, 2012.

Miscellaneous Topics

Stochastic Decomposition

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

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

1a. Draw random sample ω_k, and solve...

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

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

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

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 *subsequence* of the $x^k \to x^*$
- 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. $\mathbb{E}[-\eta^k]$ 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 $f(x)$)
- α_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 η^k that satisfies:

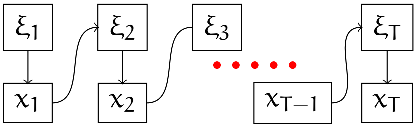
$$\mathbb{E}[\eta^k \mid x^0, x^1, \ldots, x^k] = \nabla f(x^k) + b^k,$$

  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

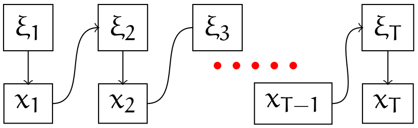
- Pioneered in paper by Nemirovski et al. [2009]
- Instead of using iteration like
$$x^{k+1} = x^k - \alpha_k \eta^k.$$
use
$$x^{k+1} = P_{x^k}(\beta \eta^k),$$
where η^k is an unbiased estimator of $\nabla(f(x^k))$
- $P_x(\cdot)$ is the so-called *prox-mapping*:
$$P_x(y) = \arg\min_{z \in X} y^T(z-x) + V(x,z),$$
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].

Multistage Decision Making



- Random vectors $\xi_1 \in \mathbb{R}^{n_1}, \xi_2 \in \mathbb{R}^{n_2}, \ldots, \xi_T \in \mathbb{R}^{n_T}$
- Make sequence of decisions $x_1 \in X_1, x_2 \in X_2, \ldots, x_T \in X_T$.

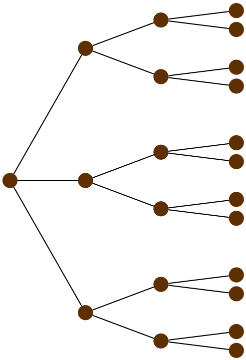
Multistage Decision Making



- Random vectors $\xi_1 \in \mathbb{R}^{n_1}, \xi_2 \in \mathbb{R}^{n_2}, \ldots, \xi_T \in \mathbb{R}^{n_T}$
- Make sequence of decisions $x_1 \in X_1, x_2 \in X_2, \ldots, x_T \in X_T$.

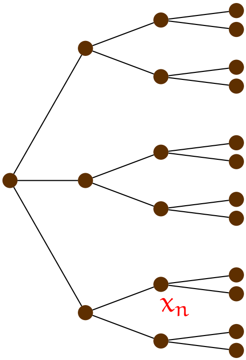
- Risk Neutral: We aim to optimize the expected value of our current decision x_t
- Linear: Assume X_t are polyhedra
- Discrete: Assume ξ_t are drawn from a discrete distribution.

Scenario Tree



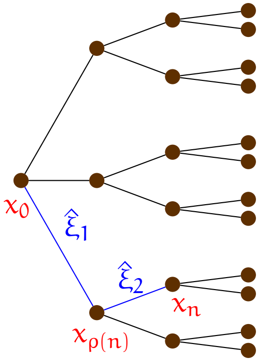
- $\mathcal{N}$: Set of nodes in the tree
- $\rho(n)$: Unique predecessor of node n in the tree
- $\mathcal{S}(n)$: Set of successor nodes of n
- q_n: Probability that the sequence of events leading to node n occurs

Scenario Tree



- $\mathcal{N}$: Set of nodes in the tree
- $\rho(n)$: Unique predecessor of node n in the tree
- $\mathcal{S}(n)$: Set of successor nodes of n
- q_n: Probability that the sequence of events leading to node n occurs
- x_n: Decision taken at node n

Scenario Tree



- N: Set of nodes in the tree
- $\rho(n)$: Unique predecessor of node n in the tree
- $\mathcal{S}(n)$: Set of successor nodes of n
- q_n: Probability that the sequence of events leading to node n occurs
- x_n: Decision taken at node n

Multistage Stochastic Programming

Deterministic Equivalent

$$z_{SP} = \min \left\{ \sum_{n \in N} q_n c_n^T x_n \;\middle|\; T_n x_{\rho(n)} + W_n x_n = h_n \;\; \forall n \in N \right\}$$

Multistage Stochastic Programming

Deterministic Equivalent

$$z_{SP} = \min \left\{ \sum_{n \in N} q_n c_n^T x_n \;\middle|\; T_n x_{\rho(n)} + W_n x_n = h_n \;\forall n \in N \right\}$$

Value Function of node n

$$\mathcal{Q}_n(x_{\rho(n)}) \stackrel{\text{def}}{=} \min_{x_n} \left\{ c_n^T x_n + \sum_{m \in \mathcal{S}(n)} \hat{q}_{mn} \mathcal{Q}_m(x_n) \mid W_n x_n = h_n - T_n x_{\rho(n)} \right\}$$

- $\hat{q}_{mn}$: conditional probability of node n given node m

Nested Decomposition

- 0: Root node of the scenario tree
- x_0: Initial state of the system

Recursive Formulation

$$z_{SP} = \mathcal{Q}_0(x_0)$$

Nested Decomposition

- 0: Root node of the scenario tree
- x_0: Initial state of the system

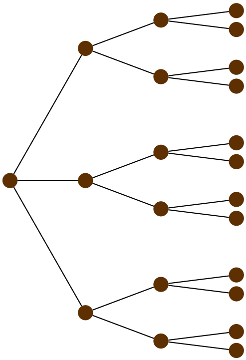
Recursive Formulation

$$z_{SP} = \mathcal{Q}_0(x_0)$$

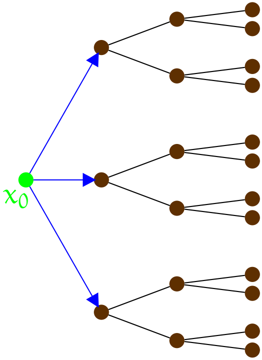
- Cost to go: $\mathcal{G}_n(x) \stackrel{\text{def}}{=} \sum_{m \in \mathcal{S}(n)} \hat{q}_{mn} \mathcal{Q}_m(x)$
- $M_n^k(x)$: Lower bound on $\mathcal{G}_n(x)$ in iteration k

$$\mathcal{Q}_n(x_{\rho(n)}) \geq \min_{x_n} \left\{ c_n^T x_n + M_n^k(x_n) \;\middle|\; W_n x_n = h_n - T_n x_{\rho(n)} \right\} \qquad ((\text{MLP}_n))$$

Action Pictures

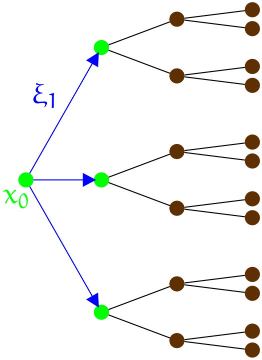


Action Pictures



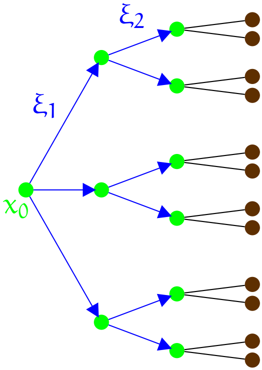
1. Solve MLP_0 to get x_0. Send policy forward

Action Pictures



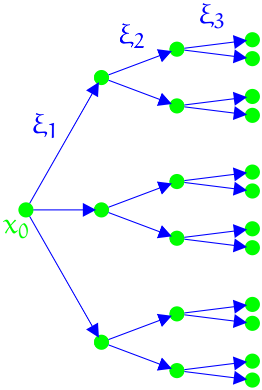
1. Solve MLP_0 to get x_0. Send policy forward
2. Solve each $\mathrm{MLP}_{\mathcal{S}_0}$ using x_0 and realizations ξ_1

Action Pictures



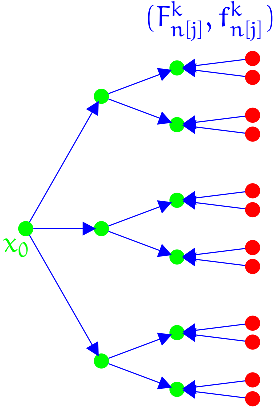
1. Solve MLP_0 to get x_0. Send policy forward
2. Solve each $\mathrm{MLP}_{\mathcal{S}_0}$ using x_0 and realizations ξ_1
3. Continue forward to end

Action Pictures



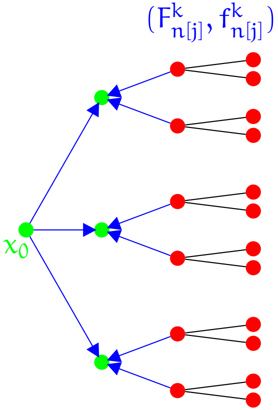
1. Solve MLP_0 to get x_0. Send policy forward
2. Solve each $\mathrm{MLP}_{\mathcal{S}_0}$ using x_0 and realizations ξ_1
3. Continue forward to end

Action Pictures



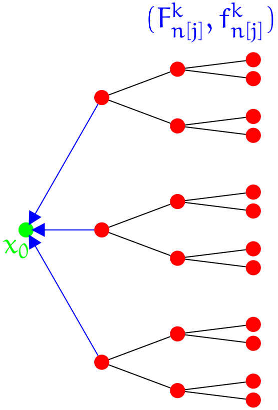
1. Solve MLP_0 to get x_0. Send policy forward
2. Solve each $MLP_{\mathcal{S}_0}$ using x_0 and realizations ξ_1
3. Continue forward to end
4. Go backwards. Send cuts from children back to parent. Update MLP_n and resolve.

Action Pictures



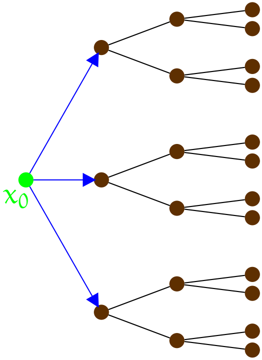
1. Solve MLP_0 to get x_0. Send policy forward
2. Solve each $\mathrm{MLP}_{\mathcal{S}_0}$ using x_0 and realizations ξ_1
3. Continue forward to end
4. Go backwards. Send cuts from children back to parent. Update MLP_n and resolve.

Action Pictures



1. Solve MLP_0 to get x_0. Send policy forward
2. Solve each $\mathrm{MLP}_{\mathcal{S}_0}$ using x_0 and realizations ξ_1
3. Continue forward to end
4. Go backwards. Send cuts from children back to parent. Update MLP_n and resolve.

Action Pictures



1. Solve MLP_0 to get x_0. Send policy forward
2. Solve each $\mathrm{MLP}_{\mathcal{S}_0}$ using x_0 and realizations ξ_1
3. Continue forward to end
4. Go backwards. Send cuts from children back to parent. Update MLP_n and resolve.
5. Lather, Rinse, Repeat.

Multistage References

- Parallel Implementation: [Ruszczyński, 1993, Birge et al., 1996]