PURDUE UNIVERSITY

Scenario Generation for Stochastic Programming

PSE Seminar 01/19/24

Presenter: Can Li

Deterministic Optimization Model

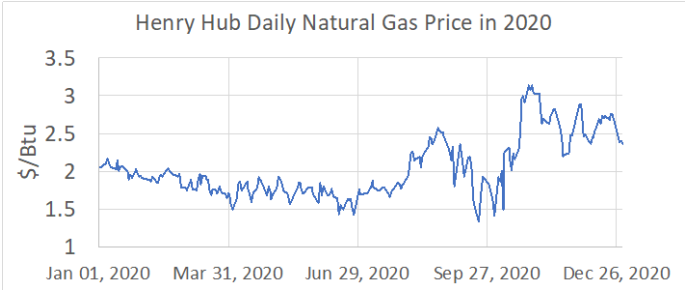
- Model **decision-making** process as an **optimization** problem

$$\begin{aligned} \min \quad & f(x, y, \theta) \\ s.t. \quad & g(x, y, \theta) \leq 0 \\ & h(x, y, \theta) = 0 \\ & x \in X, y \in Y \end{aligned}$$

- **Variables** x, y: the capacity of a process (x), whether to install a process or not (y)
- **Constraints** g, h: mass balance, to satisfy the customer demand
- **Objective** f: minimize total cost
- **Parameters** θ: Product demand, unit cost, thermal and kinetic properties
- The input parameters θ can be uncertain.

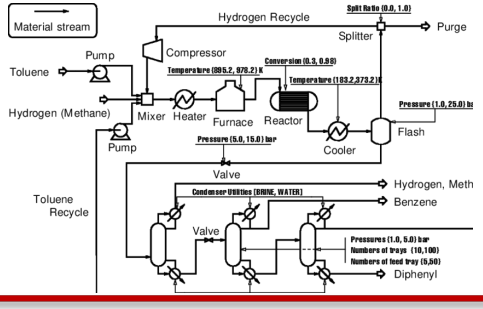
Sources of Parameter Uncertainty

- Long-term forecasts, e.g., natural gas price



- Short-term changing conditions, e.g., extreme weather

- Real-time inaccurate measurement, e.g., temperature, pressure



How Do We Model Uncertainty in Optimization Problems?

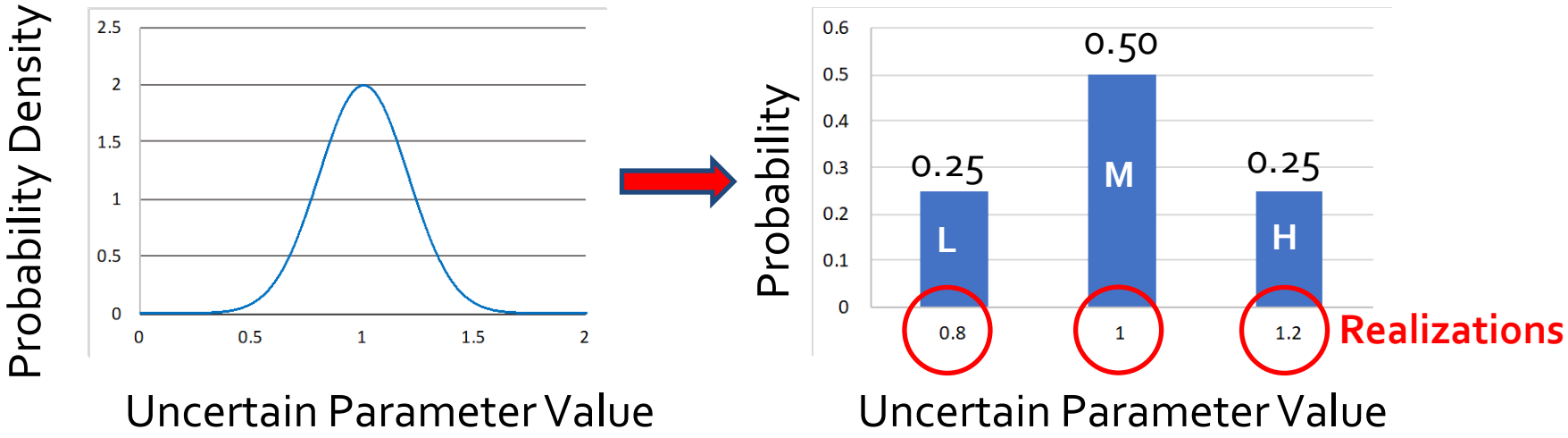
- Not a uniquely-defined problem
  - **Multiple** ways to hedge against uncertainty

The jungle of stochastic optimization

(credit: Warren Powell)

Stochastic Programming

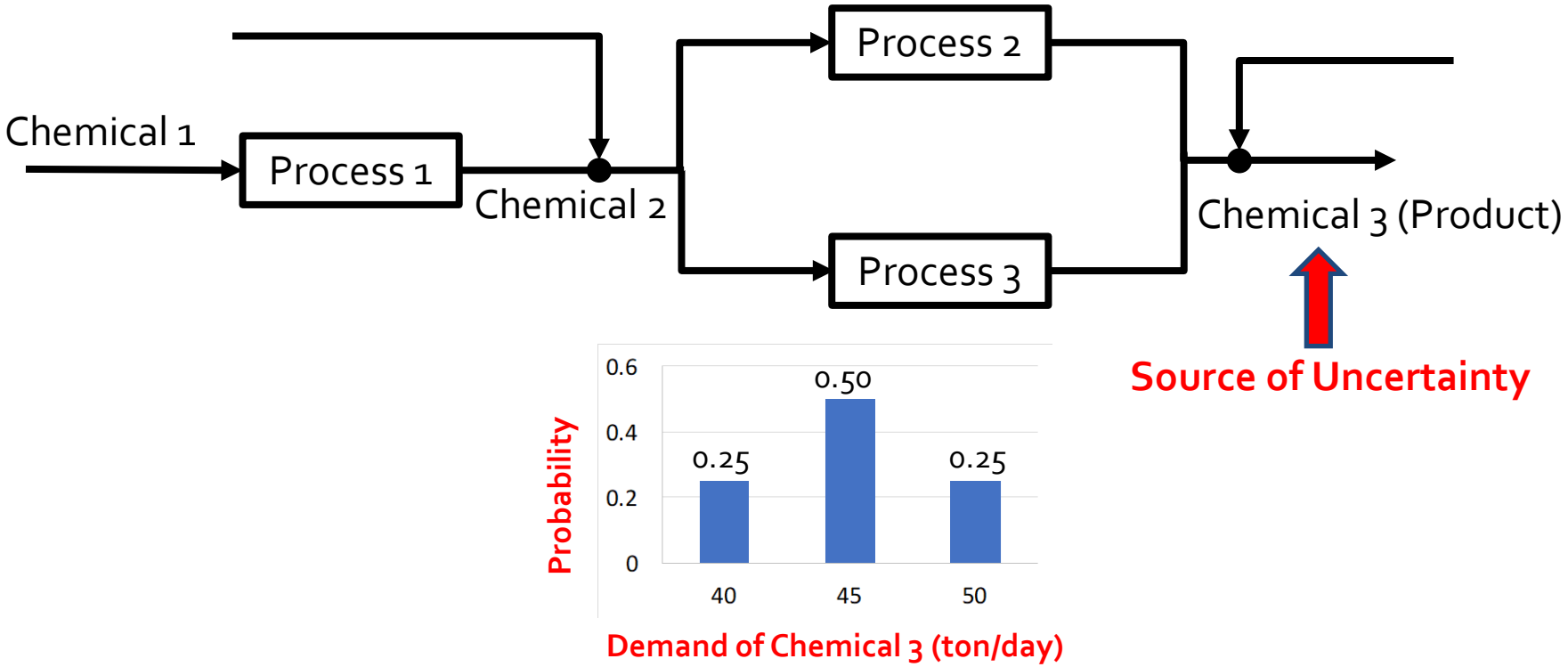
- **Stochastic programming** is a framework for modeling optimization problems that involve uncertainty (Birge and Louveaux, 2011).
- Uncertainty can be characterized by **probability distributions** known *a priori*
  - Continuous distributions ⇒ Discrete distributions



- Each realization of uncertainty parameters is called **a scenario**
- Optimize the **expected value** of the objective over **all possible scenarios**

A Motivating Example

- **Superstructure**



- **First stage decisions:** which process to install, the capacity of each process
- **Second stage decisions:** the mass flow rate of each stream
- **Constraints:** satisfy customer demands, mass balance

Two Stage Stochastic Programming

- First stage decisions: **Here and now**
- Second stage decisions: **Wait and see, Recourse decisions**

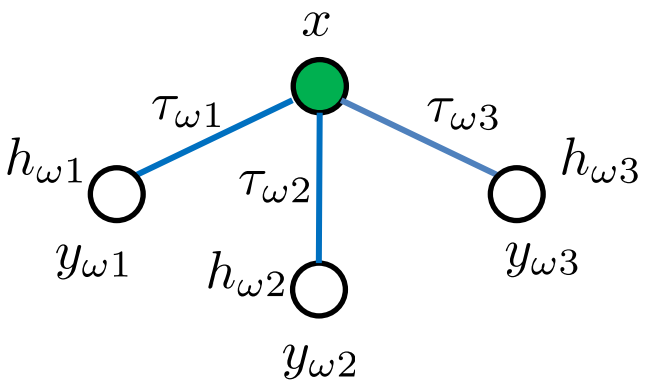
$$\min \quad c^T x + \sum_{\omega \in \Omega} \tau_\omega d_\omega^T y_\omega$$

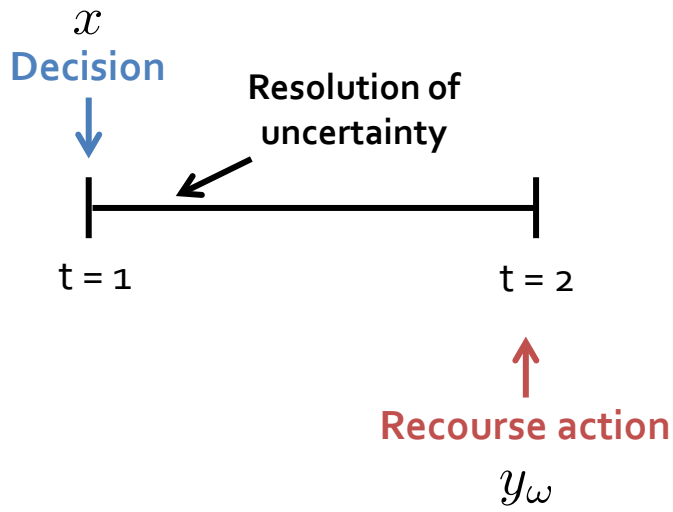
$$s.t. \quad Ax \leq b$$

$$W_\omega y_\omega \leq h_\omega - T_\omega x \qquad \forall \omega \in \Omega$$

$$x \in \{x = (x1, x2) : x_1 \in \{0,1\}^n, x_2 \geq 0\}$$

$$y_\omega \in \{y : y \geq 0\}$$

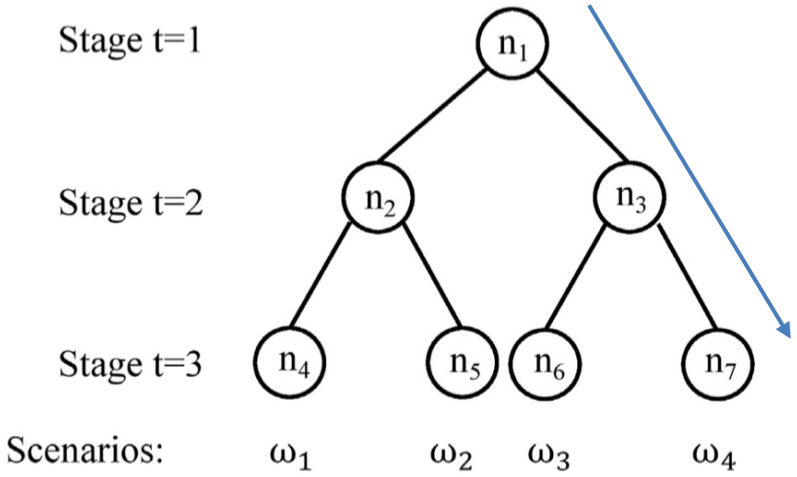




Deterministic Equivalent

Multi-stage Stochastic Program

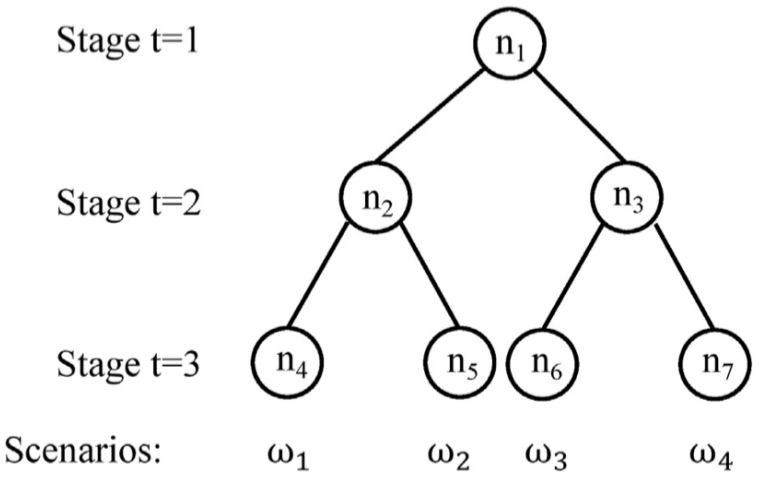
- Most practical decision problems involve a **sequence** of decisions that react to outcomes that evolve over time.
- In each stage, we have realizations of **uncertainties** at the current stage.
- Need to decide the time discretization (stage) based on the realization of uncertainty.



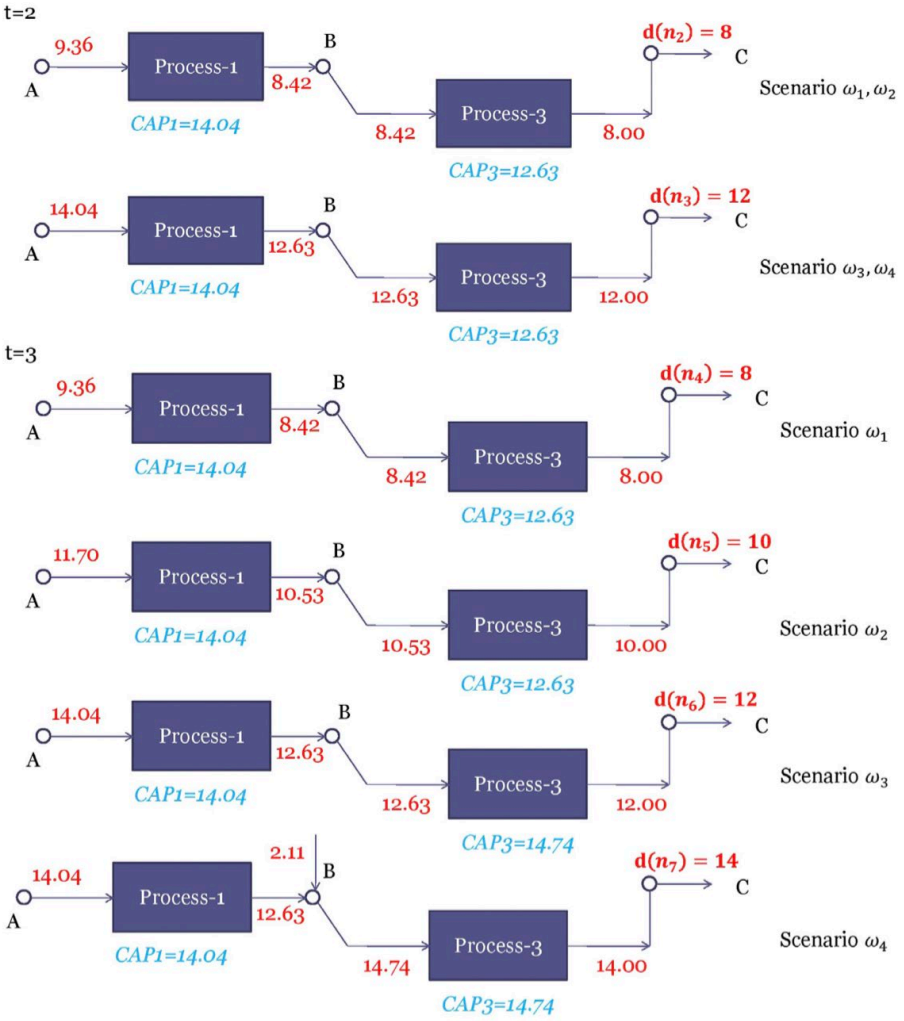
A scenario is a path from the root to on leaf

3 stages, 4 scenarios

Multi-stage Stochastic Program

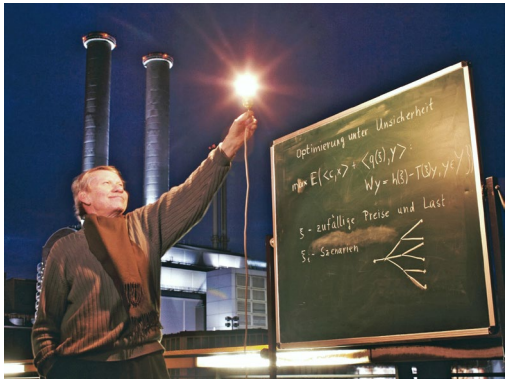


Optionally expand the capacity of the processes at stage 2



Scenario Generation

- How do we generate the scenario tree for a stochastic program?
- An area in stochastic programming community.
- Extensive literature.
- Notable researchers in this area



Werner Römisch
Department of Mathematics
Humboldt University-Berlin

Georg Pflug
Professor Statistics and OR
University of Vienna

Stein W. Wallace
Professor of Operational Research
NHH Norwegian School of Economics

Main Caveat

- This area is dominated by mathematicians, e.g.,

THEOREM 3.1. *The class $\mathcal{F}$ is P-Donsker, i.e., in $\ell^\infty(\mathcal{F})$ we have the weak convergence*

$$\sqrt{n}(P_n - P) \rightsquigarrow G_P,$$

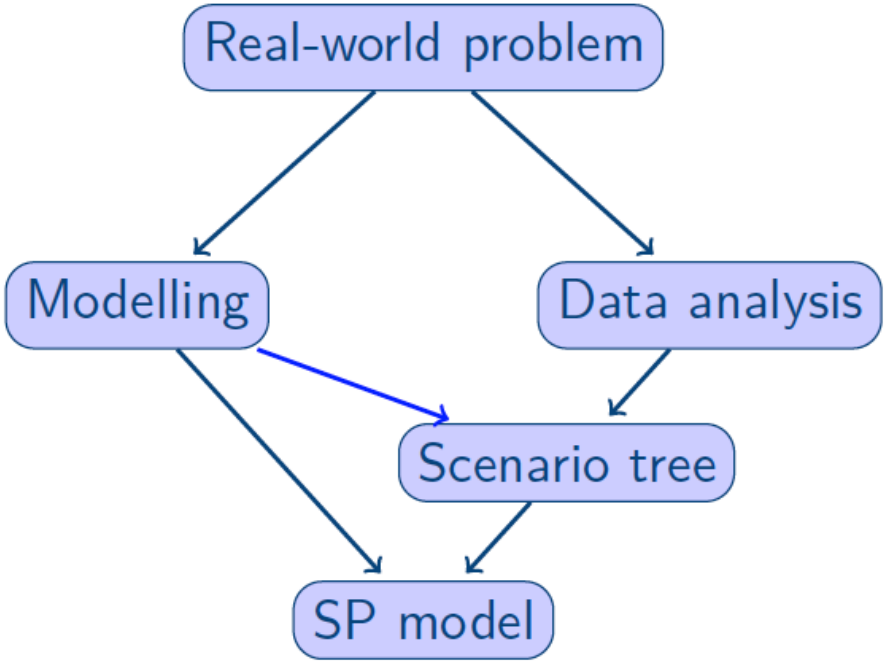
where $G_P \in \ell^\infty(\mathcal{F})^\Omega$ is a P-Brownian bridge, i.e., G_P is measurable, tight, and Gaussian:

$$G_P \sim \mathcal{N}(0, (Pfg - PfPg)_{f,g\in\mathcal{F}}).$$

Eichhorn and Römisch (Math OR 2007)

- Math is not the focus of this talk
- A tutorial of concepts and computational methods
- This talk is based on the tutorial paper by Kaut and Wallace (2004) and some recently developments on computational methods for scenario generation.

Structure of an SP problem



Note that for us, scenarios include only values of parameters (data), i.e. they do not include values of any decision variables!

What to do before scenario generation?

- Prior to scenario generation, we have to:
  - Decide the time discretization
    - Number of stages
    - Lengths of time periods
  - Know what information becomes available when, relative to the timing of decisions

    This issue does not exist in the deterministic case
  - Decide the size of the tree, i.e. the number of children for each node.

Sources of data for scenarios

- Historical data
  - ✓ Reliable past information
  - Is history a good description of the future?
- Simulation based on a mathematical/statistical/time series model, e.g., ARIMA, HMM
  - ✓ Might generalize well in the future
  - Usually need data to fit these models
- Expert opinion
  - ✓ Could be useful in some applications, e.g., forecast for oil price
  - Subjective, back-testing is not possible.
- In practice, often a combination of more than one of the above
  - Estimate the distribution from historical data, then use a mathematical model and/or an expert opinion to adjust the distribution to the current situation.

Main Criteria

- A good scenario tree should capture
  - **Distributions** of the random variables at each period
    - marginal distributions of all variables, in the very least their means and variances
    - dependence between them, typically measured by correlations
  - **Inter-temporal dependencies**
    - changes of the distributions, based on values of previous stages
    - includes things like auto-correlations, mean reversion, etc.
    - can be modelled by time-series models

Quality of scenario trees

In assessing quality of the scenario tree generation method, we consider two things

- **Stability**
  - If we generate several scenario trees, the solutions should not vary too much.
- **Error**
  - We use an approximation of the true distribution, so we are likely to find a suboptimal solution.
  - Not straightforward how to measure the error.

Some Notation

- The original problem is,

$$\min_{x \in X} F(x; \xi)$$

where x are the first stage decision variables, ξ denotes the **"true" distribution**. Note that we do not express the variables after stage 1 explicitly.

- The scenario-based problem is,

$$\min_{x \in X} F(x; \eta)$$

where η denotes the **scenario tree approximation** of the "true" distribution.

- The scenario tree generation method is usually **stochastic,** e.g, sampling-based method. In stability tests, we generate several scenario trees η_k, $k = 1, \dots, n$, leading to solutions

$$x_k^* = \arg\min_{x \in X} F(x; \eta_k)$$

Discretization Error

- Pflug (2001) defines **an approximation error** caused by η_k (also called an **optimality gap**) as:

$$e_f(\xi, \eta_k) = F\Big(\arg\min_x F(x; \eta_k); \xi\Big) - F\Big(\arg\min_x F(x; \xi); \xi\Big)$$

$$= F(x_k^*; \xi) - \min_x F(x; \xi) \geq 0$$

- To evaluate $e_f(\xi, \eta_k)$ we need to
  - Evaluate the "true" objective function $F(x; \xi)$.
    - Can sometimes be done using a "simulator"
  - Solve the original problem, i.e., $\min_x F(x; \xi)$
    - Impossible. Otherwise, we would not use the scenario tree approximation

Tests Using a Simulator

- Assume that we have a "simulator" for evaluating $F(x;\xi)$, i.e. the true performance of a solution x
- This allows us to:
  - Compare two solutions x_1^*, x_2^*
  - Compare two different scenario-generation methods.
  - Test an out-of-sample stability of a given method:
    1. Generate a set of trees $\eta_k, k = 1, \dots, n$
    2. Solve problems using the trees $\longrightarrow$ solutions x_k^*.
    3. Test whether $F(x_k^*;\xi) \approx F(x_l^*;\xi)$
       - The test is equivalent to $e_f(\xi,\eta_k) \approx e_f(\xi,\eta_l)$
       - Without stability, we have a problem!
- $e_f(\xi;\eta_k) \approx 0$, implies $e_f(\xi,\eta_k) \approx e_f(\xi,\eta_l)$ and stability

Out of Sample Tests without a Simulator

- Instead of using a simulator, we can **"cross test",** i.e. test

$$F(x_k^*; \eta_l) \text{ for } l \neq k$$

for all $k = 1, \dots, n, l = 1, \dots, n$

- It is still an out-of-sample test, as we test the solutions on different trees than were used to find them.
- If we have to choose one of the solutions x_k^*, we choose the most stable one, i.e., $F(x_k^*; \eta_l)$ for $l \neq k$ for all $k = 1, \dots, n$ has the least variance

In-sample Stability

- Instead of the true performance, we look at the optimal objective values reported by the problems themselves. Check if,

$$F(x_k^*; \eta_k) \approx F(x_l^*; \eta_l)$$

or equivalently,

$$\min_x F(x; \eta_k) \approx \min_x F(x; \eta_l)$$

  - No direct connection to out-of-sample stability
  - Without this, we cannot trust the reported performance of the scenario-based solutions.

What If We Do Not Have Stability?

- What does it mean:
  - No stability → decision depends on the choice of the tree.
- What to do:
  - Change/improve the scenario generation method.
  - Increase the number of scenarios.
  - Generate several trees, get the solutions and then "somehow" choose the best solution.

A proper mathematical treatment of stability can be found in Dupačová and Römisch (1998); Fiedler and Römisch (2005); Heitsch et al. (2006).

Upper Bound for the Error

- Recall **an approximation error** caused by η_k

$$e_f(\xi, \eta_k) = F\Big(\arg\min_x F(x; \eta_k); \xi\Big) - F\Big(\arg\min_x F(x; \xi); \xi\Big)$$

$$= F(x_k^*; \xi) - \min_x F(x; \xi) \geq 0$$

- How to obtain an upper bound for the error?

Define

$$z^* = \min_{x \in X} F(x; \xi) = F(x^*; \xi)$$

$$z_k^* = \min_{x \in X} F(x; \eta_k) = F(x_k^*; \eta_k)$$

We have

$$e_f(\xi, \eta_k) = F(x_k^*; \xi) - z^*$$

To overestimate the error, we need an **under estimator** for z^*

Upper Bound for the Error

$$z^* = \min_{x \in X} F(x; \xi) = F(x^*; \xi)$$

$$z_k^* = \min_{x \in X} F(x; \eta_k) = F(x_k^*; \eta_k)$$

Error:

$$e_f(\xi, \eta_k) = F(x_k^*; \xi) - z^*$$

Mak et al. (1999) prove that if η_k are generated using sample average approximation, i.e., generate i.i.d. samples from the "true" distribution. We have,

$$\mathbb{E}[z_k^*] \leq z^*$$ **under estimator**

If the samples in η_k are unbiased

$$\frac{1}{n} \sum_{i=1}^{n} F(x_k^*; \eta_i) \approx F(x_k^*; \xi)$$

Upper Bound for the Error

By combining under estimator for z^* and an over estimator for $F(x_k^*; \xi)$.

$$e_f(\xi, \eta_k) = F(x_k^*; \xi) - z^*$$

$$\lessapprox \frac{1}{n} \sum_{i=1}^{n} F(x_k^*; \eta_i) - z_k^*$$

- This is a **stochastic** upper bound, it can even be **negative.**
- It is possible to compute a **confidence interval** for the upper bound, based on t-distribution.
- See Mak et al. (1999) for details, including variance-reduction techniques.

Scenario Generation Methods-Standard Sampling

- Univariate random variable
  - This is a standard random number generation.
  - Methods exist for all possible distributions.
- Independent multivariate random vector
  - Generate one margin at a time, combine all against all
    - guaranteed independence
    - grows exponentially with the dimension
    - trees need often some "pruning" to be usable
- General multivariate case
  - Special methods for some distributions.
    - e.g., normal distribution via Cholesky decomposition
  - Use principal components to get "independent" (uncorrelated) variables.

Multiple Periods Sampling

- Generate one single-period subtree at a time. Start in the root, move to its children, and so on.
- Inter-temporal independence
  - Easy, as the distributions do not change.
- Distribution depends on the history.
  - Distribution of children of a node depends on the values on the path from the root to that node.
  - The dependence is modeled using stochastic processes like ARMA (autoregressive moving average)
  - New value X_t is a function of values of previous time periods and random disturbance $\epsilon_t \sim N(0, \sigma^2)$

$$X_t = f(X_{t-1}, X_{t-2}, \dots; \epsilon_{t-1}, \epsilon_{t-2}, \dots; \epsilon_t)$$

ARMA(p,q) process: $X_t = \sum_{i=1}^{p} p_i X_{t-i} + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$

Sampling Methods

- Pros
  - Easy to implement.
  - Distribution converges to the true one
  - SAA provides a lower bound for z^*
- Cons
  - Bad performance/stability for small trees.
  - Have to know the distribution to sample from.

Scenario Reduction

- The idea is to reduce size of a given scenario tree ξ into a smaller tree η, with as little impact on the solution as possible.
- It is based on the theory of stability of stochastic programs w.r.t. changes in the probability measures; see Römisch (2003)

Scenario Reduction

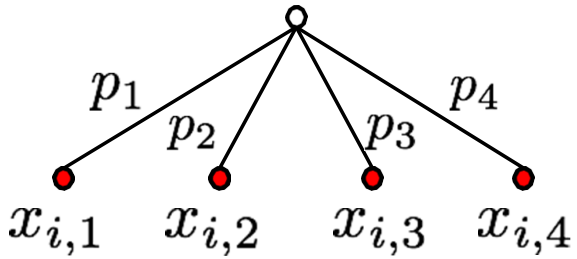
- Dupačová et al. (2003); Heitsch and Römisch (2003, 2007)
  - The goal is to reduce a tree from N to k scenarios.
  - It turns out the problem is NP-hard. The authors propose heuristics:
- backward reduction
  - find the scenario whose removal will cause the smallest error
  - remove the scenario and redistribute its probability
  - repeat until we have only k scenarios left
- forward selection
  - start with an empty tree
  - find the scenario whose addition will cause the biggest improvement
  - add the scenario and redistribute its probability
  - repeat until we have k scenarios

Property Matching Methods

- These methods construct the scenario trees in such a way that a given set of properties is matched.
- The properties are, for example, moments of the marginal distributions and covariances/correlations.
- Typically, the properties do not specify the distributions fully; the rest is left to the method.
  - Different methods produce very different results.
  - The issue is very significant for bigger trees, with many more degrees of freedom.

Degrees of Freedom in the Tree

- Simple two-stage scenario tree with structure 1-4



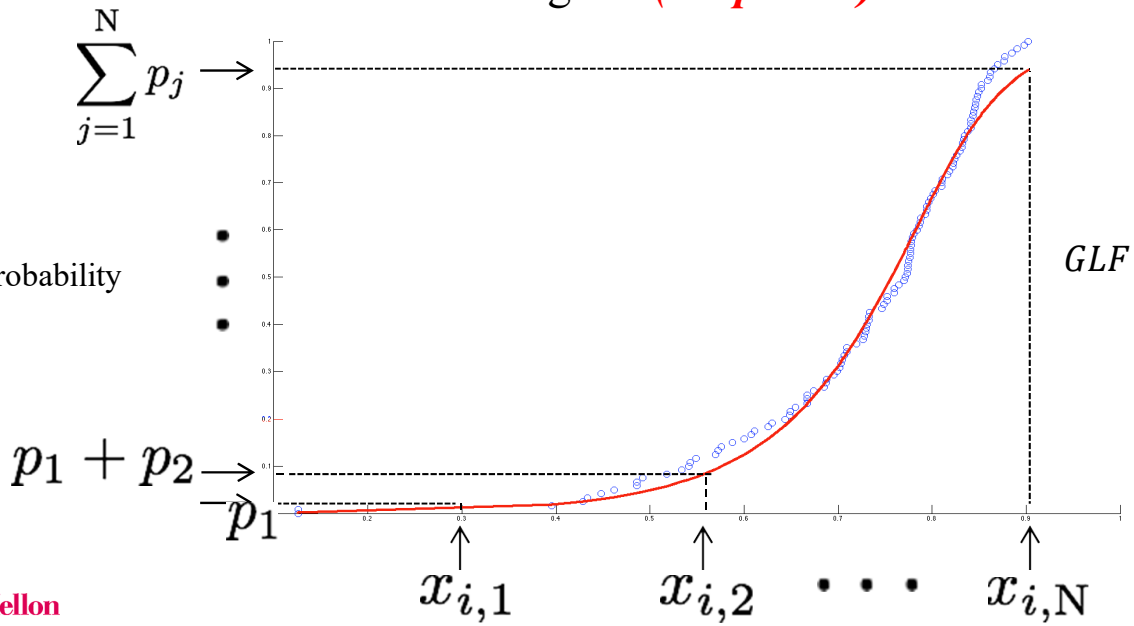
- Decision variables in an optimization formulation
  - *Probabilities of the outcomes* ($\boldsymbol{p_i}$)
  - *Values of the outcomes* ($\boldsymbol{x_i}$)
- **Moment Matching Method** (Høyland & Wallace, 2001)
  - **Determine *p* and *x*** to **match** (marginal) **moments** calculated from tree and those estimated from the data
  - Over- and under-specification issues
    - **Under-specification** is common and increases with number of outcomes for fixed number of moments

Carnegie Mellon

Slides credit to:
Bruno Calfa and Ignacio Grossmann
Calfa et al. (2014)

Mitigating Under-Specification

- Each outcome has two sets of variables: $x_{i,j}$ and p_j
- Each moment specification has one piece of information
- Consequences:
  - Multiple combinations of *x* and *p* satisfy moments
  - Probabilities may not capture the shape of the underlying distribution
- Additional information: marginal ***(Empirical) Cumulative Distribution***



$$GLF(x) = \beta_0 + \frac{\beta_1 - \beta_0}{(1 + \beta_2 e^{-\beta_3 x})^{1/\beta_4}}$$

L^2 Moment Matching Problem (L^2 MMP)

$$\min_{x,\,p} \quad z^{L^2}_{\mathrm{MMP}} = \sum_{i\in I}\sum_{k\in K} \mathrm{w}_{i,k}(m_{i,k} - \mathrm{M}_{i,k})^2 + \sum_{\substack{(i,\,i')\in I \\ i<i'}} \mathrm{w}_{i,i'}(c_{i,i'} - \mathrm{C}_{i,i'})^2$$

Min weighted error between tree and data

s.t.

$$\sum_{j=1}^{\mathrm{N}} p_j = 1$$

Probabilities add up to 1

$$m_{i,1} = \sum_{j=1}^{\mathrm{N}} x_{i,j}p_j \qquad \forall\, i \in I$$

$$m_{i,k} = \sum_{j=1}^{\mathrm{N}} (x_{i,j} - m_{i,1})^k p_j \qquad \forall\, i \in I,\ k > 1$$

Moments calculated from the tree

$$c_{i,i'} = \sum_{j=1}^{\mathrm{N}} (x_{i,j} - m_{i,1})(x_{i',j} - m_{i',1})p_j \qquad \forall\, (i,\ i') \in I,\ i < i'$$

Covariances calculated from the tree

$$x_{i,j} \in [\mathrm{x}^{\mathrm{LB}}_{i,j},\ \mathrm{x}^{\mathrm{UB}}_{i,j}] \qquad \forall\, i \in I,\ j = 1, \ldots, \mathrm{N}$$

$$p_j \in [0,\ 1] \qquad \forall\, j = 1, \ldots, \mathrm{N}$$

Bounds on variables

Nonlinear, nonconvex optimization problem.

L^1 formulation can be reformulated as an LP for fixed node values (Ji *et al.*, 2005)

L^2 Distribution Matching Problem (L^2 DMP)

$$\min_{x,\,p} \quad z^{L^2}_{\mathrm{MMP}} = \sum_{i\in I}\sum_{k\in K} \mathrm{w}_{i,k}(m_{i,k} - \mathrm{M}_{i,k})^2 + \sum_{\substack{(i,\,i')\in I \\ i<i'}} \mathrm{w}_{i,i'}(c_{i,i'} - \mathrm{C}_{i,i'})^2 + \sum_{i\in I}\sum_{j=1}^{\mathrm{N}} \omega_{i,j}\delta^2_{i,j}$$

Min weighted error between tree and data

s.t.

$$\sum_{j=1}^{\mathrm{N}} p_j = 1$$

Probabilities add up to 1

$$m_{i,1} = \sum_{j=1}^{\mathrm{N}} x_{i,j}p_j \qquad \forall\, i \in I$$

$$m_{i,k} = \sum_{j=1}^{\mathrm{N}} (x_{i,j} - m_{i,1})^k p_j \qquad \forall\, i \in I,\ k > 1$$

Moments calculated from the tree

$$c_{i,i'} = \sum_{j=1}^{\mathrm{N}} (x_{i,j} - m_{i,1})(x_{i',j} - m_{i',1})p_j \qquad \forall\, (i,\ i') \in I,\ i < i'$$

Covariances calculated from the tree

$$x_{i,j} \in [\mathrm{x}^{\mathrm{LB}}_{i,j},\ \mathrm{x}^{\mathrm{UB}}_{i,j}] \qquad \forall\, i \in I,\ j = 1, \ldots, \mathrm{N}$$

$$p_j \in [0,\ 1] \qquad \forall\, j = 1, \ldots, \mathrm{N}$$

Bounds on variables

$$\widehat{ECDF}(x_{i,j}) - \sum_{j'=1}^{j} p_{j'} = \delta_{i,j} \qquad \forall\, i \in I,\ j = 1, \ldots, \mathrm{N}$$

$$x_{i,j} \le x_{i,j+1} \qquad \forall\, i \in I,\ j = 1, \ldots, \mathrm{N} - 1$$

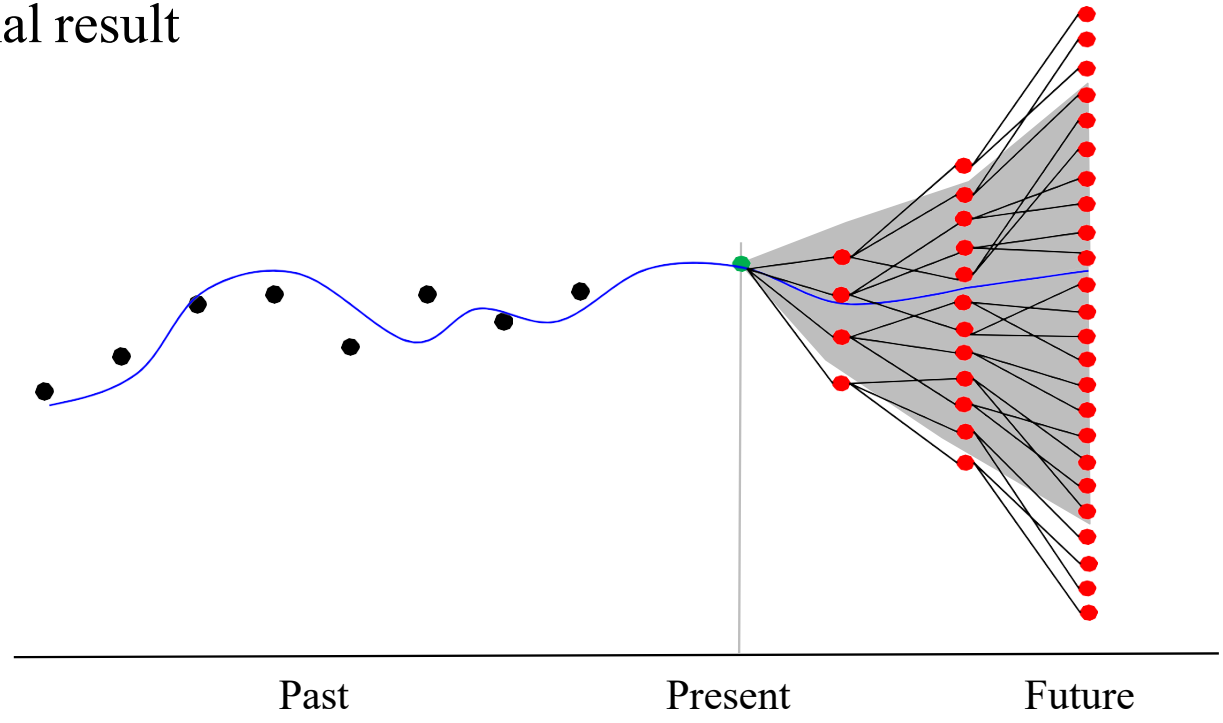
ECDF information

Nonlinear, nonconvex optimization problem.

L^1 formulation can be reformulated as an LP for fixed node values (Ji *et al.*, 2005)

Scenario Tree Generation and Forecasting Multistage Problems

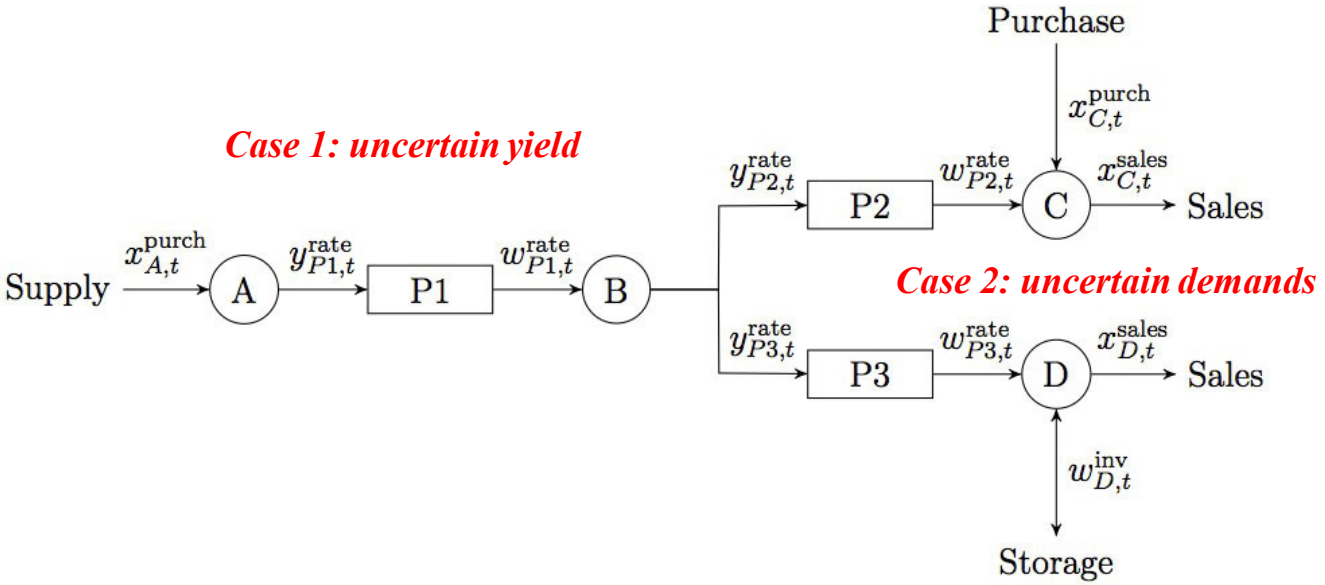
- Final result



- *NLP Approach*: calculate both probabilities and outcome values.
- *LP Approach*: fix outcome values, calculate probabilities.

Motivating Example: Process Network

- **Network of chemical plants**

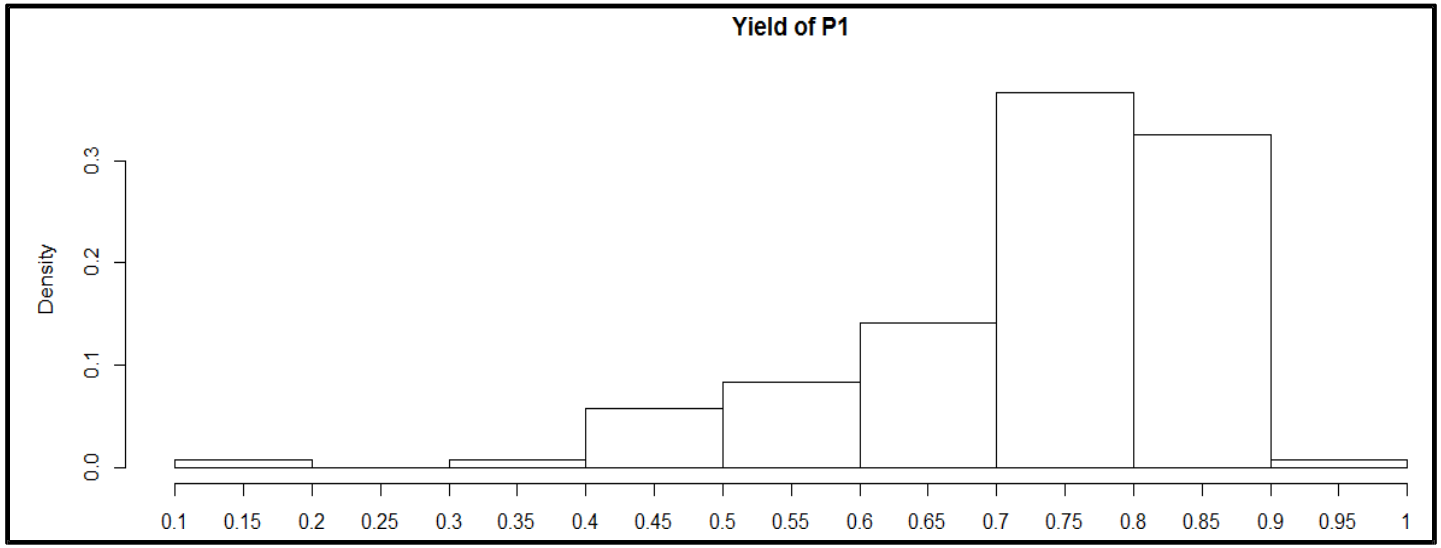


- 1 raw material (A), 1 intermediate product (B), two finished products (C and D)
- Only D can be stored and C can be purchased from elsewhere

Carnegie Mellon

Case 1: Uncertain Yield Process 1

- Historical data for production yield of facility *P1*

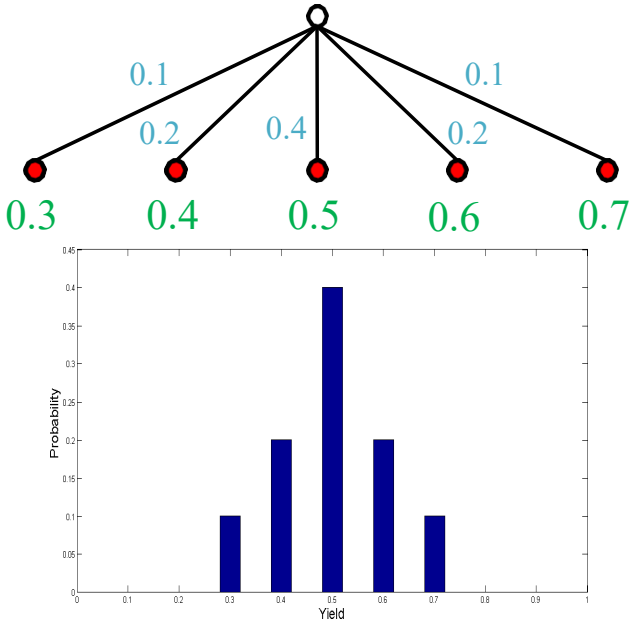


- Skewed to the right
- Tail effects (extreme values) are not negligible
- Approaches: Original MMP and L^2 DMP (2 moments + ECDF)
- TSSP, where first stage is $t = 1$ and second stage is $t = 2, \ldots, 4$

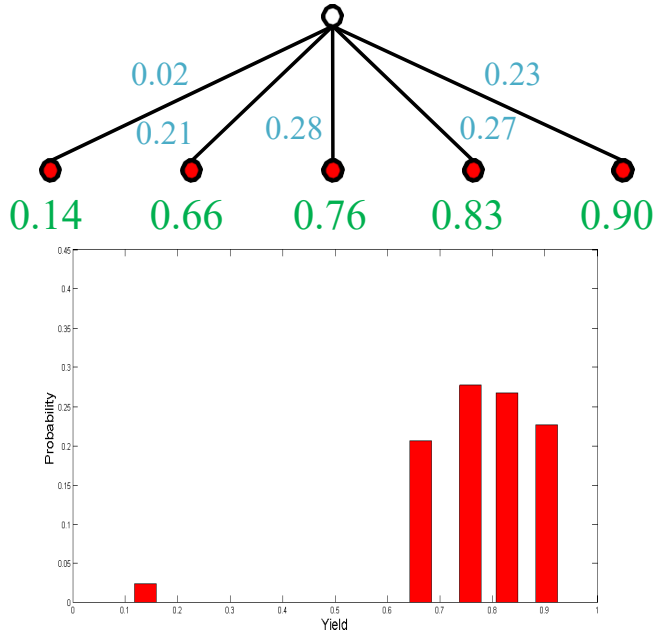
Carnegie Mellon

Two-Stage Scenario Trees

- **Heuristic Approach**



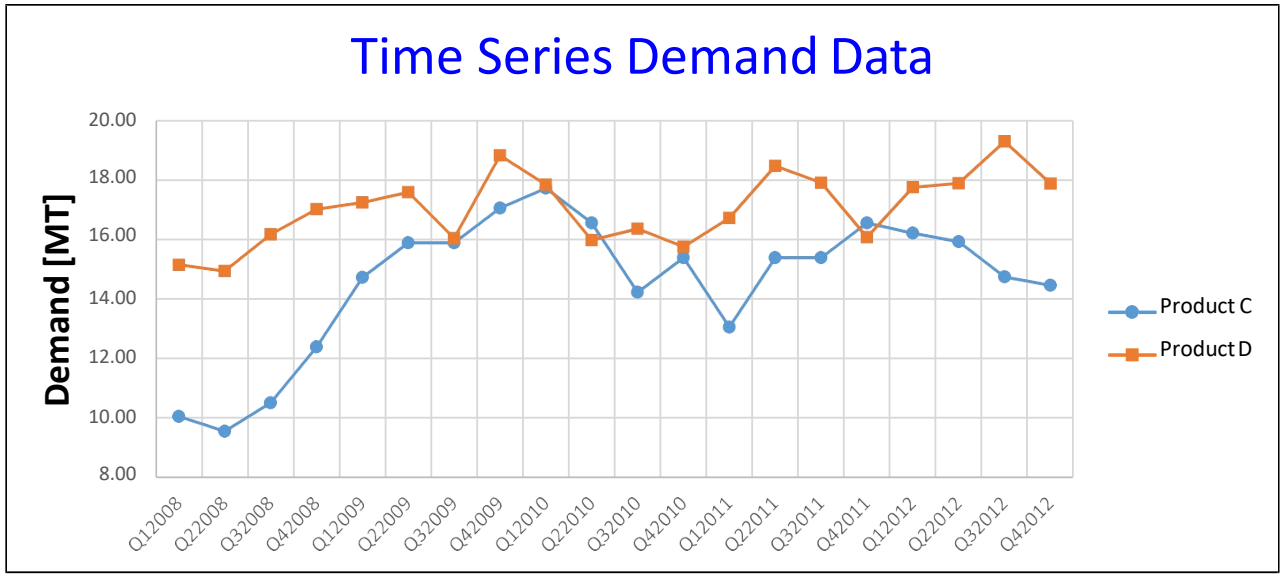
- **L² DMP Approach**



Approach	Expected Profit [$]
Heuristic	62.77
L² DMP	72.45

Carnegie Mellon

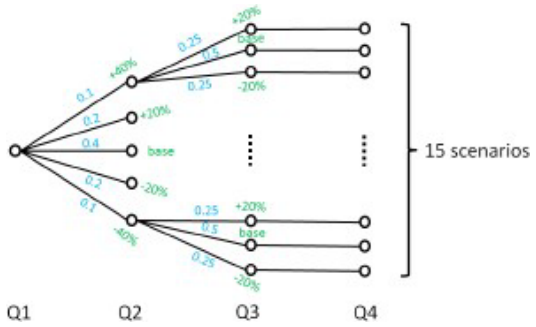
Case 2: Uncertain Product Demands *4 yrs*



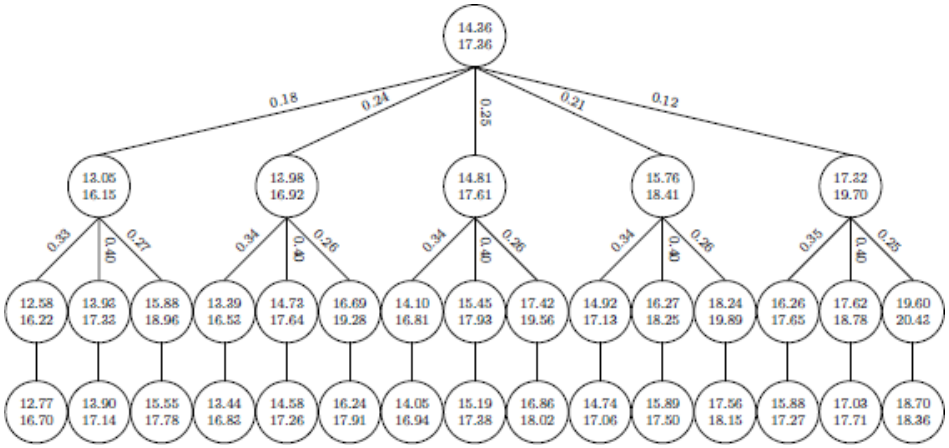
- Statistical analyses performed in R (forecast package)
- Both stochastic processes are modeled as ARIMA(1,0,0) with non-zero mean
- Approaches: Heuristic, L^2 DMP (2 moments + normal CDF)
- MSSP, where stage: time point = {1:1, 2:2, 3:3, 3:4}

Multi-Stage Scenario Trees

- **Heuristic Approach**



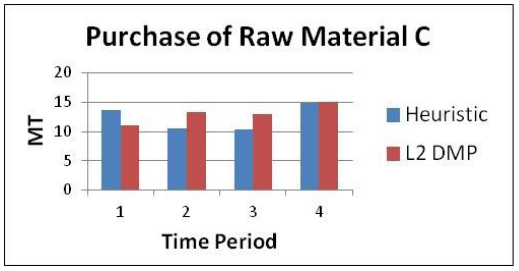
- **L^2 DMP Approach**

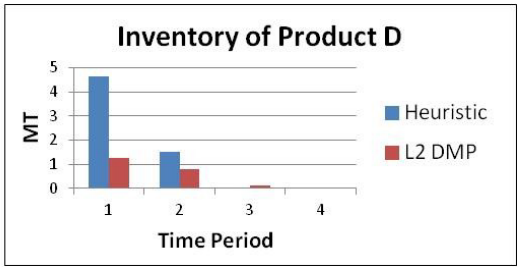


Expected profits [$]

Heuristic	L^2 DMP
79.95	82.39

NLPs: 12.3 sec (IPOPT)





Carnegie Mellon

Property Matching Method

- Pros
  - Do not have to know/assume a distribution family, only to estimate values of the required properties.
  - Can combine historical data with today's predictions.
  - The marginal distributions can have very different shapes, so the vector does not follow any standard distribution.
- Cons
  - No convergence to the true distribution.
  - If we know the distribution, we cannot utilize this information, i.e. we throw it away. Can be hard to find which properties to use.

Conclusion

- Scenario generation is an important part of the modelling/solving process for stochastic programming models.
- A bad scenario-generation method can spoil the result of the whole optimization.
- There is an increasing choice of methods, but one has to test which one works best for a given problem.