Michal Kaut

Scenario Generation for Stochastic Programming
Introduction and selected methods

SINTEF Technology and Society

September 2011
Outline

Introduction to Scenario Generation
  Scenario Trees: What? Why?
  Scenario trees – terminology etc.
Generating scenario trees
  Some general comments
Measuring Quality of Scenario Trees
  Quality and how to measure it
  Stability tests
  Estimation of upper-bound of the optimality gap
Scenario-Generation Methods
  Conditional sampling
  Property-matching methods
  “Optimal Discretization”
  Scenario reduction techniques
Outline

Introduction to Scenario Generation
  Scenario Trees: What? Why?
    Scenario trees – terminology etc.
Generating scenario trees
  Some general comments
Measuring Quality of Scenario Trees
  Quality and how to measure it
  Stability tests
  Estimation of upper-bound of the optimality gap
Scenario-Generation Methods
  Conditional sampling
  Property-matching methods
  “Optimal Discretization”
  Scenario reduction techniques
Where do scenarios come from?

A stochastic programming (SP) problem is a math. programming problem, with values of some parameters replaced by distributions. Hence, to solve the problem, we need:

- A model describing the problem.
- Values of the deterministic (known) parameters.
Where do scenarios come from?

A stochastic programming (SP) problem is a math. programming problem, with values of some parameters replaced by distributions. Hence, to solve the problem, we need:

- A model describing the problem.
- Values of the deterministic (known) parameters.
- Description of the stochasticity.
  - Known distributions, described by densities and/or CDFs.
  - Historical data, i.e. a discrete sample.
  - Only some properties of the distributions, for ex. moments.

Since SP can handle only discrete samples of limited size, we need to approximate the distribution of the stochastic parameters. The approximation is called a scenario tree.
Structure of an SP problem

Real-world problem

Modelling

Data analysis

SP model

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

- Real-world problem
  - Modelling
  - Data analysis
    - Scenario tree
  - SP model

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

Real-world problem

Modelling

Data analysis

Scenario tree

SP model

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

Actually, it is not true that we always need scenario trees.

- There are solution methods that sample the values as a part of the solution process, i.e. they create the tree ‘on the go’.
- The information where to add samples is obtained from the model, for example from the dual variables – in which case it works only for linear programs.

Examples of these methods include:

- importance sampling within Benders decomp. – Dantzig and Infanger (1992)
  – this works for convex programs, not only LPs.

Note that even if the solution methods create the scenario trees internally, we still have to decide at least the number of stages.
Outline

Introduction to Scenario Generation
  Scenario Trees: What? Why?
  Scenario trees – terminology etc.
Generating scenario trees
  Some general comments
Measuring Quality of Scenario Trees
  Quality and how to measure it
  Stability tests
  Estimation of upper-bound of the optimality gap
Scenario-Generation Methods
  Conditional sampling
  Property-matching methods
  “Optimal Discretization”
  Scenario reduction techniques
Scenario tree – terminology

Terminology:

- Stage: a moment in time, when decisions are taken, i.e., when we get new information so the last time point is not a stage.
- Period: the interval between two time points.
- Scenario: a path from the root to one leaf.

Tree above: 4 stages, 4 periods, and $3 \times 3 \times 2 = 18$ scenarios.
Scenario tree – terminology

Terminology:

- **Stage**: A moment in time when decisions are taken, i.e. when we get new information.
- **Period**: The interval between two time points.
- **Scenario**: A path from the root to one leaf.

Tree above:
- 4 stages,
- 4 periods,
- $3 \times 3 \times 2 = 18$ scenarios.
Scenario tree – terminology

Terminology:

stage is a moment in time, when decisions are taken, i.e. when we get new information – so the last time point is not a stage.
Terminology:

- **stage** is a moment in time, when decisions are taken, i.e. when we get new information – so the last time point is *not* a stage.
- **period** is the interval between two time points.
Scenario tree – terminology

**Terminology:**
- **stage** is a moment in time, when decisions are taken, i.e. when we get new information – so the last time point is *not* a stage.
- **period** is the interval between two time points.
- **scenario** is a path from the root to one leaf.
Scenario tree – terminology

Terminology:

- **stage** is a moment in time, when decisions are taken, i.e. when we get new information – so the last time point is *not* a stage.
- **period** is the interval between two time points.
- **scenario** is a path from the root to one leaf.

Tree above: 4 stages, 4 periods, and $3 \times 3 \times 2 = 18$ scenarios.
Scenario tree – importance of branching

Why a tree, why not a “fan” like this?

- Branching = arrival of new information.
- Fan above: no new information after the second stage.
- Hence, the fan represents a two-stage problem (with 3 periods)
Outline

Introduction to Scenario Generation
   Scenario Trees: What? Why?
   Scenario trees – terminology etc.

Generating scenario trees
   Some general comments

Measuring Quality of Scenario Trees
   Quality and how to measure it
   Stability tests
   Estimation of upper-bound of the optimality gap

Scenario-Generation Methods
   Conditional sampling
   Property-matching methods
   “Optimal Discretization”
   Scenario reduction techniques
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/branches for each node.
Sources of data for scenarios

- **Historical data**
  - Is history a good description of the future?

- **Simulation based on a mathematical/statistical model**
  - Parameters estimated from the real case

- **Expert opinion**
  - Subjective
  - Back-testing is not possible.

- Often a *combination* of more 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.
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
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
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
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
A good scenario tree should capture

- Inter-temporal dependencies
  - changes of the distributions, based on prev. values
  - includes things like auto-correlations, mean reversion, etc.
  - can be modelled by time-series models
A good scenario tree should capture

- Inter-temporal dependencies
  - changes of the distributions, based on prev. values
  - includes things like auto-correlations, mean reversion, etc.
  - can be modelled by time-series models
A good scenario tree should capture

- Inter-temporal dependencies
  - changes of the distributions, based on prev. values
  - includes things like auto-correlations, mean reversion, etc.
  - can be modelled by time-series models
A good scenario tree should capture

- Inter-temporal dependencies
  - changes of the distributions, based on prev. values
  - includes things like auto-correlations, mean reversion, etc.
  - can be modelled by time-series models
Outline

Introduction to Scenario Generation
  Scenario Trees: What? Why?
  Scenario trees – terminology etc.
Generating scenario trees
  Some general comments
Measuring Quality of Scenario Trees
  Quality and how to measure it
  Stability tests
  Estimation of upper-bound of the optimality gap
Scenario-Generation Methods
  Conditional sampling
  Property-matching methods
  “Optimal Discretization”
  Scenario reduction techniques
Quality of Scenario Trees and How to Measure It

In accessing the quality, we have consider two things:

**Stability**
- If we generate several scenario trees, the solutions should not vary *too much*.
- Stochastic programs tend to have flat objective functions, so we can usually only require stability of the objective values, not of the solutions themselves.

**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 (unsolvable) problem

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

is replaced by a scenario-based problem

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

In the stability tests, we generate several scenario trees $\tilde{\eta}_k$, $k = 1, \ldots, n$, leading to solutions

$$x_k^* = \arg\min_{x \in X} F(x; \tilde{\eta}_k) .$$
Error Caused by the Discretization

Pflug (2001) defines an approximation error caused by $\tilde{\eta}_k$ (also called an optimality gap) as:

$$e_f(\xi, \tilde{\eta}_k) = F\left( \arg\min_x F(x; \tilde{\eta}_k); \xi \right) - F\left( \arg\min_x F(x; \tilde{\xi}); \tilde{\xi} \right) = F\left( x_k^*; \tilde{\xi} \right) - \min_x F(x; \tilde{\xi}) \geq 0.$$  

To evaluate $e_f(\xi, \tilde{\eta}_k)$, we would need to:

• Evaluate the “true” objective function $F(x; \tilde{\eta}_k)$.
  - Sometimes can be done using a “simulator”.

• Solve the original problem, i.e. $(\arg\min)\min_x F(x; \tilde{\xi}); \tilde{\xi}$.
  - Impossible – otherwise, we would not need scenarios.
Error Caused by the Discretization

Pflug (2001) defines an approximation error caused by $\tilde{\eta}_k$ (also called an optimality gap) as:

$$e_f(\tilde{\xi}, \tilde{\eta}_k) = F\left( \arg\min_x F(x; \tilde{\eta}_k); \tilde{\xi} \right) - F\left( \arg\min_x F(x; \tilde{\xi}); \tilde{\xi} \right)$$

$$= F\left( x^*_k; \tilde{\xi} \right) - \min_x F(x; \tilde{\xi}) \geq 0.$$ 

To evaluate $e_f(\tilde{\xi}, \tilde{\eta}_k)$, we would need to:

- Evaluate the “true” objective function $F(x; \tilde{\xi})$.
  - Can sometimes be done using a “simulator”.

Scenario Generation for Stochastic Programming
Error Caused by the Discretization

Pflug (2001) defines an approximation error caused by $\tilde{\eta}_k$ (also called an optimality gap) as:

$$e_f(\tilde{\xi}, \tilde{\eta}_k) = F\left( \arg\min_x F(x; \tilde{\eta}_k); \tilde{\xi} \right) - F\left( \arg\min_x F(x; \tilde{\xi}); \tilde{\xi} \right)$$

$$= F\left( x_k^*; \tilde{\xi} \right) - \min_x F(x; \tilde{\xi}) \geq 0.$$ 

To evaluate $e_f(\tilde{\xi}, \tilde{\eta}_k)$, we would need to:

- Solve the original problem, i.e. $(\arg) \min_x F(x; \tilde{\xi})$.
  - Impossible – otherwise, we would not need scenarios.
Outline

Introduction to Scenario Generation
  Scenario Trees: What? Why?
  Scenario trees – terminology etc.
Generating scenario trees
  Some general comments
Measuring Quality of Scenario Trees
  Quality and how to measure it
  Stability tests
  Estimation of upper-bound of the optimality gap
Scenario-Generation Methods
  Conditional sampling
  Property-matching methods
  “Optimal Discretization”
  Scenario reduction techniques
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.

Without stability, we have a problem!
Tests Using a Simulator

Assume that we have a “simulator” for evaluating $F(x; \tilde{\xi})$, 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 $\tilde{\eta}_k$, $k = 1, \ldots, n$.
  2. Solve problems using the trees $\rightarrow$ solutions $x_k^*$.
  3. Test whether $F(x_k^*; \tilde{\xi}) \approx F(x_l^*; \tilde{\xi})$

- The test is equivalent to $e_f(\tilde{\xi}, \tilde{\eta}_k) \approx e_f(\tilde{\xi}, \tilde{\eta}_l)$.
- Without stability, we have a problem!
Notes on the Stability Test

- \( e_f(\tilde{\xi}, \tilde{\eta}_k) \approx 0 \) implies \( e_f(\tilde{\xi}, \tilde{\eta}_k) \approx e_f(\tilde{\xi}, \tilde{\eta}_l) \) and stability.
- Stability test assumes that we get a different tree on each run of the scenario-generation method.
- Otherwise, we can run it with different tree sizes.
Notes on the Stability Test

- $e_f(\tilde{\xi}, \tilde{\eta}_k) \approx 0$ implies $e_f(\tilde{\xi}, \tilde{\eta}_k) \approx e_f(\tilde{\xi}, \tilde{\eta}_l)$ and stability.
- Stability test assumes that we get a different tree on each run of the scenario-generation method.
- Otherwise, we can run it with different tree sizes.

Another issues:

- Only the root variables can be moved from one tree to another, as the scenarios do not coincide.
  → To evaluate $F(x; \tilde{\xi})$, we have to fix the root part of $x$ and (re)solve the problem.
- The root solution $x$ may be infeasible with scenarios $\tilde{\xi}$ — one can try to move constraints to the objective
Out-of-Sample Tests Without a Simulator

Instead of using a simulator, we can “cross test”, i.e. test

\[ F(x^*_k; \tilde{\eta}_l) \quad \text{for} \quad l \neq k \]

for all \( k = 1, \ldots, n \).

- 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 would choose the most stable one.
In-Sample Stability

Instead of the true performance, we look at the optimal objective values reported by the problems themselves:

\[ F(x_k^*; \tilde{\eta}_k) \approx F(x_l^*; \tilde{\eta}_l), \]

or, equivalently,

\[ \min_x F(x; \tilde{\eta}_k) \approx \min_x F(x; \tilde{\eta}_l). \]

- No direct connection to out-of-sample stability.
  - Can even have \( e_f(\tilde{\xi}, \tilde{\eta}) = 0 \), without in-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 $\rightarrow$ 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.

Note: A proper mathematical treatment of stability can be found in Dupačová and Römisch (1998); Fiedler and Römisch (2005); Heitsch et al. (2006).
Example: What Is the Best Method and/or Solution?

In-sample stability of three different methods.

Shows the optimal objective values for different sizes of scenario trees.
Example: What Is the Best Method and/or Solution?

Out-of-sample of three different methods.

Shows a level of infeasibility of the solutions for different sizes of scenario trees.
Outline

Introduction to Scenario Generation
  Scenario Trees: What? Why?
  Scenario trees – terminology etc.
Generating scenario trees
  Some general comments

Measuring Quality of Scenario Trees
  Quality and how to measure it
  Stability tests
  Estimation of upper-bound of the optimality gap

Scenario-Generation Methods
  Conditional sampling
  Property-matching methods
  “Optimal Discretization”
  Scenario reduction techniques
Stochastic upper bound for the optimality gap

Let us assume that the scenario trees are sampled from the true distribution, so they are unbiased and denote

\[ z^* = \min_{x \in X} F(x; \tilde{\xi}) = F(x^*; \tilde{\xi}) \ldots \text{true minimum} \]

\[ z_k^* = \min_{x \in X} F(x; \tilde{\eta}_k) = F(x_k^*; \tilde{\eta}_k) \ldots \text{in-sample objective,} \]

so we have

\[ e_f(\tilde{\xi}, \tilde{\eta}_k) = F(x_k^*; \tilde{\xi}) - z^*. \]

Then, under some convexity assumptions,

\[ \mathbb{E} [z_k^*] \leq z^*, \]
Stochastic upper bound for the optimality gap

i.e. the in-sample objective values are too optimistic.

If we, in addition, have $F(x; \tilde{\xi}) = \mathbb{E}^{\tilde{\xi}}[f(x, \tilde{\xi})]$, then

$$\mathbb{E} \left[ F(x; \tilde{\eta}_k) \right] = F(x; \tilde{\xi}) ,$$

since sampling is unbiased. With our $n$ scenario trees we get an estimate

$$\frac{1}{n} \sum_{i=1}^{n} F(x; \tilde{\eta}_i) \approx F(x; \tilde{\xi}) .$$
Stochastic upper bound for the optimality gap III

This allows us to estimate the optimality gap $e_f(\tilde{\xi}, \tilde{\eta}_k)$ as

$$e_f(\tilde{\xi}, \tilde{\eta}_k) = F(x_k^*; \tilde{\xi}) - z^*$$

$$\approx \frac{1}{n} \sum_{i=1}^{n} F(x_k^*; \tilde{\eta}_i) - z_k^*.$$

Notes:

- 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.
In addition, Bayraksan and Morton (2006) provides methods for estimating the optimality gap using only one or two scenario trees.
Outline

Introduction to Scenario Generation
  Scenario Trees: What? Why?
  Scenario trees – terminology etc.
Generating scenario trees
  Some general comments
Measuring Quality of Scenario Trees
  Quality and how to measure it
  Stability tests
  Estimation of upper-bound of the optimality gap
Scenario-Generation Methods
  Conditional sampling
  Property-matching methods
  “Optimal Discretization”
  Scenario reduction techniques
One-Period Case - Standard Sampling I.

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

- Generate one margin at a time, then join together, first with first, second with second...
  - independent only in the limit
  - size independent on the dimension
One-Period Case - Standard Sampling II.

General multivariate case

- Special methods for some distributions.
  - Ex.: normal distribution via Cholesky decomposition

- Use principal components to get “independent” variables.
  - Components are independent only for normal variables.
  - Generally, they are only uncorrelated.
One-Period Case - Standard Sampling II.

**General multivariate case**
- Special methods for some distributions.
  - Ex.: normal distribution via Cholesky decomposition
- Use principal components to get “independent” variables.
  - Components are independent only for normal variables.
  - Generally, they are only uncorrelated.

**Bootstrapping / Sampling from historical data**
- Does not need any distributional assumptions.
- Needs historical data.
- Are historical data a good description of the future?
Handling Multiple Periods

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, GARCH, ...*
- Effects we might want to consider/model:
  - mean reversion
  - variance increase after a big jump
Stochastic processes – ARMA etc.

A new value $X_t$ is generated as

$$X_t = f(X_{t-1}, X_{t-2}, \ldots; \varepsilon_{t-1}, \varepsilon_{t-2}, \ldots; \varepsilon_t),$$

where $\varepsilon_t$ is a random disturbance, usually $\varepsilon_t \sim N(0, \sigma^2)$.

Standard examples:

**AR($p$) process:** $X_t = c + \sum_{i=1}^{p} p_i X_{t-i} + \varepsilon_t$

**MA($q$) process:** $X_t = \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$

**ARMA($p, q$) process:** $X_t = \varepsilon_t + \sum_{i=1}^{p} p_i X_{t-i} + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$
Stochastic processes – GARCH etc.

Sometimes, we might need to handle heteroskedasticity, i.e. non-constant variance. This is done using

$$\varepsilon_t = \sigma_t z_t, \quad z_t \sim N(0, 1),$$

where $\sigma_t$ follows a ARCH (autoregressive conditional heteroskedasticity) or GARCH (generalized autoregressive conditional heteroskedasticity) process, where

- **GARCH($p, q$)** is defined as

  $$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2,$$

  i.e. $\sigma_t^2$ follows an ARMA process.

- **ARCH($q$)** process is a GARCH($0, q$) process.

- Many different generalizations exist.
Stochastic processes – standard use

\[ t - 3 \]
\[ t - 2 \]
\[ t - 1 \]
\[ t \]
Stochastic processes – standard use

$t - 3$
$t - 2$
$t - 1$
$t$

Scenario Generation for Stochastic Programming

37
Stochastic processes – standard use
Stochastic processes – standard use

Scenario Generation for Stochastic Programming
Stochastic processes – standard use
Stochastic processes – standard use

$t - 3$
$t - 2$
$t - 1$
$t$
Stochastic processes – standard use

Scenario Generation for Stochastic Programming
Stochastic processes – standard use
Stochastic processes – standard use
Stochastic processes – creating a tree

Using several values of $\varepsilon_t$ at each node:

$$t - 3$$
$$t - 2$$
$$t - 1$$
$$t$$
Stochastic processes – creating a tree

Using several values of $\varepsilon_t$ at each node:
Stochastic processes – creating a tree

Using several values of $\varepsilon_t$ at each node:
Stochastic processes – creating a tree

Using several values of $\varepsilon_t$ at each node:
Stochastic processes – creating a tree

Using several values of $\varepsilon_t$ at each node:
Stochastic processes – creating a tree

Using several values of $\varepsilon_t$ at each node:
Stochastic processes – creating a tree

Using several values of $\varepsilon_t$ at each node:
Stochastic processes – creating a tree

Using several values of $\varepsilon_t$ at each node:
Sampling Methods – Summary

Pros
- Easy to implement.
- Distribution converges to the true one.

Cons
- Bad performance/stability for small trees.
  - This can be improved by using corrections or some special techniques, such as low-discrepancy sequences (see for example Pennanen, 2007).
- Have to know the distribution to sample from.
Outline

Introduction to Scenario Generation
  Scenario Trees: What? Why?
  Scenario trees – terminology etc.
Generating scenario trees
  Some general comments
Measuring Quality of Scenario Trees
  Quality and how to measure it
  Stability tests
  Estimation of upper-bound of the optimality gap

Scenario-Generation Methods
  Conditional sampling
  Property-matching methods
  “Optimal Discretization”
  Scenario reduction techniques
Property-Matching Methods – Basic Info

- These methods **construct** the scenario trees in such a way that a given set of properties is matched.
- The properties are for ex. **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.
Example 1 – from Høyland and Wallace (2001)

- An optimization problem with values of the random variables and scenario probabilities as variables.
- The measured properties are expressed as function of these variables.
- The objective is to minimize a distance (usually $L_2$) of these properties from their target values.
- Leads to highly non-linear, non-convex problems.

- Works well for small trees, otherwise very slow.
- The optimization is often underspecified & no control what the solver does about the extra degrees of freedom.

- Developed as a fast approximation to the previous method, in the case of *four marginal moments + correlations*.
- Build around two transformations:
  1. Correcting correlations
     - Multiply the random vector by a Cholesky component
     - Changes also the marginal distributions (except normal)
  2. Correcting the marginal distributions
     - A *cubic transformation* of the margins, one margin at a time
     - Changes the correlation matrix

- The two transformations are repeated alternately.
- Starting point can be, for ex., a correlated normal vector.

- Works well for *large trees* (creates *smooth* distributions).
- Needs pre-specified probabilities (usually equiprobable).
Property-Matching Methods – Summary

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 can not utilize this information, i.e. we throw it away.
- Can be hard to find which properties to use.
Outline

Introduction to Scenario Generation
- Scenario Trees: What? Why?
- Scenario trees – terminology etc.

Generating scenario trees
- Some general comments

Measuring Quality of Scenario Trees
- Quality and how to measure it
- Stability tests
- Estimation of upper-bound of the optimality gap

Scenario-Generation Methods
- Conditional sampling
- Property-matching methods
- “Optimal Discretization”
- Scenario reduction techniques
“Optimal Discretization” by I.

Starts with the approximation error $e_f(\tilde{\xi}, \tilde{\eta}_k)$:

$$e_f(\tilde{\xi}, \tilde{\eta}_k) = F\left( \arg\min_x F(x; \tilde{\eta}_k); \tilde{\xi} \right) - F\left( \arg\min_x F(x; \tilde{\xi}); \tilde{\xi} \right)$$

$$= F\left( x^*_k; \tilde{\xi} \right) - \min_x F(x; \tilde{\xi}) \geq 0.$$ 

Pflug (2001) shows that, under certain Lipschitz conditions,

$$e_f(\tilde{\xi}, \tilde{\eta}_k) \leq 2 \sup_x \left| F(x; \tilde{\eta}_k) - F(x; \tilde{\xi}) \right| \leq 2 L d(\tilde{\eta}_k, \tilde{\xi}),$$

where $L$ is a Lipschitz constant of $f()$, with $F(x; \tilde{\xi}) = \mathbb{E}^{\tilde{\xi}} \left[ f(x, \tilde{\xi}) \right]$ and $d(\tilde{\eta}_k, \tilde{\xi})$ is a Wasserstein (transportation) distance of distribution functions of $\tilde{\eta}_k$ and $\tilde{\xi}$. 

Scenario Generation for Stochastic Programming
The method then creates a scenario tree that minimizes the transportation distance $d(\tilde{\eta}_k, \tilde{\xi})$.

- Whole multi-period tree is generated at once.
- The tree is “optimal” in a clearly specified sense.

- Difficult to both understand and use.

Outline

Introduction to Scenario Generation
  Scenario Trees: What? Why?
  Scenario trees – terminology etc.
Generating scenario trees
  Some general comments
Measuring Quality of Scenario Trees
  Quality and how to measure it
  Stability tests
  Estimation of upper-bound of the optimality gap

Scenario-Generation Methods
  Conditional sampling
  Property-matching methods
  “Optimal Discretization”
  Scenario reduction techniques
Scenario Reduction

- The idea is to reduce size of a given scenario tree $\tilde{\mathcal{G}}$ into a smaller tree $\tilde{\eta}$, 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)
- The theory shows that the change in solution can be approximated using a Fortet-Mourier-type metric.
  - metric on probability spaces, independent on the optimization problem
- This leads to a Monge Kantorovich mass transportation problem
‘Classical Scenario Reduction Algorithms I.’

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 $\rightarrow$ need 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

The results of one of their numerical examples were:
- $50\%$ scenarios give $90\%$ relative accuracy.
- $2\%$ scenarios give $50\%$ relative accuracy.
‘Classical Scenario Reduction Algorithms I.’

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 $\rightarrow$ need 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

The results of one of their numerical examples were:
- 50\% scenarios give 90\% relative accuracy.
- 2\% scenarios give 50\% relative accuracy.
‘Classical Scenario Reduction Algorithms I.’

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 $\rightarrow$ need 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

The results of one of their numerical examples were:
- 50\% scenarios give 90\% relative accuracy.
- 2\% scenarios give 50\% relative accuracy.
‘Classical Scenario Reduction Algorithms I.’

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 $\Rightarrow$ need 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

- The results of one of their numerical examples were:
  - 50% scenarios give 90% relative accuracy.
  - 2% scenarios give 50% relative accuracy.
‘Classical Scenario Reduction Algorithms II.’

Dupačová et al. (2003); Heitsch and Römisch (2003, 2007)

- The forward selection algorithm gives better results, but is very slow for big $N$ and $k$.
- Heitsch and Römisch (2007) presents improved versions of the heuristics
‘Classical Scenario Reduction Algorithms II.’

Dupačová et al. (2003); Heitsch and Römisch (2003, 2007)

- The forward selection algorithm gives better results, but is very slow for big $N$ and $k$.
- Heitsch and Römisch (2007) presents improved versions of the heuristics

**Problem**

- People use these techniques for multistage trees, which is *not appropriate*, as pointed out in Heitsch and Römisch (2009)
- In addition, the algorithms are used to reduce a fan to a tree, which is also not supported by the theory!
Multistage Scenario Reduction

Heitsch and Römisch (2009)

- Based on stability results for multistage stochastic programs from Heitsch et al. (2006)
  - They find out that in the multi-stage case, one has to use a filtration distance, in addition to the Fortet-Mourier-type metric.
  - This filtration distance measures the difference between the \( \sigma \)-algebras implied by the scenario trees.

- The reduction algorithm is similar to the backward reduction from the two-stage case: at each step, find a pair of nodes with the same parent that are ‘close’ and merge them.
Multistage Scenario Reduction

Heitsch and Römisch (2009)

- Based on stability results for multistage stochastic programs from Heitsch et al. (2006)
  - They find out that in the multi-stage case, one has to use a *filtration distance*, in addition to the Fortet-Mourier-type metric.
  - This filtration distance measures the difference between the $\sigma$-algebras implied by the scenario trees.

- The reduction algorithm is similar to the backward reduction from the two-stage case: at each step, find a pair of nodes with the same parent that are ‘close’ and merge them.

- Note that also this method is *not* suitable to produce a tree out of a fan—simply because the filtration of the fan is wrong to start with.
Outline

Introduction to Scenario Generation
   Scenario Trees: What? Why?
   Scenario trees – terminology etc.

Generating scenario trees
   Some general comments

Measuring Quality of Scenario Trees
   Quality and how to measure it
   Stability tests
   Estimation of upper-bound of the optimality gap

Scenario-Generation Methods
   Conditional sampling
   Property-matching methods
   “Optimal Discretization”
   Scenario reduction techniques
Summary

- 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.

Open questions:
- Is there a universally good scenario-generation method?
- What is the optimal structure of a tree (deep vs. wide)?
For Further Reading I


For Further Reading II


For Further Reading III


For Further Reading IV


For Further Reading V

The End
Example of the Optimization-Based Moment Matching

2 variables \( x, y \) + node probabilities \( p \)

Specifications:
- \( \mathbb{E}[x], \mathbb{E}[y]; \mathbb{E}[x^2], \mathbb{E}[y^2]; \) \( \text{Cov}(x, y) \)
- Possibly other functions of \( x, y, p \).

$$\min_{x,y,p} \left( \sum_i p_i x_i - \mathbb{E}[x] \right)^2 + \left( \sum_i p_i y_i - \mathbb{E}[y] \right)^2$$
$$+ \left( \sum_i p_i x_i^2 - \mathbb{E}[x^2] \right)^2 + \left( \sum_i p_i y_i^2 - \mathbb{E}[y^2] \right)^2$$
$$+ \left( \sum_i p_i (x_i - \mathbb{E}[x])(y_i - \mathbb{E}[y]) - \text{Cov}(x, y) \right)^2$$

s.t.: \( \sum_i p_i = 1 \) and \( p_i \geq 0, \ i = 1, \ldots, 3 \).
More Info on Transformation-Based Moment Matching

Correction of the correlations

- The target correlation matrix is $R_* = L_*L_*^T$.
- The correlation matrix at step $k$ is $R_k = L_kL_k^T$.
- Then $Y = L_*L_k^{-1}X$ has correlation matrix $R_*$. 

The cubic transformation

- For each margin $i$: $Y_i = a + bX_i + cX_i^2 + dX_i^3$
- To find the coefficients $a, b, c, d$, we have to:
  - express the moments of $Y_i$ as a function of $a, b, c, d$ and the moments of $X$;
  - find the values of $a, b, c, d$ that minimize the $L_2$ distance of the moments from their target values.

- This is a non-linear, non-convex optimization problem fortunately with only four variables.