SDDP for multistage stochastic programs: Preprocessing via scenario reduction

Jitka DUPAČOVÁ and Václav KOZMÍK

Dept. of Probability and Math. Statistics, Charles University, Prague, Czech Republic

Abstract:
Even with recent enhancements, computation times for large-scale multistage problems with risk-averse objective functions can be very long. Therefore, preprocessing via scenario reduction could be considered as a way to significantly improve the overall performance. Stage-wise backward reduction of single scenarios applied to a fixed branching structure of the tree is a promising tool for efficient algorithms like SDDP. We provide computational results which show an acceptable precision of the results for the reduced problem and a substantial decrease of the total computation time.

Keywords and phrases: Multistage stochastic programs, Stochastic dual dynamic programming, multiperiod CVaR, scenario reduction

1 Multistage stochastic programs

When formulating multistage stochastic programs an important requirement is that decisions must be nonanticipative, i.e., in any stage of the decision process the decisions cannot depend on the future observations and decisions, whereas the past information as well as the knowledge of the probability distribution of the data process can be exploited.

1.1 Risk-neutral multistage stochastic programs

Let us first formulate an expectation-based linear multistage stochastic program with a given horizon T and with a fixed sequence of times, stages, at which decisions will be made. We suppose that the model has random parameters in stages $t = 2, \ldots, T$, denoted $\boldsymbol{\xi}_t$, which contain cost coefficients $\boldsymbol{c}_t$, constraint matrix $\boldsymbol{A}_t$, recourse matrix $\boldsymbol{B}_t$ and constraint coefficients $\boldsymbol{b}_t$. All random elements are governed by a known, or well-estimated, distribution. The parameters of the first stage, $\boldsymbol{\xi}_1 = (\boldsymbol{c}_1, \boldsymbol{A}_1, \boldsymbol{b}_1)$, are assumed to

be known when we make decision $\boldsymbol{x}_1$, but only a probability distribution governing future realizations, $\boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_T$, is known. The realization of $\boldsymbol{\xi}_2$ is known when decisions $\boldsymbol{x}_2$ must be made and so on up to stage T. The term $\boldsymbol{B}_t\boldsymbol{x}_{t-1}$ captures the state of the system. For example, in an asset allocation model, the state could be the total value of the assets in our portfolio. The components $\boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_T$ of $\boldsymbol{\xi}$ and the decisions $\boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$ are assumed to be random vectors, not necessarily of the same dimension, defined on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$, while $\boldsymbol{\xi}_1$ is deterministic and $\boldsymbol{x}_1$ is a nonrandom vector-valued variable. The sequence of decisions and observations is

$$\boldsymbol{x}_1, \boldsymbol{\xi}_2, \boldsymbol{x}_2(\boldsymbol{x}_1, \boldsymbol{\xi}_2), \ldots, \boldsymbol{x}_T(\boldsymbol{x}_{T-1}, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_T). \tag{1}$$

In a mathematical way, let $\mathcal{F}_t \subseteq \mathcal{F}$ be the σ-field generated by the projection $\Pi_t\boldsymbol{\xi} = \boldsymbol{\xi}_{[t]} := (\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_t)$ of the stochastic data process $\boldsymbol{\xi}$ that includes data up to stage t, $\mathcal{F}_1 = \{\emptyset, \Omega\}$ is the trivial σ-field. The dependence of the t-th stage decision $\boldsymbol{x}_t$ only on the available information means that $\boldsymbol{x}_t$ is $\mathcal{F}_t$-measurable. Similarly we let $\Pi_t\boldsymbol{x} = \boldsymbol{x}_{[t]} := (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)$ denote the sequence of decisions at stages $1, \ldots, t$, $\mathbb{P}$ the probability distribution of $\boldsymbol{\xi}$, $\mathbb{P}_t$ denotes the marginal probability distribution of $\boldsymbol{\xi}_t$, and $\mathbb{P}_t\left[\cdot|\boldsymbol{\xi}_{[t-1]}\right]$, $t = 2, , \ldots, T$, its conditional probability distribution.

In applications one mostly approximates the true probability distribution $\mathbb{P}$ of $\boldsymbol{\xi}$ by a discrete probability distribution carried by a finite number of atoms (scenarios), say, $\boldsymbol{\xi}^1, \ldots, \boldsymbol{\xi}^K$. They are organized in form of a scenario tree and in principle, the optimal policy can be obtained by solving a large deterministic program. See e.g. [19] for details and more general cases. An illustrative scheme of a scenario tree can be found in Figure 1.

The first stage decisions consist of all decisions that have to be selected before further information is revealed whereas the second stage decisions are allowed to adapt to this information, etc. In each of the stages, the decisions are limited by constraints that may depend only on the previous decisions and observations. *Stages do not have to coincide with the timing of observations of the random parameters, they should rather correspond to steps in the decision process.* Besides that, time spans between stages do not have to be equal, which facilitates effective representation of problems with far horizon. The decision process (1) has a random outcome, represented by cost $f(\boldsymbol{x}, \boldsymbol{\xi})$, and the basic goal is to find a nonanticipative decision or policy $\boldsymbol{x}(\boldsymbol{\xi})$ which minimizes the expectation $\mathbb{E}_{\mathbb{P}}\left[f(\boldsymbol{x}, \boldsymbol{\xi})\right]$ and satisfies all prescribed constraints. The optimal decision will be denoted $\boldsymbol{x}^*$. If there

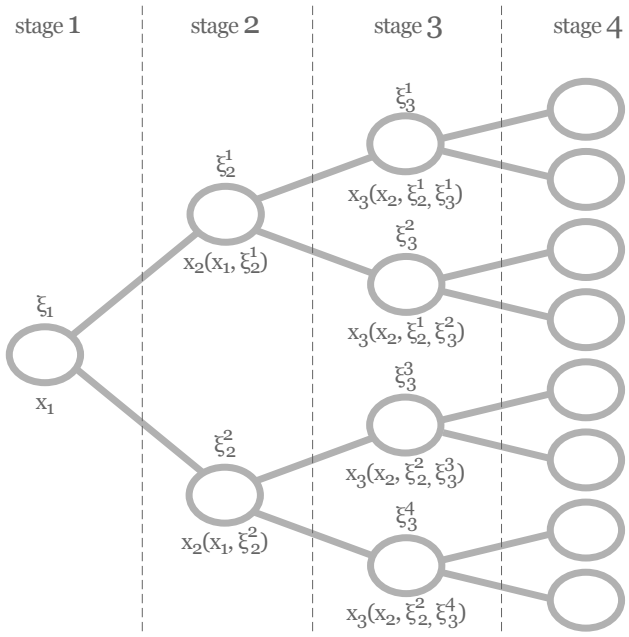

Figure 1: Scenario tree with associated decisions

are more optimal solutions, $\boldsymbol{x}^*$ is used to represent any member of the set of optimal solutions.
The nested form of the multistage stochastic linear program (MSLP) resembles the backward recursion of stochastic dynamic programming with an additive overall cost function:

$$\min_{\boldsymbol{x}_1 \in \mathcal{X}_1} \boldsymbol{c}_1^\top \boldsymbol{x}_1 + \mathbb{E}_{\mathbb{P}} \left[Q_2(\boldsymbol{x}_1, \boldsymbol{\xi}_{[2]}) \right] \text{ with } \mathcal{X}_1 := \{ \boldsymbol{x}_1 | \, \boldsymbol{A}_1 \boldsymbol{x}_1 = \boldsymbol{b}_1, \, \boldsymbol{x}_1 \geq 0 \}, \quad (2)$$

and $Q_t(\boldsymbol{x}_{t-1}, \boldsymbol{\xi}_{[t]})$, $t = 2, \ldots, T$, defined recursively as

$$Q_t(\boldsymbol{x}_{t-1}, \boldsymbol{\xi}_{[t]}) = \min_{\boldsymbol{x}_t} \boldsymbol{c}_t(\boldsymbol{\xi}_{[t-1]})^\top \boldsymbol{x}_t + \mathbb{E}_{\mathbb{P}_{t+1}[\cdot | \boldsymbol{\xi}_{[t]}]} \left[Q_{t+1}(\boldsymbol{x}_t, \boldsymbol{\xi}_{[t+1]}) \right] \quad (3)$$

subject to constraints $\boldsymbol{x}_t \in \mathcal{X}_t(\boldsymbol{x}_{t-1}, \boldsymbol{\xi}_{[t]})$, e.g.

$$\boldsymbol{A}_t(\boldsymbol{\xi}_{[t-1]}) \boldsymbol{x}_t = \boldsymbol{b}_t(\boldsymbol{\xi}_{[t-1]}) - \boldsymbol{B}_t(\boldsymbol{\xi}_{[t-1]}) \boldsymbol{x}_{t-1}, \, \boldsymbol{x}_t \geq 0 \, \text{a.s.},$$

and $Q_{T+1}(\cdot)$ is explicitly given, e.g. $Q_{T+1}(\cdot) \equiv 0$.
Matrices $\boldsymbol{A}_t$ are of a fixed (m_t, n_t) type, and the remaining vectors and matrices are of consistent dimensions. For the first stage, known values of

all elements of $\boldsymbol{c}_1, \boldsymbol{A}_1, \boldsymbol{b}_1$ are assumed, and the main decision variable is $\boldsymbol{x}_1$ that corresponds to the first stage. The first stage problem (2) has the form of the expectation-type stochastic program with the set of feasible decisions *independent* of $\mathbb{P}$.
One can rewrite (2)–(3) briefly as

$$\min_{\boldsymbol{x}_1} \boldsymbol{c}_1^\top \boldsymbol{x}_1 + \mathbb{E}\left[\min_{\boldsymbol{x}_2} \boldsymbol{c}_2(\boldsymbol{\xi}_{[1]})^\top \boldsymbol{x}_2 + \mathbb{E}\left[\cdots + \mathbb{E}\left[\min_{\boldsymbol{x}_T} \boldsymbol{c}_T(\boldsymbol{\xi}_{[T-1]})^\top \boldsymbol{x}_T\right]\right]\right] \quad (4)$$

with corresponding conditional expectations as in (3) and subject to linear constraints $\boldsymbol{x}_t \in \mathcal{X}_t(\boldsymbol{x}_{t-1}, \boldsymbol{\xi}_{[t]})$, $t = 1, \ldots, T$, on decision variables. The set of all feasible decisions $\mathcal{X}(\boldsymbol{\xi})$ is given by

$$\left\{\boldsymbol{x} \in \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_T} : \boldsymbol{x}_1 \in \mathcal{X}_1,\ \boldsymbol{x}_2 \in \mathcal{X}_2(\boldsymbol{x}_1, \boldsymbol{\xi}_{[2]}),\ \ldots\ , \boldsymbol{x}_T \in \mathcal{X}_T(\boldsymbol{x}_{T-1}, \boldsymbol{\xi}_{[T]})\right\}.$$

Constraints involving random elements are supposed to hold almost surely and for simplicity we will assume that *all optimal solutions exist*. A common assumption of *relatively complete recourse* requires that for each decision of the stage t, there exists some feasible decision for the stage $t+1$. Moreover, we will suppose that all conditional expectations exist. In the case of *stage-wise independence* the conditional probability distributions boil down to marginal distributions $\mathbb{P}_t$ of $\boldsymbol{\xi}_t$.
An example of a more complicated stochastic program could be an *expectation-based convex stochastic program*. The cost fuction $f(\boldsymbol{x}, \boldsymbol{\xi})$ is supposed to be an inf-compact convex normal integrand whose finite expectation exists and the set of feasible decisions $\mathcal{X}(\boldsymbol{\xi})$ should be closed, convex-valued, nonanticipative and uniformly bounded mapping, i.e. the assumption of relatively complete recourse.
Recently, there have been many enhancements in algorithms which exploit a special structure of the problem, such as the Stochastic Dual Dynamic Programming (SDDP), see Section 2. For large scale problems, computation times can be still very long. A natural idea is to reduce the solved problem by a suitable preprocessing. In Section 3 we shall present an extension of the scenario reduction method suggested in [2] for two-stage stochastic linear programs to multistage problems with a stage-independent structure.

1.2 Risk-averse multistage stochastic programs

Maximization of expected gains or minimization of expected losses means to get decisions that are optimal on average while possible risks are neglected.

This need not be an acceptable goal. The present tendency is to spell out explicitly the concern for risk monitoring and control. There are various types of risk and the choice of a suitable risk definition depends on the context, on the decision maker's attitude, the company goals, etc.
To reflect risks in the stochastic programming formulation, it is necessary to quantify them. Both in theoretical considerations and in applications, rational properties of risk measures are requested. A risk measure is a functional which assigns a real value to the random outcome $f(\boldsymbol{x}, \boldsymbol{\xi})$. Similarly as the risk-neutral expected value criterion, risk measures ρ should not depend on individual realizations of $\boldsymbol{\xi}$, but they depend on decisions and probability distribution $\mathbb{P}$. Moreover, they should also reflect the structure of the filtration $\mathcal{F}_1 \subset \cdots \subset \mathcal{F}_t \cdots \subseteq \mathcal{F}$.
Similarly as in our previous paper [3] we shall focus on multiperiod extensions of conditional value at risk (CVaR) – a special case of polyhedral risk measures of [5] – using importance sampling to efficiently solve the model by SDDP; see Kozmík and Morton [9] for details. We are going to use scenario reduction method suggested in [2] which applies to this class of risk measures and it does not destroy the stage-independence property, which is essential for SDDP.
We introduce the following operator which forms a weighted sum of conditional expectation and risk associated with random loss Z :

$$\rho_{t,\boldsymbol{\xi}_{[t-1]}}[Z] = (1-\lambda_t)\,\mathbb{E}\left[Z \,\middle|\, \boldsymbol{\xi}_{[t-1]}\right] + \lambda_t\,\mathrm{CVaR}_{\alpha_t}\left[Z \,\middle|\, \boldsymbol{\xi}_{[t-1]}\right]. \tag{5}$$

We suppose $\lambda_t \in [0,1]$, with $\lambda_t = 0$ it covers the risk neutral problems, whereas $\lambda_t = 1$ puts emphasis on risk control only. The case of $\lambda_t = 0$ for $t < T$ and $\lambda_T \neq 0$ models importance of risk only at the final stage.
We can write the corresponding risk-averse linear multistage model with T stages in the following form:

$$\begin{aligned} \min_{\substack{\boldsymbol{A}_1\boldsymbol{x}_1=\boldsymbol{b}_1\\ \boldsymbol{x}_1\geq 0}} \boldsymbol{c}_1^\top \boldsymbol{x}_1 + \rho_{2,\boldsymbol{\xi}_{[1]}}\Bigg[\min_{\substack{\boldsymbol{A}_2\boldsymbol{x}_2=\boldsymbol{b}_2-\boldsymbol{B}_2\boldsymbol{x}_1\\ \boldsymbol{x}_2\geq 0}} \boldsymbol{c}_2^\top \boldsymbol{x}_2 + \cdots \\ \cdots + \rho_{T,\boldsymbol{\xi}_{[T-1]}}\left[\min_{\substack{\boldsymbol{A}_T\boldsymbol{x}_T=\boldsymbol{b}_T-\boldsymbol{B}_T\boldsymbol{x}_{T-1}\\ \boldsymbol{x}_T\geq 0}} \boldsymbol{c}_T^\top \boldsymbol{x}_T \right] \Bigg]. \end{aligned} \tag{6}$$

Note that the operators $\rho_{t,\boldsymbol{\xi}_{[t-1]}}$ are conditional with respect to the current state of the system, denoted by $\boldsymbol{\xi}_{[t-1]}$.

2 Stochastic dual dynamic programming

In general, multistage stochastic programs are very hard to solve and a discrete approximation of the true distribution $\mathbb{P}$ has to be built. This approximating distribution is represented by a scenario tree and we are able to form a single deterministic optimization problem, which considers all nodes of this tree. However, such problems are usually too large to be solved by the standard software and we need to exploit the special structure to obtain efficient algorithms. In particular, our risk-averse multistage programs have convex cost functions which can be approximated by polyhedral functions from below. We solve our models using the stochastic dual dynamic programming algorithm (SDDP) which originated in the work of Pereira and Pinto [11]. SDDP-style algorithms rely on a form of independence, mostly on the stagewise independence to provide good performance for problems with multiple stages. Otherwise, memory issues arise even for a modest number of stages. There are several suggestions how to weaken the inter-stage independence requirement; we refer to additive dependence models of [4], or to a Markov dependence exploited in [14].

The algorithm performs series of forward and backward iterations until a satisfactory solution is found, meaning that some stopping rule, given lower and upper bound, is fulfilled; see Kozmík and Morton [9] and references therein. Computation of the upper bound is straightforward in the risk-neutral case, but it has to be enhanced to handle the case of CVaR. We apply here the importance sampling procedure derived in Kozmík and Morton [9].

During a typical iteration of the SDDP algorithm, cuts have been accumulated at each stage. These cuts represent a piece-wise linear approximation of the future cost functions. On a forward pass we sample a number of linear paths through the tree. As we solve a sequence of problems along these forward paths, the cuts that have been accumulated so far are used to form decisions at each stage. The costs incurred along all the sampled forward paths through the tree can be used to estimate the expected cost of the current policy, thus providing an upper bound.

In the backward pass of the algorithm, we add cuts to the collection defining the current approximation of the future cost functions. We do this by solving programs corresponding to the descendant nodes of each node in the linear paths from the forward pass, except in the final stage T. To form a cut, we use the objective values and subgradients of the descendant nodes to calculate the subgradient of the future cost function. The cuts collected at any node

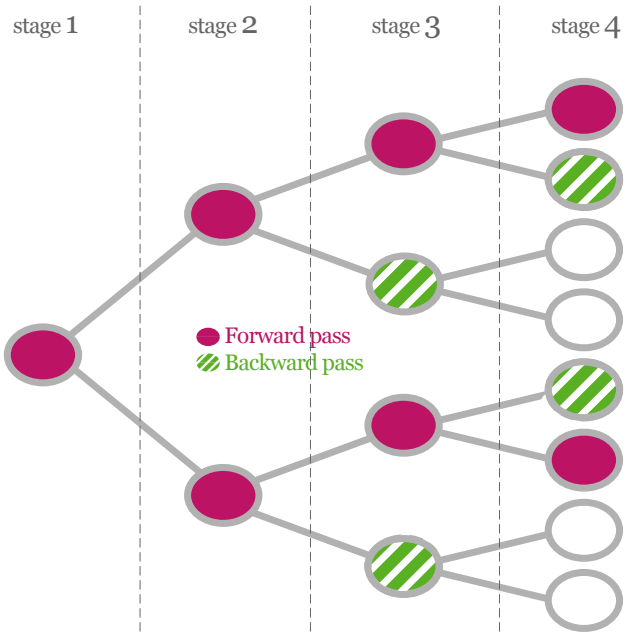

Figure 2: SDDP algorithm scheme

in stage τ apply to all the nodes in that stage, therefore only one set of cuts is maintained for each stage. This complexity reduction is possible because of the stage-wise independence assumption. The optimal value of the first-stage problem provides the lower bound. We refer to articles by Philpott and Matos [14] and Shapiro [18] for a thorough description and details about the SDDP algorithm.

3 Scenario reduction

There are several suggestions how to reduce the number of scenarios when solving stochastic programming problems, see for instance [7, 13, 17].
Here we apply scenario reduction technique based on quantitative stability results for problems of the form

$$\varphi(\mathbb{P}) := \min_{\boldsymbol{x}\in\mathcal{X}} \int_{\Omega} f(\boldsymbol{x}, \boldsymbol{\xi})\mathbb{P}(\mathrm{d}\boldsymbol{\xi}), \tag{7}$$

with:

- $\mathbb{P}$ a known probability distribution of $\boldsymbol{\xi}$ on $(\Omega, \mathcal{F})$, Ω closed subset of $\mathbb{R}^s$,
- a given, nonempty, closed set $\mathcal{X} \subset \mathbb{R}^n$ of decisions $\boldsymbol{x}$ which does not depend on $\mathbb{P}$, and at the same time, $\mathbb{P}$ does not depend on $\boldsymbol{x}$,
- preselected random objective f from $\mathcal{X} \times \Omega$ to the extended reals — loss or cost caused by decision $\boldsymbol{x}$ when scenario $\boldsymbol{\xi}$ occurs. As a function of $\boldsymbol{\xi}$, f is measurable for each fixed $\boldsymbol{x} \in \mathcal{X}$ and such that its expectation $\mathbb{E}_{\mathbb{P}}[f(\boldsymbol{x}, \boldsymbol{\xi})]$ exists; the structure of f may be quite complicated (e.g. for multistage problems). For convex $\mathcal{X}$, a frequent assumption is f lower semicontinuous and convex with respect to $\boldsymbol{x}$.

The success and applicability of quantitative stability results depends essentially on an appropriate choice of probability metrics d used to measure perturbations in $\mathbb{P}$. It should be closely tailored to the structure of the considered stochastic program and/or to the particular type of approximation of $\mathbb{P}$. The desired results are e.g. a Lipschitz property

$$d(\mathbb{P}, \mathbb{Q}) < \eta \quad \Longrightarrow \quad |\varphi(\mathbb{P}) - \varphi(\mathbb{Q})| < K\eta \tag{8}$$

and possibly also Lipschitz property of the Hausdorff distance of the corresponding solution sets or sets of ε-optimal solutions, see Römisch [16].
In context of scenario reduction one assumes that $\mathbb{P}$ is discrete probability distribution carried by a finite number of scenarios and $\mathbb{Q}$ is discrete probability distribution carried by a subset of atoms of $\mathbb{P}$.

3.1 Scenario reduction for two stage SLP

Under relatively complete fixed recourse and dual feasibility, the recommended class of probability metrics is based on Kantorovich-Rubinstein functional with continuous function $c : \mathbb{R}^s \times \mathbb{R}^s \to \mathbb{R}^1_+$ and a probability distribution η on $\Omega \times \Omega$:

$$d_c(\mathbb{P}, \mathbb{Q}) = \inf \left\{ \int_{\Omega \times \Omega} c(\boldsymbol{\xi}, \tilde{\boldsymbol{\xi}}) \eta(\mathrm{d}\boldsymbol{\xi}, \mathrm{d}\tilde{\boldsymbol{\xi}}) : \right.$$
$$\left. \eta(B \times \Omega) - \eta(\Omega \times B) = \mathbb{P}(B) - \mathbb{Q}(B) \, \forall B \in \mathcal{F} \right\}.$$

An example is $c(\boldsymbol{\xi}, \tilde{\boldsymbol{\xi}}) = ||\boldsymbol{\xi} - \tilde{\boldsymbol{\xi}}||$ with $||\cdot||$ a norm in $\mathbb{R}^s$. It applies to two stage problems with random right-hand sides and technology matrices.

Evaluation and optimization of CVaR is problem of this type; see Example 16 of Römisch [16].
For $\mathbb{P}$ a discrete probability distribution carried by a finite number of scenarios, say, $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_N$ with probabilities $p_i > 0, i = 1, \ldots, N$, $\sum_i p_i = 1$, scenario reduction means to construct discrete probability distribution $\mathbb{Q}$ by deleting scenarios $j \in J \subset \{1, \ldots, N\}$ so that $d(\mathbb{P}, \mathbb{Q})$ is small.
With $c_{ij} = c(\boldsymbol{\xi}_i, \boldsymbol{\xi}_j)$ for $i, j \in \{1, \ldots, N\}$, for given set J and $\mathbb{Q}$ carried by remaining scenarios, to get $d(\mathbb{P}, \mathbb{Q})$ means to solve transportation problem

$$D(J, q) := \min \left\{ \sum_{i,j} c_{ij}\eta_{ij} : \eta_{ij} \geq 0, \sum_{j \notin J} \eta_{ij} = p_i \ \forall i, \sum_i \eta_{ij} = q_j \ j \notin J \right\} \quad (9)$$

For fixed J, minimal distance D_J is obtained according to the optimal redistribution rule:

$$D_J = \min \left\{ D(J, q) : q_j \geq 0, \ \sum_{j \notin J} q_j = 1 \right\} = \sum_{i \in J} p_i \min_{j \notin J} c_{ij}.$$

Minimum is attained at $q_j^* = p_j + \sum_{i \notin J_j} p_i \ \forall j \notin J$, where $J_j := \{i \in J : j = j(i)\}$ and $j(i) \in \arg\min_{j \notin J} c_{ij}$. See Theorem 2 of [2].
The optimal choice of set J of a given cardinality is a difficult combinatorial problem. It can be derived for $\#J = 1$ or $N - 1$. Heuristic algorithms were constructed using bounds for $\min\{D_j : j \subset \{1, \ldots, N\}, \#J = k\}$; see [2].

3.1.1 Backward reduction

When deleting a single scenario, e.g. $J = \{l\}$, $D_J = p_l \min_{j \neq l} c_{lj}$ is minimized with respect to l. When minimum is attained for l^*, i.e. scenario $\boldsymbol{\xi}_{l^*}$ is deleted, the optimal redistribution rule allocates probability q_{l^*} to that $\boldsymbol{\xi}_j$ which is the nearest one to $\boldsymbol{\xi}_{l^*}$.

3.2 Scenario reduction for multistage stochastic programs

It turns out that quantitative stability studies for multistage stochastic programs cannot be in general based solely on the probability metrics

$$d_c(\mathbb{P},\mathbb{Q}) = \inf\left\{\int_{\Omega\times\Omega} c(\boldsymbol{\xi},\tilde{\boldsymbol{\xi}})\eta(\mathrm{d}\boldsymbol{\xi},\mathrm{d}\tilde{\boldsymbol{\xi}}) : \right.$$
$$\left. \eta(B\times\Omega) - \eta(\Omega\times B) = \mathbb{P}(B) - \mathbb{Q}(B)\,\forall B\in\mathcal{F}\right\}.$$

One should consider also differences in structure of the original and reduced scenario tree which can be quantified by the filtration distance, see e.g. [6]. However, as pointed out in [8], a good approximation can be obtained if $||\boldsymbol{\xi}-\tilde{\boldsymbol{\xi}}||$ is small and $\tilde{\boldsymbol{\xi}}$ is adapted to filtration $\mathcal{F}_t$ of $\boldsymbol{\xi}$. Such situation occures for stage-wise independent multistage stochastic programs with a fixed structure of stages. Hence, we focus on the stage-wise backward reduction of single scenarios or scenario groups applied to the fixed branching structure of the tree.

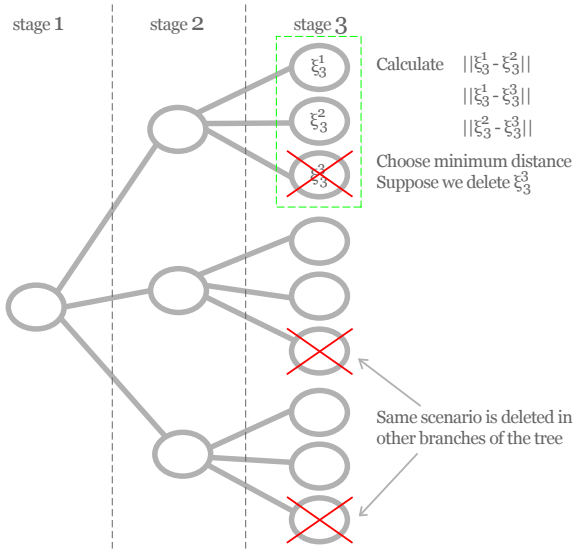

Figure 3: Scenario tree reduction

4 Scenario reduction scheme

We assume that for each stage $t = 2, \ldots, T$ there is a known (possibly continuous) distribution $\mathbb{P}_t$ of $\boldsymbol{\xi}_t$ and that we have a procedure to sample i.i.d. observations from this distribution. Given horizon T and a branching structure of the scenario tree, we obtain a single empirical distribution for each stage, denoted $\hat{\mathbb{P}}_t$, $t = 2, \ldots, T$, and the associated empirical scenario tree is interstage independent. The scenarios generated by this procedure are equally probable, but this is not required to apply SDDP or our reduction scheme.

Algorithm 1 (Sampling under interstage independence)

1. *Let $\boldsymbol{\xi}_1$ denote the deterministic first stage parameters.*
2. *Sample D_2 i.i.d. observations $\boldsymbol{\xi}_2^1, \ldots, \boldsymbol{\xi}_2^{D_2}$ from $\mathbb{P}_2$. These are the descendants of the first stage scenario (node) $\{\boldsymbol{\xi}_1\}$.*
3. *Sample D_3 i.i.d. observations $\boldsymbol{\xi}_3^1, \ldots, \boldsymbol{\xi}_3^{D_3}$ from $\mathbb{P}_3$, independent of those formed in stage 2. Let these denote the same set of descendant nodes for each of the $N_2 = D_2$ nodes $\{\boldsymbol{\xi}_1\} \times \left\{\boldsymbol{\xi}_2^1, \ldots, \boldsymbol{\xi}_2^{D_2}\right\}$.*

   $\vdots$

t. *Sample D_t i.i.d. observations $\boldsymbol{\xi}_t^1, \ldots, \boldsymbol{\xi}_t^{D_t}$ from $\mathbb{P}_t$, independent of those formed in stages $2, \ldots, t-1$. Let these denote the same set of descendant nodes for each of the $N_{t-1} = \prod_{i=2}^{t-1} D_i$ nodes $\{\boldsymbol{\xi}_1\} \times \left\{\boldsymbol{\xi}_2^1, \ldots, \boldsymbol{\xi}_2^{D_2}\right\} \times \cdots \times \left\{\boldsymbol{\xi}_{t-1}^1, \ldots, \boldsymbol{\xi}_{t-1}^{D_{t-1}}\right\}$.*

   $\vdots$

T. *Sample D_T i.i.d. observations $\boldsymbol{\xi}_T^1, \ldots, \boldsymbol{\xi}_T^{D_T}$ from $\mathbb{P}_T$, independent of those formed in stages $2, \ldots, T-1$. Let these denote the same set of descendant nodes for each of the $N_{T-1} = \prod_{i=2}^{T-1} D_i$ nodes $\{\boldsymbol{\xi}_1\} \times \left\{\boldsymbol{\xi}_2^1, \ldots, \boldsymbol{\xi}_2^{D_2}\right\} \times \cdots \times \left\{\boldsymbol{\xi}_{T-1}^1, \ldots, \boldsymbol{\xi}_{T-1}^{D_{T-1}}\right\}$.*

In the following algorithm, we apply the stage-wise backward reduction of single scenario until the desired number of scenarios R_t is reached.

Algorithm 2 (Backward multistage scenario reduction)

Suppose stage-wise independent scenario structure $\mathbb{P}_t$
For stage $t \in \{2, \ldots, T\}$ {
    Set $\mathbb{Q}_t = \mathbb{P}_t = \{\boldsymbol{\xi}_t^1, \ldots, \boldsymbol{\xi}_t^{N_t}\}$;
    While $|\mathbb{Q}_t| > R_t$
        Select $\boldsymbol{\xi}_t^k = \arg\min_l \, p_t^l \min_{j \neq l} ||\boldsymbol{\xi}_t^l - \boldsymbol{\xi}_t^j||$;
        Set $\boldsymbol{\xi}_t^l = \arg\min_{j \neq k} \, ||\boldsymbol{\xi}_t^k - \boldsymbol{\xi}_t^j||$;
        Set $\mathbb{Q}_t = \mathbb{Q}_t \setminus \{\boldsymbol{\xi}_t^k\}$;
        Set $p_t^l = p_t^l + p_t^k$;
    }
}

Although we stop short of a computational comparison, let us discuss similarities and differences between our reduction scheme and that based on the nested distance presented in [12]. The main distinction is that our procedure works with stage-independent trees, while the nested distance operates on any scenario tree with a general structure. However, we point out that fully general forms of interstage dependency lead to inherent computational intractability as even the memory requirements to store a general sampled scenario tree grow exponentially in the number of stages. While it may seem that some comparison might be done on stage-independent trees, this is not the case, because after application of the scenario reduction with nested distance, the resulting tree does not have to be stage-independent, even if we started with such property. That said, we believe that one of the advantages of our scheme is that we produce trees with stage-independent structure, which can be used in efficient algorithms like SDDP. Moreover, computation time of our reduction scheme grows only linearly with number of stages, while the reduction based on general nested distance would require exponential effort.

4.1 Numerical experiments

We present a simple asset allocation model without transaction costs and emphasize that our primary purpose is to illustrate the performance of the proposed reduction scheme as opposed to building a high-fidelity model for practical use. At stage t the decisions $\boldsymbol{x}_t$ denote the allocations, and $\boldsymbol{r}_t$

denotes gross return per stage; i.e., the ratio of the price at stage t to that in stage $t-1$. These represent the only random parameters in the model. Using the definition of conditional value at risk from [15],

$$\mathrm{CVaR}_\alpha\left[Z\right] = \min_u \left(u + \frac{1}{\alpha}\mathbb{E}\left[Z-u\right]_+ \right), \tag{10}$$

where $\left[\,\cdot\,\right]_+ \equiv \max\{\,\cdot\,, 0\}$, the nested CVaR model (6) without transaction costs specializes to:

$$\begin{aligned} Q_t(\boldsymbol{x}_{t-1}, \boldsymbol{\xi}_t) = \min_{\boldsymbol{x}_t, u_t} \quad & -\mathbf{1}^\top \boldsymbol{x}_t + \lambda_{t+1} u_t + \mathcal{Q}_{t+1}(\boldsymbol{x}_t, u_t) && \text{(11a)} \\ \text{s.t.} \quad & \mathbf{1}^\top \boldsymbol{x}_t = \boldsymbol{r}_t^\top \boldsymbol{x}_{t-1} && \text{(11b)} \\ & \boldsymbol{x}_t \geq 0, && \text{(11c)} \end{aligned}$$

where

$$\mathcal{Q}_{t+1}(\boldsymbol{x}_t, u_t) = \mathbb{E}\left[\left(1-\lambda_{t+1}\right) Q_{t+1}(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1}) + \frac{\lambda_{t+1}}{\alpha_{t+1}} \left[Q_{t+1}(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1}) - u_t \right]_+ \right]. \tag{12}$$

except that: in the first stage (i) the right-hand side of (11b) is instead equal to 1 and (ii) because $-\mathbf{1}^\top \boldsymbol{x}_1$ is then identically -1, we drop this constant from the objective function; (iii) we take $\mathcal{Q}_{T+1}(\cdot) \equiv 0$ and $\lambda_{T+1} \equiv 0$.

We used monthly price data of the most important assets traded on the Prague Stock Exchange, January 2009 to February 2012. We have fitted a multidimensional correlated log-normal distribution to the price ratios to obtain the original distribution $\mathbb{P}$. Scenario trees were constructed by sampling $\hat{\mathbb{P}}$ from this distribution, using the polar method for normal distribution sampling. The CVaR levels α_t were was always set to 5% and we evaluated the model with risk coefficients $\lambda_t = 0.5$.

We applied stage-wise backward reduction of single scenario on the 10-stage asset allocation problem. The original problem has been sampled with $1,000$ scenarios for each stage, replicating those scenarios to every node in the corresponding stage, which gives in total 10^{27} scenarios. It takes 24 hours to solve the original problem on a single 2 GHz thread, with a gap between lower and upper bound of 1.5%. We repeated every experiment for 5 times and conclude that for reduced problems, the gap between lower and upper bound remains the same, approximately 1.5%. In Table 1 we report differences between the lower bounds of the full problem and the problem with reduced scenarios.

scenarios per stage	total scenarios	opt. value gap	opt. sol'n distance	comput. time
500	$\sim 10^{24}$	-0.88%	0.1480	52%
250	$\sim 10^{21}$	-2.06%	0.1595	26%
100	$\sim 10^{18}$	-3.71%	0.2761	13%

Table 1: Performance of the scenario reduction scheme

We observe that the reduction in computation time is almost proportional to the reduction in the number of scenarios. When reducing the number of scenarios to one quarter, we are still getting results with acceptable precision and the computation time is four times lower.
Suppose you can only afford to compute problem with a limited number of scenarios. Scenario reduction is a useful technique even if you are sampling from a continuous distribution. Instead of sampling the prescribed number of scenarios, consider the case when we sample more scenarios and then apply our reduction technique to get back to the prescribed number of scenarios.

Algorithm 3 (Optimized sampling scheme)

1. *Sample with Algorithm 1 under interstage independence, $N_t > R_t$.*
2. *Apply backward scenario reduction under stage-wise independence (Algorithm 2) to get R_t scenarios for each stage.*
3. *Solve the problem with SDDP algorithm with importance sampling applied to the reduced scenario tree, see Section 2.*
4. *Validate the results, for example by assessing solution quality as in [1] or by the contamination technique, as in [3].*

We now compare results of Algorithm 3 with those for the standard sampling. We report results for two setups – 250 and 100 scenarios. In the first setup, we sample 250 scenarios directly, in the second setup, we first sample 1000 scenarios and then reduce their number to 250. Since the scenarios differ, we get solutions that do not match, but we can measure their stability. We report results for 5 iterations in Table 2 and conclude that decisions obtained using our optimized sampling scheme are more stable. The computation overhead associated with sampling of additional scenarios and running scenario reduction is usually negligible compared to the solution times of the multistage stochastic programs.

scenarios	reduction	avg. sol'n deviation
250	–	0.0475
1000	250	0.0279
100	–	0.0628
1000	100	0.0495

Table 2: Performance of the optimized sampling scheme

Conclusion

We have proposed a simple backward reduction scheme for stage-wise independent multistage problems, which allows to improve existing procedures with a relatively small effort. Our computational results show significant reduction of the computation time while keeping a good approximation of solutions. The proposed scenario reduction procedure has a low impact on computation times in optimization problems and it can be used to improve sampling procedures which provide inputs to the appropriate algorithms as well.

In our future research, we would like to focus on applications such as hydroelectric scheduling under inflow uncertainty, see for instance Oliviera et al. [10] for a case without optimization. Approximation techniques which reduce more scenarios in each step of the procedure may be an interesting improvement for our procedure as well. Moreover, we would like to apply scenario reduction to other types of stochastic programs such as mixed-integer programs or programs with chance constraints, we refer to Römisch [17] for a case of mixed-integer two-stage program.

Acknowledgements

The research was partly supported by the project of the Czech Science Foundation P/402/12/G097 'DYME/Dynamic Models in Economics'.

References

[1] BAYRAKSAN, G. and MORTON, D. P. (2009): *Assessing solution quality in stochastic programs via sampling.* In: OSKOOROUCHI, M.,

GRAY, P. and GREENBERG, H., eds. Tutorials in operations research, Hannover, Informs, pp. 102–122, ISBN 978-1-877640-24-7.

[2] DUPAČOVÁ, J., GRÖWE-KUSKA, N. and RÖMISCH, W. (2003): *Scenario reduction in stochastic programming: an approach using probability metrics*, Math. Program. vol. 95, pp. 493–511.

[3] DUPAČOVÁ, J. and KOZMÍK, V. (2015): *Structure of Risk-Averse Multistage Stochastic Programs*, OR Spectrum, vol. 37, pp. 559–582.

[4] INFANGER, G. and MORTON, D. P. (1996): *Cut sharing for multistage stochastic linear programs with interstage dependency*, Math. Program. vol. 75 pp. 241–256.

[5] EICHHORN, A. and RÖMISCH, W. (2005): *Polyhedral risk measures in stochastic programming*, SIAM J. Optimiz. vol. 16, pp. 69–95.

[6] HEITSCH, H., RÖMISCH, W. and STRUGAREK, C. (2006): *Stability of multistage stochastic programs*, SIAM J. Optimiz. vol. 17, pp. 511–525.

[7] HEITSCH, H. and RÖMISCH, W. (2009): *Scenario tree reduction for multistage stochastic programs*, Comput. Manag. Sci. vol. 6, pp. 117–133.

[8] HEITSCH, H. and RÖMISCH, W. (2009): *Scenario tree modeling for multistage stochastic programs*, Math. Program. vol. 118, pp. 371–406.

[9] KOZMÍK, V. and MORTON, D. (2015): *Evaluating policies in risk-averse multi-stage stochastic programming*, Math. Program. vol. 152, pp. 275–300.

[10] OLIVEIRA, W.L., SAGASTIZÁBAL, C., PENNA, D.D.J., MACEIRA, M.E.P. and DAMÁZIO, J.M. (2010): *Optimal Scenario Tree Reduction for Stochastic Streamflows in Power Generation Planning Problems*, Optimization Methods and Software vol. 25, pp. 917–936.

[11] PEREIRA, M. V. F. and PINTO, L. M. V. G. (1991): *Multi-stage stochastic optimization applied to energy planning*, Math. Program. vol. 52, pp. 359–375.

[12] PFLUG, G. CH. and PICHLER, A. (2012): *A Distance For Multistage Stochastic Optimization Models*, SIAM J. Optimiz. vol. 22,pp. 1–23.

[13] PFLUG, G. CH. and PICHLER, A. (2011): *Approximations for Probability Distributions and Stochastic Optimization Problems*. In: BERTOCCHI, M., CONSIGLI, G. and DEMPSTER, M. A. H., eds. Stochastic Optimization Methods in Finance and Energy, New York, Springer, ISBN 978-1-4419-9585-8.

[14] PHILPOTT, A. B. and DE MATOS, V. L. (2012): *Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion*, Eur. J. Oper. Res., vol. 218, pp. 470–483.

[15] ROCKAFELLAR, R. T. and URYASEV, S. (2002): *Conditional value at risk for general loss distributions*, J. Bank. Financ., vol. 26, pp. 1443–1471.

[16] RÖMISCH, W. (2003): *Stability of stochastic programming problems, Chapter 8*. In: RUSZCZYNSKI, A., SHAPIRO, A. (eds.), *Handbook on Stochastic Programming*, Elsevier, Amsterdam, pp. 483–554.

[17] RÖMISCH, W. (2009): *Scenario Reduction Techniques in Stochastic Programming*. In: WATANABE, O., ZEUGMANN, T. (eds.), *Stochastic Algorithms: Foundations and Applications, Lecture Notes in Computer Science* vol. 5792, Springer, Sapporo, pp. 1–14.

[18] SHAPIRO, A. (2011): *Analysis of stochastic dual dynamic programming method*, Eur. J. Oper. Res., vol. 209, pp. 63–72.

[19] SHAPIRO, A., DENTCHEVA, D. and RUSZCZYŃSKI, A. (2009): *Lectures on Stochastic Programming: Modeling and Theory*, Philadelphia, SIAM-Society for Industrial and Applied Mathematics, ISBN 978-1107025127.