Multi-stage Stochastic Programming, Stochastic Decomposition, and Connections to Dynamic Programming: An Algorithmic Perspective

Suvrajeet Sen
Data Driven Decisions Lab, ISE Department
Ohio State University
Columbus, OH

The Plan – Part I (70 minutes)

- The Mathematical Models
  - 2-stage Stochastic Programs with Recourse
  - T-stage Stochastic Programs with Recourse
  - Non-anticipativity and Measurability of Decisions
- Energy Infrastructure Applications of SP
- Foundations for 2-stage Programs
  - Subgradients of the Expected Recourse Function
  - Subgradient and Stochastic Quasi-gradient Methods
  - Deterministic Decomposition (Kelley/Benders/L-shaped)
  - Stochastic Decomposition

The Plan – Part II (70 minutes)

- Algorithms for T-stage Programs
  - Scenario Decomposition
    - Progressive Hedging Algorithm
  - Tree-traversal (time-staged) methods
    - Nested Benders' Decomposition & Variants
  - Discrete-time Dynamic Systems (not covered)
  - Simulating Optimization (not covered)
    - Multi-stage Extensions Stochastic Decomposition
- Comparative Remarks and Conclusions

The Mathematical Models

2-stage Stochastic Program with Recourse

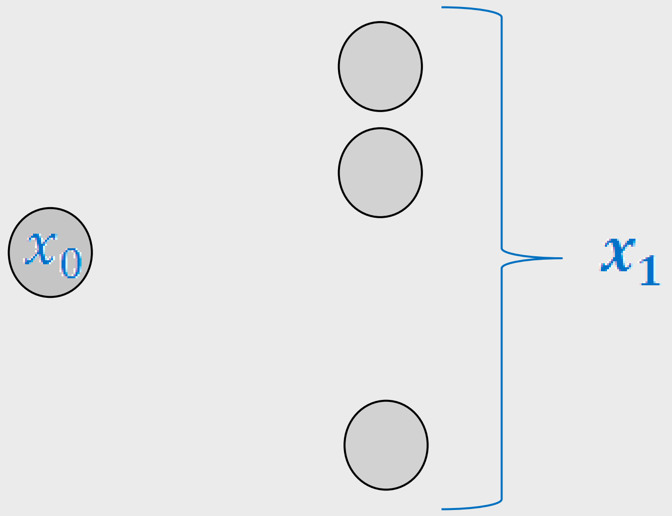
Deterministic Data

$$\text{Min}\ \ c_0 x_0 + \overbrace{\boldsymbol{E}[\boldsymbol{c}_1 \boldsymbol{x}_1]}$$

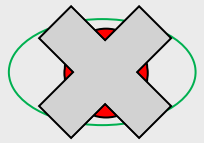
$$\boxed{C_0 x_0 \qquad\qquad = r_0}$$

$$\boxed{\boldsymbol{D}_0 x_0 + \boldsymbol{C}_1 \boldsymbol{x}_1 \quad = \boldsymbol{r}_1 \quad a.s.}$$

Stochastic Data



Avoids having to deal
with Zero Prob. Events



$$(c = c_0, p_1 c_{11} \dots, p_N c_{1N})$$

$$A = \begin{pmatrix} C_0 & 0 & \cdots & 0 \\ D_{01} & C_{11} & & \vdots \\ \vdots & 0 & \ddots & 0 \\ D_{0N} & \vdots & & C_{1N} \end{pmatrix}, b = \begin{pmatrix} r_0 \\ r_{11} \\ \vdots \\ r_{1N} \end{pmatrix}$$

SP as a Large-Scale Linear Program

T-stage Stochastic Program with Recourse

Deterministic Data

Nested Expectation Formulation … a la DP

$$\text{Min } \boxed{c_0} x_0 + \overbrace{\boldsymbol{E}[\boldsymbol{c}_1\boldsymbol{x}_1 + \cdots + \boldsymbol{E}[\boldsymbol{c}_{(T-2)}\boldsymbol{x}_{T-2} + \boldsymbol{E}[\boldsymbol{c}_{T-1}\boldsymbol{x}_{T-1}]] \ldots]}$$

$$\text{s.t. } \boxed{C_0 x_0 = r_0}$$

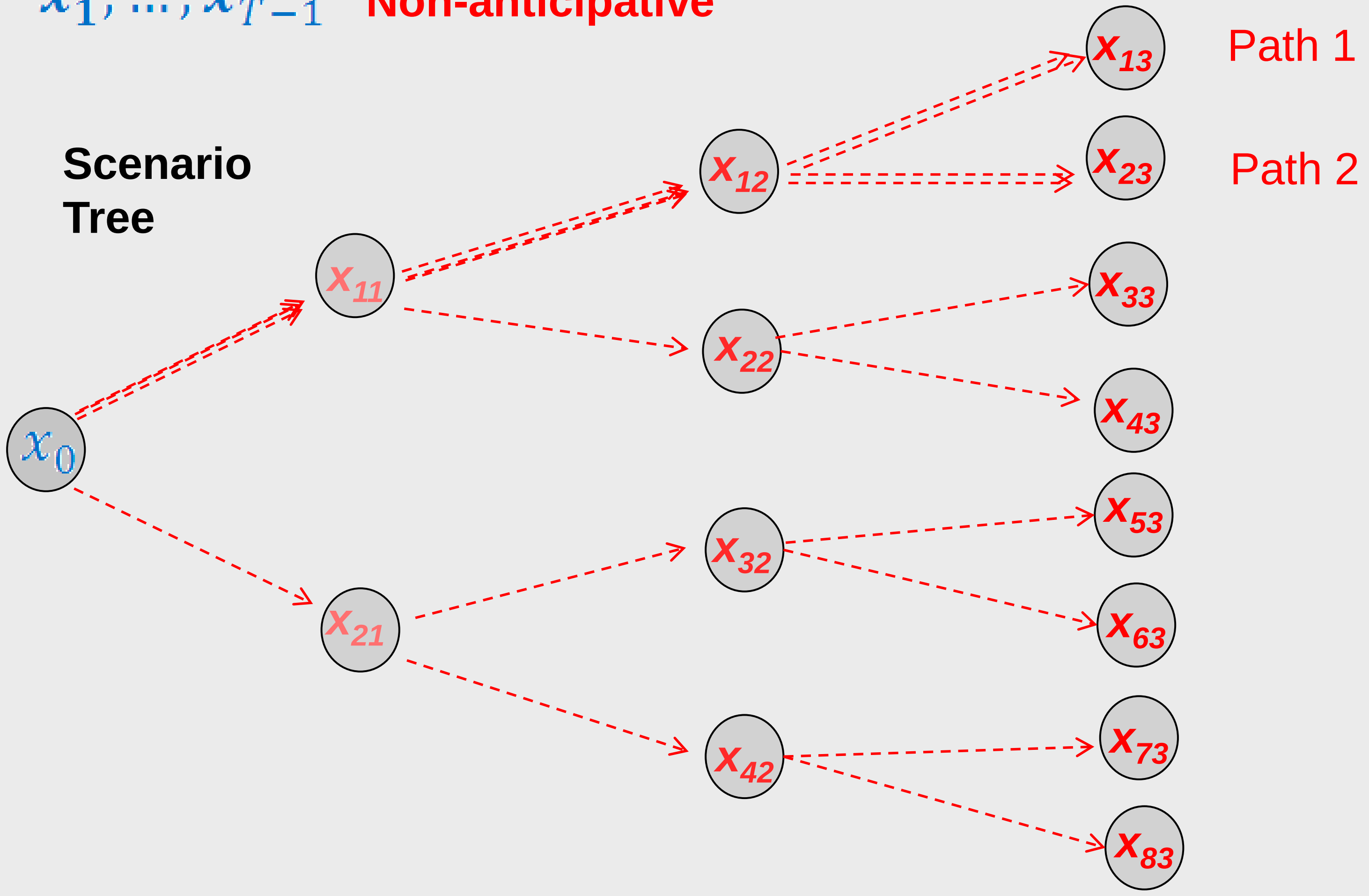
$$\boxed{\begin{array}{llll} \boldsymbol{D}_0 x_0 + \boldsymbol{C}_1\boldsymbol{x}_1 & & = \boldsymbol{r}_1 & a.s. \\ & \boldsymbol{D}_1\boldsymbol{x}_1 + \boldsymbol{C}_2\boldsymbol{x}_2 & = \boldsymbol{r}_2 & a.s. \\ & \ddots & & \\ & \boldsymbol{D}_{T-2}\boldsymbol{x}_{T-2} + \boldsymbol{C}_{T-1}\boldsymbol{x}_{T-1} & = \boldsymbol{r}_{T-1} & a.s. \end{array}}$$

Stochastic Data

$\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T-1}$ **Non-anticipative**

What do we mean by **Non-anticipativity?**

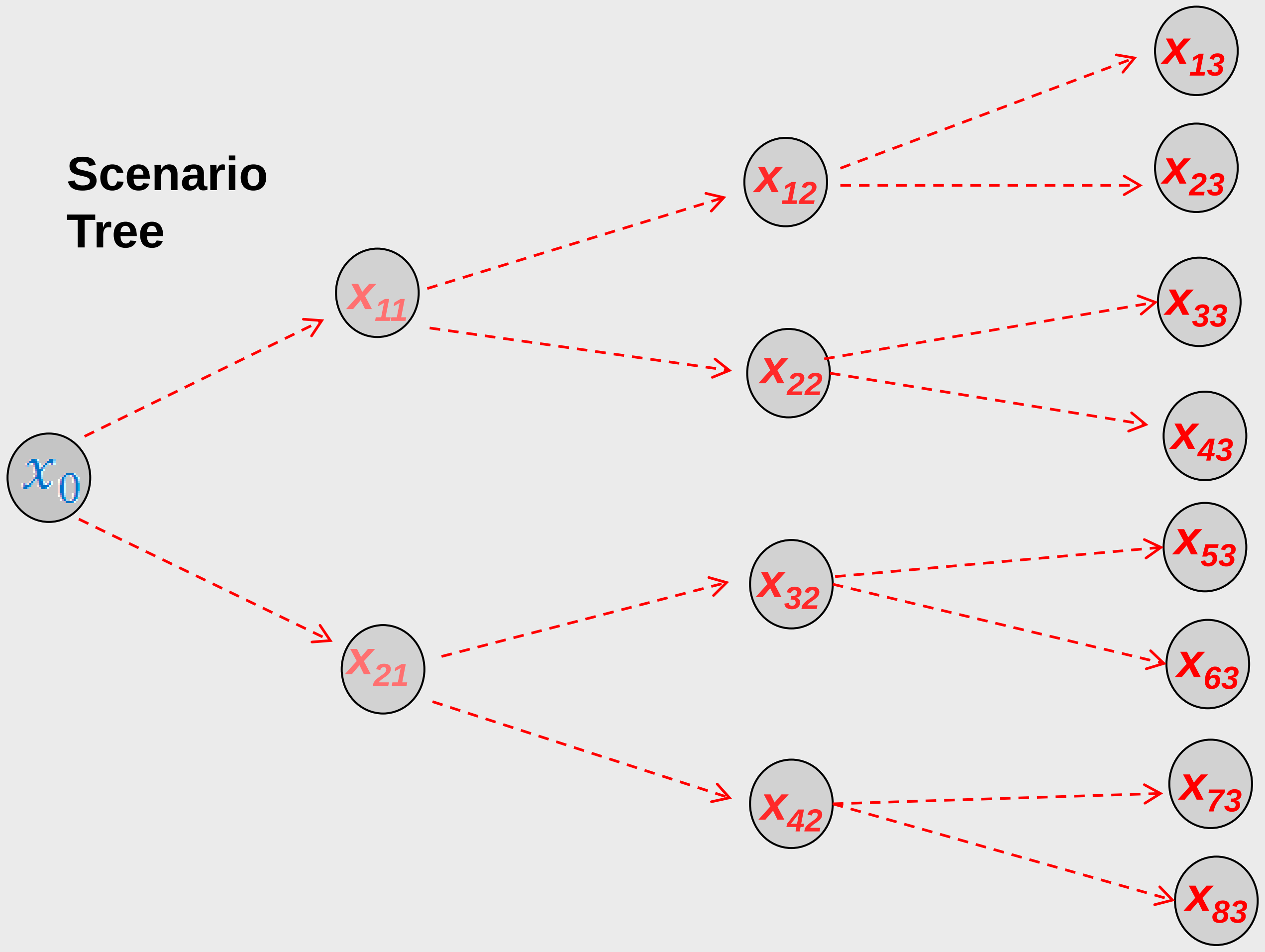
$x_1, \ldots, x_{T-1}$ **Non-anticipative**



Since paths 1 and 2 share the **same history until t=2**
They must also share the **same decisions until t=2**

Observations of the Decision Process

Scenario Tree



$(x_0, x_{11}, x_{12}, x_{13})$

$(x_0, x_{11}, x_{12}, x_{23})$

$(x_0, x_{11}, x_{22}, x_{33})$

$(x_0, x_{11}, x_{22}, x_{43})$

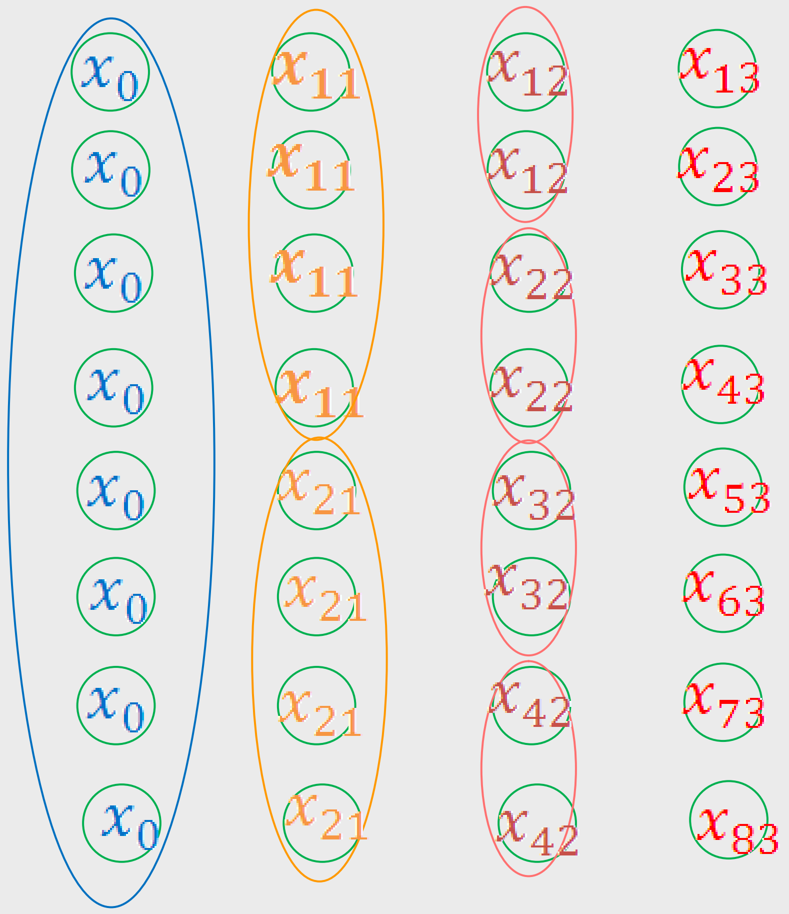
$(x_0, x_{21}, x_{32}, x_{53})$

$(x_0, x_{21}, x_{32}, x_{63})$

$(x_0, x_{21}, x_{42}, x_{73})$

$(x_0, x_{12}, x_{42}, x_{83})$

Probabilistic Language: **Measurability**

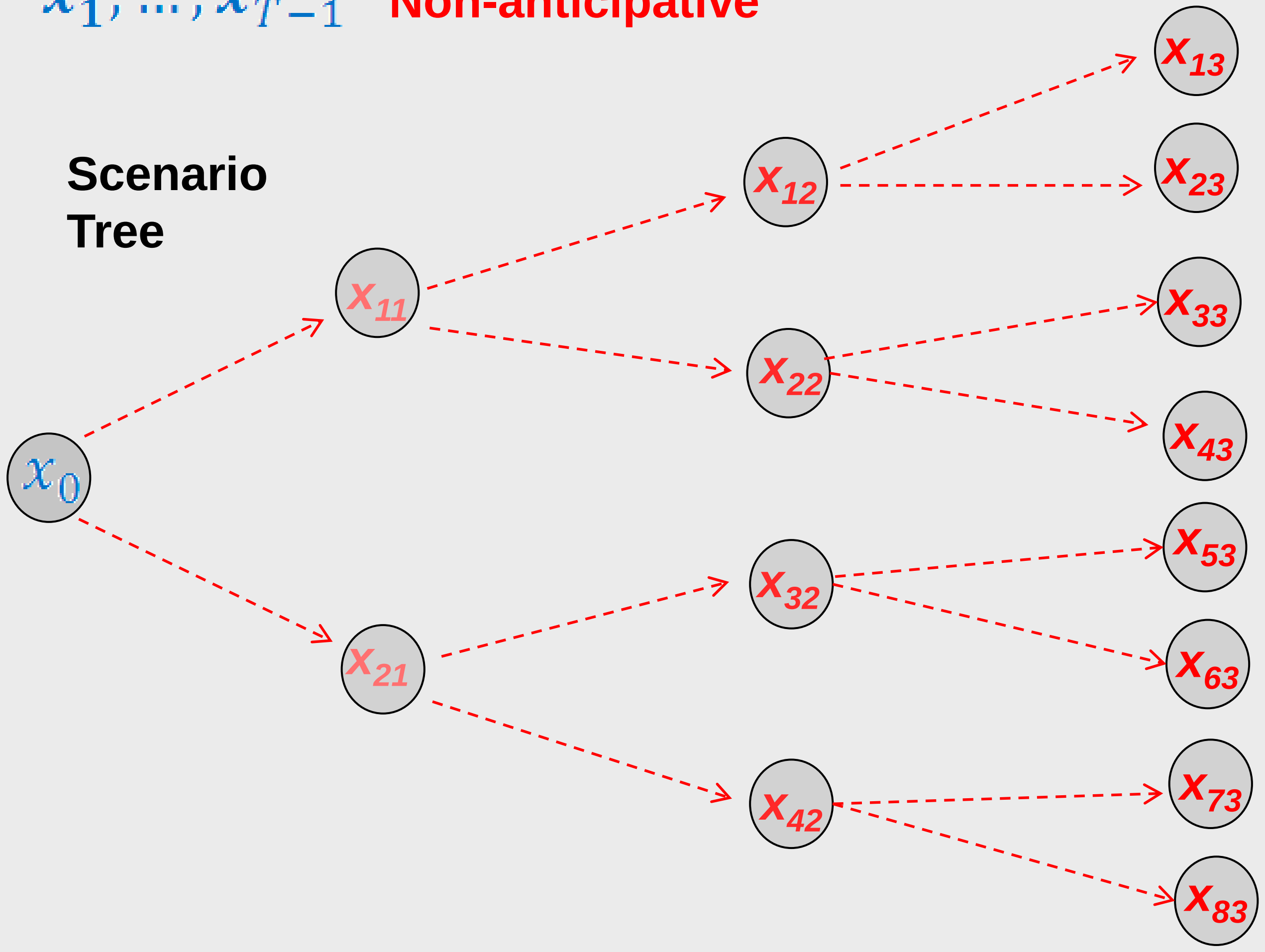


If we look at x_t as a stochastic process then one can assign probability measures (on decisions) that are consistent with the stochastic process embedded in the scenario tree.

x_t is measureable wrt to σ-algebra $\mathcal{F}_t$

What does this mean for algorithms?

$x_1, \ldots, x_{T-1}$ **Non-anticipative**



For Multi-stage SP Models, it is necessary to track decisions at each node of the scenario tree. So, for **Multi-stage SD, we will track decisions by node number.**

Matching mathematical models with algorithms

- Stochastic LPs as Linear Programs
  - Specialized Factorization Methods for Simplex and Interior Point Methods (few scenarios)
- Scenario Decomposition
  - Progressive Hedging Algorithm
- Tree-traversal (time-staged) methods
  - Nested Benders' Decomposition & Variants
- Simulating Optimization (time permitting)
  - Stochastic Decomposition

Energy Infrastructure Applications of SP

Why Stochastic Programming in Power Systems? Long History

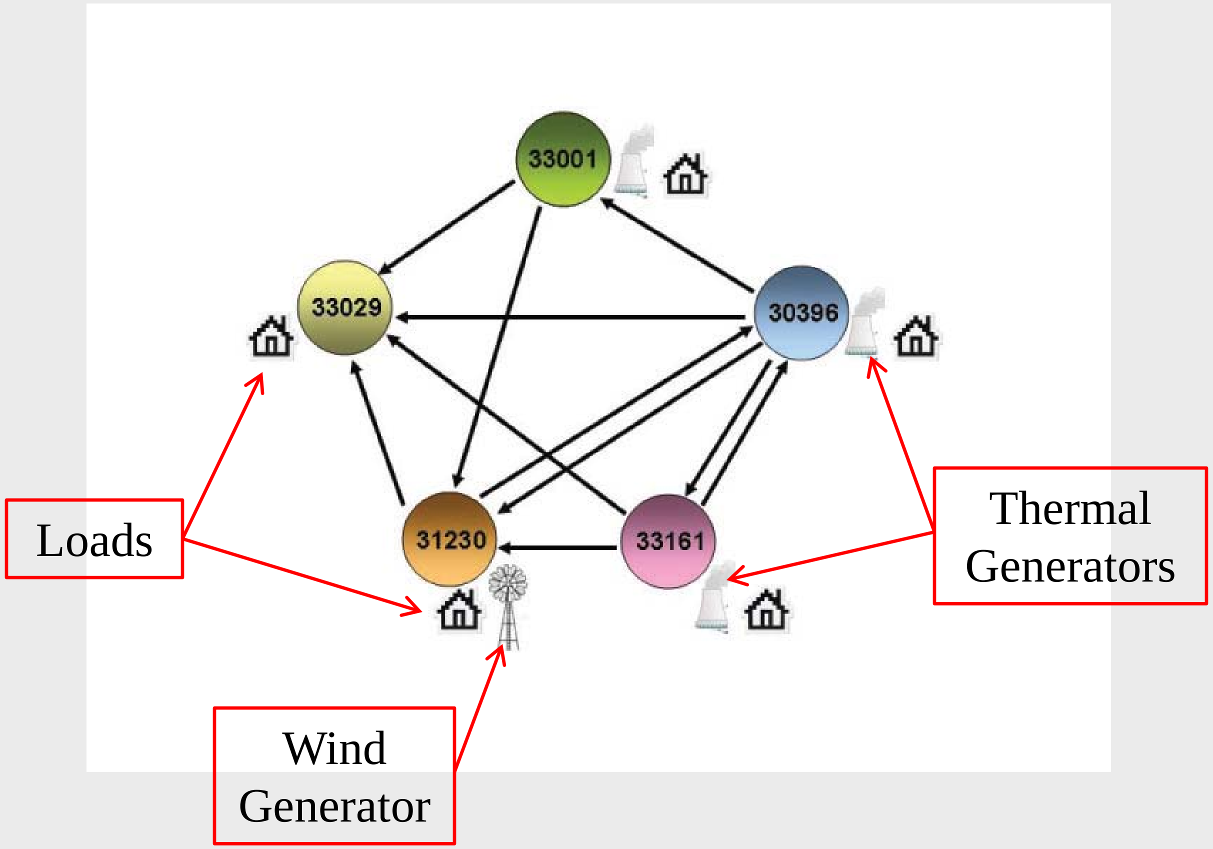
- Reliability Metrics in wide-spread use
  - Loss of Load Probability (LOLP)
  - Loss of Load Expectation (LOLE)
  - Expected Loss of Load Duration (ELOLD)
- Some SP References Prior to 2000
  - Murphy, Sen and Soyster (1981) … Generation Planning
  - Louveaux and Smeers (1981) … Generation Planning
  - Sherali, Sen and Soyster (1984) … Electricity Prices
  - Prekopa and Boros (1989) … System Reliability
  - Hobbs and Maheshwari (1990) … System Planning
  - Frauendorfer, Glavitsch, and Bacher (1992) … System Operations
  - Takriti, Birge and Long (1996) … Stochastic UC … SMIP
  - Takriti , Kassenbrink and Wu (2000) … Electricity Contracts
- Conference of Probabilistic Methods Applied to Power Systems (since the early 1990's)

Economic Dispatch Problem

- Next generation power grids highly dynamic
  - Distributed Storage
  - Cogeneration
  - Large Scale Renewable Generation
  - Real-time Pricing
- Mandates more proactive and fast operational systems like Economic Dispatch.

- The ED system updates the output levels of the committed generators to match the load demands in a cost-optimal manner.
  - Present ED systems uses forecast of the order of 2 hours (myopic).
  - Steep trends like wind ramping deteriorate the ED system.
  - Increasing the foresight and resolution of the ED problem comes at the expense of additional computational complexity.

Economic Dispatch Problem



ED Problem Formulation Features (V. Zavala, Argonne National Labs)

- 2-stage model of a T-period application
  - Decisions for first stage … played out over the next T-1 periods as the second stage
  - Randomness in second-stage is wind
- Each stage has Cost of Generation and following constraints
  - Generation capacity constraints
  - Power Flow
  - Flow Balance
  - Power Flow Bounds
  - Bus-angle Bounds
  - Wind ramp constraints (randomness)
  - Generation ramp constraints
- First stage only has generation ramp constraints (bounds)

Foundations for 2-stage Programs: A Review of Basics

The commonly stated 2-stage SLP (will be stated again, as needed)

- Let $\tilde{\omega}$ be a random variable defined on a probability space $(\Omega, \mathcal{F}, \mathcal{P})$
- Then "static" formulation of a stochastic program is given by

$$\text{Min} \left\{ \begin{array}{c} c_0(x_0) + E[h(x_0, \tilde{\omega})]: \\ x_0 \in X_0 \end{array} \right\}$$

- Why call it "static"?
  - ω (the outcome) is revealed once, and the rest of the decision-model becomes deterministic
  - This process may be repeated many times.

The Recourse Function and its Expectation

$$\begin{aligned} h(x_0;\omega) = \min\, & c_1(\omega)^\top x_1 \\ \text{s.t.}\ \ & C_1 x_1 = r_1(\omega) - D_0(\omega)x_0, \\ & x_1 \geq 0 \end{aligned}$$

- Usually, the matrix C_1 is NOT fixed … For our presentation, we will make this assumption. This is called the "Fixed-Recourse" assumption.
- Assuming that $h(x_0;\omega)$ is finite, LP duality implies

$$\begin{aligned} h(x_0;\omega) = \max\, & [r_1(\omega) - D_0(\omega)x_0]^\top \pi_1(\omega) \\ \text{s.t.}\ \ & C_1^\top \pi_1(\omega) \leq c_1(\omega) \end{aligned}$$

- Also LP theory => $h(\bullet;\omega)$ is piecewise linear, convex and moreover,

$$E[h(x_0;\tilde{\omega})] \geq E[\pi_1(\tilde{\omega})^\top r_1(\tilde{\omega})] - E[\pi_1(\tilde{\omega})^\top D_0(\tilde{\omega})]x_0$$

Subgradient Method (Shor/Polyak/Nesterov/Nemirovsky…)

Interchange of Expectation and Subdifferentiation is required here

- At iteration *k* let x_0^k be given
- Let $\beta^k = -E[\pi_1(\tilde{\omega})^\top D_0(\tilde{\omega})] \in \partial E\left[h\left(x_0^k;\, \tilde{\omega}\right)\right]$
- Then, $x_0^{k+1} \leftarrow \mathbb{P}_{X_0}\left(x_0^k - \theta_k \beta^k\right)$ where $\mathbb{P}_{X_0}$ denotes the projection operator on the set X_0 of the decisions x_0 and,

$$\theta_k \geq 0, \theta_k \to 0, \sum_k \theta_k \to \infty$$

- Note that β^k is very difficult to compute! …. Enter SQG! Use an unbiased estimate of β^k
- How? Use a sample size of *N*N_k:$\{\omega^1, \omega^2, \ldots, \omega^{N_k}\}$

Stochastic Quasi-gradient Method (SQG) (Ermoliev/ Gaivoronski/…)

- At iteration *k* let x_0^k be given
- Replace β^k of the previous slide with its unbiased estimate

$$\hat{\beta}^k = -1/N_k \sum_{t=1}^{N_k} \pi_1(\omega^t) D_0(\omega^t)$$

- Then, $x_0^{k+1} \leftarrow \mathbb{P}_{X_0}\left(x_0^k - \theta_k \hat{\beta}^k\right)$

  with $\theta_k \geq 0, \theta_k \to 0, \sum_k \theta_k \to \infty$

and, in addition: $\sum_k \theta_k^2 < \infty$

Strengths and Weaknesses of Subgradient Methods

- Strengths
  - Easy to Program, no master problem, and easily parallelizable
- Weaknesses
  - Non-adaptive step-sizes (e.g. *Constant/k*)
    - Needs a lot of fine-tuning to determining step-size (e.g. *Constant*)
  - Convergence
    - Method makes good progress early on, but like other steepest-descent type methods, there is zig-zagging behavior
  - Need ways to stop the algorithm
    - Difficult because upper and lower bounds on objective values are difficult to obtain

Kelley's Cutting Plane/Benders'/L-shaped Decomposition for 2-stage SLP (Recall Problem)

- Let $\tilde{\omega}$ be a random variable defined on a probability space $(\Omega, \mathcal{F}, \mathcal{P})$
- Then "static" formulation of a stochastic program is given by

$$\text{Min} \left\{ \begin{matrix} c_0(x_0) + E[h(x_0, \tilde{\omega})]: \\ x_0 \in X_0 \end{matrix} \right\}$$

$$\begin{aligned} h(x_0; \omega) = \min\, & c_1(\omega)^\top x_1 \\ \text{s.t. } & C_1 x_1 = r_1(\omega) - D_0(\omega) x_0, \\ & x_1 \geq 0 \end{aligned}$$

KBL Decomposition (J. Benders/Van Slyke/Wets)

- At iteration *k* let x_0^k , and $f_{k-1}(x_0)$ be given. Recall

$$E[h(x_0;\tilde{\omega})] \geq E[\pi_1(\tilde{\omega})^\top r_1(\tilde{\omega})] - E[\pi_1(\tilde{\omega})^\top D_0(\tilde{\omega})]x_0$$

- Then define

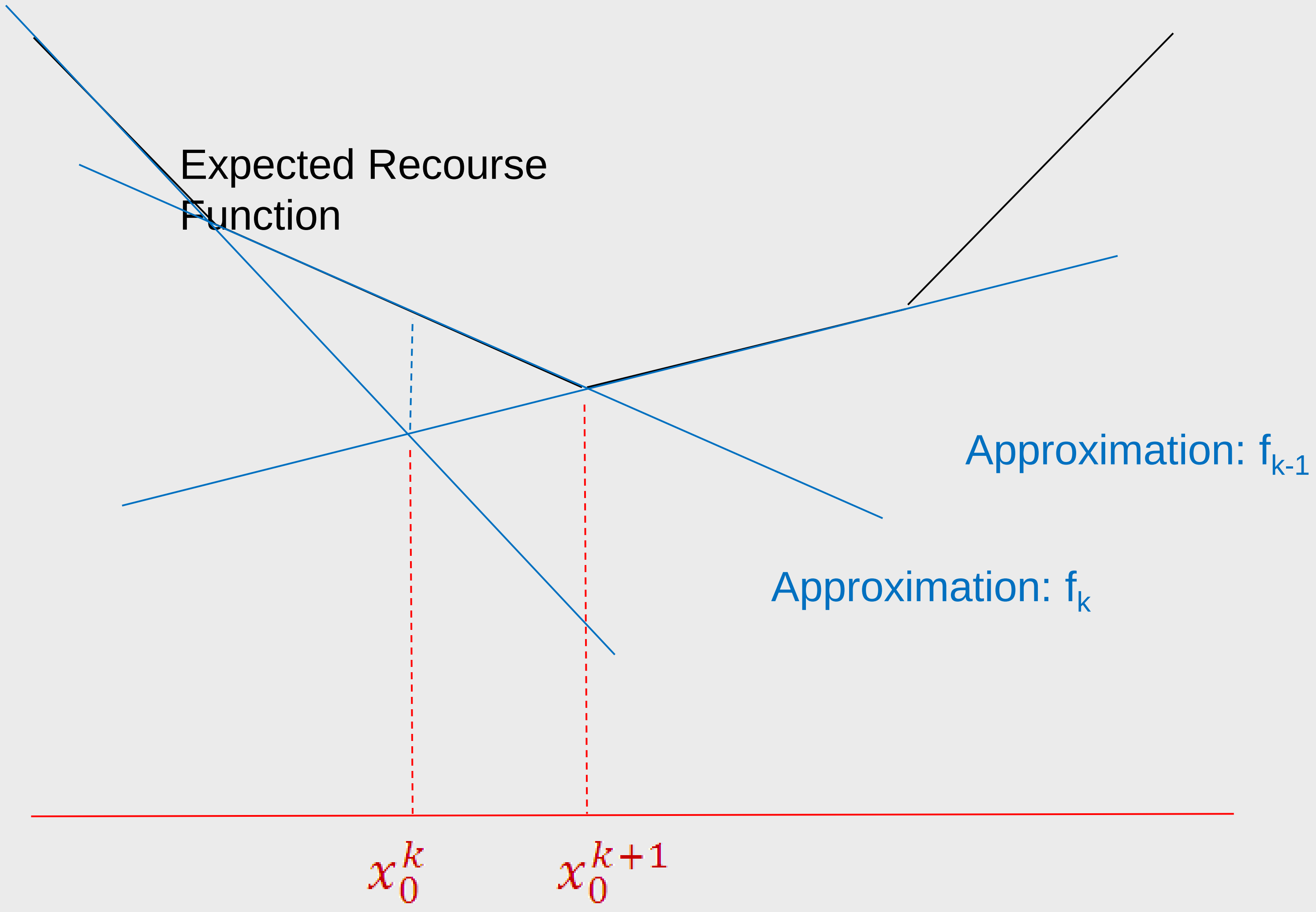
$$\alpha^k = E[\pi_1(\tilde{\omega})^\top r_1(\tilde{\omega})] \qquad \beta^k = -E[\pi_1(\tilde{\omega})^\top D_0(\tilde{\omega})]$$

Constant Term of the Supporting Hyperplane

"Normal" of the Supporting Hyperplane (same as the subgradient method)

- Let $f_k(x_0) = c_0(x_0) + \text{Max}_{1\leq t\leq k}\{\alpha^t + \beta^t x_0\}$
- Then, $x_0^{k+1} \in \text{argmin}\{f_k(x_0) : x_0 \in X_0\}$

KBL Graphical Illustration



Comparing Subgradient Method and KBL Decomposition

- Both evaluate subgradients

$$\beta^k = -E[\pi_1(\tilde{\omega})^\top D_0(\tilde{\omega})] \in \partial E\left[h\left(x_0^k;\, \tilde{\omega}\right)\right]$$

- Expensive Operation (requires solving as many second-stage LPs as there are scenarios)
- Step size in KBL is implicit (user need not worry)
- Master program grows without bound and looks unstable in the early rounds
- Stopping rule is automatic (Upper Bound – Lower Bound ≤ ε)
- KBL's use of master can be a bottleneck for parallelization

Regularization of the Master Problem (Ruszczynski/Kiwiel/Lemarechal …)

- Addresses the following issue:
  - Master program grows without bound and looks unstable in the early rounds
- Include an incumbent $\hat{x}_0^k$ and a proximity measure from the incumbent, using σ >0 as a weight: $\frac{\sigma}{2}\left\|x_0 - \hat{x}_0^k\right\|^2$
- Particularly useful in case of Stochastic Decomposition.

Where do we stand at this point in the Lecture?

Feature\Method	Subgradient Method	SQG Algorithm	KBL Decomposition
Subgradient or Estimation	Accurate	Estimation	Accurate
Step Length Choice Required	Yes	Yes	No
Stopping Rules	Unknown	Unknown	Known
Parallel Computations	Good	Good	Not so good-Good
Continuous Random Variables	No	Yes	No
First-stage Integer Variables	No	No	Yes
Second-stage Integer Variables	No	No	No

Of course for small instances, we can always try deterministic equivalents!

Stochastic Decomposition (Sequential Sampling) Higle/Sen

- Allow arbitrarily many outcomes (scenarios) including continuous random variables
- Requirement: can provide a simulator
- Assume: cost coefficients are deterministic (although random costs will be allowed soon)

$$\text{Min} \begin{Bmatrix} c_0(x_0) + E[h(x_0, \tilde{\omega})]: \\ x_0 \in X_0 \end{Bmatrix}$$

$$h(x_0;\omega) = \min c_1^\top x_1 \qquad h(x_0;\omega) = \max\ [r_1(\omega) - D_0(\omega)x_0]^\top \pi_1$$

$$\text{s.t.}\quad C_1 x_1 = r_1(\omega) - D_0(\omega)x_0, \qquad \text{s.t.}\quad \Pi_1 = \{\pi_1 | C_1^\top \pi_1 \le c_1\}.$$

$$x_1 \ge 0$$

Central Question: Scalability of each iteration

- If the number of scenarios is large, can we afford to solve **all** second-stage LPs to obtain accurate subgradient estimates?
  - No!
- Put another way: What is the smallest number of LPs we can solve in each iteration, and yet guarantee asymptotic convergence?
  - The SD answer: 1!
- How?

Approximating the recourse function in SD

- At the start of iteration *k*, sample one more outcome … say ω^k independently of $\{\omega^t\}_{t=1}^{k-1}$
- Given x_0^k solve the following LP

$$\pi_{1k}^{k} \in \operatorname{argmax} \left[r_1(\omega^k) - D_0(\omega^k)x_0^k\right]^\top \pi_1$$

$$\text{s.t. } \pi_1 \in V_1 = \text{vertices}\,(\Pi_1)$$

- Define $V_1^k \leftarrow V_1^{k-1} \cup \pi_k^k$ and calculate for $\{\omega^t\}_{t=1}^{k-1}$

$$\pi_{1t}^{k} \in \operatorname{argmax} \left\{\left[r_1(\omega^t) - D_0(\omega^t)x_0^k\right]^\top \pi_1 : \pi_1 \in V_1^k\right\}$$

- Notice the mapping of outcomes $\{\omega^t\}_{t=1}^{k}$ to finitely many dual vertices.

Forming the approximation of the Expected Recourse Function.

- The estimated "cut" in SD is given by

$$\hat{\alpha}^k = \frac{1}{k}\sum_{t=1}^{k}\left(\pi_{1t}^k\right)^{\top} r_1(\omega^t), \qquad \hat{\beta}^k = -\frac{1}{k}\sum_{t=1}^{k}\left(\pi_{1t}^k\right)^{\top} D_0(\omega^t)$$

- To calculate this "cut" requires one LP corresponding to the most recent outcome ω^k and the "argmax" operations at the bottom of the previous slide
- In addition all previous cuts need to be updated … to make old cuts consistent with the changing sample size over iterations.

Many more SD details (are skipped) … see Higle and Sen (1999)

- Updating previously generated subgradients
- Defining incumbents
- Using regularized approximations
- Dropping subgradients (finite master)
- Stopping rules … three phases
  - Set of dual vertices stops changing
  - Incumbent objective stabilizes
  - Bootstrapped estimate of distribution of duality gap is acceptably small

Further comparisons including 2-stage SD

Feature\Method	SQG Algorithm	KBL Decomposition	Stochastic Decomposition
Subgradient or Estimation	Estimation	Accurate	Estimation
Step Length Choice Required	Yes	No Needed	Not Needed
Stopping Rules	Unknown	Well Studied	Partially Solved
Parallel Computations	Good	Not so good-Good	Not known
Continuous Random Variables	Yes	No	Yes
First-stage Integer Variables	No	Yes	Yes
Second-stage Integer Variables	No	No	Available in a Dissertation

Of course for small instances, we can always try deterministic equivalents!

Comparisons between SD and SAA (Higle/Zhao accepted for publication)

Prob	Approx.	Value	Seconds
20Term (40 rvs)	Reg. SD	254,581 (79)	259.30 (31.85)
20Term	SAA	254,512 (55)	approx. 10,000
Fleet20_3 (200 rvs)	Reg. SD	141,749 (18)	293.57 (2.45)
Fleet20_3	SAA	141,654 (6.5)	approx. 12,000
SSN (80 rvs)	Reg. SD	10.26 (0.14)	7491.81 (3728.81)
SSN	SAA	10.57 (0.28)	approx. 100,000

Why the difference in computational times between SD and SAA?

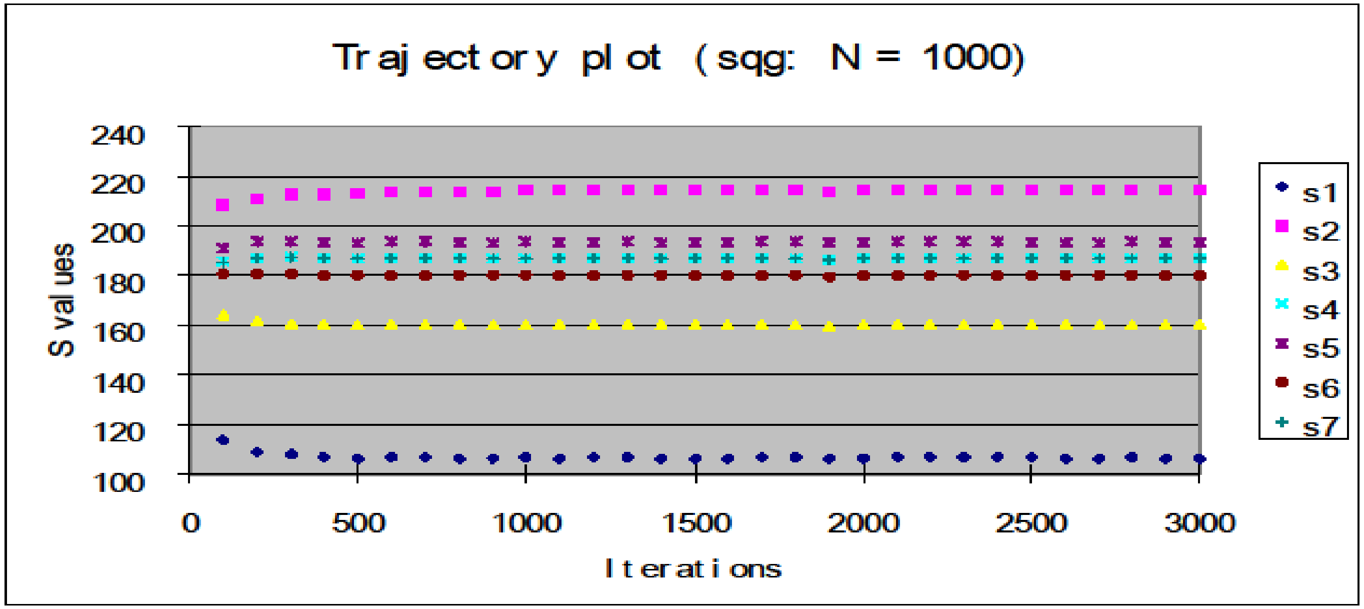
- If there are 1000 outcomes in the SAA approximations, it requires the subproblem LP to be solved for 1000 outcomes in every iteration.
- Unlike SAA, the subproblem in SD is solved for only one outcome, while approximations are used for other outcomes (from previous iterations). This explains the difference in computational times.

Comparisons between SQG and SD

- Inventory coordination instance (Herer et al 2006)
- SQG method to solve a small inventory transshipment problem
  - Find order quantities to minimize total cost of inventory and transshipment
  - Example has 7 outlets which can ship goods among themselves, if necessary
  - Demand is normally distributed
  - Herer et al ran the SQG method for K=3000 iterations, using subgradient estimates with N=1000 simulated outcomes in each iteration

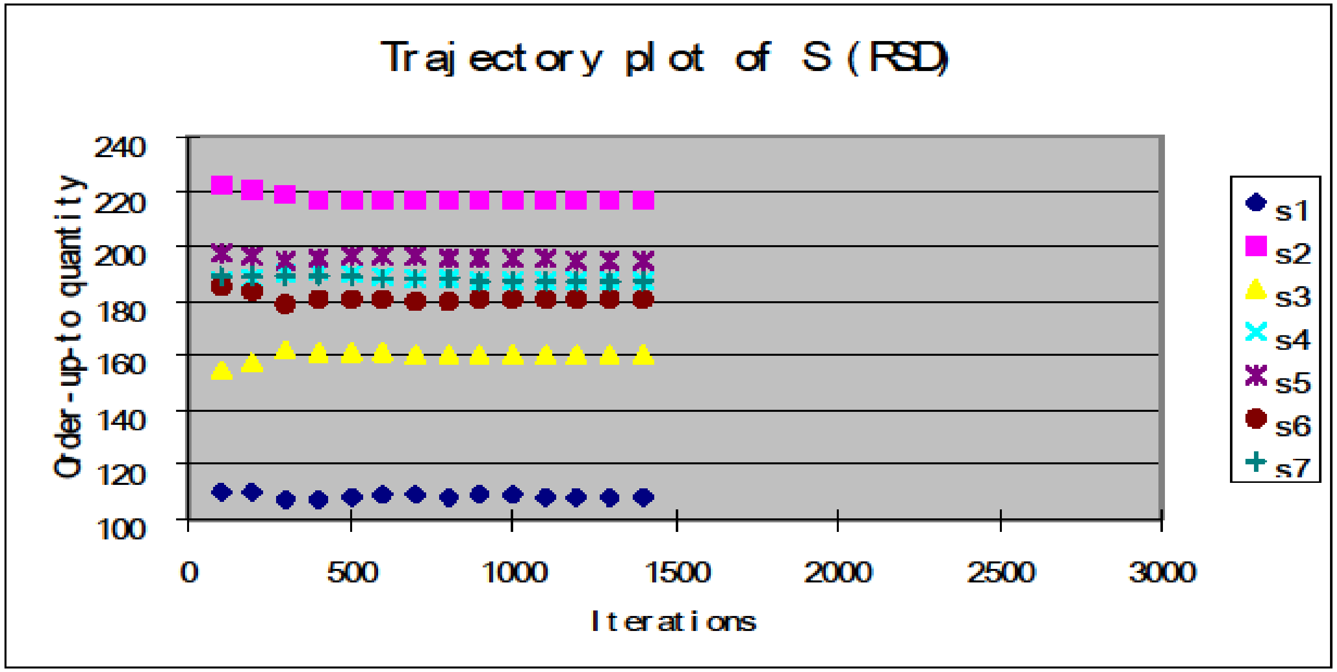
SQG trajectory of order quantities

- Here is the trajectory for $N = 1000$



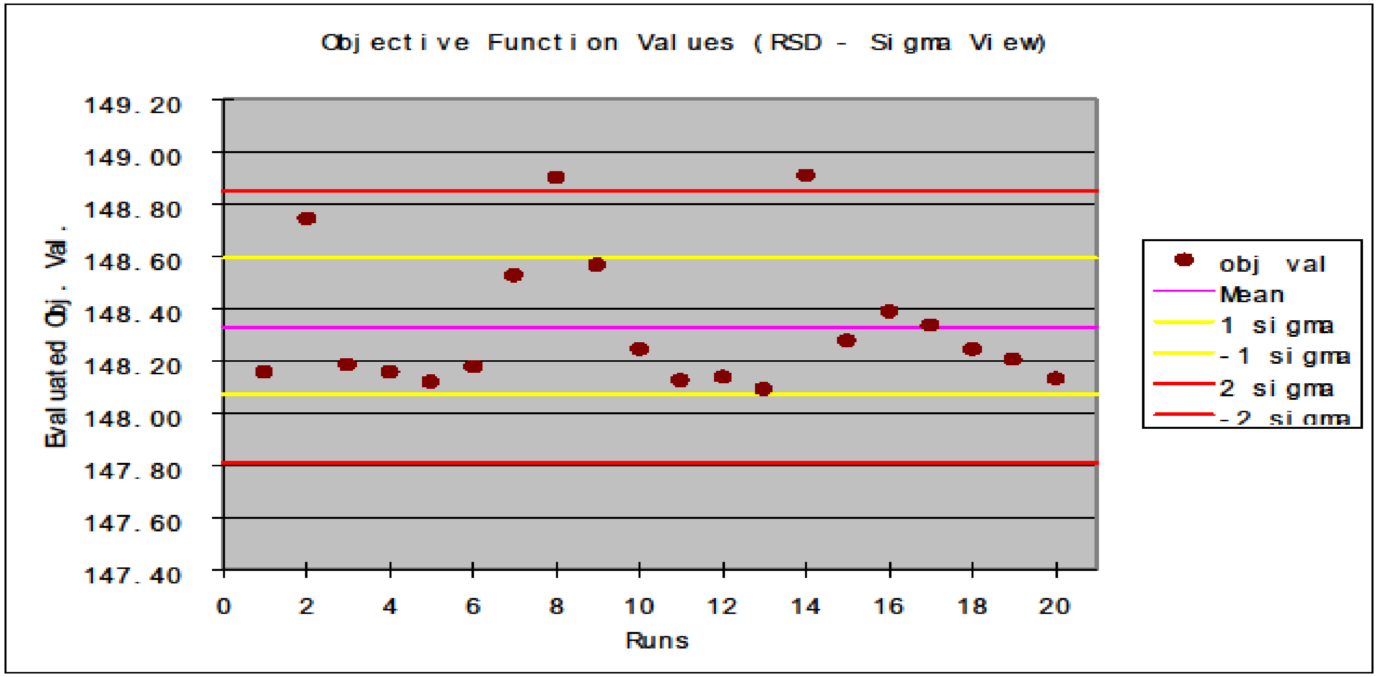
SD Trajectory of Order Quantities

- Here is the trajectory for Regularized Stochastic Decomposition (RSD)



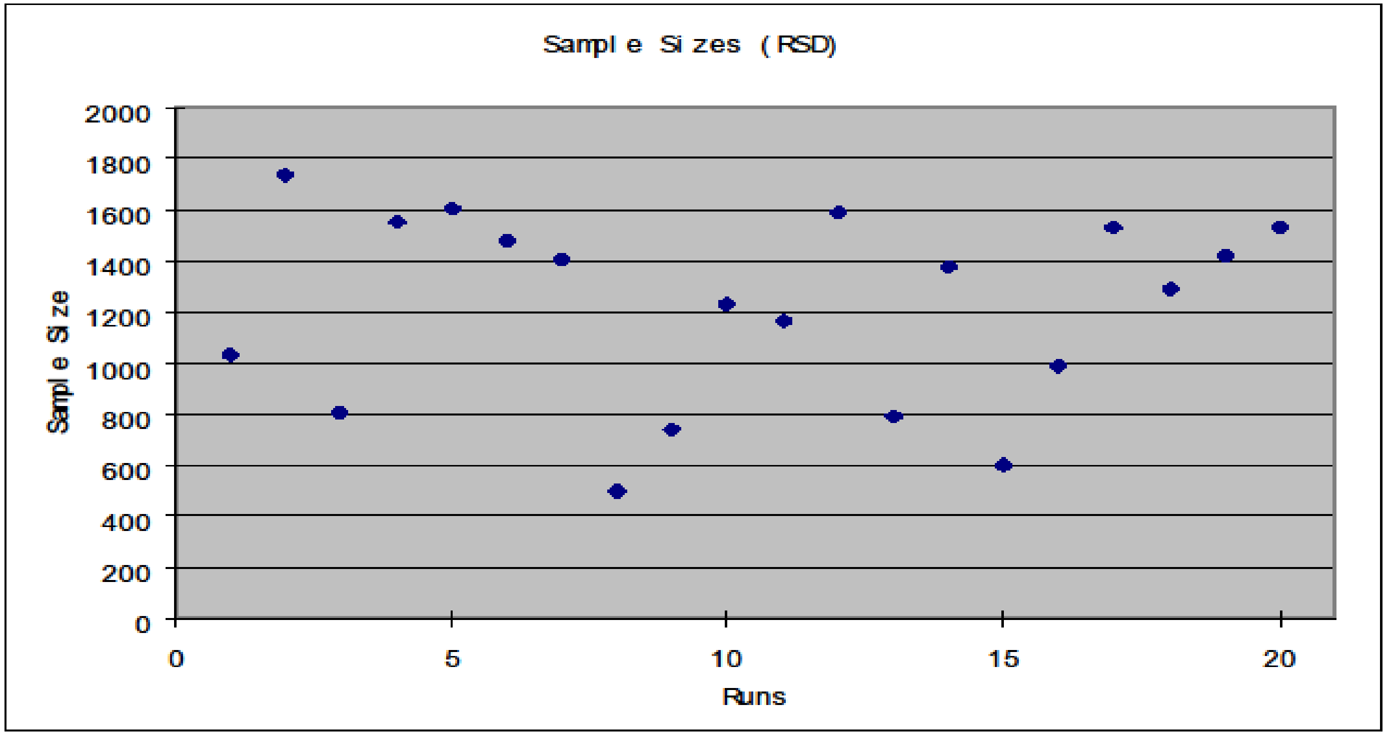
Optimal Values from 20 SD Runs

- Replications are a must for both SD and IPA/SQG



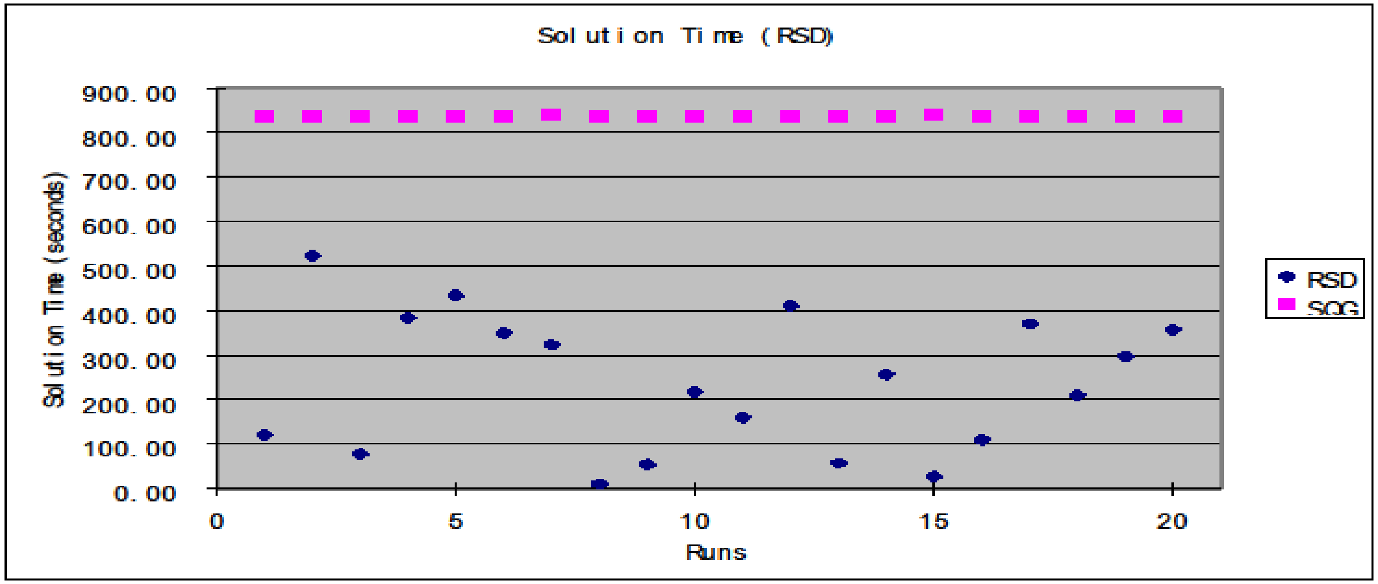
Iterations may vary depending on the seed

- RSD Sample Sizes adapt to the seed



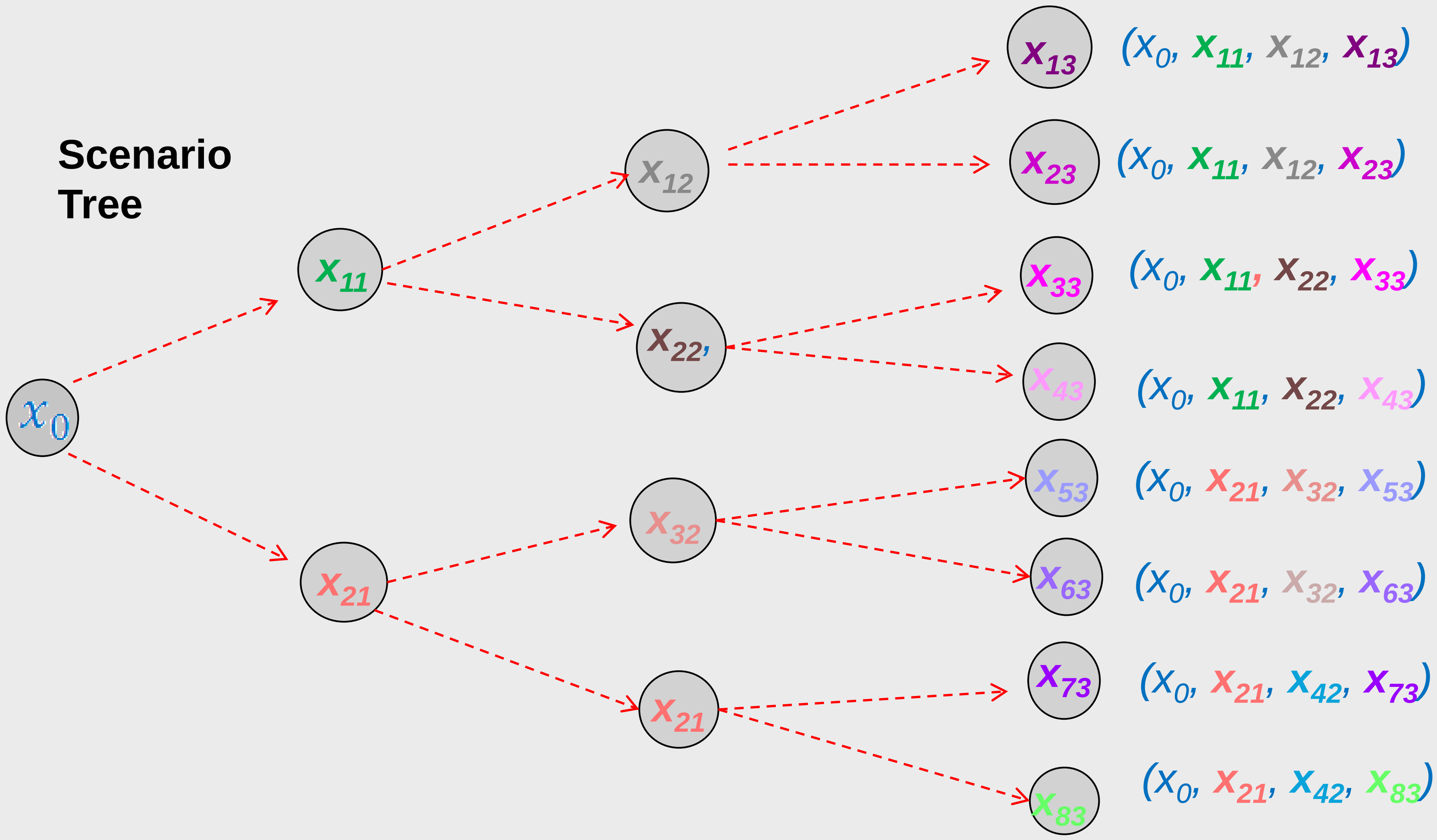
Running times for SD and SQG

- Running time of RSD is much lower than IPA/SQG



Multi-stage Stochastic Programming Algorithms

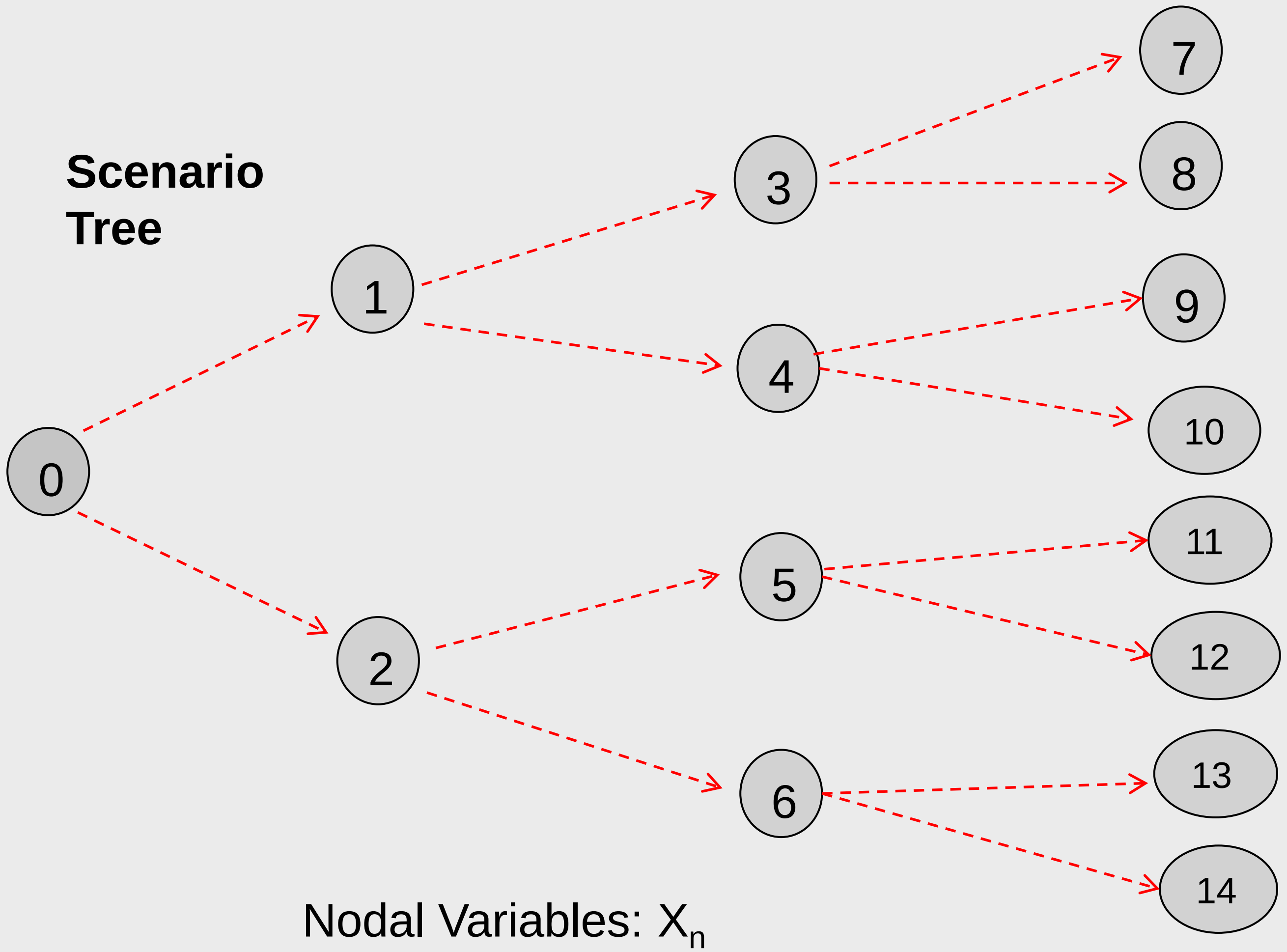
Non-anticipative Solutions by Scenario (Rockafellar/Wets)



Scenario Tree
x_0
x_{11}
x_{21}
x_{12}
x_{22},
x_{32}
x_{21}
x_{13}
x_{23}
x_{33}
x_{43}
x_{53}
x_{63}
x_{73}
x_{83}
$(x_0, x_{11}, x_{12}, x_{13})$
$(x_0, x_{11}, x_{12}, x_{23})$
$(x_0, x_{11}, x_{22}, x_{33})$
$(x_0, x_{11}, x_{22}, x_{43})$
$(x_0, x_{21}, x_{32}, x_{53})$
$(x_0, x_{21}, x_{32}, x_{63})$
$(x_0, x_{21}, x_{42}, x_{73})$
$(x_0, x_{21}, x_{42}, x_{83})$

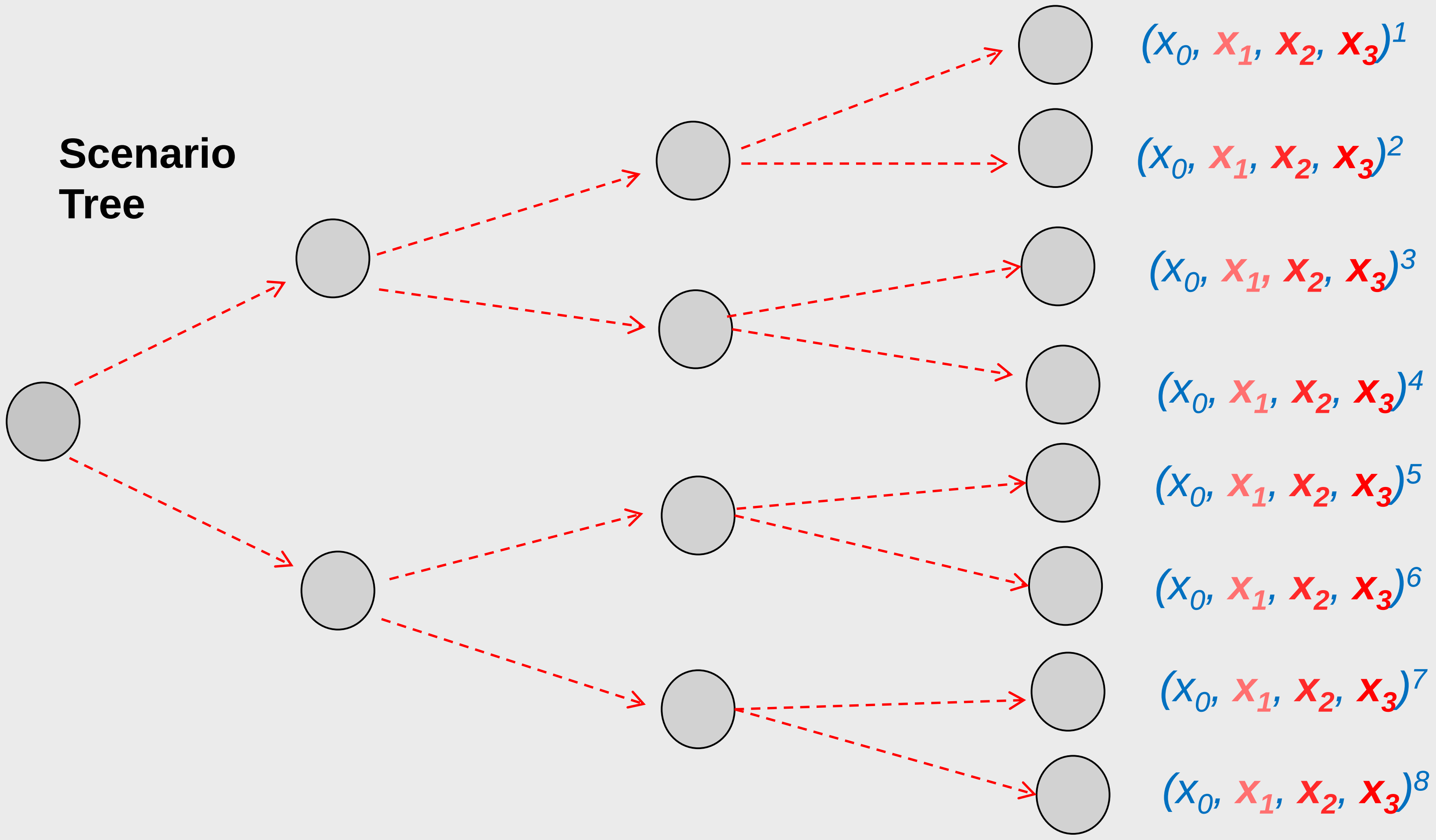
Non-anticipative Solutions by Node

Scenario Tree



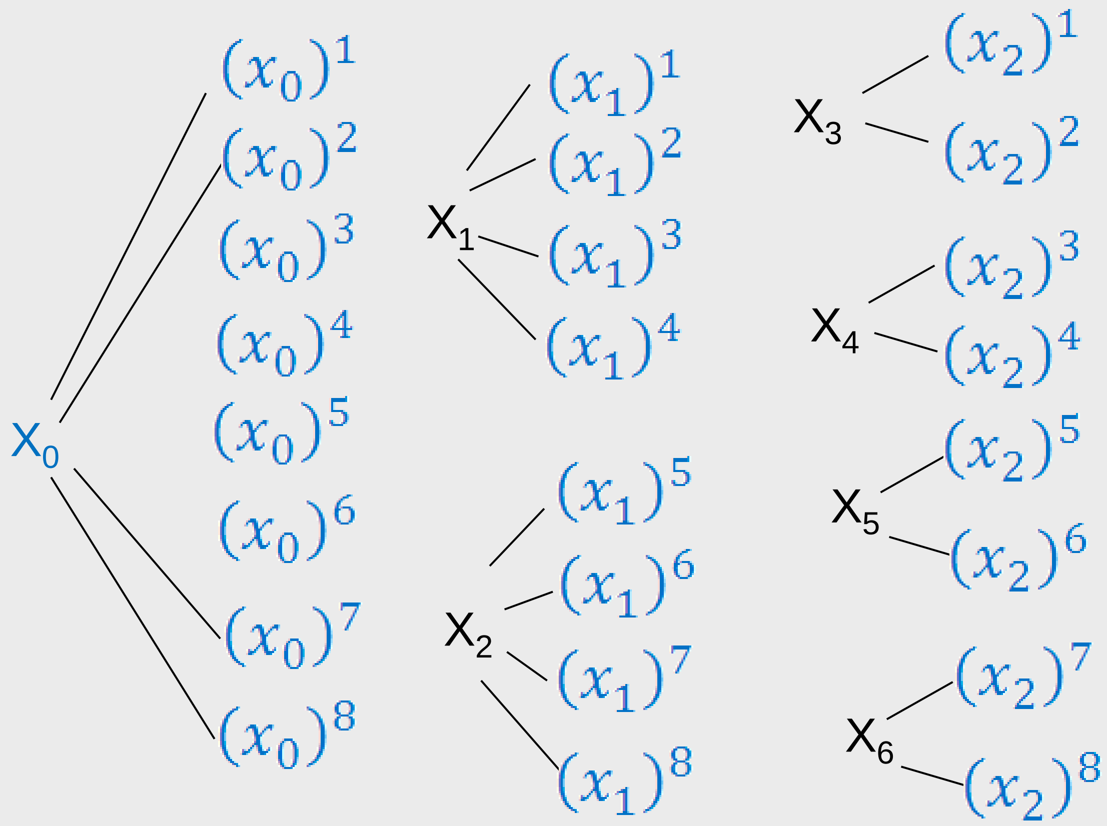
Nodal Variables: X_n

Relaxing Nonanticipativity Creates Clairvoyant Decisions



Progressive Hedging Algorithm: Coordination of Clairvoyant Decisions

- The constraints may be considered a graph



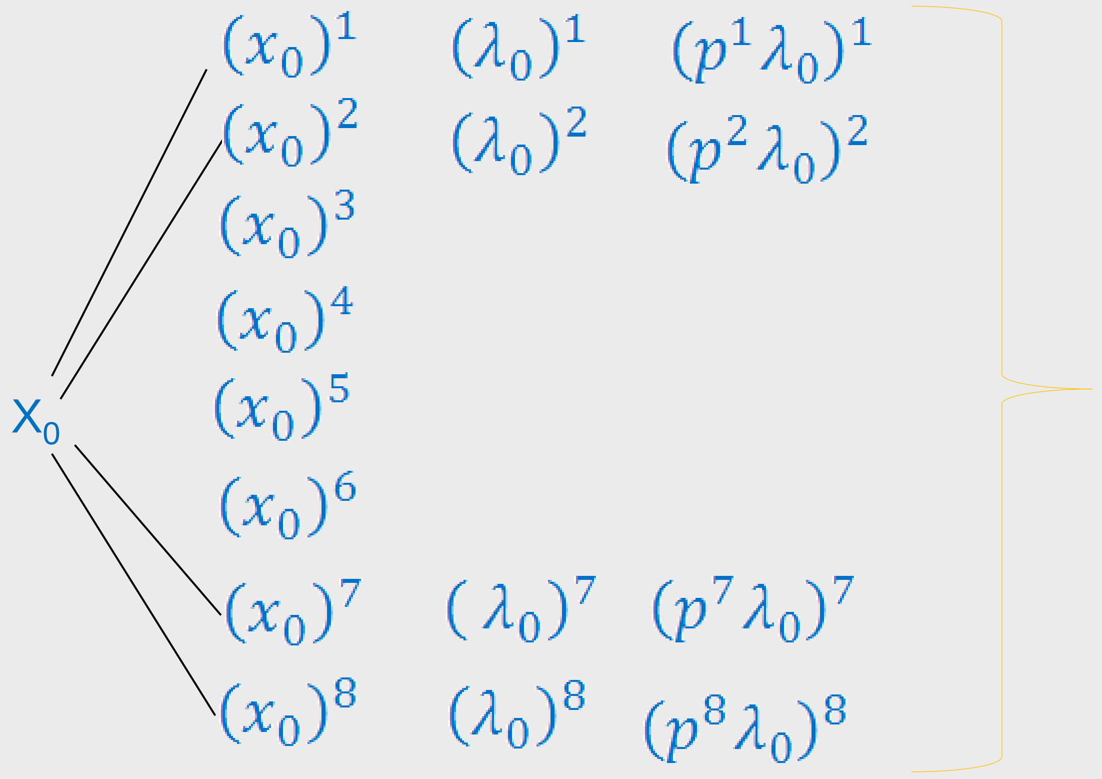
No Non-anticipativity Constraints for the Terminal Stage

$$X_n - (x_t)^{\omega} = 0$$

Dualizing for Progressive Hedging

Primal Constraints

$X_n - (x_t)^{\omega} = 0 \quad X_n$ Free



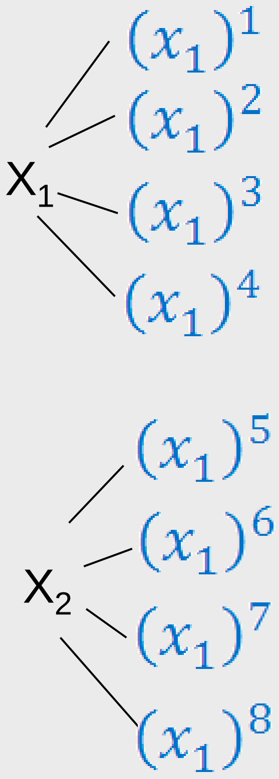
X_0
$(x_0)^1 \quad (\lambda_0)^1 \quad (p^1\lambda_0)^1$
$(x_0)^2 \quad (\lambda_0)^2 \quad (p^2\lambda_0)^2$
$(x_0)^3$
$(x_0)^4$
$(x_0)^5$
$(x_0)^6$
$(x_0)^7 \quad (\lambda_0)^7 \quad (p^7\lambda_0)^7$
$(x_0)^8 \quad (\lambda_0)^8 \quad (p^8\lambda_0)^8$

Dual Constraints

$E[\lambda_{t(n)}(\tilde{\omega})|n] = 0$

Time of node *n*

Dualizing for Progressive Hedging



Primal Problem

$$Min_{\{x^\omega \in \mathbb{X}^\omega\}\omega\in\Omega} \sum_\omega p^\omega f_\omega(x^\omega)$$

$$x^\omega - N(\omega)X = 0 \quad \forall\omega \in \Omega, \mathrm{a.s}$$

$$E[\lambda_{t(n)}(\tilde{\omega})|n] = 0$$

Time of node *n*

Lagrangian Dual Problem

$$Max_{E[\lambda_{t(n)}(\tilde{\omega})|n]=0,\forall n}\; Min_{\{x^\omega \in \mathbb{X}^\omega\}\omega\in\Omega} \sum_{\omega\in\Omega} p^\omega \left[f_\omega(x^\omega) - \lambda(\omega)x^\omega + \lambda(\omega)N(\omega)X\right]$$

Regularized of the Lagrangian Dual

Regularized Lagrangian Dual Problem

$$Max_{E[\lambda_{t(n)}(\tilde{\omega})|n]=0,\forall n}\ Min_{\{x^{\omega}\in\mathbb{X}^{\omega}\}_{\omega\in\Omega}} \sum_{\omega\in\Omega} p^{\omega}\left[f_{\omega}(x^{\omega}) - \lambda(\omega)x^{\omega} + \lambda(\omega)N(\omega)X\right]$$
$$+p^{\omega}\left[\frac{1}{2}\|x^{\omega} - N(\omega)\bar{X}\|^{2} - \frac{1}{2}\left\|\lambda(\omega) - \bar{\lambda}(\omega)\right\|^{2}\right].$$

Where $\bar{X}$ and $\bar{\lambda}$ are given at the start of any iteration.
Also assume that $\bar{\lambda}$ satisfies dual feasibility

The Progressive Hedging Strategy is Really Simple: Fix Two of the Three Categories of Variables, and Optimize the Third in the following order: Primal x^{ω} , followed by the Primal X (the conditional mean) and finally solve for $\lambda(\omega)$. Now Repeat this Procedure until the change in estimated solution is within acceptable range.

Summary of the PHA Process

- Let $x_{k+1}^{\omega} \in \operatorname{argmin} \{ f_{\omega}(x^{\omega}) - \bar{\lambda}(\omega) x^{\omega} + \frac{1}{2} \|x^{\omega} - N(\omega)\bar{X}\|^2 : x^{\omega} \in \mathbb{X}^{\omega} \}.$
- Note that this minimization only involves data for the outcome ω
- Next "minimizing" with respect to X, gives a new estimate $\bar{X}$ for the conditional expectations. This is simply the conditional expectation of the new vectors x_{k+1}^{ω}
- Finally, update the dual multipliers:

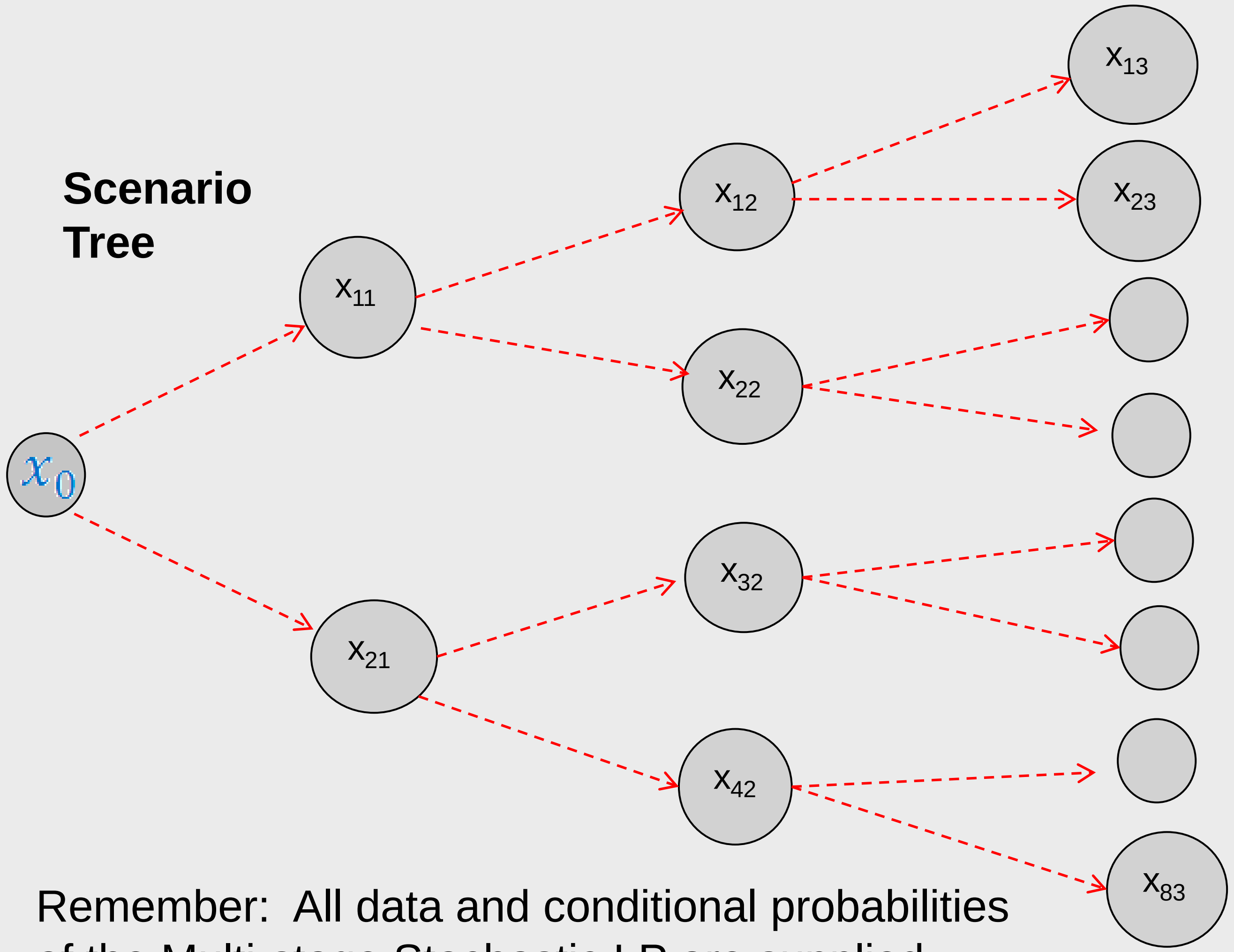
$$\bar{\lambda}(\omega) \leftarrow \bar{\lambda}(\omega) + (N(\omega)\bar{X} - x^{\omega})$$

Comments on the PHA Process

- Coordination process has no master problem – making it highly suited for parallelization
- The Lagrange Multipliers provide ex-post estimates of prices or subsides for every scenario
  - But very large search spaces because of exponentially many dual variables.
- The method has also been used as a heuristic for Stochastic Mixed-Integer Programs (see Watson, Wets and Woodruff, as well as PySP (part of Coopr at Sandia).

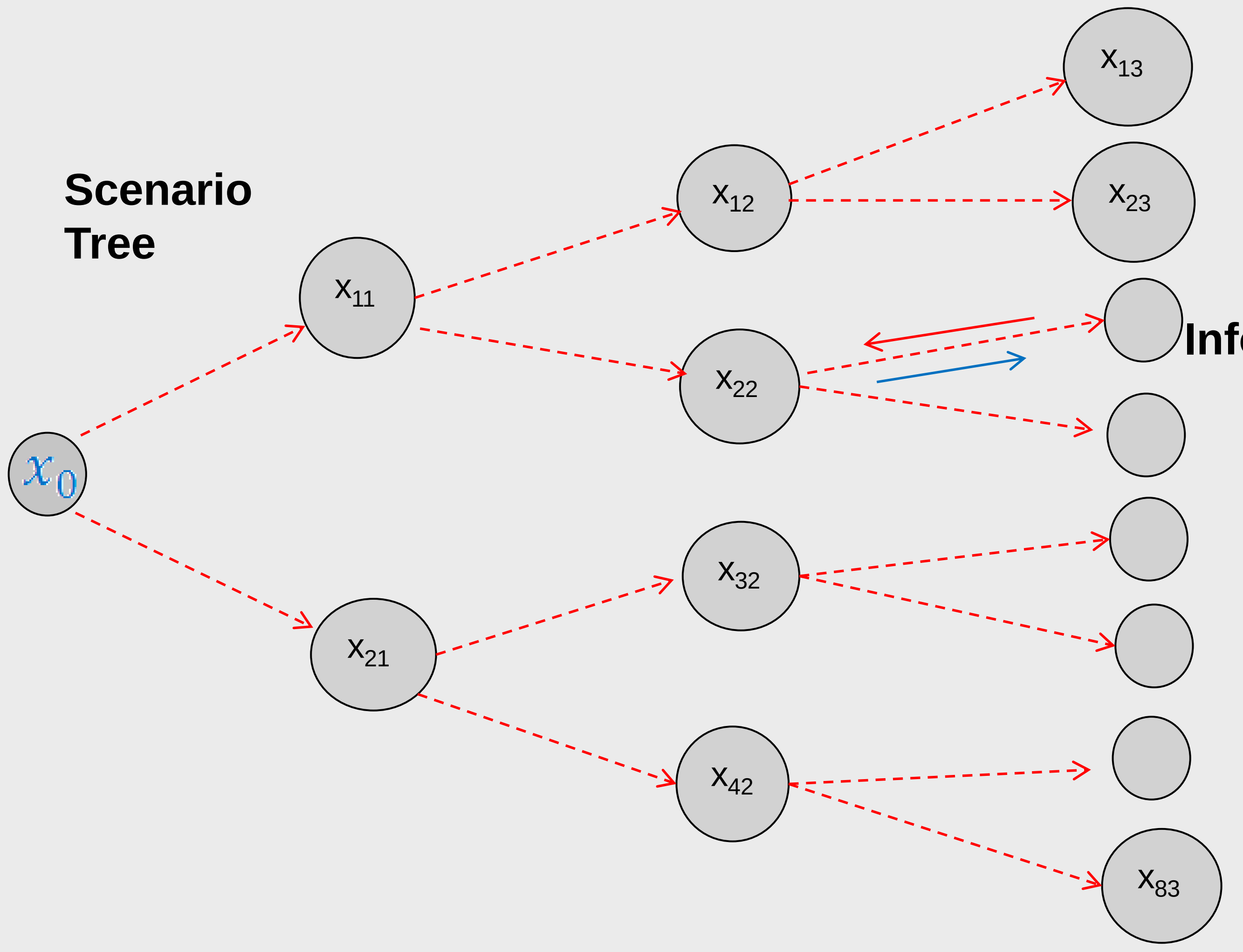
Nested Benders' Decomposition (Birge/Gassman/Dempster …)

Scenario Tree



Remember: All data and conditional probabilities of the Multi-stage Stochastic LP are supplied

Nested Benders' Decomposition (Birge/Gassman/Dempster …)



Information Visualization

- Upstream nodes place "orders" based on a local decision (e.g. x_{22})
- Downstream nodes respond with prices (i.e. subgradients) and feasibility facets

Each Node of the Tree will "House" an LP

Notation: *j is an index for a stage*

i is an index of a node in stage j

i- ("i minus") is an upstream node.

$$h_{i,j}(x_{i-,j-1}) = Min\ c_{i,j}x_{i,j} + \eta_{i,j}$$

$$s.t.\ C_{i,j}x_{i,j} = r_{i,j} - D_{i-,j-1}x_{i-,j-1}$$

Prices (i.e. subgradients) supplied by downstream nodes $\longrightarrow$

$$-E_{i,j}x_{i,j} + \eta_{i,j} \geq e_{i,j}$$

Feasibility facets supplied by downstream nodes $\longrightarrow$

$$F_{i,j}x_{i,j} \geq f_{i,j}$$

$$x_{i,j} \geq 0$$

"Orders" from upstream $\uparrow$

“Nested” Benders’ Method

- Traverse the tree solving LPs whenever feasible. In this case, pass a Subgradient to the upstream node
- If any LP is infeasible, pass a “Feasibility Facet” to the upstream node.
- Question?
  - Can this algorithm be run via asynchrounous processing, and still converge?

Comments on "Nested" Benders'

- Has been extended to sampling the tree (but you still work the same "probability"). So, asymptotic convergence does NOT rely on subgradients that are stochastic (Philpott and Guan 2008)
- Extensions to Stochastic Subgradients are right around the corner (under revision)