Risk Neutral and Risk Averse Multistage Stochastic Programming.

A. Shapiro
School of Industrial and Systems Engineering,
Georgia Institute of Technology,
Atlanta, Georgia 30332-0205, USA

SESO 2014 International Thematic Week
"Smart Energy and Stochastic Optimization"

Consider a multistage decision process of the from

$$\begin{array}{c} \text{decision } (x_1) \rightsquigarrow \text{observation } (\xi_2) \rightsquigarrow \text{decision } (x_2) \rightsquigarrow \\ ..... \rightsquigarrow \text{observation } (\xi_T) \rightsquigarrow \text{decision } (x_T). \end{array} \tag{1}$$

Here $\xi_t \in \mathbb{R}^{d_t}$, $t = 1, ...,$ is a sequence of vectors with $\xi_{[t]} := (\xi_1, ..., \xi_t)$ representing history of the data process up to time t. At time period $t \in \{1, ..., T\}$ we observe the past, $\xi_{[t]}$, but future observations $\xi_{t+1}, ...,$ are uncertain. So our decision at time t should only depend on information available at that time, i.e., $x_t = x_t(\xi_{[t]})$ should be a function of $\xi_{[t]}$ and should not depend on future observations. This is the basic requirement of *nonanticipativity* of the decision process. A sequence $x_1, x_2(\xi_{[2]}), ...$ of such decisions is called a *policy* or a decision rule.

Nested formulation the linear multistage problem

$$\operatorname*{Min}_{\substack{A_1x_1=b_1\\ x_1\geq 0}} c_1^{\mathsf{T}}x_1+\mathbb{E}\left[\operatorname*{Min}_{\substack{B_2x_1+A_2x_2=b_2\\ x_2\geq 0}} c_2^{\mathsf{T}}x_2+\cdots+\mathbb{E}\left[\operatorname*{Min}_{\substack{B_Tx_{T-1}+A_Tx_T=b_T\\ x_T\geq 0}} c_T^{\mathsf{T}}x_T\right]\right].$$

Equivalent formulation

$$\begin{array}{ll}\operatorname*{Min}\limits_{x_1,x_2(\cdot),...,x_T(\cdot)} & \mathbb{E}\left[c_1^{\mathsf{T}}x_1+c_2^{\mathsf{T}}x_2(\xi_{[2]})...+c_T^{\mathsf{T}}x_T(\xi_{[T]})\right]\\ \text{s.t.} & A_1x_1=b_1,\ x_1\geq 0,\\ & B_tx_{t-1}(\xi_{[t-1]})+A_tx_t(\xi_{[t]})=b_t,\\ & x_t(\xi_{[t]})\geq 0,\ t=2,...,T.\end{array}$$

Here $\xi_t=(c_t,B_t,A_t,b_t)$, $t=2,...,T$, is considered as a random process, $\xi_1=(c_1,A_1,b_1)$ is supposed to be known, $\xi_{[t]}:=(\xi_1,...,\xi_t)$ denotes history of the data process up to time t.

Optimization is performed over feasible policies (also called decision rules). A policy is a sequence of (measurable) functions $x_t = x_t(\xi_{[t]})$, $t = 1, ..., T$. Each $x_t(\xi_{[t]})$ is a function of the data process up to time t, this ensures the *nonanticipative* property of a considered policy. The constraints should be satisfied for almost every realization of the random data process.

The equivalence of two formulations is based on the decomposition property of the expectation operator

$$\mathbb{E}[\cdot] = \mathbb{E}_{|\xi_{[1]}}\Big[\mathbb{E}_{|\xi_{[2]}}\Big[\cdots \mathbb{E}_{|\xi_{[T]}}[\cdot]\Big]\Big]$$

and interchangeability of the expectation and minimization operators. The nested formulation can be used to derive dynamic programming equations.

Ideally, the multistage problem is solved if one can construct a feasible policy minimizing the expected value of the (total) cost.

This formulation assumes that: (i) the probability distribution of the data process is known (specified), (ii) the optimization is performed *on average* (both, with respect to different realizations of the random process, and with respect to time).

Numerical difficulties in solving multistage problems.

From a modeling point of view typically it is natural to assume that the random data process has a *continuous* distribution. This raises the question of how to compute the involved expectations (multivariate integrals). A standard approach is to discretize the random process by generating a finite number of possible realizations (called scenarios). These scenarios can be represented by the corresponding *scenario tree*.

How many scenarios are needed in order to approximate the "true" distribution of the random data process?

Note that solving the deterministic equivalent for the constructed scenario tree does not produce by itself an implementable policy for the "true" problem (with continuous distributions). This is because an actual realization of the data process could, and with probability one (w.p.1) will, be different from scenarios used in the constructed tree. In that case policy constructed for scenarios of the tree does not tell what decision to make. Of course, one can use only the first stage solution which is deterministic (does not depend on future observations) and update it as new observations become available - this is a rolling horizon approach. Such a rolling horizon approach requires resolving the corresponding multistage problem at every stage as new realization of the data becomes available.

"Any model is wrong but some are useful".

From a modeling point of view it is natural to assume that the random data process has a *continuous* distribution. We refer to such model as "true" or "continuous".

Simplifying assumption: we assume that the data process is stagewise independent. That is random vector ξ_{t+1} is independent of $\xi_{[t]} = (\xi_1, ..., \xi_t)$, $t = 1, ..., T-1$.

In some cases stagewise dependent problems can be reformulated in a stagewise independent form at the price of increasing number of state variables.

For example, suppose that only the right hand side vectors b_t are random and can be modeled as a (first order) autoregressive process

$$b_t = \mu + \Phi b_{t-1} + \varepsilon_t,$$

where μ and Φ are (deterministic) vector and regression matrix, respectively, and the error process ε_t, $t = 1, ..., T$, is stagewise independent. The corresponding feasibility constraints can be written in terms of x_t and b_t as

$$B_t x_{t-1} + A_t x_t \leq b_t, \;\; \Phi b_{t-1} - b_t + \mu + \varepsilon_t = 0.$$

That is, in terms of decision variables (x_t, b_t) this becomes a linear multistage stochastic programming problem governed by the stagewise independent random process $\varepsilon_1, ..., \varepsilon_T$.

Discretization by Monte Carlo sampling

Independent of each other random samples $\xi_t^j = (c_t^j, B_t^j, A_t^j, b_t^j)$, $j = 1, ..., N_t$, of respective ξ_t, $t = 2, ..., T$, are generated and the corresponding scenario tree is constructed by connecting every ancestor node at stage $t-1$ with the same set of children nodes $\xi_t^1, ..., \xi_t^{N_t}$. In that way the stagewise independence is preserved in the generated scenario tree. We refer to the constructed problem as the Sample Average Approximation (SAA) problem.

The total number of scenarios of the SAA problem is given by the product $\mathcal{N} = \prod_{t=2}^{T} N_t$ and quickly becomes astronomically large with increase of the number of stages even for moderate values of sample sizes N_t.

If we measure computational complexity, of the "true" problem, in terms of the number of scenarios required to approximate true distribution of the random data process with a reasonable accuracy, the conclusion is rather pessimistic. In order for the optimal value and solutions of the SAA problem to converge to their true counterparts all sample sizes $N_2, ..., N_T$ should tend to infinity. Furthermore, available estimates of the sample sizes required for a first stage solution of the SAA problem to be ε-optimal for the true problem, with a given confidence (probability), sums up to a number of scenarios which grows as $O(\varepsilon^{-2(T-1)})$ with decrease of the error level $\varepsilon > 0$. This indicates that from the point of view of the number of scenarios, complexity of multistage programming problems grows exponentially with increase of the number of stages.

Because of the exponential growth of the number of scenarios $\mathcal{N}$ it is hopeless to try to solve multistage stochastic programs by enumerating all scenarios. An alternative approach is suggested by the dynamic programming.

Dynamic programming equations for the SAA problem

Going backward in time the so-called *cost-to-go* functions are defined recursively for $t = T, ..., 2$, as follows

$$Q_t^j(x_{t-1}) = \inf_{\substack{B_t^j x_{t-1} + A_t^j x_t = b_t^j \\ x_t \geq 0}} \left\{ (c_t^j)^{\mathsf{T}} x_t + \mathcal{Q}_{t+1}(x_t) \right\},$$

$j = 1, ..., N_t$, with $\mathcal{Q}_{T+1}(\cdot) \equiv 0$ and

$$\mathcal{Q}_{t+1}(x_t) = \frac{1}{N_{t+1}} \sum_{j=1}^{N_{t+1}} Q_{t+1}^j(x_t).$$

At the first stage the following problem should be solved

$$\operatorname*{Min}_{x_1} c_1^{\mathsf{T}} x_1 + \mathcal{Q}_2(x_1) \text{ s.t. } A_1 x_1 = b_1,\ x_1 \geq 0.$$

The optimal value of the first stage problem gives the optimal value of the corresponding multistage problem.

A policy $\bar{x}_t = \bar{x}_t(\xi_{[t]})$, $t = 1, \ldots, T$, is *optimal* if $\bar{x}_1$ is an optimal solution of the first stage problem and for $t = 2, \ldots, T$, it holds for every realization of the data process that

$$\bar{x}_t(\xi_{[t]}) \in \arg \min_{\substack{B_t \bar{x}_{t-1} + A_t x_t = b_t \\ x_t \geq 0}} \left\{ c_t^{\mathsf{T}} x_t + \mathcal{Q}_{t+1}(x_t) \right\}.$$

In the dynamic programming formulation the problem is reduced to solving a sequence of finite dimensional problems, indexed by t and depending on $\xi_{[t]}$. In fact, because of the stagewise independence, $\bar{x}_t$ is a function of $\bar{x}_{t-1}$ and ξ_t.

For linear programs the cost-to-go (value) functions $Q_t^j(x_{t-1})$ and $\mathcal{Q}_{t+1}(x_t)$ are *convex*.

Because of the stagewise independence condition, the cost-to-go functions $\mathcal{Q}_{t+1}(x_t)$ are functions of x_t and do not depend on the data process ξ_t.

Curse of dimensionality

One of the main difficulties in solving the dynamic programming equations is how to represent the cost-to-go functions in a computationally feasible way.

For dimension of x_t say greater than 3 and large number of stages it is practically impossible to solve the dynamic programming equations with high accuracy. Several alternatives were suggested in recent literature.

Approximate dynamic programming

Basic idea is to approximate the cost-to-go functions by a class of computationally manageable functions. Since functions $\mathcal{Q}_t(\cdot)$ are convex it is natural to approximate these functions by piecewise linear functions given by maximum of cutting hyperplanes.

Stochastic Dual Dynamic Programming (SDDP) method (Pereira and Pinto, 1991).

For trial decisions $\bar{x}_t$, $t = 1, ..., T-1$, at the backward step of the SDDP algorithm, piecewise linear approximations $\mathfrak{Q}_t(\cdot)$ of the cost-to-go functions $\mathcal{Q}_t(\cdot)$ are constructed by solving problems

$$\operatorname*{Min}_{x_t \in \mathbb{R}^{n_t}} (c_t^j)^{\mathsf{T}} x_t + \mathfrak{Q}_{t+1}(x_t) \text{ s.t. } B_t^j \bar{x}_{t-1} + A_t^j x_t = b_t^j, \ x_t \geq 0,$$

$j = 1, ..., N_t$, and their duals, going backward in time $t = T, ..., 1$.

Denote by v^0 and $\widehat{v}_N$ the respective optimal values of the true and SAA problems.

By construction

$$\mathcal{Q}_t(\cdot) \geq \mathfrak{Q}_t(\cdot),\ t = 2, ..., T.$$

Therefore the optimal value of

$$\operatorname*{Min}_{x_1 \in \mathbb{R}^{n_1}} c_1^{\mathsf{T}} x_1 + \mathfrak{Q}_2(x_1) \ \text{ s.t. } A_1 x_1 = b_1,\ x_1 \geq 0$$

gives a lower bound for the optimal value $\widehat{v}_N$ of the SAA problem.

We also have that

$$v^0 \geq \mathbb{E}[\widehat{v}_N].$$

Therefore *on average* $\widehat{v}_N$ is also a lower bound for the optimal value of the true problem.

The approximate cost-to-go functions $\mathfrak{Q}_2, ..., \mathfrak{Q}_T$ and a feasible first stage solution $\bar{x}_1$ define a feasible policy. That is for a realization (sample path) $\xi_1, ..., \xi_T$ of the data process, $\bar{x}_t = \bar{x}_t(\xi_{[t]})$ are computed recursively in $t = 2, ..., T$ as a solution of

$$\underset{x_t}{\text{Min}}\, c_t^\top x_t + \mathfrak{Q}_{t+1}(x_t) \text{ s.t. } B_t \bar{x}_{t-1} + A_t x_t \leq b_t.$$

In the *forward step* of the SDDP algorithm M sample paths (scenarios) are generated and the corresponding $\bar{x}_t$, $t = 2, ..., T$, are used as trial points in the next iteration of the backward step.

It is essential for convergence of this algorithm that at each iteration in the forward step the paths (scenarios) are *resampled*, i.e., generated independently of the previous iteration.

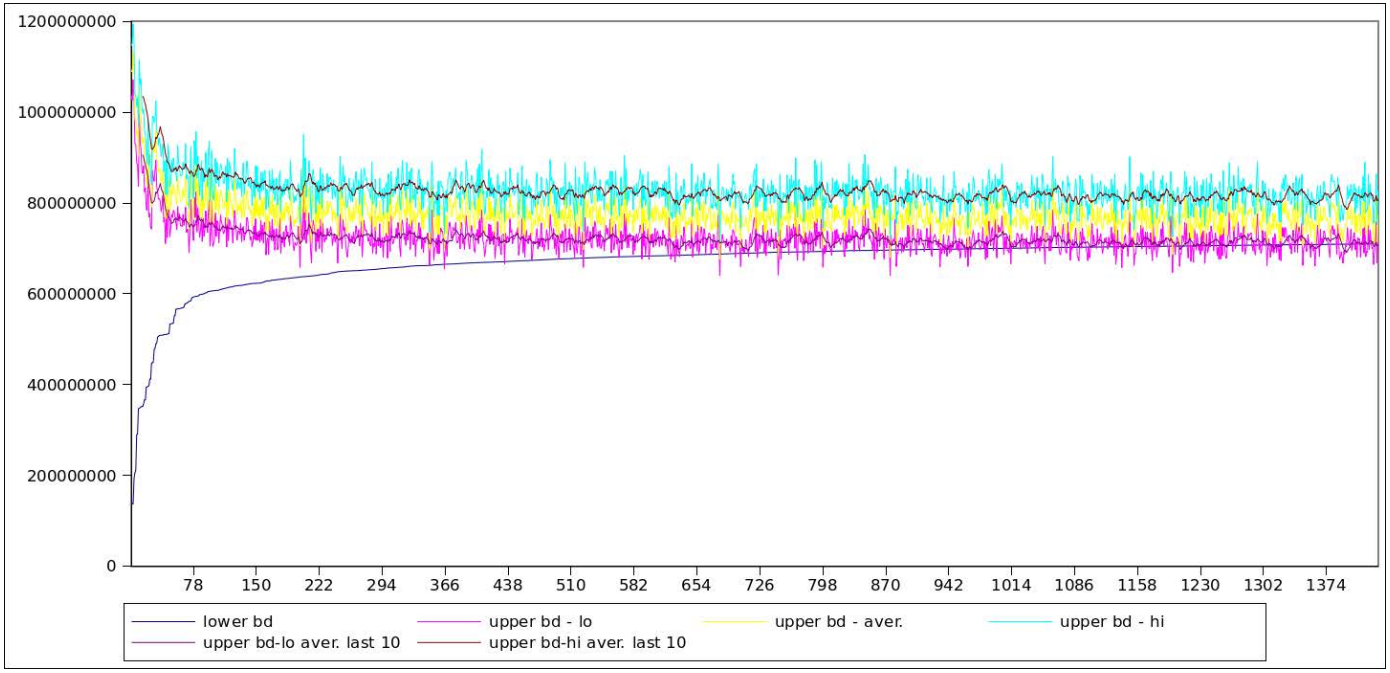
Note that the functions $\mathfrak{Q}_2, ..., \mathfrak{Q}_T$ and $\bar{x}_1$ define a feasible policy also for the *true* problem.

Convergence of the SDDP algorithm

It is possible to show that, under mild regularity conditions, the SDDP algorithm converges as the number of iterations go to infinity. That is, the computed optimal values and generated policies converge w.p.1 to their counterparts of the considered SAA problem. However, the convergence can be very slow and one should take such mathematical proofs very cautiously.

Moreover, it should be remembered that the SAA problem is just an approximation of the "true" problem. It is possible to show that, in a certain probabilistic sense, the SAA problem converges to the "true" problem as all sample sizes N_t, $t = 2, ..., T$, tend to infinity. Theoretical analysis and numerical experiments indicate that computational complexity of the SDDP algorithm grows fast with increase of the number of state variables.

Typical example of behavior of the lower and upper bounds produced by the SDDP algorithm for an SAA problem



120 stages, 1 cut per iteration and 95% upper bound estimation with 100 scenarios

Risk averse approach

How to control risk, i.e., to reduce chances of extreme costs, at every stage of the time process.

Value-at-Risk of a random outcome (variable) Z at level $\alpha \in (0,1)$:

$$\text{V@R}_\alpha(Z) = \inf\{t : F_Z(t) \geq 1-\alpha\},$$

where $F_Z(t) = \Pr(Z \leq t)$ is the cdf of Z. That is, $\text{V@R}_\alpha(Z)$ is the $(1-\alpha)$-quantile of the distribution of Z.

Note that $\text{V@R}_\alpha(Z) \leq c$ is equivalent to $\Pr(Z > c) \leq \alpha$. Therefore it could be a natural approach to impose constraints (chance constraints) of $\text{V@R}_\alpha(Z) \leq c$ for $Z = cost$, chosen constant c and significance level α at every stage of the process.

There are two problems with such approach. It is difficult to handle chance constraints numerically and could lead to infeasibility problems.

Average Value-at-Risk (also called *Conditional Value-at-Risk*)

$$\mathsf{AV@R}_\alpha(Z) = \inf_{t\in\mathbb{R}} \left\{ t + \alpha^{-1}\mathbb{E}[Z-t]_+ \right\}$$

Note that the minimum in the above is attained at $t^* = \mathsf{V@R}_\alpha(Z)$. If the cdf $F_Z(z)$ is continuous, then

$$\mathsf{AV@R}_\alpha(Z) = \mathbb{E}\Big[Z | Z \geq \mathsf{V@R}_\alpha(Z) \Big].$$

It follows that $\mathsf{AV@R}_\alpha(Z) \geq \mathsf{V@R}_\alpha(Z)$. Therefore the constraint $\mathsf{AV@R}_\alpha(Z) \leq c$ is a conservative approximation of the chance constraint $\mathsf{V@R}_\alpha(Z) \leq c$.

In the problem of minimizing expected cost $\mathbb{E}[Z]$ subject to the constraint $\mathsf{AV@R}_\alpha(Z) \leq c$, we impose an infinite penalty for violating this constraint. This could result in infeasibility of the obtained problem. Instead we can impose a finite penalty and consider problem of minimization of $\mathbb{E}[Z] + \kappa \mathsf{AV@R}_\alpha(Z)$ for some constant $\kappa > 0$. Note that this is equivalent to minimization of $\rho(Z)$, where

$$\rho(Z) = (1 - \lambda)\mathbb{E}[Z] + \lambda \mathsf{AV@R}_\alpha(Z)$$

for $\lambda \in (0, 1)$ and $\kappa = \frac{\lambda}{1-\lambda}$.

This leads to the following (nested) formulation of risk averse multistage problem.

$$\begin{aligned}\operatorname*{Min}_{A_1x_1\leq b_1} \quad & c_1^{\mathsf{T}}x_1+\rho_{2|\xi_1}\Big[\inf_{\substack{B_2x_1+A_2x_2=b_2\\ x_2\geq 0}} c_2^{\mathsf{T}}x_2+\ldots \\ & +\rho_{T-1|\xi_{[T-2]}}\Big[\inf_{\substack{B_{T-1}x_{T-2}+A_{T-1}x_{T-1}=b_{T-1}\\ x_{T-1}\geq 0}} c_{T-1}^{\mathsf{T}}x_{T-1} \\ & +\rho_{T|\xi_{[T-1]}}\big[\inf_{\substack{B_Tx_{T-1}+A_Tx_T=b_T\\ x_T\geq 0}} c_T^{\mathsf{T}}x_T\big]\Big]\Big],\end{aligned}$$

with

$$\rho_{t|\xi_{[t]}}(\cdot):=(1-\lambda)\mathbb{E}_{|\xi_{[t]}}[\cdot]+\lambda\mathsf{AV@R}_{\alpha|\xi_{[t]}}(\cdot)$$

being conditional analogue of $\rho(\cdot)$.

We can write the risk averse multistage programming problem as

$$\begin{array}{cl} \underset{x_1, x_2(\cdot), \ldots, x_T(\cdot)}{\text{Min}} & \bar{\rho}\Big[F_1(x_1) + F_2(x_2(\xi_{[2]}), \xi_2) + \cdots + F_T\left(x_T(\xi_{[T]}), \xi_T\right)\Big] \\ \text{s.t.} & x_1 \in \mathcal{X}_1,\ x_t(\xi_{[t]}) \in \mathcal{X}_t(x_{t-1}(\xi_{[t-1]}), \xi_t),\ t = 2, \ldots, T, \end{array}$$

where $F_t(x_t, \xi_t) = c_t^{\mathsf{T}} x_t$ and

$$\mathcal{X}_t(x_{t-1}, \xi_t) = \{x_t : B_t x_{t-1} + A_t x_t = b_t,\ x_t \geq 0\}.$$

$$\begin{aligned} \bar{\rho}(Z_1 + \ldots + Z_T) &= \rho_{|\xi_1}\Big(\rho_{|\xi_{[2]}}\Big(\cdots \rho_{|\xi_{[T-1]}}(Z_1 + \ldots + Z_T)\Big)\Big) \\ &= Z_1 + \rho_{|\xi_1}\Big(Z_2 + \rho_{|\xi_{[2]}}\Big(+ \cdots \rho_{|\xi_{[T-1]}}(Z_T)\Big)\Big) \end{aligned}$$

is the corresponding composite risk measure. The optimization is performed over (nonanticipative) policies $x_1, x_2(\xi_{[2]}), \ldots, x_T(\xi_{[T]})$ satisfying the feasibility constraints.

With some modifications the SDDP algorithm can be applied to the above multistage problem. Assuming the stagewise independence, the dynamic programming equations for the adaptive risk averse problem take the form

$$Q_t\left(x_{t-1},\xi_t\right) = \inf_{x_t\in\mathbb{R}^{n_t}} \left\{c_t^{\mathsf{T}} x_t + \mathcal{Q}_{t+1}(x_t) : B_t x_{t-1} + A_t x_t = b_t,\ x_t \geq 0\right\},$$

$t = T, ..., 2$, where $\mathcal{Q}_{T+1}(\cdot) \equiv 0$ and

$$\mathcal{Q}_{t+1}\left(x_t\right) := \rho_{t+1|\xi_{[t]}}\left[Q_{t+1}\left(x_t,\xi_{t+1}\right)\right].$$

Since ξ_{t+1} is independent of $\xi_{[t]}$, the cost-to-go functions $\mathcal{Q}_{t+1}\left(x_t\right)$ do not depend on the data process. In order to apply the backward step of the SDDP algorithm we only need to know how to compute subgradients of the cost-to-go functions.

The value of this problem corresponds to the total objective

$$\begin{aligned}\bar{\rho}(Z_1 + ... + Z_T) &= \rho_{|\xi_{[1]}}\Big(\cdots \rho_{|\xi_{[T-1]}}(Z_1 + ... + Z_T)\Big) \\ &= Z_1 + \rho_{|\xi_{[1]}}\Big(Z_2 + \cdots + \rho_{|\xi_{[T-1]}}(Z_T)\Big)\end{aligned}$$

The dynamic programming equations of the risk averse formulation of the SAA program take the form

$$Q_t^j(x_{t-1}) = \inf_{x_t}\Big\{(c_t^j)^{\mathsf{T}}x_t + \mathcal{Q}_{t+1}(x_t) : B_t^j x_{t-1} + A_t^j x_t = b_t^j,\ x_t \geq 0\Big\},$$

$j = 1, ..., N_t$, $t = T, \ldots, 2$, and

$$\mathcal{Q}_{t+1}(x_t) = \rho\left(Q_{t+1}^1(x_t), ..., Q_{t+1}^{N_{t+1}}(x_t)\right),$$

with $\mathcal{Q}_{T+1}(\cdot) \equiv 0$ and the first stage problem

$$\operatorname*{Min}_{A_1 x_1 \leq b_1} c_1^{\mathsf{T}}x_1 + \rho\left(Q_2^1(x_1), ..., Q_2^{N_2}(x_1)\right).$$

There are formulas for construction of cuts in the backward step of the SDDP algorithm. In the forward step trial points are generated in the same way as in the risk neutral case.

Remarks

Unfortunately there is no easy way for evaluating value of the risk objective of generated policies, and hence constructing a corresponding upper bound. Some suggestions were made in the recent literature. However, in larger problems the optimality gap (between the upper and lower bounds) never approaches zero in any realistic time. Therefore stopping criteria based on stabilization of the lower bound (and may be optimal solutions) could be reasonable. Also it should be remembered that there is no intuitive interpretation for the risk objective $\bar{\rho}(cost)$ of the total cost. Rather the goal is to control risk at *every* stage of the process.

Time consistency of stochastic programming problems

Consider a multiperiod stochastic program

$$\begin{array}{ll} \underset{x_1, x_2(\cdot), \ldots, x_T(\cdot)}{\text{Min}} & \varrho\Big(F_1(x_1), F_2(x_2(\xi_{[2]}), \xi_2), \ldots, F_T\left(x_T(\xi_{[T]}), \xi_T\right)\Big) \\ \text{s.t.} & x_1 \in \mathcal{X}_1,\ x_t(\xi_{[t]}) \in \mathcal{X}_t(x_{t-1}(\xi_{[t-1]}), \xi_t),\ t = 2, \ldots, T, \end{array}$$

where $\varrho : \mathcal{Z}_1 \times \mathcal{Z}_2 \times \cdots \times \mathcal{Z}_T \to \mathbb{R}$ is a multiperiod risk measure.

Is it time consistent? For example is

$$\begin{array}{ll} \underset{x_1, x_2(\cdot), \ldots, x_T(\cdot)}{\text{Min}} & \mathsf{AV@R}_\alpha\Big(F_1(x_1) + \cdots + F_T\left(x_T(\xi_{[T]}), \xi_T\right)\Big) \\ \text{s.t.} & x_1 \in \mathcal{X}_1,\ x_t(\xi_{[t]}) \in \mathcal{X}_t(x_{t-1}(\xi_{[t-1]}), \xi_t),\ t = 2, \ldots, T, \end{array}$$

time consistent? Note that for $\alpha \in (0, 1)$,

$$\mathsf{AV@R}_\alpha(\cdot) \neq \mathsf{AV@R}_{\alpha|\xi_1}\Big(\mathsf{AV@R}_{\alpha|\xi_{[2]}}\Big(\cdots \mathsf{AV@R}_{\alpha|\xi_{[T-1]}}(\cdot)\Big)\Big).$$