Abstract. We provide an extensive review on stochastic dual dynamic programming (SDDP), as one of the state-of-the-art solution methods for multistage stochastic programs. Since introduced about 30 years ago for solving large-scale multistage stochastic linear programming problems in a hydrothermal context, SDDP has been applied to practical problems from several fields and is enriched by various improvements and enhancements to broader problem classes. We give a detailed introduction to SDDP, with special focus on its motivation and required assumptions. Then, we present and discuss in depth the existing enhancements as well as current research trends, which allow for an alleviation of those assumptions.

Key words. Stochastic Dual Dynamic Programming, multistage stochastic programming,

1. Stochastic Dual Dynamic Programming. In this section, we present the stochastic dual dynamic programming (SDDP) method. We start introducing the considered form of decision processes and multistage stochastic programming problems. Then, we explain nested Benders decomposition and derive SDDP as a sampling-based variant. In particular, we point out nine assumptions which are crucial for the presented SDDP method to work.

In later sections, we consider several enhancements and generalizations of SDDP with special focus on how these assumptions can be relaxed.

1.1. Problem Definition. We consider a multistage decision process where decisions \( x_t \) have to be taken over several stages \( t = 1, \ldots, T \) under some objective function and constraints. The data in this decision process can be subject to uncertainty, which is modeled by some probability space \((\Xi, F, \mathcal{P})\) with universe of outcomes \(\Xi\), sigma-algebra \(F\) and probability measure \(\mathcal{P}\) along with a real-valued random vector \(\xi: \Xi \to \mathbb{R}^p\). The uncertainty can also be interpreted stagewise by defining a stochastic process \((\xi_t)_{t=1}^T\) with random vectors \(\xi_t = (c_t, T_{t-1}, W_t, h_t) \in \Xi_t \subseteq \mathbb{R}^{p_t}\). For the first stage, the data is assumed deterministic, i.e., \(\Xi_1\) is a singleton. Then, \(\xi = (\xi_1, \ldots, \xi_T)\).

Note that, for simplicity, we do not introduce a separate symbol \(\omega_t\) for the realizations of \(\xi_t\). This distinction becomes clear from context.

At stage 1, the here-and-now decision \(x_1\) is taken to hedge against the uncertainty in the following stages. At those stages, recourse decisions \(x_t(\xi_{[t]}))\) can be taken under knowledge of the history of the data process up to time \(t\), which is denoted by \(\xi_{[t]} := (\xi_1, \ldots, \xi_t)\). Thus, the paradigm is that those decisions can be taken after the uncertainty corresponding to stage \(t\) has unfolded (so-called wait-and-see decisions or hazard-decision process), making \(x_t\) a function of \(\xi_{[t]}\). Note that \(x_t\) does only depend on realizations up to stage \(t\), but does not anticipate future events or decisions. Future events are only considered using distributional information. More formally, let \(\mathcal{F}_t\) denote the sigma-algebra generated by \(\xi_{[t]}\) and let those sigma-algebras form a filtration \(\mathcal{F}_1 \subseteq \mathcal{F}_2 \cdots \subseteq \mathcal{F}_T = \mathcal{F}\). Then \(x_t\) is assumed to be \(\mathcal{F}_t\)-measurable.

The decision process is depicted in Fig. 1, which is loosely based on the policy graphs introduced in [20].

A sequence of functions \((x_t(\xi_{[t]}))_{t=1}^T\) is called policy and provides a decision rule for all stages and any realization of the data process \(\xi = (\xi_1, \ldots, \xi_T)\). By the previous arguments, such policy is non-anticipative, modeling a sequence of nested conditional...
decisions. The aim is to determine an optimal policy.

The decision process can be modeled by a multistage stochastic programming problem \((P)\). For SDDP, the following assumptions for \((P)\) are standard.

**Assumption 1.** Problem \((P)\) is a multistage stochastic linear program (MSLP).

**Assumption 2.** Problem \((P)\) has a finite number \(T \in \mathbb{N}\) of stages.

**Assumption 3.** The constraints in problem \((P)\) have a block-diagonal structure, i.e., only consecutive stages can be linked by constraints.

**Assumption 4.** An optimal risk-neutral policy is to be determined.

Under Assumptions 1-4, \((P)\) can be formulated as follows:

\[
(P) \quad v^* := \min_{x_1} \begin{bmatrix} c_1^\top x_1 + \mathbb{E}_{\xi_2}(\xi_1) \left[ \min_{x_2} c_2(\xi_2)^\top x_2(\xi_2) + \mathbb{E}_{\xi_3}(\xi_2) \left[ \cdots \right. \right. \\
\left. \left. + \mathbb{E}_{\xi_T}(\xi_{T-1}) \left[ \min_{x_T} c_T(\xi_T)^\top x_T(\xi_T) \right] \cdots \right] \right] \end{bmatrix}
\]

s.t. \[
\begin{align*}
W_1 x_1 &= h_1 \\
T_2(\xi_2) x_1 + W_2(\xi_2) x_2(\xi_2) &= h_2(\xi_2) \quad \forall \xi_2 \in \xi_2 \text{ a.s.} \\
&\vdots \\
T_{T-1}(\xi_T) x_{T-1}(\xi_{T-1}) + W_T(\xi_T) x_T(\xi_T) &= h_T(\xi_T) \quad \forall \xi_{T-1} \in \xi_{T-1}, \xi_T \in \Xi_T | \xi_{T-1} \text{ a.s.} \\
x_1 &\geq 0 \\
x_t(\xi_t) &\geq 0 \quad \forall t = 1, \ldots, T, \xi_t \in \Xi_t | \xi_{t-1} \text{ a.s.}
\end{align*}
\]

with decision variables \(x_t \in \mathbb{R}^{n_t}\), vectors \(c_t \in \mathbb{R}^{n_t}\) and \(h_t \in \mathbb{R}^{m_t}\) and \((m_{t-1} \times n_t)\)-matrices \(T_t\) and \((m_t \times n_t)\)-matrices \(W_t\).

The block-diagonal structure of the constraints of \((P)\), see Assumption 3, is visualized in Fig. 2.

As stated before, some of (or all) data in \(\xi_t = (c_t, T_t, W_t, h_t)\) can be subject to uncertainty for \(t = 2, \ldots, T\), while the first stage is assumed to be deterministic with \(T_1 \equiv 0\). Since decisions are taken successively and a risk-neutral decision is targeted (see Assumption 4), conditional expectations are considered in the objective function. They are taken with respect to the filtration generated by \(\xi_t\).

Problem \((P)\) can be straightforwardly enhanced with deterministic polyhedral constraints \(x_t \in X_t, t = 1, \ldots, T\), and local decision variables (controls) \(y_t \in Y_t\), but for simplicity we focus on the decision variables and structural constraints which are important to illustrate SDDP.

A policy \((x_t(\xi_t))_{t=1}^T\) is called *feasible* if it satisfies the constraints in \((P)\) for almost every realization of the random data, i.e., with probability 1.

As the equality constraints in \((P)\) have a block-diagonal structure (see Assump-
For $t = T, \ldots, 2$, the dynamic programming equations are

$$Q_t(x_{t-1}, \xi_{[t-1]}, \xi_t) := \begin{cases} \min_{x_t} & c_t^\top (\xi_t)x_t + Q_{t+1}(x_t, \xi_t) \\ \text{s.t.} & W_t(\xi_t)x_t = h_t(\xi_t) - T_{t-1}(\xi_t)x_{t-1} \\ & x_t \geq 0 \end{cases}$$

where

$$Q_{T+1}(x_T, \xi_{[T]}) := E_{\xi_{[T]}} \left[ Q_{T+1}(x_T, \xi_{[T]}, \xi_{T+1}) \right]$$

and $Q_{T+1}(x_T) \equiv 0$. $Q_t(\cdot)$ is called value function and $Q_t(\cdot)$ is called expected value function, cost-to-go function, future cost function or recourse function. For the first stage, we obtain

$$v^* = \begin{cases} \min_{x_1} & c_1^\top x_1 + Q_2(x_1, \xi_{[1]}) \\ \text{s.t.} & W_1 x_1 = h_1 \\ & x_1 \geq 0 \end{cases}$$

Importantly, in the dynamic programming equations (1.1) and (1.3), the variables $x_t$ are deterministic, since for each subproblem (1.1) a fixed event $\xi_t$ is considered.

To ensure the existence of feasible solutions, different recourse assumptions are considered in stochastic programming. In the SDDP setting most commonly used is relatively complete recourse:

**Assumption 5.** Problem (P) has relatively complete recourse, i.e. for any $x_{t-1}$ feasible for stage $t-1$, the stage $t$-subproblem is feasible for any realization of $\xi_t$ almost surely.

Additionally, assuming boundedness and feasibility of (P), it is assured that problems (1.1) and (1.3) are bounded and feasible as well, i.e., that all $Q_t(\cdot), t = 1, \ldots, T$, are finite-valued.

Assumption 5 can often be ensured by introducing slack variables and penalizing them in the objective function with appropriate weights.

In case of a continuous random vector $\xi$, the solution of (P) requires the evaluation of (multidimensional) integrals in order to determine the expected values. Apart from some low-dimensional cases, this is computationally intractable. Thus, for now, we assume that the stochastic process has a finite number of realizations (called scenarios), which means that $\xi_t$ is a discrete and finite random variable for all $t = 1, \ldots, T$. Moreover, we require the specific distribution $F_\xi$ of $\xi$ to be known.
Assumption 6. The probability distribution $F_\xi$ of the process $(\xi_t)^T_{t=1}$ is known.

Assumption 7. There is a finite number of realizations of $\xi_t$ for all $t = 1, \ldots, T$.

Additionally, we assume that the probability distribution does not depend on the chosen policy. Otherwise, the stochastic optimization problems are called decision-dependent [45].

Assumption 8. The probability distribution $F_\xi$ of the data process $(\xi_t)^T_{t=1}$ does not depend on the considered policies.

Under these assumptions, the possible realizations of the data process $(\xi_t)^T_{t=1}$ can be represented in form of a finite scenario tree, as visualized in Fig. 3. We index the different paths $\xi^s$ through the scenario tree, i.e., the stage-$t$ scenarios, by $s \in S$.

![Scenario tree with 8 stage-$t$ scenarios $\xi^t$ with $\xi^2$ being highlighted.](image)

In earlier stages, several scenarios can be equal and, hence, share the same nodes in the scenario tree. For example, in Fig. 3, scenarios $\xi^1, \xi^2, \xi^3$ and $\xi^4$ share the same node on stage 2. To cope with this, we define by $S_t$ the set of all scenarios distinguishable for stage $t$. Additionally, by $\Delta(\xi^s_t)$ we denote the set of stage-$(t+1)$ successor scenarios of stage-$t$ scenario $\xi^s_t$.

To simplify such relations, an equivalent but different formulation of the dynamic programming equations is quite established in multistage stochastic programming, putting focus on the nodes of the scenario tree. Here, adjacency of nodes is used to relate subproblems to each other [11, 83, 85]. In this review, we stick to the first perspective though, as it is more common in the literature on SDDP.

We obtain the following useful properties for the value functions $Q_t(\cdot)$.

**Theorem 1.1 ([11]).** Let $h_t, T_{t-1}, c_t$ and $x_{t-1}$ be elements of some convex sets. Then, under Assumptions 1-5, for given $\xi_{[t-1]}$ and a given realization $\xi_t$, the value function $Q_t(\cdot, \xi_{[t-1]}, \xi_t)$

a) is piecewise linear and convex in $(h_t, T_{t-1})$,

b) is piecewise linear and concave in $c_t$,

c) is piecewise linear and convex in $x_{t-1}$.

The main idea here is that $h_t, T_{t-1}$ and $x_{t-1}$ do only appear in the right-hand side of problem (1.1). Therefore, the dual feasible set is independent of those elements.
and possesses finitely many extreme points. This assures piecewise linearity of $Q_t(\cdot)$, as known from parametric optimization. The convexity follows with the linearity (Assumption 1) and all vectors and matrices being part of convex sets.

Theorem 1.1 directly implies the piecewise linearity and convexity of $Q_t(\cdot,\xi_{[t-1]})$.

**Corollary 1.2** ([11]). **Under Assumption 7 and the premises of Theorem 1.1, for any history $\xi_{[t-1]}$ of the data process, the expected value function $Q_t(\cdot,\xi_{[t-1]})$ is piecewise linear and convex in $x_t$.**

### 1.2. Nested Benders Decomposition.

Even for a finite number of scenarios, problem $(P)$ becomes a large-scale linear programming problem. If it is too large to be solved by off-the-shelf solvers, a common approach is to use tailored solution techniques which exploit its sequential and block-diagonal structure and decompose $(P)$ into smaller subproblems. One such decomposition method is Benders decomposition, introduced by Benders in 1962 [6] for linear programs. Its extension to multistage stochastic linear programs, called *nested Benders decomposition* (NBD), was proposed by Birge [8]. NBD can be considered the basis of SDDP. Therefore, in order to introduce the latter, we explain NBD first.

The basic idea is to solve $(P)$ by taking advantage of its dynamic programming equations (1.1)-(1.3). With this comes the main challenge how to express the expected value functions [77], since they are not known in an analytical form in advance. In stochastic dynamic programming (SDP), usually the state variables $x_t$ are discretized and then $Q_t(\cdot)$ is evaluated at all such states, e.g., by backward recursion. However, for all but very small dimensions of $x_t$, this is computationally intractable. This *curse-of-dimensionality* is a well-known drawback of SDP and gave rise to the development of approximate dynamic programming methods [64].

In NBD, instead, the expected value functions are only evaluated at well-chosen trial points and approximated iteratively with polyhedral outer approximations constructed by cutting planes (or short cuts). This way, the above curse-of-dimensionality is broken. Such approximation of $Q_t(\cdot)$ is possible, since with Corollary 1.2 its piecewise linearity and convexity in $x_{t-1}$ are assured. This result also guarantees that finitely many cuts are sufficient to completely reproduce $Q_t(\cdot)$ for all $t$. We notice that this kind of approximation of $Q_t(\cdot)$ makes NBD a special variant of Kelley’s cutting plane method from convex optimization [43].

NBD consists of two main steps in each iteration $i$, a *forward pass* and a *backward pass* through the stages $t = 1, \ldots, T$. In the forward pass, for each $t$, an approximation of the stage-$t$ subproblem (1.1) is solved, yielding a trial solution $x_t^i(\xi_t^i)$, which is passed to the following stage where it serves as the fixed parameter for the subproblem. After the forward pass, thus, a feasible policy $(x_t^i(\xi_t^i))_{t=1}^T$ is identified and an upper bound $\bar{v}$ for $v^*$ can be determined by evaluating it in the objective function.

In the backward pass, the approximations of the stage-$t$ subproblems are improved iteratively by constructing cuts for $Q_t(\cdot)$ and passing them back to the previous stage. In this step also a lower bound $\underline{v}$ for $v^*$ is determined. This way, if not optimal, the current policy is amended.

We now provide a more detailed and technical look at the algorithmic steps.

**Forward pass (NBD).** For each stage $t = 1, \ldots, T$ and each scenario $\xi^s \in S_t$, the following problem is solved in the forward pass of some iteration $i$:

$$
Q^i_t(x_t^{i-1}, \xi_{[t-1]}, \xi^s_t) := \begin{cases} 
\min_{x_t} & c_t^\top (\xi^s_t)x_t + \Omega_{t+1}^i(x_t, \xi^s_t) \\
\text{s.t.} & W_t(\xi^s_t)x_t = h_t(\xi^s_t) - T_{t-1}(\xi^s_t)x_{t-1}^{i-1} \\
& x_t \geq 0.
\end{cases}
$$


Here, the function $\Omega_{t+1}^i(\cdot, \xi_{t-1}^s)$ denotes a polyhedral convex lower approximation of $Q_{t+1}(\cdot, \xi_{t-1}^s)$, which we refer to as cut approximation of $Q_{t+1}(\cdot, \xi_{t-1}^s)$. This approximation is constructed in the backward pass. For stage $T$, we have $\Omega_{T+1}^i(\cdot) = 0$.

We should mention that, therefore, apart from $x_{t-1}, \xi_{t-1}$ and $\xi_t$, the approximate value function $Q_{t+1}^i(\cdot)$ is also a function of the current cut approximation $\Omega_{t+1}^i(\cdot)$. For notational simplicity, we do not state this explicitly. However, the iteration index $i$ indicates the incorporation of $\Omega_{t+1}^i(\cdot)$.

Solving the forward pass problem recursively starting from stage 1, trial solutions $x_t^{is} = x_t^{is}(\xi_{t-1}^s)$ are computed for all $t$ and all $s \in S_t$. Since $(x_t^{is})_{t=1}^T$ for any $s \in S$ is a feasible policy for $(P)$, an upper bound $\pi$ for $v^*$ can be determined by

$$
\pi^i := \mathbb{E} \left[ \sum_{t=1}^T c_t^T(\xi_t)x_t^i(\xi_t) \right] = \frac{1}{|S|} \sum_{s \in S} \sum_{t=1}^T c_t^T(\xi_t)x_t^{is},
$$

assuming that all scenarios have the same probability.

**Backward pass (NBD).** The backward pass starts at stage $T$. Here, for all scenarios $\xi_T^s$, $s \in S = S_T$, the following subproblem is solved, using the trial solution $x_T^{is}$ determined in the forward pass:

$$(1.6) \quad Q_{t+1}^i(x_{t-1}^{is}, \xi_{[t-1]}^s, \xi_T^s) : = \begin{cases} \min_{x_T} & c_T^T(\xi_T^s)x_T \\ \text{s.t.} & W_T(\xi_T^s)x_T = h_T(\xi_T^s) - T_{T-1}(\xi_T^s)x_T^{is} \\ & x_T \geq 0 \end{cases}$$

Note that we use the superscript $i+1$ for $Q_{T}^i(\cdot)$ to emphasize that this function is used in the forward pass of iteration $i+1$ and that (as will become apparent for stages $T-1, \ldots, 1$) updated cut approximations $\Omega_{t+1}^{i+1}(\cdot)$ are incorporated in the backward pass subproblems.

The corresponding dual problem to (1.6) is

$$
\begin{aligned}
\max_{\pi_T} & \quad (h_T(\xi_T^s) - T_{T-1}(\xi_T^s)x_T^{is})^\top \pi_T \\
\text{s.t.} & \quad W_T(\xi_T^s)\pi_T \leq c_T(\xi_T^s).
\end{aligned}
$$

Let $\pi_T^{is} = \pi_T^i(\xi_T^s)$ be an optimal dual solution. Then, by strong duality,

$$
Q_{t+1}^i(x_{t-1}^{is}, \xi_{[t-1]}^s, \xi_T^s) = (h_T(\xi_T^s) - T_{T-1}(\xi_T^s)x_T^{is})^\top \pi_T^{is}
$$

$$
= -(\pi_T^{is})^\top T_{T-1}(\xi_T^s)x_T^{is} + (\pi_T^{is})^\top h_T(\xi_T^s).
$$

Since the dual feasible set does not depend on $x_{t-1}$, it is the same for all trial solutions. In particular, $\pi_T^{is}$ is feasible, but not necessarily optimal for all $x_{t-1}$. Therefore, it follows

$$
Q_{t+1}^i(x_{t-1}^{is}, \xi_{[t-1]}^s, \xi_T^s) \geq -(\pi_T^{is})^\top T_{T-1}(\xi_T^s)x_{t-1} + (\pi_T^{is})^\top h_T(\xi_T^s).
$$

As $Q_{T}(\cdot)^{i+1} = Q_{T}(\cdot)$, this directly implies

$$
Q_{T}(x_{t-1}, \xi_{[t-1]}^s, \xi_T^s) \geq -(\pi_T^{is})^\top T_{T-1}(\xi_T^s)x_{t-1} + (\pi_T^{is})^\top h_T(\xi_T^s).
$$

By taking expected values over all scenarios $\xi_T^s$ with $s \in A(\xi_{[t-1]}^s)$, we obtain

$$
\beta_T(\xi_{[t-1]}^s) := \mathbb{E}_{\xi_T|\xi_{[t-1]}^s} \left[ -(\pi_T^{is})^\top T_{T-1}(\xi_T^s) \right],
$$
and using both we define
\[
\phi_{T}(x_{T-1}, \xi_{[T-1]}^{s}) := \beta_{T}^{T}(\xi_{[T-1]}^{s})x_{T-1} + \alpha_{T}(\xi_{[T-1]}^{s}).
\]
\(\beta_{T}\) is called cut gradient and \(\alpha_{T}\) is called cut intercept.

It follows
\[
Q_{T}(x_{T-1}, \xi_{[T-1]}^{s}) \geq \phi_{T}(x_{T-1}, \xi_{[T-1]}^{s})
\]
\[
= Q_{T}(x_{T-1}, \xi_{[T-1]}^{s}) + \beta_{T}^{T}(\xi_{[T-1]}^{s})x_{T-1} - x_{T-1}^{s} + \alpha_{T}(\xi_{[T-1]}^{s}),
\]
i.e., (1.9) defines a cut for the expected value function \(Q_{T}(\cdot, \xi_{[T-1]}^{s})\). Moreover, \(\beta_{T}\)
denotes a subgradient of \(Q_{T}(\cdot, \xi_{[T-1]}^{s})\).

By Assumption 5, the dual feasible set is also bounded for any scenario. Therefore,
there exist only finitely many dual extreme points, which can be attained for the
considered scenario. Hence, by Assumption 7, only finitely many different extreme
values \((\beta_{T}, \alpha_{T})\) are possible.

The cut approximation \(\Omega_{T}^{+}(\cdot, \xi_{[T-1]}^{s})\) of \(Q_{T}(\cdot, \xi_{[T-1]}^{s})\) is then updated to
\[
\Omega_{T}^{+}(x_{T-1}, \xi_{[T-1]}^{s}) := \max \left\{ Q_{T}^{+}(x_{T-1}, \xi_{[T-1]}^{s}), \phi_{T}(x_{T-1}, \xi_{[T-1]}^{s}) \right\}.
\]
Thus, essentially \(\Omega_{T}^{+}(\cdot, \xi_{[T-1]}^{s})\) consists of finitely many linear functions \(\phi_{T}(\cdot)\), indexed by \(r \in R_{T_{i}}\),
for each \(s \in S_{T-1}\).

Using backward recursion, analogously, cuts can be derived for any stage \(t = T-1, \ldots, 2\) using the already updated cut approximation \(\Omega_{T+1}^{+}(\cdot)\). At stage \(t\), the
following problem is solved for all scenarios \(s \in S_{t}\):
\[
Q_{t}^{+}(x_{t-1}^{s}, \xi_{[t-1]}^{s}, \xi_{t}^{s}) := \begin{cases} 
\min_{x_{t}} & c_{t}^{T}(\xi_{t}^{s})x_{t} + \Omega_{t+1}^{+}(x_{t}, \xi_{[t]}^{s}) \\
\text{s.t.} & W_{t}(\xi_{t}^{s})x_{t} = h_{t}(\xi_{t}^{s}) - T_{t-1}(\xi_{t}^{s})x_{t-1}^{s} \\
x_{t} \geq 0 
\end{cases}
\]
(1.10)
\[
= \begin{cases} 
\min_{x_{t}, \theta_{t+1}} & c_{t}^{T}(\xi_{t}^{s})x_{t} + \theta_{t+1} \\
\text{s.t.} & W_{t}(\xi_{t}^{s})x_{t} = h_{t}(\xi_{t}^{s}) - T_{t-1}(\xi_{t}^{s})x_{t-1}^{s} \\
& \quad - (\beta_{t+1}^{T}(\xi_{[t]}^{s}))x_{t} + \theta_{t+1} \geq \alpha_{t+1}(\xi_{[t]}^{s}), \\
x_{t} \geq 0
\end{cases}
\]
As for stage \(T\), with optimal dual multipliers \(\pi^{s}_{T}\) and \(\rho^{irs}_{T}\), \(r = 1, \ldots, R_{t+1,s}\), the
latter corresponding to the cut constraints, we can define
\[
\beta_{t}(\xi_{[t-1]}^{s}) := E_{\xi_{[t]}^{s}|[T-1]} \left[ -(\pi^{s}_{T})^{T}T_{t-1}(\xi_{t}^{s}) \right]
\]
and
\[
\alpha_{t}(\xi_{[t-1]}^{s}) := E_{\xi_{[t]}^{s}|[T-1]} \left[ (\pi^{s}_{T})^{T}h_{t}(\xi_{t}^{s}) + \sum_{r \in R_{t+1}} \rho^{irs}_{T} \alpha_{r+1}(\xi_{[t]}^{s}) \right]
\]
by taking expectations over all scenarios \(\xi_{t}^{s}\) with \(s \in \Delta(\xi_{[T-1]}^{s})\).

Then, we obtain
\[
\phi_{t}(x_{t-1}, \xi_{[t-1]}^{s}) := \beta_{t}^{T}(\xi_{[t-1]}^{s})x_{t-1} + \alpha_{t}(\xi_{[t-1]}^{s}).
\]
and the cut

\[(1.12) \quad Q_t(x_{t-1}, \xi_{t-1}^s) \geq \phi_t(x_{t-1}, \xi_{t-1}^s)\]

As for stage \(T\), these type of cuts have three important properties:

**Lemma 1.3.** For any stage \(t = 2, \ldots, T\) and \(\xi_{t-1}^s\), the functions \(\phi_t(\cdot, \xi_{t-1}^s)\) are

(a) valid lower approximations of \(Q_t(\cdot, \xi_{t-1}^s)\),

(b) tight for the approximate expected value function \(Q_t^*(\cdot, \xi_{t-1}^s)\) at \(x_{t-1}^*\),

(c) finite, i.e., only finitely many different cuts can be generated.

**Proof.** Property (a) is given in (1.12). (b) follows from solving the dual of subproblem (1.10) for all \(s \in S_t\), strong duality and taking expected values over the obtained optimal values. Property (c) follows by induction with the same arguments as for stage \(T\).

Note that \(\phi_t(\cdot)\) is not necessarily tight for \(Q_t(\cdot)\) in early iterations for \(t \neq T\), though, but might provide a loose cut only. However, by the finiteness and tightness properties it can be shown recursively, that eventually the derived cuts become tight for \(Q_t(\cdot)\) as well. An exemplary approximation of \(Q_t(\cdot)\) with cutting planes is depicted in Fig. 4.

\[\text{Fig. 4: Expected value function } Q_t(\cdot) \text{ exactly represented by maximum of three cutting planes.}\]

In some cases, it may be beneficial to summarize all previously derived cuts in one formula by row-wise appending all cut gradients in a matrix \(\vec{G}_t\) and all cut intercepts in a vector \(\vec{g}_t\) [42, 65]:

\[(1.13) \quad -\vec{G}_t(\xi_{t-1}^s) + e\theta_{t+1} \geq \vec{g}_t(\xi_{t-1}^s)\]

where \(e\) denotes a vector of ones with appropriate dimension.

Again, the cut approximation \(\Omega_t^*(\cdot, \xi_{t-1}^s)\) for \(Q_t(\cdot, \xi_{t-1}^s)\) is updated to

\[\Omega_{t+1}(x_{t-1}, \xi_{t-1}^s) := \max \left\{ \Omega_t^*(x_{t-1}, \xi_{t-1}^s), \phi_t(x_{t-1}, \xi_{t-1}^s) \right\} .\]
At the first stage, the following problem is solved, yielding a lower bound \( v \) to the optimal value \( v^* \):

\[
\begin{align*}
\mathcal{L}^i & := \min_{x_1} \ c_1^T x_1 + \Omega_{2}^{i+1}(x_1, \xi[1]) \\
\text{s.t.} & \quad W_1 x_1 = h_1 \\
& \quad x_1 \in X_1
\end{align*}
\]

(1.14)

(1.14)

If the upper bound \( \pi^i \) and the lower bound \( \mathcal{L}^i \) are equal, NBD terminates with an optimal solution of \( (P) \). Otherwise, a new iteration \( i + 1 \) is started. Since the derived cuts are valid, tight and finite, finite convergence to an optimal solution of \( (P) \) can be proven.

**Theorem 1.4 ([11]).** NBD converges to an optimal solution of \( (P) \) in finitely many iterations.

### 1.3. Stochastic dual dynamic programming.

The main drawback of NBD is that, despite avoiding the curse-of-dimensionality stemming from discretizing the state variables, it still becomes computationally infeasible for all but moderate problem instances, as the scenario tree grows exponentially in the number of stages. By moderate we mean instances with some hundreds or a few thousand scenarios [82].

This type of the curse-of-dimensionality is broken by SDDP, which was introduced by Pereira and Pinto in 1991 in the context of optimizing the Brazilian hydro storage system [56]. It combines the idea of NBD with sampling. The main idea is that in each iteration not all scenarios \( \xi_s, s \in S \), of the scenario tree are considered, but only a sample of those scenarios is drawn and used, with the aim to reduce the number of linear programs to be solved significantly.

**Cut-sharing.** Considering only a subset of all scenarios, not all nodes in the scenario tree are visited. Thus, not all subproblems and value functions are updated in each iteration. Assuming that information on all nodes is required for convergence, this means that the iterations are executed faster using sampling, but much more iterations are needed in comparison to NBD, which is still computationally infeasible. In the worst case, performance may be even worse than for NBD, since it can take quite some time until a specific, but possibly crucial scenario is sampled [16].

For this reason, SDDP comes along with the additional, crucial assumption that the uncertainty on different stages does not depend on each other.

**Assumption 9.** The uncertainty in \( (P) \) is stagewise independent, that is, \( \xi_t \) is independent of the history \( \xi_{[t-1]} \) of the process.

Under Assumption 9, the considered scenario tree collapses to a recombining scenario tree, as visualized in Fig. 5. The expected value \( \mathbb{E}_{\xi_{t+1} | \xi_t} \) in (1.2) simplifies to \( \mathbb{E}_{\xi_{t+1} | \xi_t} \) and \( Q_{t+1}(x_t, \xi_t) = Q_{t+1}(x_t) \), i.e., for each stage \( t \), essentially, there is only one expected value function which does not depend on the data process.

Thinking of the original scenario tree, this means that in all nodes of stage \( t \) the same realizations \( \xi_{tj}, j = 1, \ldots, q_t \), of the uncertain data may occur. This implies that \( |S| = \prod_{t=1}^T q_t \) with \( q_0 = 1 \) and that for each scenario \( \xi^* \), there exists some \( j \) such that \( \xi^*_t = \xi_{tj} \).
As a result, cuts derived for one specific sample path through the scenario tree are valid for all other scenarios as well and all nodes are updated with new information in each iteration. We refer to this property as cut-sharing [42]. As a direct consequence, the cut approximation $\Omega_i(\cdot)$ no longer depends on $\xi[1]$. 

**Forward pass (SDDP), also called forward simulation.** The main principle of the forward pass remains the same as in NBD, but only a sample of scenarios is considered in each iteration. We use the index $k \in \mathcal{K}$ to denote sampled scenarios, with $\mathcal{K} \subset \mathcal{S}$ and $|\mathcal{K}| \ll |\mathcal{S}|$. This is illustrated in Fig. 6.

We should note that the number of samples $|\mathcal{K}|$ may vary by iteration and, hence, depends on $i$. We discuss this idea in Sect. 2.2, but apart from that do not state this dependence explicitly.

The subproblem solved for iteration $i$, stage $t$ and sample $k$ then denotes

\begin{equation}
Q_i^k(x_{t-1}^k, \xi_t^k) = \min_{x_t} \begin{cases} \mathbf{c}_t^T(\xi_t^k)x_t + \Omega_{t+1}(x_t) \\
\text{s.t.} \quad W_t(\xi_t^k)x_t = h_t(\xi_t^k) - T_{t-1}(\xi_t^k)x_{t-1}^{ik} \\
x_t \geq 0. \end{cases}
\end{equation}

As in NBD, the obtained trial solutions $(x_t(\xi_t^k))_{t=1}^T$ correspond to a feasible policy for $(P)$. Note that this policy is not actually determined in the forward pass, but
implicitly determined by the cut approximations $\Omega_t(\cdot), t = 2, \ldots, T$, and only evaluated in the forward pass.

In contrast to NBD, the policy is not evaluated for all scenarios, though. For this reason, no deterministic upper bound is calculated in the forward pass. However, the sample average

$$v_i^k := \frac{1}{|K|} \sum_{k \in K} \sum_{t=1}^T c_t^\top (\xi^k_t) x^{ik}_t$$

is an unbiased estimator of the NBD upper bound (1.5). Hence, $E[v_i^k] = \bar{v}^i$ and $v_i^k$ is a statistical upper bound for $v^*$. Moreover, by the Law of Large Numbers, $\lim_{k \to \infty} v_i^k = v_i$. Together with the sample variance

$$\sigma_{v_i^k}:= \frac{1}{|K| - 1} \sum_{k \in K} (v^i(\xi^k) - v_i^k)^2,$$

an approximate 95% confidence interval $[v_i^k - 1.96 \frac{\sigma_{v_i^k}}{\sqrt{|K|}}, v_i^k + 1.96 \frac{\sigma_{v_i^k}}{\sqrt{|K|}}]$ can be determined, see Sect. 4 for more details. SDDP terminates if $v_i^k$ is included in this confidence interval [56]. We discuss shortcomings of this stopping criterion and possible alternatives in Sect. 4.

Backward pass (SDDP). The main principle of the backward pass and the cut generation remain the same as for NBD. However, not all nodes of the scenario tree are visited in the backward pass.

To be precise, for all $t = T, \ldots, 2$ and all scenarios sampled in the forward pass, i.e., for all trial solutions $x^{ik}_{t-1}, k \in K$, the following subproblems are solved for all possible stage-$t$ realizations $\xi^k_{tj} \equiv \xi_{tj}$, $j = 1, \ldots, q_t$:

$$Q_t^{i+1}(x^{ik}_{t-1}, \xi_{tj}) = \begin{cases} 
\min_{x_t} & c_t^\top (\xi_{tj}) x_t + \Omega_t^{i+1}(x_t) \\
\text{s.t.} & W_t(\xi_{tj}) x_t = h_t(\xi_{tj}) - T_{t-1}(\xi_{tj}) x^{ik}_{t-1} \\
& x_t \geq 0 
\end{cases}$$

(1.18)

$$\equiv \begin{cases} 
\min_{x_t, \theta_{t+1}} & c_t^\top (\xi_{tj}) x_t + \theta_{t+1} \\
\text{s.t.} & W_t(\xi_{tj}) x_t = h_t(\xi_{tj}) - T_{t-1}(\xi_{tj}) x^{ik}_{t-1} \\
& -(\beta_{t+1})^\top x_t + \theta_{t+1} \geq \alpha_{t+1}, \quad r \in R_{t+1} \\
& x_t \geq 0 
\end{cases}$$

This is illustrated in Fig. 7. Again, we have the stage-$T$ exception of $\Omega_{T+1}(\cdot) \equiv 0$. For the cut formula, it follows

$$\beta_t := - \sum_{j=1}^{q_t} p_{tj} (\pi_t^{ij})^\top T_{t-1}(\xi_{tj})$$

(1.19)

and

$$\alpha_t := \sum_{j=1}^{q_t} p_{tj} \left( (\pi_t^{ij})^\top h_t(\xi_{tj}) + \sum_{r \in R_{t+1}} \rho_{t+1}^{ij} \alpha_{t+1}^r \right).$$

(1.20)
For the first stage, subproblem (1.14) is solved. It yields a lower bound \( v^i \) for \( v^* \).

The number of linear programs to be solved in the backward pass is \( 1 + |\mathcal{K}| \sum_{t=2}^{T} q_t \).

Hence, in contrast to NBD, the number of problems to solve grows linearly in the number of stages \( T \) instead of exponentially [65].

![Fig. 7: SDDP backward pass. The blue subproblems have to be solved in addition to those from the forward pass.](image)

As all \( q_t \) realizations are used to construct a cut, the optimality cuts determined in SDDP are valid underestimators for \( Q_t(\cdot) \). Using Assumption 9, those cuts are shared among scenarios, i.e., cuts are constructed for all nodes of the scenario tree, even though not all of them are visited in the backward pass. This is a difference to standard NBD, where usually nodes only obtain cuts generated at their direct descendants. It also implies that \( v^i \) is a safe lower bound on \( v^* \).

A few example iterations and cuts for SDDP are illustrated in Fig. 8. The cuts for stage 1 show that the generated approximation may be loose in early iterations.

**Convergence.** The generated cuts possess the same validity, tightness and finiteness properties as for NBD. Moreover, with Corollary 1.2, the expected value function \( Q_t(\cdot) \) can be exactly represented as the maximum of finitely many linear cuts. In principle, this allows for finite convergence. However, the convergence properties of SDDP also depend on the sampling procedure which is random. Therefore, only convergence in probability can be expected.

The first SDDP related convergence results have been stated in [13] and [47], however implicitly assuming the independence of sampled random variables and convergent subsequences of algorithm iterates. A fully valid convergence proof is given by Philpott and Guan in [62] for the case where uncertainty only exists in the right-hand side of problem (P).

**Theorem 1.5 (Finite convergence of SDDP [62]).** Assuming fixed recourse, i.e., \( W_t(\xi_t) \equiv W_t \) for all \( t \), fixed objective function coefficients, i.e., \( c_t(\xi_t) \equiv c_t \) for all \( t \), fixed technology, i.e., \( T_t(\xi_t) \equiv T_t \) for all \( t \), and using an independent sampling procedure in the forward pass, SDDP converges with probability 1 to an optimal policy of (P) in a finite number of iterations.

The main idea behind the proof is that, assuming that the algorithm does not
stop, each scenario $\xi^s, s \in S$, is traversed infinitely many times with probability 1 in the forward pass under independent sampling. Philpott & Guan refer to this as the Forward Pass Sampling Property (FPSP).

A more general convergence analysis is provided by Shapiro [76]. In principle, the above idea and convergence result remain true even if $W_t, c_t$ or $T_t$ are uncertain. This
remains not true, though, if the stagewise independence Assumption 9 is dropped.

While Theorem 1.5 guarantees almost sure finite convergence of SDDP, since every scenario is eventually sampled, it is important that convergence can take way too long in large-scale applications to be computationally tractable [40, 76]. Still, SDDP has shown good performance for large-scale instances of multistage stochastic problems in many applications, such as hydrothermal scheduling.

2. Performance Improvement for SDDP. In the form presented in Sect. 1, SDDP has several drawbacks and potential for improvement.

Firstly, Assumptions 1-9 all have to be satisfied for it to be applied in a computationally feasible way. Some of these assumptions are rather restrictive and, thus, limit the application of SDDP to a narrow class of problems. Moreover, some of them are usually either not satisfied, e.g., stagewise independence or a finite distribution of \(\xi\), or not relevant in practical applications, e.g., risk-neutral decisions. Therefore, several enhancements of SDDP have been proposed since 1991 relieving some of these basic assumptions. We discuss such enhancements in the following sections.

Secondly, even if Assumptions 1-9 are satisfied, SDDP still suffers from some performance issues. As well-known for all types of cutting-plane methods, it can show slow convergence. Additionally, even with sampling only a subset of all scenarios from the scenario tree, the computational effort in each iteration can become pretty high, especially in the backward pass where the subproblems are solved for all \(t\), all \(x_{t-1}\) and all realizations of \(\xi_t\). Such high effort to solve subproblems and derive cuts may be detrimental if the current policy is still far away from the optimum.

Another aspect is the increasing number of cuts to be incorporated in the subproblems. While this grows only linearly in the number of iterations, it can increase the subproblems and slow down the algorithm significantly for large problems \((P)\), which require a high number of iterations.

In this section, we give an overview on different modifications of SDDP to address such issues and improve its performance. In general, those techniques attempt to either speed up the SDDP iterations by reducing the computational effort, or to reduce the number of iterations required by improving the progress achieved in single iterations. The latter is usually related to more sophisticated optimization strategies.

2.1. Multi-cut SDDP. In the backward pass of SDDP, for any \(t\) and any \(x_{t-1}\), subproblems are solved for all realizations \(\xi_{tj} \equiv \xi_{tj}, j = 1, \ldots, q_t\), see subproblem (1.18). By taking expected values over all such realizations, a single cut

\[
\phi_t(x_{t-1}) = \beta_t^T x_{t-1} + \alpha_t \leq Q_t(x_{t-1})
\]

is derived with \(\beta_t\) and \(\alpha_t\) defined in (1.19) and (1.20). This is also referred to as a uni-cut approach.

Such cuts are then incorporated in the stage-\((t-1)\) subproblem using an epigraph variable \(\theta_t \in \mathbb{R}\) and

\[
(\beta_r^T x_{t-1} + \alpha_r) \leq \theta_t, \quad r \in R_t.
\]

A different approach is to not aggregate the dual information to generate a single cut corresponding to a single variable \(\theta_t\), but to generate one cut for each realization \(\xi_{tj}, j = 1, \ldots, q_t\). This requires to introduce variables \(\theta_{tj}\) and cut approximations \(\Omega_{t+1, \ell}(')\) for all \(\ell = 1, \ldots, q_t\) in the stage-\(t\) subproblem. In this case, we obtain

\[
\phi_{tj}(x_{t-1}) := \beta_{tj}^T x_{t-1} + \alpha_{tj}
\]
with
\[ \beta_{tj} := - (\pi_{tj}^T T_{t-1} (\xi_{tj})) \]
and
\[ \alpha_{tj} := (\pi_{tj}^T T_{t-1} (\xi_{tj})) + \sum_{\ell=1}^{q_{t+1}} \sum_{r \in R_{t+1,\ell}} \rho_{tj}^{r \ell} \alpha_{t+1}^r. \]

The expectation is then not calculated within the cut formula, but in the objective function:
\[
Q_{t+1}^{i+1}(x_{t-1}^{i+1}, \xi_{tj}) = \begin{cases} 
\min_{x_t} & c_t^T (\xi_{tj}) x_t + \sum_{\ell=1}^{q_{t+1}} p_{t+1,\ell} Q_{t+1,\ell}(x_t) \\
\text{s.t.} & W_t(\xi_{tj}) x_t = h_t(\xi_{tj}) - T_{t-1}(\xi_{tj}) x_{t-1} \\
x_t \geq 0. \end{cases}
\]
(2.3)
\[
\equiv \begin{cases} 
\min_{x_t} & c_t^T (\xi_{tj}) x_t + \theta_{t+1} \\
\text{s.t.} & W_t(\xi_{tj}) x_t = h_t(\xi_{tj}) - T_{t-1}(\xi_{tj}) x_{t-1} \\
& - (\beta_{t+1}^r (\xi_{tj}))^T x_t + \theta_{t+1} \geq \alpha_{t+1}^r (\xi_{tj}), \\
r \in R_{t+1,\ell} \end{cases}
\]
\[ x_t \geq 0. \]

This way, more specific information about the value functions is incorporated into the subproblems, hopefully leading to less iterations. On the downside, the number of decision variables and cut restrictions grows significantly compared to the uni-cut approach, especially if many realizations \( q_t \) per stage are considered, increasing the computational effort for each iteration.

For two-stage BD and NBD, the multi-cut variant has been examined extensively [10, 24, 52]. Using a multi-cut version should be most promising when there are not too many scenarios. For the two-stage case, a rule of thumb is that a uni-cut approach should be preferred if the number of realizations is considerably larger than the number of first stage constraints [11]. A more detailed analysis and worst-case iteration bounds for uni- and multi-cut Benders decomposition as well as cut aggregation techniques in between are provided in [7].

Some of the rare occasions where multi-cut SDDP is considered are [29] and [84], the latter considering the risk-averse case. In the latter, it is proven that using the same arguments as for uni-cut SDDP, also almost sure convergence of multi-cut SDDP is assured. However, it is also identified that the multi-cut variant requires much more computational effort and that it is rarely more efficient than its uni-cut pendant.

**2.2. Tree-traversing Strategies.** As described in Sect. 1, the uncertain data process considered in SDDP can be expressed by means of a scenario tree. In contrast to NBD, not each node of this tree is visited in each iteration, though. However, the order and strategy in which nodes are visited within the algorithm has a considerable effect on its performance in practical applications.

**Sequencing Protocols.** Sequencing protocols determine at which point direction changes occur while traversing the scenario tree. Most commonly used for NBD is the *fast-forward fast-back* protocol, in which all stages are traversed either in one direction (either forward or backward) before the direction is changed again, presuming that no feasibility cuts are required. In numerical experiments, this has been identified as the most efficient sequencing protocol [9, 11, 24, 52].

For SDDP so far only the fast-forward fast-backward protocol has been applied, as it allows for a clear distinction between forward and backward passes in the algorithm.
Thus, it is also clear when a new sample of scenarios is to be drawn. In case of other protocols, this may be hindered. Still, those protocols could be applied in SDDP and combined with different sampling strategies.

**Scenario strategies.** Tree-traversing strategies also include strategies on how many and which scenarios are considered in each forward pass and backward pass, respectively.

In the original version of SDDP, see Sect. 1.3, $|K|$ out of all $|S|$ scenarios in the scenario tree are sampled in each iteration, with $|K| \ll |S|$ and $K \subset S$. Philpott & Guan even propose a method with only $|K| = 1$ for all iterations [62]. This strategy may be particularly efficient in earlier iterations in order to obtain a rough approximation of $Q_t(\cdot)$ fast without wasting too much effort in regions for $x_{t-1}$ which are likely to be far from optimal. On the other hand, this strategy faces some drawbacks. Firstly, if the current policy is already reasonably good, but convergence is not achieved yet, it should be beneficial to generate more than one new cut per stage and iteration and, thus, to sample more than 1 scenario [18].

Secondly, if only one scenario is considered per stage, it is not possible to simulate and evaluate the current policy directly within the algorithm. Therefore, the stopping criterion of SDDP has to be adapted. In particular, we do not obtain $|K|$ different realizations for $v^i(\xi^k) = \sum_{t=1}^T c_t^\top(\xi^k_t)x_{t}^k$ that can be used to estimate the expected value and the variance for $v_K$. Using values $v^i(\xi^k)$ from previous iterations instead, still yields an unbiased estimate of $E[v_K]$, but destroys the independence between the different samples. Thus, the central limit theorem cannot be applied and the classical stopping criterion of SDDP is not applicable. As a relief, it is proposed to use a predetermined number of iterations [18].

Using a *scenario incrementation* strategy in which the number of scenarios sampled is gradually increased, combines the advantages of both approaches - a small number of samples in early iterations and a higher number of samples and a valid stopping criterion in later iterations. This is proposed in [79] and tested in [18].

For similar reasons, further variations of the SDDP algorithm have been proposed, such as AND [19], CUPPS [13], DOASA [62] and ReSa [38].

**CUPPS.** In comparison to SDDP, the CUPPS (convergent cutting-plane and partial-sampling) method reduces the computational effort to determine valid cuts for the expected value function $Q_t(\cdot)$ [13]. The algorithm does not contain a backward, but only a forward pass through the stages. Therefore, contrary to SDDP, the subproblems do not have to be solved for all realizations $\xi_{tj}, j = 1, \ldots, q_t$, in order to derive cuts. Instead, the cuts are directly derived in the forward pass, in which only one scenario $\xi^t_i$ is sampled per iteration.

Once the stage-$t$ subproblem is solved for $x_{t-1}^i$ and $\xi^t_i$, the optimal dual solution $(\pi^t_i, \rho^t_i)$ is utilized to determine a new cut. To obtain a valid cut for $Q_t(\cdot)$ as tight as possible, not only this optimal dual solution is used, though, but also information from previous iterations.

To be precise, all optimal dual multipliers obtained up to iteration $i$ are saved in the set $D^i_t = D^{i-1}_t \cup \{ (\pi^t_i, \rho^t_i) \}$, where elements with smaller dimensions from earlier iterations are extended with zeros.

Then, a new cut for the following iteration $i+1$ is determined by

\begin{equation}
\phi^{i+1}_t(x_{t-1}) := (\beta^{i+1}_t)^\top x_{t-1} + \alpha^{i+1}_t,
\end{equation}
with
\[
\beta_{t+1}^i := -\sum_{j=1}^{q_t} p_{tj} (\pi_t(x_{t-1}^i, \xi_{ij}, D_t^i))^\top T_{t-1}
\]
and
\[
\alpha_{t+1}^i := -\sum_{j=1}^{q_t} p_{tj} \left( (\pi_t(x_{t-1}^i, \xi_{ij}, D_t^i))^\top h_t(\xi_{ij}) + \sum_{r \in K_{t+1}} \rho_t^r (x_{t-1}^i, \xi_{ij}, D_t^r) \alpha_{t+1}^r \right).
\]

For any \(\xi_{ij}, j = 1, \ldots, q_t\), the used dual multipliers are defined by
\[
(\pi_t(x_{t-1}^i, \xi_{ij}, D_t^i), \rho_t(x_{t-1}^i, \xi_{ij}, D_t^i)) = \arg \max \left\{ \pi_t^\top (h_t(\xi_{ij}) - T_{t-1}x_{t-1}) + \rho_{t+1} \alpha_{t+1} \mid (\pi_t, \rho_t) \in D_t^i \right\}.
\]
Thus, not necessarily optimal dual multipliers of the corresponding subproblem are used, but the previously generated dual multipliers yielding the best approximation.

This approach does only yield valid cuts if the dual multipliers generated for one scenario still correspond to dual extreme points for different scenarios. Therefore, in [13], only uncertainty in the right-hand side is considered.

While the computational effort to derive new cuts is reduced, CUPPS has the drawback that the obtained cuts are not necessarily tight for \(Q_{i+1}(\cdot)\); firstly, since possibly non-optimal dual multipliers are used for some \(\xi_{ij}\), and secondly, since no backward pass is used and, thus, a new cut for stage \(t + 1\) is not directly incorporated in deriving a new cut for stage \(t\).

**DOASA.** The dynamic outer approximation sampling algorithm (DOASA) [62] uses the same idea of choosing dual multipliers for cut generation by formula (2.5). However, instead of constructing the cuts directly in the forward pass using scenario \(\xi_{ij}\), they are constructed in a backward pass, where again one scenario \(\xi_{BP,i}^{BP,i}\) is sampled.

Thus, DOASA and CUPPS mainly differ in the order of trial point determination and cut construction: In CUPPS, trial points and cuts are directly determined for each stage-\(t\) problem before stage \(t + 1\) is reached. In DOASA first trial points for all stages \(t = 1, \ldots, T\) are determined and then cuts are constructed by traversing the stages in backward direction.

**AND.** Donohue and Birge [19] claim that the SDDP strategy is useful for narrow and long scenario trees, but not well-designed for bushier scenario trees with a higher number \(q_t\) of different realizations \(\xi_{ij}\) per stage. To obtain reasonably good policies for such trees, many scenarios have to be sampled in the forward pass. While this is not a huge problem in the forward pass itself, the computational burden in the backward pass becomes tremendous, as for all stages and sampled scenarios the subproblems are solved for all descendants \(\xi_{ij}, j = 1, \ldots, q_t\).

Donohue and Birge design a SDDP related method which should be better suited for bushier scenario trees, the so-called abridged nested decomposition method (AND). In the forward pass, again several scenarios are sampled. However, in each stage, the method does not proceed forward from the solutions \(x_{ik}^t\) of all sampled problems considered in the previous stage. Instead, branching values \(x_{ik}^t\) are sampled from all these solutions. In the backward pass, the subproblems are then only solved for all realizations \(\xi_{ij}\) using the solutions \(x_{ik}^t\) of these branching scenarios.

The main idea of AND is that this way, information on more parts of the scenario tree can be incorporated in the forward pass than for SDDP. In particular, this holds because the branching values \(x_{ik}^t\) are not required to be a subsample of all \(x_{ik}^t, k \in K,\)
but may also be a convex combination of those feasible solutions. This way, even if not all sampled scenarios are considered over the whole horizon \( t = 1, \ldots, T \), information on a lot of scenarios can be considered. Numerical results in [19] indicate that AND performs significantly better than SDDP for bushier trees.

The main drawback of AND is that the special structure of the forward pass allows no direct estimate of an upper bound [38]. Therefore, an additional sampling procedure has to be started every few iterations to obtain upper bounds and test for convergence.

ReSa. In the reduced sampling method (ReSa) it is tried to combine the advantages of SDDP and AND [38]. The obtained algorithm should be efficient for problems with many stages but only a few possible realizations of the uncertain data, as well as for problems with few stages but many random outcomes per stage.

The forward pass follows the same principle as SDDP, sampling scenarios \( \xi_k, k \in K \). In the backward pass, to reduce the number of subproblems to be solved, not all forward pass scenarios \( \xi_k \) are considered but only a subsample is drawn. This also means that less cuts are derived per iteration than in SDDP.

2.3. Regularization. Since the approximation of the expected value functions in SDDP follows the principle of general cutting-plane methods [43], it may also suffer from slow convergence, i.e., the number of trial solutions may grow exponentially with the problem dimension [81].

In convex and nonsmooth optimization, regularization techniques called bundle methods are shown to entail faster convergence than classical cutting-plane methods. In stochastic programming, a quadratic regularization term is most commonly used [69, 72] in which some quadratic deviation of \( x_t \) from an incumbent solution (or stability center) \( \bar{x}_t \) is penalized in the objective function. Therefore, the solution procedure is stabilized around the known incumbent solution and zig-zagging is mitigated.

However, an application of quadratic regularization to SDDP is not straightforward, since using a separate incumbent solution for each node of the scenario tree is computationally infeasible. The reason for this is its exponential growth, implying also exponential growth of the number of incumbent solutions [3].

Therefore, Asamov & Powell [3] propose a regularization technique, in which incumbents are considered part of the state variable and, thus, the same for all realizations of \( \xi_t \). Then, in the forward pass subproblems the objective function is

\[
c_t^\top x_t + Q_{t+1}^i(x_t) + \frac{\gamma^i}{2} (x_t - \bar{x}_t^{i-1})^\top H_t (x_t - \bar{x}_t^{i-1}),
\]

with a positive semidefinite matrix \( H_t \) and some sequence \((\gamma^i)_{i \in \mathbb{N}}\) satisfying \( \gamma^i \geq 0 \) for all \( i \) and \( \lim_{i \to \infty} \gamma^i = 0 \). The incumbents \( \bar{x}_t^{i-1} \) are chosen as the previous forward pass solution, i.e., the solution is stabilized around a “known” region of the domain of \( Q_t(\cdot) \). This idea is generalized in [30] by considering weighted averages of several previous forward pass solutions.

Using objective (2.6) a convex, continuous and linearly constrained quadratic programming problem has to be solved in each forward pass step of SDDP, hopefully, reducing the required number of iterations. As still only finitely many different cuts can be generated using dual extreme points, finite convergence is assured. In numerical tests, it is shown that this method exhibits faster convergence than SDDP, in particular for a high dimension of the state variable \( x_t \) [3].

While the above approach achieves some regularization by stabilizing the solution around a “known” region of the domain of \( Q_t(\cdot) \), in a sampling setting, it is not clear
whether this is beneficial. For the current sample $\xi^k$ a region may be identified and used for stabilization, which is no appropriate indicator for the whole scenario tree.

Additionally, as pointed out in [81], the condition $\lim_{i \to \infty} \gamma_i = 0$ may evoke that the regularization is diminished and the proposed method in [3] reduces to standard SDDP before convergence is obtained. Since it is well-known that regularization is important in areas close to optimal solutions, this is detrimental to convergence speed [81].

Van Ackooij et al. also address that convergence of bundle methods usually requires the incumbents to be feasible, which is not guaranteed for SDDP subproblems where the feasible set changes with $x^t_{i-1}$. Therefore, they propose to combine SDDP with a level bundle method, which does not face this requirement [81].

For stage $t$ and scenario $\xi_t$, trial solutions $x^t_{ik}$ are obtained as solutions of

\[(2.7)\]
\[
\begin{align*}
\min_{x_t} \quad & \psi_t(x_t) \\
\text{s.t.} \quad & x_t \in X_t(x^t_{ik}; \ell_t)
\end{align*}
\]

with $\psi_t(x_t) : \mathbb{R}^n_t \to \mathbb{R}$ a given convex function, e.g., $\psi_t(x_t) := x_t^\top x_t$, and

\[(2.8)\]
\[
X_t(x^t_{ik}; \ell_t) := \begin{cases} 
\arg \min_{x_t \in X_t} & \max \left\{ c_t^\top x_t + \Omega^t_{i+1}(x_t), \ell_t \right\} \\
\text{s.t.} & W_t x_t = h_t - T_t x^t_{i-1}. 
\end{cases}
\]

If the maximum in (2.8) is attained by the first term, then $x^t_{ik}$ obtained by solving (2.7) is an ordinary SDDP trial solution, referred to as normal iterate. Otherwise, problem (2.7) reduces to a typical level bundle method subproblem, yielding a regularized level iterate $x^t_{ik}$.

The parameter $\ell_t$ is chosen and adapted using heuristics in [81].

### 2.4. Cut Elimination and Selection.

Instead of reducing the number of cuts created in the backward pass, as proposed in Sect. 2.2, the computational burden in the backward pass of SDDP can also be reduced by careful elimination or selection of cuts. A computational advantage can only be expected if cuts are omitted only when they are redundant or dominated by other cuts and, thus, guaranteed to not contribute to the solution process. Numerical results in [80] indicate that SDDP tends to generate a large number of such cutting planes.

**Cut Elimination.** One way to reduce the number of cuts is to eliminate such cuts permanently. This can be done by solving an auxiliary problem checking feasibility of the system [80]

\[(2.9)\]
\[
\begin{align*}
\theta_t+1 &\leq \alpha^t_{i+1} + (\beta^t_{i+1})^\top x_t \\
\theta_t+1 &\geq \alpha^t_{i+1} + (\beta^t_{i+1})^\top x_t, \\
x_t &\in X_t.
\end{align*}
\]

If this system is infeasible, then the cut $\theta_t+1 \geq \alpha^t_{i+1} + (\beta^t_{i+1})^\top x_t$ is redundant and can be eliminated. The drawback of this method is that the auxiliary problem has to be solved for all cuts in the system.

A different approach is to permanently store all cuts for each stage $t$, but only select a subset of those cuts to be considered in the backward pass of the current iteration $i$. Selection techniques based on this approach are introduced in [18, 29].

**Selecting Last Cuts.** In this naive strategy only the $H$ most recently added cuts are selected. Although on average, late cuts may provide a better approximation...
Level of Dominance. This strategy is a heuristic to consider only non-dominated cuts, but avoid the computational effort of the above cut elimination approach. Using the most basic approach only cuts are selected, which yield the highest function value at one of the trial solutions considered so far within the algorithm. This is called Level 1 Dominance [18].

Let $x^\ell_t$ the trial solution corresponding to the $\ell$-th cut, $\ell \in R_{t+1}$, and $\phi^r(x^\ell_t)$ the corresponding function value of cut $r$. Then, the values $v(\ell) := \max_{r \in R_{t+1}} \{ \phi^r(x^\ell_t) \}$ and $r(\ell) := \arg \max_{r \in R_{t+1}} \{ \phi^r(x^\ell_t) \}$ can be saved in a list and be updated every time a new cut is constructed.

Similarly, a Level $H$ Dominance strategy can be used, selecting the $H$ highest cuts for all trial solutions. Using this strategy, only previous trial solutions are taken into consideration, though. Therefore, cuts may be excluded which provide a significant benefit at not yet visited feasible states.

Another downside is that this strategy draws a lot of resources to store all the required cut information. This can even be relevant for Level 1, especially if the maximum function value at the trial solutions is attained by several cuts. As a resort, in [27], the Limited Memory Level 1 strategy is introduced, selecting only the oldest of such cuts. In [29] this technique is applied to SDDP and almost sure convergence is proven.

Dynamic Cut Selection. A more dynamic, but also computationally more expensive strategy is to select cuts dynamically within the SDDP framework. In [18] it is proposed to remove all cuts at the beginning of each iteration. Then, for each stage $t$, each scenario $k$, and each cut $\phi^r$ the forward pass subproblem (1.15) is solved. If the current cut yields the highest value at the obtained trial solution, it is added to the subproblem, and the next cut is considered.

This way, only cuts are selected that contribute to the optimal solution in the current iteration. On the other hand, the additional loop may also slow down the convergence speed. The computational effort can be reduced by inheriting all added cuts from the already considered scenarios.

Numerical results for sampling about 5,000 scenarios and computing 10,000 cuts in SDDP indicate that all cut selection techniques can significantly speed-up the classical SDDP method [18]. For example, the Level 1 strategy is reported to be ten times faster than SDDP without cut selection. For dynamic cut selection, the speed-up is much smaller due to the additional computational effort. It is also shown that the cut selection strategies do not have a significant impact on the quality of the determined policies and bounds. In [29], Limited Memory Level 1 is identified as more efficient than pure Level 1.

2.5. Warm Start Techniques. One characteristic of decomposition methods for multistage stochastic programming problems is that a lot of similar subproblems are solved. In particular, this is the case if those problems only face uncertainty in the RHS $h_t$ or the technology matrix $T_{t-1}$.

In each SDDP backward pass, for example, the stage-$t$ subproblem has to be solved for the same state variable $x^k_{t-1}$ and all realizations $\xi_{t,j}, j = 1, \ldots, q_t$. If only the RHS is uncertain and some of its realizations do not differ by too much, it may prove efficient to consider warm start techniques, which for example exploit similarities in the optimal bases of such problems.
Among those techniques are bunching, sifting or initializing subproblems with basic optimal solutions from previous iterations [52]. While such techniques have reportedly been tested for NBD, there appears to be no specific research on warm starts in SDDP so far.

### 2.6. Inexact Cuts

In general, in literature on SDDP it is assumed that all subproblems within the algorithm are solved to global optimality. Moreover, to guarantee that only a finite number of different cuts can be generated, it is assumed that basic optimal solutions are determined. However, sometimes it may be more efficient to solve subproblems only approximately.

This may be the case for nonlinear subproblems, if exact global optimality takes too long to be achieved. We discuss this in more detail in Sect. 8. Solving subproblems approximately can also be beneficial in speeding up SDDP for linear problems of type $(P)$. In such case, cuts generated in early iterations are usually far away from the optimal point and, moreover, especially for the first stages, very loose. Therefore, such cuts become quickly dominated by other cuts [28].

From this perspective, it seems reasonable to solve the subproblems in early iterations, and especially early stages, more quickly and less accurately than in standard SDDP. This means, that inexact cuts are generated in those iterations. Assume that all forward pass problems are solved to $\delta^i_t$-optimality and all backward pass problems are solved to $\varepsilon^i_t$-optimality with $\delta^i_t$ and $\varepsilon^i_t$ bounded from above by $\bar{\delta}$ and $\bar{\varepsilon}$ for all $t = 1, \ldots, T$. Then, it can be shown that for all $t = 2, \ldots, T$ the condition

$$0 \leq \liminf_{i \to \infty} Q_i(x^t_i) - \limsup_{i \to \infty} Q_i(x^t_i) \leq (\bar{\delta} + \bar{\varepsilon})(T - t + 1)$$

is satisfied [28].

As shown in [28], as long as $\lim_{i \to \infty} \delta^i_t = 0$ and $\lim_{i \to \infty} \varepsilon^i_t = 0$, almost sure convergence of SDDP is still satisfied.

### 2.7. Parallelization

The performance of SDDP cannot only be improved by modifications of the algorithm itself, but also by its implementation and computational execution. Since several computational steps in SDDP are independent of each other, a significant performance improvement can be achieved by parallelization. This was specifically examined in [15, 37, 63].

In the forward pass, for all $t$, the subproblems (1.15) are solved for $|K|$ different scenarios, which are sampled independently from the whole scenario tree. The uncertain data and the trial solutions $x^k_{t-1}$ in each of those problems do only depend on scenario $k$. Therefore, the different scenarios $\xi^k, k \in K$, can be assigned to different processors. Assuming $P$ different processors, each processor is assigned $\frac{P}{|K|}$ scenarios and solves all corresponding subproblems. A master process is then used to aggregate the objective values and compute the upper bound estimate (1.16). This means that there is a synchronization point for all processors at the end of the forward pass.

In the backward pass, different parallelization concepts are possible. One approach is to parallelize the solution of the subproblems (1.18) for some stage $t$ and some $x^k_{t-1}$ for different realizations $\xi_{tj}, j = 1, \ldots, q_t$. However, this causes high communication costs even in constructing single cuts. Moreover, warm start techniques exploiting similarities between the subproblems cannot be applied [37]. Therefore, it is more convenient to follow a similar approach as in the forward pass and assign scenarios to different processors. This means that all subproblems with the same $x^k_{t-1}$ are solved by the same processor.

Evenly distributing the problems between processors, this way each processor
solves $\prod_{k} q_k$ subproblems. However, it is also possible to let the master process assign new scenarios to processes once they become idle instead of using a fixed assignment scheme [63].

After generating a new cut, each processor directly sends the cut information to the master process, which organizes and groups the cuts. In a standard parallelization, the processes are synchronized after each stage in the backward pass. This means that once the master process has received all cuts for stage $t$, it passes those cuts to all processors and the stage-$(t - 1)$ problems are solved. As stated in [37], this synchronization can be partially relaxed, though, to avoid waiting for single slow processors. Instead, the master process can assign stage-$(t - 1)$ subproblems to available processors even if not all cuts have been generated for stage $t$ yet. Numerical results show that such partial relaxation can significantly improve the computational performance of SDDP. However, the number of cuts to wait for to achieve an optimal trade-off between faster iterations and better approximation of $Q_t(\cdot)$ is problem-dependent.

3. Sampling in SDDP. Sampling is a central and crucial element of SDDP, as described in Sect. 1.3. In the forward pass, a finite number $|\mathcal{K}|$ of scenarios is sampled. Usually this sampling is done from a finite, but much larger set of scenarios, which can be represented as a finite scenario tree.

Applying SDDP to a multistage stochastic programming problem may also involve sampling in further ways, though. Sampling techniques can also be used to generate a scenario tree before applying SDDP if the probability distribution of $\xi$ is actually continuous, that is, if Assumption 7 is not satisfied. Alternatively, in this case, it is also possible to directly sample from the continuous distribution in the forward pass of SDDP instead of generating a scenario tree at all.

In this section, we discuss different sampling techniques, which can be used in SDDP for all these purposes. In our description we focus on the classical forward pass as described in Sect. 1.3. The case of continuous probability distributions is covered in Sect. 6.

The main requirement for any sample method used in the forward pass is that the samples should be independent and identically distributed (i.i.d.). This is important for two reasons:

1. Finite convergence of SDDP requires that any scenario is sampled infinitely many times with probability 1. In [62], this is referred to as the Forward Pass Sampling Property (FPSP), see Sect. 1.3. This property is satisfied if the sampling procedure produces independent samples.

2. In the stopping criterion of SDDP a confidence interval is used, which is built using the sample mean $\bar{\pi}_K$ and the sample variance $\left(\sigma_{\pi,K}^2\right)^2$. However, by the central limit theorem, even an approximate confidence interval can only be obtained for a sequence of i.i.d. random variables.

3.1. Monte Carlo Sampling. The simplest sampling method satisfying the above requirement is Monte Carlo (MC) sampling. Here, samples are drawn randomly from a given probability distribution. This is based on sampling from a uniform distribution and using appropriate transforms. Under stagewise independence (Assumption 9), this can be done independently for each stage $t = 1, \ldots, T$.

MC sampling was already used in our description of SDDP in Subsection 1.3, yielding a statistical upper bound $\bar{\pi}_K$ (1.16) and sample variance $\left(\sigma_{\pi,K}^2\right)^2$ (1.17).

The main advantage of MC sampling is that it is easy to implement and computationally tractable even for multiple variables. Moreover, the obtained upper bound
is an unbiased estimator of $\pi^i$ and according to the Law of Large Numbers converges to $\pi^i$ for $|K|$ approaching infinity. The convergence rate is $O(\sqrt{|K|})$, i.e., for a doubling of accuracy a quadrupling of the sample size is required. While this is slow, an advantage of MC sampling is that the convergence rate does not depend on the dimension of the random variable $\xi_t$.

Still, the sampling error of MC estimators can be significant. The variance of the upper bound estimator $\bar{\pi}_K^i$ can be estimated by $\frac{1}{|K|} (\sigma_{\pi, K})^2$. This means that the variance of $\bar{\pi}_K^i$ can be reduced by increasing the number of samples $|K|$ or by reducing the sample variance $(\sigma_{\pi, K})^2$.

Increasing the sample size may look promising at first glance, but may become computationally intractable in practice [54]. Recall that for every sampled scenario all subproblems have to be solved in the forward pass and additionally, the subproblems for all descendants $\xi_{tj}, j = 1, \ldots, q_t$, have to be solved on each stage in the backward pass. In fact, increasing the sample size by too much contradicts the fundamental idea of SDDP to reduce the effort of traversing the whole scenario tree in each iteration. Therefore, the more promising approach is combining MC sampling with variance reduction techniques [54].

3.2. Variance Reduction Techniques. Incorporating variance reduction techniques into sampling in SDDP is studied by Homem-de-Mello, Matos and Finardi [40] and Parpas et al. [55]. For a review on sampling techniques in stochastic programming in general, see [39].

Randomized Quasi-Monte Carlo Sampling. In [40], it is proposed to use Quasi-Monte Carlo (QMC) sampling within SDDP. In this case, instead of randomly sampling from the uniform distribution, a deterministic sequence of points $u_1, \ldots, u_N$ from $(0, 1)^{p_t}$ is chosen, with $p_t$ the dimension of random vector $\xi_t$. This is done in such way that the discrepancy between the empirical distribution of the samples and the uniform distribution is as low as possible. That means, that the sequences are chosen with the aim to obtain sampled points which fill $(0, 1)^{p_t}$ as homogeneously as possible. Thus, after an appropriate transformation they provide a better representation of the scenario tree than randomly sampled points.

A drawback of QMC methods is that the sample points are not random, the obtained estimator is biased and no confidence interval can be established. Randomized QMC (RQMC) methods, where the choice of QMC points is combined with some kind of randomness, avoid this drawback and allow for standard error estimation [40].

Compared to MC sampling, RQMC methods achieve better convergence rates of $O(|K|^{-1}(\log|K|)^{p_t})$ and, thus, are considered more efficient. However, the convergence rate depends on the dimension $p_t$ of $\xi_t$.

Latin Hypercube Sampling. In Latin Hypercube Sampling (LHS) (or stratified sampling) [50], the space $(0, 1)^{p_t}$ is divided into equidistant subintervals and then scenarios are sampled from each subinterval in such way that in each row and column of the grid only one point is sampled. This is illustrated in Fig. 9 (a).

In this way, again, a more even distribution of the sample points can be obtained. Therefore, compared to MC sampling, using LHS the variance can be reduced. On the flipside, poor space-filling or correlation between the sample points has to be ruled out, see Fig. 9 (b), which requires significant additional effort.

Incorporation into SDDP. It is important to notice that while reducing the variance compared to the classical MC estimators, scenarios sampled by RQMC and LHS are no longer i.i.d. Therefore, both sampling techniques cannot be incorporated into SDDP without modification, if convergence properties and the stopping criterion
should not be compromised. Homem-de-Mello et al. therefore suggest to build sampling blocks [40]. This means that the total number of samples \(|\mathcal{K}|\) is divided into \(M\) blocks \(\ell = 1, \ldots, M\) with \(M \geq 5\) a divisor of \(|\mathcal{K}|\). Then, for each block \(\ell\), \(|\mathcal{K}'| := |\mathcal{K}|/M\) scenarios are obtained using conditional sampling with RQMC or LHS, which are not independent. For each \(k' \in \mathcal{K}'\), values \(v^i(\xi^{k'})\) are determined and averaged to \(v_{i,\ell}^\mathcal{K}\).

This is repeated for each block \(\ell\). Then, the average \(\bar{v}_\mathcal{K}\) of all values \(v_{i,\ell}^\mathcal{K}\), \(\ell = 1, \ldots, M\), and the sample variance are determined. Since the scenarios of different blocks are independent, this still yields a useful confidence interval to stop the algorithm.

Another challenge reported in [40] is that it is computationally expensive to generate samples using RQMC for high dimensions. To reduce the computational effort, it may be reasonable to apply RQMC only to important components, e.g., to early stages of the scenario tree, and standard MC or LHS to the other ones. This strategy is called padding and applied after 6 or 12 stages in the numerical tests in [40].

The numerical results in [40] imply that RQMC and LHS both lead to upper bounds \(\bar{v}_\mathcal{K}\) oscillating around the lower bound \(\underline{v}\) more quickly compared to the MC approach.

**Importance Sampling in SDDP.** In [55], Parpas et al. propose incorporating importance sampling, which is a well-known and efficient variance reduction technique, into SDDP. Importantly, in difference to the previously described techniques, it can be used to obtain i.i.d. samples in the forward pass.

The main idea of importance sampling in general is to attach different importance to subregions of the sample space and to sample more often from subregions of higher importance. In context of SDDP, this means that it is sampled with priority from scenarios that contribute more to the value of the expected value functions \(Q_1(\cdot)\).

This is achieved by sampling from a different than the original distribution, the so-called importance sampling distribution, but correcting the bias introduced by this
different sampling. Then, an importance sampling estimate of $\overline{v}$ can be calculated as

$$\overline{v}^{IS,i}_{K} := \frac{1}{|K|} \sum_{k \in K} v^i(\xi^k) \Lambda(\xi^k)$$

with $\Lambda(\xi) := \frac{f(\xi)}{g(\xi)}$, where $f$ denotes the original distribution and $g$ the importance sampling distribution. The likelihood function $\Lambda(\xi)$ is used to correct for sampling from the wrong distribution.

It can be shown that importance sampling can reduce the variance of sampling estimators significantly. In the SDDP case, as shown in [55], the variance is minimized for choice

$$g^*(\xi) := \frac{|Q(x^i_t)|}{E_f|Q(x^i_t)|} f(\xi).$$

However, clearly, this zero-variance distribution is a theoretical construct and not known, which is referred to as curse of circularity. Therefore, Parpas et al. propose the following approach to obtain importance sampled scenarios [55]:

1. A Markov Chain Monte Carlo (MCMC) algorithm is used to generate samples from the zero-variance distribution $g^*$, which is possible even without actually knowing it.
2. Using Kernel Density Estimation an approximate zero-variance distribution is constructed from the samples obtained in (1).
3. The approximate distribution from (2) is used for the actual importance sampling.

In their numerical experiments, SDDP with importance sampling is shown to outperform MC and QMC sampling based methods, in case that it is difficult to sample from the original probability distribution and that the original problem has moderate or high variance.

Importance sampling is also applied in SDDP in other context, for example to obtain upper bounds for $v^*$ in a risk-averse setting, see also Sect. 7.

4. Stopping Criterions for SDDP. In each iteration $i$ of SDDP, a valid lower bound $\overline{v}^i$ and an estimate $\overline{v}^i_K$ of the upper bound $\overline{v}$ are determined. Since the upper bound is only statistical, an important question is when to consider an obtained policy $(x^i_{t|t}(\xi^k))_{t=1}^{T}$ as (approximately) optimal and to stop the SDDP method. If the stopping criterion is too conservative, the algorithm may iterate much longer than required, if it is too optimistic, then SDDP may stop prematurely.

The original confidence criterion. In their seminal work on SDDP, Pereira and Pinto propose to use a confidence interval based stopping criterion [56], which we already addressed in Sect. 1.3. An approximate confidence interval for $\overline{v}$ is determined as follows.

The values $v^i(\xi^k)$ from (1.16) are i.i.d. random variables with expected value $\overline{v}^i$ and variance $\sigma^2$. Moreover, we know the sample mean $\overline{v}^{i,k}_{K}$ (1.16) and sample variance $(\sigma^2)^2$ (1.17). We can then define a standardized random variable

$$Z^i_{K} := \frac{\overline{v}^{i,k}_{K} - \overline{v}^i}{\frac{\sigma^2}{\sqrt{K}}}.$$  

According to the Central Limit Theorem, this random variable asymptotically, that is, for $|K| \to \infty$, follows a standard normal distribution $N(0,1)$. This implies
that for sufficiently large $|\mathcal{K}|$, $Z^i_k$ is approximately standard normal distributed. This remains also true, if $\sigma^i$ is replaced by the sample standard distribution $\sigma^i_{v,K}$.

Due to symmetry of the standard normal distribution, it follows

$$\mathbb{P}(-z_{1-\alpha/2} \leq Z^i_k \leq z_{1-\alpha/2}) \approx 1 - \alpha,$$

where $z_{1-\alpha/2}$ denotes $(1 - \frac{\alpha}{2})$-quantiles of $\mathcal{N}(0,1)$. Inserting (4.1) and rearranging yields

$$\mathbb{P}\left(\frac{v^i_k - z_{1-\alpha/2}}{\sigma_{v,K}} \leq \frac{\overline{v}^i - \overline{v}^i_k}{\frac{\sigma_{v,K}}{\sqrt{|\mathcal{K}|}}} \right) \approx 1 - \alpha.$$

Choosing $\alpha = 0.05$, we have $z_{1-0.05/2} = 1.96$ and the confidence interval follows as stated in Sect. 1.3.

If $\overline{v}^i$ is included in this confidence interval, $i.e.$, exceeds its lower bound, the SDDP method terminates in iteration $i$.

As pointed out in literature, this stopping criterion has several flaws. Shapiro analyzes that the higher the sample variance $(\sigma^i_{v,K})^2$, the earlier $\overline{v}^i$ exceeds the lower end of the confidence interval, which provides a misguided incentive to increase the sample variance. The same is true for increasing the confidence $1 - \alpha$, which contradicts the intuition behind $\alpha$ [76].

Additionally, the above stopping criterion may favor premature stopping, as it is rather unlikely that $\overline{v}^i$ is located exactly at the lower bound of the confidence interval.

**Alternative criterions.** For those reasons, different stopping criterions have been put forward in literature. As a starting point, an early discussion of statistical stopping criterions for sampling-based stochastic programming methods is provided in [53].

Instead of using statistical criterions, it may be convenient to stop SDDP after a fixed number of iterations or if the lower bounds $\overline{v}^i$ have stabilized. Neither guarantees that the considered problem $(P)$ is solved to optimality [76], but both approaches are prominently used in practical applications, especially when the problems become too large to observe convergence in reasonable time.

In [76], Shapiro proposes a statistical, but more conservative stopping criterion as the original one, which incorporates the upper instead of the lower bound of the confidence interval. To be precise, SDDP stops if the difference between the upper bound $\overline{v}^i_k + 1.96 \frac{\sigma_{v,K}}{\sqrt{|\mathcal{K}|}}$ and $\overline{v}^i$ is sufficiently small.

Homem-de-Mello et al. take a different perspective [40]: Since hypothesis tests and confidence intervals can be regarded as two sides of the same coin, they interpret the original SDDP stopping criterion in terms of a hypothesis test with hypotheses:

$$H_0 : \overline{v}^i = \overline{v}'^i, \quad \text{against} \quad H_1 : \overline{v}^i \neq \overline{v}'^i.$$\nonumber

The null hypothesis $H_0$ is tested using the test statistic $\overline{v}_{K}$, which is assumed to be approximately normal distributed. This can be reasoned using the Central Limit Theorem for sufficiently large $|\mathcal{K}|$.

It can be shown that $H_0$ is rejected in iteration $i$ if and only if $\overline{v}^i_k > \overline{v}'^i + c$ or $\overline{v}^i_k < \overline{v}'^i - c$ with

$$c = \Phi^{-1}\left(1 - \frac{\alpha}{2}\right) \frac{\sigma_{v,K}}{\sqrt{|\mathcal{K}|}}.$$

Here, $\Phi$ denotes the cumulative distribution function of the standard normal distribution $\mathcal{N}(0,1)$ and $\alpha$ the significance level. Thus, the region of acceptance for
$H_0$ can be determined to

$$v^i \in [\tau_{v^i}^K - c, \tau_{v^i}^K + c],$$

which for $\alpha = 0.05$ equals the confidence interval from above.

By choosing $\alpha$, the type I error (rejecting optimality although SDDP has converged) can be controlled. However, this comes at the cost of a possibly high type II error (stopping the algorithm prematurely) [40]. In fact, it is shown that the power of the test may be low when $\tau_{v^i}^K$ is larger than but relatively close to $v^*$ or when the sample size $|K|$ is small, which implies that a type II error is likely.

To avoid stopping prematurely, the authors propose a modified hypothesis test controlling type I and type II errors simultaneously [40]. First, it is checked whether

$$\rho_i^K := \frac{\tau_{v^i}^K - \Phi(1 - \alpha)\frac{\sigma_{v^i,K}}{\sqrt{|K|}}}{\tilde{\zeta}^i},$$

is larger than 1. $\rho_i^K$ describes the ratio between the lower bound of the region of acceptance related to a one-sided hypotheses test with $H_0 : \tilde{\zeta}^i \leq v^i$, and the lower bound $v^i$. If it is larger than 1, then optimality is rejected. This is completely in line with the original SDDP hypothesis test.

However, if $\rho_i^K \leq 1$, optimality is not directly retained. Instead, the idea is that the probability of a type II error is bounded by some scalar $\gamma$ when the true upper bound exceeds the lower bound by more than a percentage $\delta$. For given $\gamma$ and $\alpha$, this $\delta$ can be calculated to

$$\delta_i = (z_{1-\alpha} + z_{1-\gamma})\frac{\sigma_{v^i,K}}{\tilde{\zeta}^i \sqrt{|K|}}.$$

If $\rho_i^K \leq 1$ and $\delta_i$ is below some predefined threshold $\overline{\delta}$, the algorithm stops. This is done, as in case that the true relative optimality gap is larger than $\delta$, the type II error is bounded by $\gamma$ and, thus, controlled. Otherwise, the algorithm proceeds.

Numerical experiments with $\overline{\delta} = 0.1$ and $\gamma = 0.05$ indicate that this stopping criterion is effective in preventing SDDP from premature stopping [40].

Summarizing, various attempts have been made to address the aforementioned issues of stopping criterions for SDDP, yielding significant computational improvements over the original confidence criterion. However, still to this day, stopping SDDP remains a challenge.

5. Exact upper bounds in SDDP. Since the vague stopping criterion for SDDP stems from estimate $\tau_{v^i}^K$ providing a statistical upper bound only, an alternative way to address the issues discussed in the previous section, is to determine exact valid upper bounds for $v^*$. This idea has drawn interest in the research community recently.

In principle, an intuitive way to determine valid upper bounds for the expected value functions $Q_t(\cdot)$, and thus also for $v^*$, can be deduced from the characteristics of convex functions. It is well-known that for a convex function, all tangents lie below or on the graph of this function. Cutting-plane methods are essentially rooted in this property [43], which also SDDP uses to construct cuts in the backward pass. However, similarly, all secants of a convex function lie above or on the graph of this function. Therefore, an inner approximation of a convex function is possible by a convex combination of a given set of function values. Outside of the convex hull of the given points, the inner approximation can be extended for example by some overestimation with Lipschitz constants. This is illustrated in Fig. 10.
In principle, there are two different approaches to apply this idea and determine inner approximations for $Q_t(\cdot)$, $t = 1, \ldots, T$. One uses the usual SDDP perspective, which we refer to as primal, and one is related to some dual perspective on SDDP and its value functions \cite{46}.

A primal inner approximation approach. Since the expected value functions $Q_t(\cdot)$ are not known, they cannot be evaluated directly. However, similar to the cut approximations $\Omega_t(\cdot)$ constructed in NBD or SDDP, which under-approximate $Q_t(\cdot)$, we can determine inner approximations $\overline{Q}_t(\cdot)$, which over-estimate the expected value functions. Thus, in the primal inner approximation approach we define over-approximating subproblems in the dynamic programming equations by replacing $Q_t(\cdot)$ with $\overline{Q}_t(\cdot)$ for all $t = 1, \ldots, T$. This idea was first used by Philpott, de Matos and Finardi \cite{61}. As they consider only the right-hand side of $(P)$ to be uncertain, we adopt this assumption, although it is not required.

For stages $t = T - 1, \ldots, 2$, each element $m$ in a given set of points $x_t^1, \ldots, x_t^{M_t-1}$ and each $\xi_{tj}$, $j = 1, \ldots, q_t$, the following subproblem can be solved by backward recursion:

\begin{equation}
\overline{Q}_t(x_{t-1}^m, \xi_{tj}) := \min_{x_t} c_t^\top x_t + \overline{Q}_{t+1}(x_t) \quad \text{s.t.} \quad W_t x_t = h_t(\xi_{tj}) - T_{t-1} x_{t-1}^m \quad x_t \geq 0.
\end{equation}

Here, $\overline{Q}_{t+1}(\cdot)$ denotes an upper bound function, which is defined by

\begin{equation}
\overline{Q}_{t+1}(x_t) := \min_{w} \sum_{m=1}^{M_t} w_m \overline{Q}_{t+1}(x_t^m) \quad \text{s.t.} \quad \sum_{m=1}^{M_t} w_m x_t^m = x_t \quad \sum_{m=1}^{M_t} w_m = 1 \quad w_m \geq 0, \quad m = 1, \ldots, M_t,
\end{equation}
with \( \mathcal{Q}_{t+1}(x_{t}^m) = E \left[ \mathcal{Q}_{t+1}(x_{t}^m, \xi_{t+1,j}) \right] \). That is, the upper bound function \( \mathbf{\Sigma}_{t+1}(\cdot) \) is determined by a convex combination of the \( M_t \) values \( \mathcal{Q}_{t+1}(x_{t}^m) \) obtained for the following stage.

By recursion, it can be shown that

\[
\mathcal{Q}_t(x_{t-1}^m, \xi_{t,j}) \geq Q_t(x_{t-1}^m),
\]

for all \( m = 1, \ldots, M_{t-1} \) and \( j = 1, \ldots, q_t \). This implies

\[
\mathcal{Q}_t(x_{t-1}^m) \geq Q_t(x_{t-1}^m).
\]

The first stage problem then yields

\[
\psi^A := \begin{cases} 
\min_{x_t} & c_1^\top x_1 + \mathbf{\Sigma}_2(x_1) \\
\text{s.t.} & W_1 x_1 = h_1 \\
& x_1 \geq 0,
\end{cases}
\]

with \( \psi^A \) an exact valid upper bound to \( v^* \).

The main challenge with this approach is how to choose the set of points \( x_{t-1}^m \) with \( m = 1, \ldots, M_{t-1} \). In principle, the points \( x_1^1, \ldots, x_1^{M_t} \) should be chosen such that as much of the state space is spanned as possible. However, choosing (at least some of) those points as extreme points leads to \( M_t \geq 2^n \) points, i.e., the number of required points grows exponentially in the dimension of the state. For this reason, Philpott et al. suggest to simply use the trial points from the SDDP forward pass, which is reported to give reasonable results [61].

Even using the SDDP trial points, the computational effort may become excessive, though. Similarly to the SDDP backward pass, subproblems (5.1) have to be solved for each stage \( t \), each point \( x_{t-1}^m \) and each descendant scenario \( \xi_{t,j} \). However, in the usual backward pass, the number of points \( x_{k-1}^m \) depends on the number of samples \( |K| \) drawn in the forward pass and, thus, remains stable, unless the sampling is changed. On the contrary, here, the number of points \( x_{t-1}^m \) grows with each iteration, as it is built by all previous trial solutions. For this reason, it is suggested to only use the upper bound computation every few iterations (in numerical experiments it is used every 200, 600 or 1000 cuts) and not to incorporate it into the backward pass [61]. This hinders using the upper bounds \( \psi^A \) in the stopping criterion of SDDP, though. Moreover, no convergence proof for the upper bounds is provided in [61].

Bauke et al. provide a different perspective on this idea [4]. Instead of (5.2), they use its dual representation:

\[
\mathbf{\Sigma}_{t+1}(x_t) = \begin{cases} 
\max_{\mu, \lambda} & x_t^\top \lambda + \mu \\
\text{s.t.} & (x_t^m)^\top \lambda + \mu \leq \eta_t^m, \quad m = 1, \ldots, M_t
\end{cases}
\]

with \( \eta_t^m := \mathcal{Q}_{t+1}(x_{t}^m) \).

This dual representation shows that, essentially, \( \mathbf{\Sigma}_{t+1}(x_t) \) can be equivalently described by maximizing over the coefficients of all supporting hyperplanes for all points \( (x_t^m, \mathcal{Q}_{t+1}(x_{t}^m)) \), \( m = 1, \ldots, M_t \).

In [4], the dual problem is additionally regularized, i.e., it is enhanced by constraint

\[
\|\lambda\|_q \leq L_t,
\]
with $L_t$ denoting a Lipschitz constant of $\overline{Q}_t(\cdot)$ with respect to the norm $\| \cdot \|_p$ with $\frac{1}{p} + \frac{1}{q} = 1$ and $p, q > 0$. This way, a reasonable approximation is also achieved for points outside of the convex hull of the set defined by the points $x_t^m, m = 1, \ldots, M_t$.

Using this expression for the inner approximation functions, Baucke et al. propose a deterministic algorithm for multistage stochastic convex programs. In their case, subproblems (5.1) are solved in each backward pass iteration, and $\overline{Q}_{t+1}(x_t)$ is updated by adding constraint $(x_t^m)^\top \lambda + \mu \leq \overline{Q}_{t+1}(x_t^m)$ for the current iterate $x_t^m$.

The proposed algorithm differs in many regards from SDDP, though. Most significantly, the method is deterministic, as it contains no sampling at all. Instead, a sample path has to be sampled to obtain a further improvement in the upper and lower bounds, which avoid the stopping issues of SDDP, and the deterministic state optimum $\overline{Q}_t(\cdot)$ is updated deterministically in each iteration $i$ by choosing the realization $\xi^i$ and corresponding trial solution, for which the difference between upper bound function $\overline{Q}_{t+1}(\cdot)$ and cut approximation $\overline{Q}_{t+1}(\cdot)$ is the largest. Therefore, in each forward pass, only one scenario path $\xi^i$ with trial solutions $x_t^i, t = 1, \ldots, T$, is determined.

Moreover, the proposed method uses a multi-cut approach, see Sect. 2, and the considered decision process differs, as decisions have to be taken before the stage-$t$ uncertainty realizes.

While the scenario path $\xi^i$ is chosen such that it promises the largest improvement, this may also favor getting stuck or improving the bounds at regions far away from optimum [46]. The main advantages of the proposed method are the deterministic upper bounds, which avoid the stopping issues of SDDP, and the deterministic state trajectory, which avoids sampling and thus ensures finite deterministic convergence [4]. Numerical results indicate that especially in later iterations, when a specific sample path has to be sampled to obtain a further improvement in the upper and lower bounds, a deterministic scenario choice may prove beneficial.

A dual inner approximation approach. Recently, Leclère et al. proposed an SDDP extension with deterministic upper and lower bounds based on employing duality concepts [46]. Basically, the proposed method also relates to the definition of an inner approximation function $\overline{Q}_{t+1}(x_t)$ by the regularized dual problem (5.4). However, a totally different perspective is chosen.

Let $f$ be an extended real-valued function. Then its convex conjugate, also called Fenchel conjugate, is defined as

\[
(5.5) \quad f^*(\lambda) := \sup_{x \in \mathbb{R}^n} \lambda^\top x - f(x).
\]

Using convex conjugates, several duality concepts can be generalized, see [66].

We consider the convex conjugates $D_t(\cdot) := Q_t^*(\cdot)$ of the value functions $Q_t(\cdot), t = 1, \ldots, T$, of problem $(P)$, which are also referred to as dual value functions in [46]. As shown by the authors, similar to the value functions $Q_t(\cdot)$, also the dual value functions $D_t(\cdot)$ satisfy some dynamic recursion with linear subproblems on each stage. In our setting, this recursion denotes

\[
(5.6) \quad D_t(\lambda_{t-1}) := \begin{cases} 
\min_{\mu_t, \lambda_t} & \mathbb{E} \left[ -h_t(\xi_t)^\top (\mu_{t1} - \mu_{t2}) + D_{t+1}(\lambda_t) \right] \\
\text{s.t.} & -W_t^\top (\mu_{t1} - \mu_{t2}) = \lambda_t \\
& T_{t-1}^\top \mathbb{E} [\mu_{t1} - \mu_{t2}] - \mathbb{E} [\mu_{t3}] + \lambda_{t-1} = 0 \\
& \|\lambda_t\|_\infty \leq L_t \\
& \mu_t \leq 0.
\end{cases}
\]

For more general formulations, for example if local control variables are introduced,
Using this dynamic recursion, it is possible to apply an SDDP-type algorithm to $D_t(\cdot)$, using iteratively improving outer approximations $\mathcal{D}_i^t(\cdot)$ for $D_t(\cdot)$. Analogously to SDDP, this iterative method yields a converging deterministic lower bound for the first stage optimal value, i.e., $\mathcal{D}_i^t(x_0) \leq D_t(x_0)$, assuming some first stage parameter $x_0$. Applying duality theory again, we obtain

$$\mathcal{D}_i^t(x_0) \geq \mathcal{D}_i^t(x_0) = Q_t^{**}(x_0) = Q_t(x_0) = v^*.$$ 

Hence, $(\mathcal{D}_i^t(x_0)$ defines a sequence of converging deterministic upper bounds for $v^*$.

This result can be used to derive inner approximations of the primal value functions $Q_t(\cdot)$ as Lipschitz regulations of the convex conjugate of the outer approximations $\mathcal{D}_i^t(\cdot)$ [46]. To be precise, such inner approximation $\mathcal{D}_i^{t+1}(\cdot)$ can be computed by solving a regularized version of dual problem (5.4), but with the breakpoints $x_m^{t-1}$ (and the corresponding function values $\eta_m^t$) determined by the slopes of the dual outer approximation [46]. Thus, the dual SDDP method reveals a fundamental connection between inner approximations of $Q_t(\cdot)$ and outer approximations of its dual function $D_t(\cdot)$.

It can also be shown that the inner approximations yield a converging policy whose expected value $C_I^{A,i}$ is bounded by $v^*$ and $\bar{v}^{A,i} := \mathcal{D}_0(x_0)$ in each iteration [46]. As also discussed in [61], the upper bound is not guaranteed to bound the expected value $C^{O,A,i}$ of the standard SDDP forward pass policy $(x_{ik}^t)_{t=1}^T$, though.

To incorporate their dual SDDP method and inner approximations into SDDP, Leclère et al. propose the following framework [46]:

1. Run a forward pass of (primal) SDDP, yielding trial solutions $(x_i^t)_{t=1}^T$ for the sampled scenario path (the authors choose $|\mathcal{K}| = 1$).
2. Run a backward pass of SDDP using the trial solutions $x_i^t$, obtaining new slopes $\pi_i^t$ from the cuts.
3. Run a backward pass of dual SDDP using the slopes $\lambda_i^t = \pi_i^t$, obtaining new cuts for the dual problem.
4. Run a forward pass of dual SDDP, to obtain a new dual trajectory of trial solutions and update the cuts along this trajectory.

One computational drawback of this approach is that dual SDDP is much more computationally expensive than (primal) SDDP, as for the chosen sample path, for all $t = 1, \ldots, T$, all realizations $\xi_{t,j}$, $j = 1, \ldots, q_t$, have to be considered in the forward as well as backward pass due to coupling constraints in (5.6). This also hampers the application of a dual stopping criterion solely based on the deterministic bounds for $v^*$, at least for problems requiring a very large number of iterations [46].

6. SDDP for Continuous Uncertainty [relaxing Assumption 7]. So far, we assumed the uncertainty in $(P)$ to be modeled by some discrete and finite random process, see Assumption 7. This allows for a finite scenario representation. However, in many practical applications, this assumption is not justified. For example, if the stochastic process governing the uncertain data is modeled by some time series model like ARIMA, the random error terms are usually assumed to follow a continuous distribution [76].

As pointed out in Sect. 1.1, for problems with sizes of practical interest, relaxing Assumption 7 yields a computationally intractable problem, which we denote by $(\tilde{P})$ for distinction from $(P)$. Therefore, if the true distribution of the random data is continuous, usually, a discretization is used. That is, the stochastic process $(\xi_t)_{t=1}^T$ is
approximated using a scenario tree with finitely many, say \( N \), realizations. This yields an approximating problem \((\tilde{P}_N)\), which then can be handled by SDDP, as presented in Sect. 1.

Approximating the original data process \((\xi_t)_{t=1}^T\) with a very large and fine scenario tree may yield a very good representation of the original process, but not improve the computational tractability by much. On the other hand, a very rough discretization may yield a problem \((\tilde{P}_N)\), which can be solved efficiently by SDDP, but does not provide any information about the solution to the true problem \((\tilde{P})\) [44]. Therefore, a central question arising with this approach is how the approximating scenario tree can be determined appropriately. Related are the questions how close the obtained solution for \((\tilde{P}_N)\) is to the solution of \((\tilde{P})\) and how policies obtained by SDDP can be evaluated with respect to the original problem.

6.1. Scenario Tree Generation. To generate scenario trees from given continuous distributions various strategies exist. An overview and an empirical comparison of different techniques is provided by Lönhdorf [49]. The techniques are assigned to different categories such as sampling, probability metrics, Vronoi cell sampling and Moment matching. In context of SDDP, so far, mostly the first two groups have been considered.

Scenario tree generation based on probability metrics includes techniques where the approximation error of the scenario tree with respect to the original process is controlled based on theoretical stability results for stochastic programs. Such results, in turn, make use of probability metrics to measure distances between random objects [49]. Important contributions for this type of scenario generation methods are [35, 36, 51, 58]. Dupacova et al. [21] as well as Heitsch and Römisch [35] propose scenario reduction techniques, which attempt to reduce the number of scenarios in a tree without compromising the quality of the obtained solutions.

The most frequently used scenario tree generation category for SDDP is sampling. Here, conditional sampling from the true probability distribution is used to create an approximating scenario tree. Often, this technique is referred to as Sample Average Approximation (SAA), especially, if classical Monte Carlo sampling is used. SAA and its properties are discussed in detail in the following subsection.

The obtained SAA problem is then solved by SDDP to obtain an approximate solution of the original problem \((\tilde{P})\). This means, that within SDDP a fixed scenario tree is considered as usual.

We note that it is also possible to sample from the true distribution in the forward pass of SDDP instead of creating a scenario tree first, see for example [55]. However, in such case, the convergence properties from Sect. 1.3 are not guaranteed.

6.2. Sample Average Approximation. In this subsection, we present SAA for multistage stochastic optimization problems of form \((\tilde{P})\) and discuss some of its main properties. As SAA is a broad field of research on its own, we focus on central results, relevant in the context of SDDP and omit technical derivations. Instead, we refer the interested reader to [78].

SAA and SDDP. A sample average approximation of the original problem \((\tilde{P})\) is constructed by replacing the distributions \(F_\xi\) of the true continuous random vectors \(\xi_t\) by some empirical distribution \(F_N\) [76]. As stated before, the empirical distributions over all stages \(t = 1, \ldots, T\) can be represented by a finite scenario tree, which approximates the true data process \((\xi_t)_{t=1}^T\).

In the multistage setting, conditional sampling has to be applied to set up such scenario tree. Otherwise, the statistical lower bounds obtained by SAA are not guar-
anteed to be consistent \[73\]. The main reason for this is that the conditional expected value functions are not approximated adequately. Conditional sampling, in contrast, assures that each conditional expectation is approximated by a sample average \[40\]. In general, conditional sampling is achieved as follows: \(\xi_1\) is deterministic. So first, finitely many realizations \(\tilde{\xi}_2, \ldots, \tilde{\xi}_{N_2}\), with \(\ell = 1, \ldots, N_2\), are sampled from the original distribution of \(\xi_2\). Then, for each \(\tilde{\xi}_2\), again finitely many realizations \(\tilde{\xi}_3, \ldots, \tilde{\xi}_{N_3}\), with \(j = 1, \ldots, N_3\), are sampled. This is repeated for all stages up to stage \(T\). In literature, this is called \textit{independent conditional sampling} \[76\].

If we assume stagewise independence of the true data process \((\xi_t)_{t=1}^T\), i.e., under Assumption 9, it is desirable to preserve this property in the SAA problem. This is particularly relevant, since stagewise independence is a crucial ingredient of SDDP, as explained in Sect. 1. To preserve stagewise independence, the conditional sampling scheme can be applied, but using the same realizations \(\tilde{\xi}_t, \ell = 1, \ldots, N_t\), for all nodes of the same stage. This means that in the corresponding scenario tree each stage-\(t\) node has the same branches \[76\]. In literature, this kind of sampling is called \textit{identical conditional sampling} \[78\].

Either way, the obtained SAA problem and the corresponding scenario tree have a total number of \(N = \sum_{t=2}^T N_t\) scenarios, i.e., the number of scenarios is exponentially growing in the number of stages \[76\].

For the SAA problem \((P_N)\), for each stage \(t = 2, \ldots, T\) and sample \(\ell = 1, \ldots, N_t\), the dynamic programming equations can be written as

\[
\bar{Q}_t(x_{t-1}, \tilde{\xi}_t^t) := \min_{x_t} c_t^\top (\tilde{\xi}_t^t)x_t + \bar{Q}_{t+1}(x_t) \quad \text{s.t.} \quad W_t(\tilde{\xi}_t^t)x_t = h_t(\tilde{\xi}_t^t) - T_{t-1}(\tilde{\xi}_t^t)x_{t-1} \geq 0
\]

where

\[
\bar{Q}_t+1(x_t) := \frac{1}{N_{t+1}} \sum_{\ell=1}^{N_{t+1}} \bar{Q}_{t+1}(x_t, \xi_{t+1})
\]

and \(\bar{Q}_{T+1} \equiv 0\).

For the first stage, we obtain

\[
\bar{v}_N := \min_{x_1} c_1^\top x_1 + \bar{Q}_2(x_1) \quad \text{s.t.} \quad W_1x_1 = h_1, \quad x_1 \geq 0.
\]

The problems defined by (6.1), (6.2) and (6.3) can be solved by SDDP, see Sect. 1. Note, however, that in contrast to the problems considered there, the SAA problems are random as they depend on a sample \((\tilde{\xi}_t^1, \ldots, \tilde{\xi}_t^T)\) from the true data process \((\xi_t)_{t=1}^T\).

**SAA Properties.** Since the aim is to solve the original problem \((P)\), the central question is how the solution and the bounds obtained by applying SDDP to the SAA problem \((P_N)\) relate to the solution of \((P)\). We denote the optimal value of \((P)\) by \(v^*\) and the bounds obtained by SDDP in iteration \(i\) with \(\bar{v}_i\) and \(\bar{v}_Ki\). We summarize important properties of SAA.

(P.6.1) It can be shown that the optimal value \(\bar{v}_N\) provides a consistent estimator of the true optimal value \(v^*\), i.e., \(\lim_{N \to \infty} E[\bar{v}_N] = v^*\) with probability 1 \[76\]. This can be proven using the Law of Large Numbers \[78\]. The intuition
behind this is that asymptotically, the scenario tree structure of the true process \((\xi_t)_{t=1}^T\) is recovered. In practical applications, increasing \(N_t\) to infinity is computationally intractable.

\[\Pr[x_t \text{ is determined}, \text{yielding a reasonable approximation of } \bar{x}]\]

is a feasible and appropriate policy \((P_N)\).

Another problem identified by Shapiro \[76\] is that for large sample sizes, it can be even impossible to solve the SAA problem \((P_N)\) to optimality in reasonable time. The accuracy of the solution cannot be guaranteed anymore \[73\].

Since each iteration \(i\) of SDDP, we have \(\bar{v}^i \leq \bar{v}_N\), it follows \(\Pr[\bar{v} \leq \bar{v}^i] \leq \bar{v}^* \[76\].

Thus, the SDDP lower bound is also a statistical lower bound for \(\bar{v}^*\). Note, however, that both, \(\bar{v}_N\) and \(\bar{v}^i\), are lower bounds in expectation only, whereas this is not clear for one specific sample and SAA problem \((P_N)\).

Applying SDDP to (6.1), (6.2) and (6.3), a feasible policy for \((P_N)\) is determined. Thus, as shown in Sect. 1.3, \(E_{\xi}[(\sum_{t=1}^{T} c_t^T (\xi_t)x_t^i(\xi_t))]\) is an upper bound to the optimal value \(\bar{v}_N\) of the SAA problem \((P_N)\) conditional on the chosen sample. \(E_{\xi}\) denotes the expectation taken with respect to the corresponding finite scenario tree.

Since the policy defined by SDDP is feasible for \((\bar{P})\) as well, the expectation \(E_{\xi}[(\sum_{t=1}^{T} c_t^T (\xi_t)x_t^i(\xi_t))]\) yields an upper bound for \(\bar{v}^*\).

The sample mean \(\bar{v}_N^i\) determined in the forward pass of iteration \(i\) in SDDP is an unbiased and consistent estimator of \(E[(\sum_{t=1}^{T} c_t^T (\xi_t)x_t^i(\xi_t))]\). Hence, \(E[\bar{v}_N^i]\) is an upper bound for \(\bar{v}^*\).

Despite these properties, solving multistage problems \((\bar{P})\) using SAA may be computationally intractable. The bias and the variance of \(\bar{v}_N\) can grow fast in the number of stages and hence, may be already very high even for a mild number of stages. Consequently, the obtained lower bounds by applying SDDP to SAA can be quite loose \[78\]. This also affects the quality of the upper bound estimators in SDDP. Moreover, the variability of the upper bound estimators often increases with the number of stages \[78\].

In fact, Shapiro shows that even under relatively complete recourse (Assumption 5) and stagewise independence of the true data process \((\xi_t)_{t=1}^T\) (Assumption 9), the total number of scenarios required in SAA problem \((P_N)\) to solve the true problem \((\bar{P})\) with a reasonable accuracy \(\varepsilon > 0\) grows exponentially in the number of stages \[74\]. Therefore, he proposes to use smaller sample sizes \(N_t\) for later stages, although the accuracy of the solution cannot be guaranteed anymore \[75\].

Another problem identified by Shapiro \[76\] is that for large sample sizes, it can be even impossible to solve the SAA problem to optimality in reasonable time. The convergence results in Sect. 1 prove almost sure finite convergence of SDDP, but for large scenario trees it may take too long until all scenarios are eventually sampled.

Still, the hope is that by solving the SAA problem with SDDP in reasonable time and with reasonable accuracy, a feasible and appropriate policy \((x_t(\xi_t))_{t=1}^T\) can be determined, yielding a reasonable approximation of \(\bar{v}^*\) \[76\].

6.2.1. Assessing policy quality. As it is computationally intractable, to solve a SAA problem \((\bar{P})\) with a sample size that guarantees a predetermined accuracy, in practice, usually moderate sample sizes are used. For example, in \[17\], sample sizes with branching numbers \(N_t\) between 5 and 200 are tested.

Since the bounds \(\bar{v}^i\) and \(\bar{v}_N^i\) in SDDP are determined using the finite scenario tree associated to one specific sample of the true data process, they only measure the in-sample performance of the determined feasible policy \((x_t(\xi_t))_{t=1}^T\). Therefore, to
assess its quality for the original problem \( (\bar{P}) \), i.e., its \textit{out-of-sample} performance, it is required to evaluate it with respect to the original data process \( (\xi_t)_{t=1}^T \). Such evaluation also allows to compare policies obtained for different SAA problems and to design scenario trees of appropriate size [17].

Various techniques have been proposed in stochastic programming to measure the performance of feasible policies, such as analyzing optimality conditions, assessing solution stability or estimating the optimality gap [17]. Many of these techniques focus on two-stage stochastic programs. For multistage programs, Morton et al. have made substantial contributions [14, 17, 44], which are based on estimating the optimality gap. [17] and [44] in particular deal with policy assessment in context of SDDP, the latter considering a risk-averse variant of SDDP, see Sect. 7. We discuss this approach thoroughly in the remainder of this subsection.

\textbf{Estimating the optimality gap.} For some feasible policy \( (x_t(\xi_{[t]}))_{t=1}^T \), let again \( \bar{v}(\xi) = \sum_{t=1}^T c_t x_t(\xi_{[t]}) \) denote the random cost for scenario path \( \xi = (\xi_1, \ldots, \xi_T) \).

From (P.6.5) we have \( E[\bar{v}(\xi)] \geq \bar{v}^* \). Therefore, the optimality gap induced by policy \( (x_t(\xi_{[t]}))_{t=1}^T \) can be expressed as

\begin{equation}
\Delta := E[\bar{v}(\xi)] - \bar{v}^* \geq 0.
\end{equation}

This gap cannot be directly evaluated because the optimal value \( \bar{v}^* \) is not known. Using some lower bound for \( \bar{v}^* \), \( \Delta \) can be overestimated though. Such lower bound is given by \( E[\bar{v}] \), see (P.6.3). This yields

\begin{equation}
E[\bar{v}(\xi)] - E[\bar{v}] \geq \Delta \geq 0.
\end{equation}

Still, the left-hand side of (6.5) is computationally infeasible to evaluate. It requires excessive computational effort to evaluate policy \( (x_t(\xi_{[t]}))_{t=1}^T \) for all possible scenarios to obtain \( E[\bar{v}(\xi)] \). Furthermore, from SDDP only one specific realization of \( \bar{v} \) is known. Therefore, the policy assessment strategies proposed in [17] use estimators for both terms to derive an approximate one-sided confidence interval bounding \( \Delta \) from above.

\textbf{Upper bound estimation.} Since the policy \( (x_t(\xi_{[t]}))_{t=1}^T \) is feasible, it can be evaluated for any sampled scenario from the original data process to assess its out-of-sample performance. Let us sample \( M_u \) i.i.d. scenario paths from \( (\xi_t)_{t=1}^T \). For each of those sampled scenarios \( \xi^\ell, \ell = 1, \ldots, M_u \), the SDDP subproblems (1.15) are solved in forward direction, yielding \( x_t(\xi^\ell_{[t]}) \) and \( \bar{v}(\xi^\ell) \) [17]. An upper bound estimator is then defined by the sample mean

\begin{equation}
U_{M_u} := \frac{1}{M_u} \sum_{\ell=1}^{M_u} \bar{v}(\xi^\ell).
\end{equation}

Similarly to the in-sample estimator, this estimator is an unbiased and consistent estimator of \( E[\bar{v}(\xi)] \). Its sample variance is given by [17]

\begin{equation}
\sigma_U^2 := \frac{1}{M_u - 1} \sum_{\ell=1}^{M_u} (\bar{v}(\xi^\ell) - U_{M_u})^2.
\end{equation}

Alternatively, an upper bound estimator can be obtained by sampling a finite number of different scenario trees, i.e., different SAA problems, and applying policy
(x_t(ξ_t))_{t=1}^T to each of them [14]. This comes at the cost of increased computational effort.

**Lower bound estimation with several scenario trees.** From SDDP, only one single realization of ˜v is known. Hence, it is not directly possible to determine a sampling error for this point estimate and to derive a confidence interval for E[˜v].

One approach to derive a lower bound estimator is to solve a finite number of SAA problems for different scenario trees with SDDP and to determine the mean of the lower bounds ˜v. To be precise, M_t different scenario trees are constructed, each by sampling N_t realizations per stage from (ξ_t)_{t=1}^T. Then SDDP is run, yielding the lower bounds ˜v^ℓ, ℓ = 1, . . . , M_t [17].

The sample mean

\[ L_{M_t} := \frac{1}{M_t} \sum_{\ell=1}^{M_t} ˜v^\ell \]

then defines an estimator for E[˜v] with sample variance

\[ \sigma^2_\ell := \frac{1}{M_t - 1} \sum_{\ell=1}^{M_t} ( ˜v^\ell - L_{M_t} )^2. \]

Note that instead of lower bounds ˜v^\ell, also the optimal values ˜v^\ell_N could be used in estimator (6.8). We already discussed in Sect. 6.2 that it may be computationally intractable to solve one single SAA problem to optimality, though. For this reason, it is proposed to only use sufficiently good lower bounds [17].

In principle, applying SDDP to not only one, but several SAA problems and building the mean of the obtained bounds seems very reasonable from a statistical perspective, as the outcome of one SAA problem is random. Additionally, this implies another possible benefit: If SDDP is run for M_t different SAA problems ( ˜P^\ell_N ), each of these problems yields a different feasible policy. By calculating the upper bound estimator U_{M_t} (6.6) for each of them, directly M_t different policies could be compared.

However, for problems with multiple stages and for sufficiently high N_t, this becomes computationally intractable, even without solving ( ˜P^\ell_N ) exactly. Therefore, de Matos et al. follow the strategy to run SDDP once for some SAA problem with larger branch size N_t to determine a high quality policy and then, afterwards, to run SDDP for M_t SAA problems with smaller branch size N_t only to produce the lower bound estimate L_{M_t} and assess the quality of that policy [17]. In their numerical tests, they choose values between 5 and 200 for N_t and 5 for ˜N_t. However, it is not clear in general how to choose ˜N_t to reach a reasonable trade-off between computational tractability and an appropriate quality of the lower bound estimator.

**Lower bound estimation with one scenario tree.** An alternative lower bound estimator, avoiding the computational effort stemming from using several scenario trees, is proposed in [17]. Here, the estimator is derived by only using the already existing scenario tree, which has been applied to determine the considered policy.

The idea is then to use the SDDP outcome ˜v as the point estimate for the lower bound. To estimate the unknown sampling error of ˜v, the sampling error of the in-sample upper bound estimator is used. This upper bound estimator is determined by sampling M_t i.i.d. scenarios from the existing scenario tree and using formulae (6.6)
where (6.7), which yields

\[ \bar{v}_{M'} := \frac{1}{M'} \sum_{\ell=1}^{M'} \bar{v}(\xi^\ell) \]

and

\[ (\sigma')^2 := \frac{1}{M' - 1} \sum_{\ell=1}^{M'} (\bar{v}(\xi^\ell) - \bar{v}_{M'})^2. \]

As described in [17], a more conservative estimator is obtained if the point estimate is chosen as

\[ L'_{M'} = \min \left\{ \bar{v}, \bar{v}_{M'} \right\} \]

The idea behind applying the sampling error \((\sigma')^2\) to \(\bar{v}\) is that \(\bar{v}\) and \(E[\bar{v}_{M'}]\) are equal if SDDP has been run to optimality. However, this also implies that if SDDP has not converged (and also if \(N_t\) is not sufficiently large) the sampling error of \(L'_{M'}\) may be underestimated and, thus, the confidence intervals drawn from this become overly optimistic.

Confidence intervals. Using the bound estimators and their sample variances, asymptotically valid confidence intervals can be derived [17].

\[ \left( -\infty, U_{M_u} + t_{M_u-1,\alpha} \frac{\sigma_U}{\sqrt{M_u}} \right) \]

is an asymptotically valid, for finite \(M_u\) approximate, \((1 - \alpha)\%\) confidence interval for \(E[\bar{v}(\xi)]\). Here, \(t_{M_u-1,\alpha}\) denotes the \((1 - \alpha)\)-level quantile of a student’s \(t\) distribution with \(M_u - 1\) degrees of freedom.

Similarly,

\[ \left[ L_{M_l} - t_{M_l-1,\alpha} \frac{\sigma_l}{\sqrt{M_l}}, \infty \right) \]

is an asymptotically valid, for finite \(M_l\) approximate, \((1 - \alpha)\%\) confidence interval for \(\bar{v}^*\).

An equivalent confidence interval can be derived using lower bound estimator (6.10) with \(M'_l\). This confidence interval is only valid if SDDP has converged and if \(N_t\) is sufficiently large for the SAA problem.

Combining both confidence intervals yields

\[ \left[ 0, (U_{M_u} - L_{M_l})^+ + t_{M_l-1,\alpha} \frac{\sigma_l}{\sqrt{M_l}} + t_{M_u-1,\alpha} \frac{\sigma_U}{\sqrt{M_u}} \right) \]

as a one-sided approximate confidence interval for the optimality gap \(\Delta\) [17].

The sample size choices from [17] are summarized in Table 1.

Policy assessment for finite uncertainty. It should be mentioned that policy assessment is also important in case that Assumption 7 holds, i.e., for discrete and finite random variables \(\xi\) in the true data process \((\xi_i)_{i=1}^T\). In this case, it is much easier, though, as a deterministic lower bound \(\bar{v}\) for \(v\), an upper bound estimator \(\bar{v}_K\) (1.16) with associated sample variance (1.17) and an approximate confidence interval are directly determined within SDDP. Since \(|K|\) may be chosen rather small within the forward pass of SDDP, however, it is reasonable to assess the determined policy \(x_t(\xi[i])\) more precisely by running a forward simulation with a much higher number of samples from the scenario tree after SDDP has terminated.
### 6.3. Variance reduction techniques

Instead of MC sampling, also the variance reduction techniques which are introduced in Sect. 3.2, can be applied for scenario tree generation to reduce the bias and the variance of the obtained SAA estimators.

In [40], numerical tests comparing MC, LHS and RQMC for scenario tree generation indicate that the error from sampling scenario trees may be significant. In comparison with MC and LHS, RQMC yields the most promising results to obtain more representative scenario trees.

In [17] also MC, LHS and RMC are compared for different branch sizes and policy evaluation strategies. The results indicate that with both LHS and RQMC, a reduction of bias and sampling error, a higher policy quality and tighter confidence intervals can be achieved in comparison with MC sampling, especially for smaller branch sizes \( N_t \). For smaller branch sizes LHS appears to be superior, while RQMC yields better results for larger branch sizes.

While showing higher variability for MC sampling, if combined with RQMC and LHS sampling, the lower bound estimator (6.10) using only in-sample scenarios from the existing tree yields comparable results to the approach solving several SAA problems [17]. In particular, the computational effort can be reduced significantly.

Importance sampling may be particularly useful if it is difficult to sample from the true distribution directly [55].

### 7. Risk-averse SDDP [relaxing Assumption 4]

In SDDP, as described in Sect. 1, a risk-neutral optimal policy is determined, which we formalize by Assumption 4. Therefore, the formulation of problem \( (P) \) contains conditional expectations, such that a nested expectation is minimized in the objective function, subject to non-anticipativity and satisfaction of all feasibility constraints. Due to the tower property of expected values, \( E_{\xi_t} [Z_t(\xi_t)] = E_{\xi_{t-1}} [E_{\xi_t|\xi_{t-1}} [Z_t(\xi_t)]] \) for some random variable \( Z_t \), this is equivalent to minimizing the expectation of the total objective values of all scenarios, i.e.,

\[
E \left[ \sum_{t=1}^{T} c_t(\xi_t)^\top x_t(\xi_t) \right].
\]

under the same constraints [77]. Thus, SDDP yields a policy \( (x_t(\xi_t))_{t=1}^{T} \) which provides the optimal objective value, e.g., minimum costs, in expectation.

Recall that for any scenario \( \xi^s, s \in S \), the objective value \( \sum_{t=1}^{T} c_t(\xi_t^s)^\top x_t(\xi_t^s) \) is random, though. Therefore, for some specific realization of the data process \( (\xi_t)_{t=1}^{T} \), the SDDP policy may produce an objective value which widely deviates from its expectation. For this reason, in practice, decision makers are often anxious not only to find a policy yielding low costs on average, but also to avoid the risk of extremely high costs. This motivates to consider risk-averse approaches in stochastic programming. Specifically for multistage stochastic programming, risk-aversion has been a popular

\begin{table}[h]
\centering
\begin{tabular}{|l|c|c|}
\hline
Parameter & Estimator & Suggested values \\
\hline
\( N_t \) (no. of samples) & SAA & \[5, 200\] \\
\hline
\( M_u \) (no. of out-of-sample scenarios) & Upper bound (6.6) & 5000 \\
\hline
\( M_l \) (no. of scenario trees) & Lower bound (6.8) & 5 \\
\hline
\( M_l' \) (no. of in-sample scenarios) & Lower bound (6.10) & 5000 \\
\hline
\end{tabular}
\caption{Sample size choices for policy evaluation in [17].}
\end{table}
research topic in the last decade. This includes theoretical fundamentals on risk measures [70] as well as algorithmic developments, such as rolling horizon approaches with chance and CVaR constraints [33, 34] or SDDP.

Mathematically, a risk-averse formulation of problem \((P)\), which we denote by \((P_{R})\) can be achieved by replacing the expected value in (7.1) with some multi-period risk measure \(R[\cdot]\):

\[
R \left[ \sum_{t=1}^{T} c_t(\xi_t)^\top x_t(\xi_t) \right].
\]

### 7.1. Risk measures for multistage programs.

We briefly discuss risk measures and some properties required for SDDP now. Since our focus is on algorithmic aspects of SDDP, we keep this short and refer to the relevant literature for more details such as technical definitions and derivations [70, 71, 78].

In general, a (one-period) risk measure is a function \(\rho : \mathcal{Z} \rightarrow \mathbb{R}\) from the space \(\mathcal{Z}\) of random variables \(\mathcal{Z}\) to \(\mathbb{R}\) (since random variables are functions themselves, it is actually a functional). Two well-known risk measures are the value-at-risk (VaR\(_\alpha\)) and conditional value-at-risk (CVaR\(_\alpha\)) to some level \(\alpha \in (0, 1)\).

VaR is defined as the left-side \((1 - \alpha)\)-quantile of the distribution of some random variable \(Z\), i.e.,

\[
\text{VaR}_\alpha[Z] := \inf \{ u : \mathbb{P}(z \leq u) \geq 1 - \alpha \}.
\]

CVaR, also referred to as average value-at-risk (AVaR) [79], reflects the expected value of random variable \(Z\) conditioned on values greater or equal than VaR\(_\alpha[Z]\] [1, 67]. Formally,

\[
\text{CVaR}_\alpha[Z] := \mathbb{E}[Z|Z \geq \text{VaR}_\alpha[Z]].
\]

It can be shown that an equivalent formulation is given by [76]

\[
\text{CVaR}_\alpha[Z] = \text{VaR}_\alpha[Z] + \frac{1}{\alpha} \mathbb{E}[Z - \text{VaR}_\alpha[Z]]_+.
\]

CVaR\(_\alpha[\cdot]\) has some beneficial properties compared to VaR\(_\alpha[\cdot]\). It does not only consider the probability mass beyond VaR\(_\alpha[\cdot]\), but also its distribution, e.g. if it has fat or long tails. Moreover, it allows to retain convexity of optimization problems, as we discuss later on. VaR\(_\alpha[Z]\) and CVaR\(_\alpha[Z]\) are illustrated in Fig. 11. An optimization-based intuition for using VaR or CVaR is presented in [76, 79].

In the multistage setting of (7.2), traditional single-period risk measures have to be extended to several periods, that is, to a sequence of random variables \(Z_1, \ldots, Z_T\). We denote such multi-period risk measures by \(R[\cdot]\).

With regard to optimization, such risk measures have to be carefully chosen. First of all, it is not clear how risk should be measured in a multistage setting, that is, if it should be measured once over the whole time horizon from stage 1 to \(T\), stage by stage or in a nested way [41]. Secondly, a risk measure should be chosen in such way that the resulting risk-averse multistage stochastic programming problem possesses some desirable properties.

**Coherence.** One desired property for risk measures applied in optimization is coherence, a concept introduced by Artzner et al. [2]. We employ a slightly different definition from [78]:
Definition 7.1. A risk measure $\rho : \mathcal{Z} \to \mathbb{R}$ is called coherent, if it satisfies

- Convexity: for any $Z_1, Z_2 \in \mathcal{Z}$ and all $\lambda \in [0, 1]$ it holds
  \[ \rho(\lambda Z_1 + (1 - \lambda)Z_2) \leq \lambda \rho(Z_1) + (1 - \lambda) \rho(Z_2), \]

- Monotonicity: If $Z_1 \leq Z_2$ for almost all realizations, then $\rho(Z_1) \leq \rho(Z_2),$

- Translation Equivariance: If $a \in \mathbb{R}$ and $Z \in \mathcal{Z}$, then $\rho(Z + a) = \rho(Z) + a,$

- Positive Homogeneity: If $\lambda > 0$ and $Z \in \mathcal{Z}$, then $\rho(\lambda Z) = \lambda \rho(Z).$

The main advantage of a coherent risk measure is that it is convex and, thus, convex objective functions, such as the value functions $Q_t(\cdot)$ occurring in the dynamic programming equations of $(P)$, remain convex if $\rho(\cdot)$ is applied to them. $\text{VaR}_\alpha[Z]$ is no coherent risk measure, but $\text{CVaR}_\alpha[Z]$ is [57]. Therefore, in optimization $\text{CVaR}_\alpha[Z]$ is usually preferred.

**Time-consistency.** A second crucial property for risk-averse multistage programs is time-consistency. In the literature, different definitions of this property exist. A common definition is that a risk-measure is called time-consistent, if an optimal solution of (7.2) is still optimal at every stage of the decision process with respect to the conditional risk measure. We do not discuss this in more detail and refer to [22, 41, 77, 78] and references within for technical definitions and discussions on different concepts of time-consistency.

In the risk-neutral case, time-consistency is naturally satisfied by the equivalence of (7.1) and the nested formulation in (P). The reason is that the nested formulation can be expressed by means of dynamic programming equations (1.1)-(1.3), which naturally yield time-consistent solutions. The mentioned equivalence does only hold because of the decomposability and (strict) monotonicity of the expectation operator $\mathbb{E}$, though, which allow for a decomposition of (7.1) and interchangeability of the min and $\mathbb{E}[\cdot]$ operators [77]. As risk measures $\mathcal{R}[\cdot]$ do not necessarily possess those properties, time-consistency is not assured.

For this reason, it can be shown that even very intuitive risk measures $\mathcal{R}$ may lead to time-inconsistent decisions. For instance, some simple examples in [22, 41]
show that taking an unconditional risk measure $\rho[\cdot]$, such as VaR or CVaR, taken as a multiperiod risk measure over the whole horizon $t = 1, \ldots, T$, i.e.,

$$
\mathcal{R} \left[ \sum_{t=1}^{T} c_t(\xi_t) x_t(\xi_t) \right] = \rho \left[ \sum_{t=1}^{T} c_t(\xi_t) x_t(\xi_t) \right],
$$

leads to time-inconsistent decisions. The same is true if risk is measured separately in each stage, that is, if we set

$$
\mathcal{R} \left[ \sum_{t=1}^{T} c_t(\xi_t) x_t(\xi_t) \right] = c_1^T x_1 + \rho_2(c_2(\xi_2)^T x_2(\xi_2)) + \cdots + \rho_T(c_T(\xi_T)^T x_T(\xi_T)).
$$

**Nested conditional risk measures.** To obtain a risk-averse program ($P_\mathcal{R}$) with time-consistent solutions, it is often proposed to use one-period conditional risk measures $\rho^t[\cdot]$ for all $t = 1, \ldots, T$ in a nested fashion [77]. This idea goes back to Ruszczyński and Shapiro [71], where the risk measure is taken as a function of the recourse value [44]. Such *nested risk measures* have the drawback that they are less amenable to suitable interpretations, although some economic interpretations are possible [68].

Applying a nested risk measure to the objective in problem ($P_\mathcal{R}$) yields

$$
\mathcal{R} \left[ \sum_{t=1}^{T} c_t(\xi_t) x_t(\xi_t) \right] = c_1^T x_1 + \rho_2(c_2(\xi_2)^T x_2(\xi_2)) + \cdots + \rho_T(c_T(\xi_T)^T x_T(\xi_T)).
$$

(7.6)

By using this nested risk measure and additionally denoting the stage-$t$ feasible set by $M_t(x_{t-1}, \xi_t)$, the risk-averse problem ($P_\mathcal{R}$) can be formulated as

$$
(P_\mathcal{R}) \quad v^*_\mathcal{R} := \left\{ \begin{array}{l}
\min_{x_1} \ c_1^T x_1 + \rho_2^t \left[ \min_{x_2} c_2(\xi_2)^T x_2(\xi_2) + \rho_3^t \left[ \min_{x_3} c_3(\xi_3)^T x_3(\xi_3) + \cdots \right] \right] \\
\text{s.t.} \quad x_1 \in M_1, \quad x_t(\xi_t) \in M_t(x_{t-1}, \xi_t) \quad \forall t = 1, \ldots, T, \ \xi_t \in \Xi_t | \xi_{t-1} \text{ a.s.}
\end{array} \right.
$$

As indicated before, we can derive an equivalent formulation using dynamic programming equations [78]. Under stagewise independence (Assumption 9), the conditional risk measures $\rho^t[\cdot]$ coincide with the associated coherent risk measures $\rho_t[\cdot]$ and thus, the dynamic programming equations denote

$$
Q_{\mathcal{R},t}(x_{t-1}, \xi_t) := \left\{ \begin{array}{l}
\min_{x_t} c_t^T(\xi_t)x_t + Q_{\mathcal{R},t+1}(x_t) \\
\text{s.t.} \quad W_t(\xi_t)x_t = h_t(\xi_t) - T_{t-1}(\xi_t)x_{t-1} \\
x_t \geq 0
\end{array} \right.
$$

(7.7)

with some *risk value function*

$$
Q_{\mathcal{R},t+1}(x_t) := \rho_{t+1}[Q_{\mathcal{R},t+1}(x_t, \xi_{t+1})]
$$

(7.8)
and \( Q_{\mathcal{R},T+1}(\cdot) \equiv 0 \). The first stage remains

\[
(7.9) \quad v_{\mathcal{R}}^* = \begin{cases} \min_{x_1} & c_1^T x_1 + Q_{\mathcal{R},2}(x_1) \\ \text{s.t.} & W_1 x_1 = h_1 \\ & x_1 \geq 0. \end{cases}
\]

Fortunately, for coherent conditional risk measures \( \rho^t_{\xi(t)}[\cdot], t = 1, \ldots, T \), also the nested risk measure \( \mathcal{R}[\cdot] \) in (7.6) preserves convexity of \( Q_{\mathcal{R},t+1}(x_t) \).

**Expected conditional risk measures.** Another class of multiperiod risk measures are expected conditional risk measures [22, 41]. Here, conditional expectations are used to avoid the risk measure nesting, which proves beneficial in determining upper bounds in SDDP. The risk measure \( \mathcal{R}[\cdot] \) applied to the objective function can be expressed as

\[
(7.10) \quad \mathcal{R} \left[ \sum_{t=1}^{T} c_t(\xi_t) \right] = c_1^T x_1 + \rho_2 \left[ c_2(\xi_2) \right] + \mathbb{E}_{\xi_2} \left[ \rho_3 \left[ \ldots \right] \right] + \cdots + \mathbb{E}_{\xi_{T-1}} \left[ \rho_T \left[ c_T(\xi_T) \right] \right].
\]

Under some assumptions, e.g., for coherent risk measures \( \rho_t[\cdot] \), expected conditional risk measures satisfy some notion of time-consistency [41].

**Polyhedral risk measures.** Multiperiod polyhedral risk measures are a special type of risk measure, which can be formulated as the optimal value of certain \( T \)-stage linear stochastic programs [23]. The arguments of the risk measure enter these multistage programs on the right-hand side.

In [31], multiperiod extended polyhedral risk measures are introduced, for which the corresponding linear program even has a slightly more general form.

Formally, a risk measure \( \mathcal{R}[\cdot] \) is called multiperiod extended polyhedral if there exist \( \tilde{k}_t \in \mathbb{N} \), vectors \( \tilde{a}_t, \tilde{c}_t \), matrices \( A_t \) and \( B_t \) and functions \( \tilde{h}_t, t = 1, \ldots, T, \tau = 0, \ldots, t-1 \) of appropriate size such that

\[
(7.11) \quad \mathcal{R}[Z_1, \ldots, Z_T] := \inf_{y_1, \ldots, y_T} \mathbb{E} \left[ \sum_{t=1}^{T} \tilde{c}_t^T y_t \right] \text{ s.t. } y_t \in L^\alpha(\Xi, \mathcal{F}_t, P; \mathbb{R}^{\tilde{k}_t}), \quad t = 1, \ldots, T \\
A_t y_t \leq a_t \quad \text{a.s.}, \quad t = 1, \ldots, T \\
\sum_{\tau=0}^{t-1} B_{t,\tau} y_{t-\tau} = \tilde{h}_t(Z_t) \quad \text{a.s., } \quad t = 1, \ldots, T.
\]

Multiperiod extended polyhedral risk measures comprise polyhedral risk measures, spectral risk measures and also CVaR. They can be shown to be convex and coherent under certain assumptions [31].

The main advantage of risk measures (7.11) is that they can naturally be used in a multistage stochastic programming setting. For example, in our context, a risk-averse
form of problem \((P_R)\) can be formulated as

\[
\inf_{y_1, \ldots, y_T} \mathbb{E} \left[ \sum_{t=1}^{T} \hat{c}_t^\top y_t \right]
\]

\[
\text{s.t. } \begin{align*}
& y_t \in L_p(\mathcal{F}, P_t, \mathbb{R}_t), & t = 1, \ldots, T \\
& A_t y_t \leq a_t \quad \text{a.s.,} & t = 1, \ldots, T \\
& \sum_{\tau=0}^{t-1} B_{t,\tau} y_{t-\tau} = \tilde{h}_t \left( \sum_{\tau=1}^{t} c_\tau(\xi_t)^\top x_\tau \right) \quad \text{a.s.,} & t = 1, \ldots, T \\
& T_{t-1}(\xi_t) x_{t-1}(\xi_{t-1}) + W_t(\xi_t) x_t(\xi_t) = h_t(\xi_t) \quad \text{a.s.,} & t = 1, \ldots, T \\
& x_t(\xi_t) \geq 0 \quad \text{a.s.,} & t = 1, \ldots, T.
\end{align*}
\]

In the following subsections, we discuss how SDDP can be applied to risk-averse multistage problems for the above risk measures. While some articles on this topic also cover SAA [44, 76, 79], we restrict to finite random variables here.

### 7.2. SDDP with polyhedral risk measures.

An extension of SDDP to extended polyhedral risk measures is proposed in [31], a more specific one to spectral risk measures in [32].

For extended polyhedral risk measures, a risk-averse pendant \((P_{R})\) of problem \((P)\) can be expressed by (7.12). This is a large-scale linear programming problem, which allows for a reformulation by means of dynamic programming equations [31]:

\[
Q_{R,t}(x_{t-1}, \xi_t, z_{t-1}, y_1, \ldots, y_{t-1}) = \min_{x_t, y_t, z_t} \hat{c}_t^\top y_t + Q_{R,t+1}(x_t, z_t, y_1, \ldots, y_t)
\]

\[
\text{s.t. } (1 - \delta_t) \left( \sum_{\tau=0}^{t-1} B_{t,\tau} y_{t-\tau} = \tilde{h}_t(z_t) \right) = 0
\]

\[
\begin{align*}
& A_t y_t \leq a_t \\
& z_t = z_{t-1} - c_t(\xi_t)^\top x_t \\
& T_{t-1}(\xi_t) x_{t-1} + W_t(\xi_t) x_t = h_t(\xi_t) \\
& x_t(\xi_t) \geq 0.
\end{align*}
\]

with some risk value function

\[
Q_{R,t+1}(x_t, z_t, y_1, \ldots, y_t) := \mathbb{E}_{\xi_{t+1}} \left[ Q_{R,t}(x_{t-1}, \xi_t, z_{t-1}, y_1, \ldots, y_{t-1}) \right].
\]

Then, SDDP can be applied to these dynamic programming equations in a natural way, yielding lower bounds and statistical upper bounds for \(v_{R,t}^\ast\). Guigues and Römisch derive the cut formulae and give a convergence proof for the special case of \(\tilde{h}_t(z) = zb_t + b_t^\ast\) for some \(b_t, b_t^\ast \in \mathbb{R}^{a_t}\), with \(a_t\) the dimension of \(y_t\) [31].

The main drawback of this approach is the introduction of additional state variables \(z_{t-1}\) and \(y_1, \ldots, y_{t-1}\), which are required to store the history of previous decisions. As this leads to a considerable increase of the state space dimension, it can be considered computationally prohibitive in practice [61].

### 7.3. SDDP with nested conditional CVaR.

Nested conditional risk measures are by far the most frequently chosen approach for risk-averse extensions of SDDP [22, 41, 44, 60, 61, 76, 79]. Usually, a convex combination of conditional expectation and conditional CVaR is used as the conditional risk measure [76]:

\[
\rho_t^{\xi_{t-1}}[Z] := (1 - \lambda_t) \mathbb{E}_{\xi_{t-1}}[Z] + \lambda_t \text{CVaR}_{\alpha_t}[Z|\xi_{t-1}]
\]
with parameters $\lambda_t \in [0, 1]$ and $\alpha_t \in (0, 1)$; they control the risk-aversion. Choosing $\lambda_t = 0$ yields the standard risk-neutral model for $(P)$. The conditional risk measure (7.15) is coherent [76].

### 7.3.1. Dynamic programming equations.

The general dynamic programming equations for nested conditional risk measures are formulated in (7.7)-(7.9). To determine $Q_t(\cdot), t = 1, \ldots, T,$ for (7.15) specifically, the conditional CVaR of $Q_t(\cdot)$ has to be evaluated, which is quite challenging using formula (7.5).

However, CVaR can also be formulated as the optimal value of an optimization problem with decision variable $u \in \mathbb{R}$ [67]. To be precise,

$$\text{CVaR}_\alpha[Z] = \inf_{u \in \mathbb{R}} \left\{ u + \frac{1}{\alpha} E[Z - u]^+_+ \right\},$$

Note that an optimal solution $u^* = \text{VaR}_\alpha[Z]$ [76].

Using this definition, we are able to derive different equivalent representations of the dynamic programming equations.

#### Additional state variable approach.

With relation (7.16), the risk value function (7.8) can be expressed as

$$Q_{R,t+1}(x_t) = \min_{u_t \in \mathbb{R}} \mathbb{E}_{\xi_{t+1}} \left[ (1 - \lambda_{t+1})Q_{R,t+1}(x_t, \xi_{t+1}) \right. + \lambda_{t+1} \left. \left( u_t + \frac{1}{\alpha_{t+1}} [Q_{R,t+1}(x_t, \xi_{t+1}) - u_t]^+_+ \right) \right],$$

The minimization over $u_t$ can be incorporated into the stage-$t$ subproblems [76], which yields

$$Q_{R,t}(x_{t-1}, \xi_t) = \left\{ \begin{array}{ll} \min_{x_t, u_t} & c^T_t (\xi_t) x_t + \lambda_{t+1} u_t + Q_{R,t+1}(x_t, u_t) \\ \text{s.t.} & W_t(\xi_t) x_t = h_t(\xi_t) - T_{t-1}(\xi_{t-1}) x_{t-1} \\ & x_t \geq 0 \end{array} \right\}$$

with some modified risk value function (7.19)

$$Q_{R,t+1}(x_t, u_t) = \mathbb{E}_{\xi_{t+1}} \left[ (1 - \lambda_{t+1})Q_{R,t+1}(x_t, \xi_{t+1}) + \frac{\lambda_{t+1}}{\alpha_{t+1}} [Q_{R,t+1}(x_t, \xi_{t+1}) - u_t]^+_+ \right],$$

$Q_{R,T+1}(\cdot) \equiv 0$ and $\lambda_{T+1} \equiv 0$ [76]. The first stage changes to

$$v^*_R = \left\{ \begin{array}{ll} \min_{x_t, u_t} & c^T_1 x_t + \lambda_2 u_t + Q_2(x_t, u_t) \\ \text{s.t.} & W_1 x_t = h_1 \\ & x_t \geq 0 \end{array} \right\}.$$

The functions $Q_{R,t+1}(x_t, u_t)$ differ from the original risk value functions $Q_{R,t+1}(x_t)$, but remain convex.

With equations (7.18)-(7.20), the risk measures $\rho_t[\cdot]$ are incorporated into the subproblems such that only expectations have to be evaluated in the dynamic programming equations. However, in comparison with the dynamic programming equations of the risk-neutral case (1.1)-(1.3), we still observe some fundamental differences: Firstly, an additional state variable $u_t \in \mathbb{R}$ is introduced at each stage to estimate the VaR-level, augmenting the state space. Secondly, the risk value functions do not
only depend on $x_t$, but also on $u_t$ and parameters $\lambda_t, \alpha_t$. Thirdly, the weighted value functions contain the nonlinear, i.e., piecewise linear, function $[\cdot]_+$. Philpott and de Matos provide an alternative reformulation of the dynamic programming equations, eliminating the nonlinear expression via an epigraph reformulation [60]. To this end, the random term in the brackets in (7.19) is fully incorporated into the value functions.

This yields for $t = 2, \ldots, T - 1$

$$Q_{R,t}(x_{t-1}, u_{t-1}, t)$$

(7.21)

$$\begin{align*}
\min_{x_t, u_t, w_t} & \quad (1 - \lambda_t)\left(c_t^\top(\xi_t)x_t + \lambda_{t+1}u_t\right) \\
& \quad + Q_{R,t+1}(x_{t}, u_{t}) + \lambda_t w_t \\
\text{s.t.} & \quad W_t(\xi_t)x_t = h_t(\xi_t) - T_{t-1}(\xi_t)x_{t-1} \\
& \quad w_t - c_t^\top(\xi_t)x_t - \lambda_{t+1}u_t - Q_{R,t+1}(x_{t}, u_{t}) \geq u_{t-1} \\
& \quad w_t, x_t \geq 0.
\end{align*}$$

Using this formulation, the risk value function is defined more naturally as

$$Q_{R,t+1}(x_t, u_t) = \mathbb{E}\left[Q_{R,t+1}(x_t, \xi_{t+1})\right].$$

Again, $Q_{R,T+1}(\cdot) \equiv 0$ and $\lambda_{T+1} \equiv 0$.

The first stage problem reads then

$$v^*_R = \begin{cases}
\min_{x_1, u_1, w_1} & \quad c_1^\top x_1 + \lambda_2 u_1 + Q_{R,2}(x_1, u_1) \\
\text{s.t.} & \quad W_1 x_1 = h_1 \\
& \quad x_1 \geq 0.
\end{cases}$$

(7.23)

In comparison to the first approach by Shapiro [76], additional variables and constraints have to be introduced. Both approaches allow application of SDDP, but share the drawback of augmenting the state space. Since the computational effort of SDDP grows exponentially in the state space dimension, such increase should be avoided.

**Modifying the probability measure.** The idea is to avoid the introduction of an additional state variable $u_t$ by directly determining the $(1 - \alpha)$-quantile of the value functions, i.e., VaR$_\alpha[Q_{R,t+1}(\cdot)]$, as first introduced by Shapiro et al. [79].

Without loss of generality, assume that for all $t = 2, \ldots, T$ and any fixed trial solution $\bar{x}_{t-1}$ the value functions $Q_{R,t}(\bar{x}_{t-1}, \xi_{t,j})$ are ordered for all $j = 1, \ldots, q_t$. That means, we have $Q_{R,t}(\bar{x}_{t-1}, \xi_{t,1}) \leq \cdots \leq Q_{R,t}(\bar{x}_{t-1}, \xi_{t,q_t})$. Then, in (7.17) the variable $u_t$ can be replaced by the $(1 - \alpha)$-quantile $Q_{R,t}(\bar{x}_{t-1}, \xi_{t,j^*})$ with $j^*$ chosen such that

$$\sum_{j=1}^{j^*} p_{t,j} \geq 1 - \alpha.$$
In [61], the same idea is discussed and used to reformulate the risk measure (7.15) even before formulating the dynamic programming equations. In this context, the so-called dual representation of CVaR
\[ \rho^{\alpha}(\xi) = \min_{\lambda \geq 0} \lambda + \frac{1}{\alpha} \mathbb{E}_{\lambda}[\xi - \lambda] \]
is used, with
\[ \zeta_{tj} = \begin{cases} 
(1 - \lambda_t), & j < j^*, \\
(1 - \lambda_t) + \frac{1}{p_{tj}^*} \left( \lambda_t - \frac{\lambda_t}{\alpha_t} \sum_{n=j^*+1}^{q_t} p_{tn} \right), & j = j^*, \\
(1 - \lambda_t) + \frac{1}{\alpha_t}, & j > j^*.
\end{cases} \]

With that, the expectations within the dynamic programming equations (7.7)-(7.9) can be used as usual, having only a changed probability measure. As \( \zeta_{tj} \) changes with \( \bar{x}_{t-1} \), the modified probabilities have to be recalculated for each stage \( t \), iteration \( i \) and sample \( k \) in SDDP. This principle is also extended to general coherent risk measures in [61].

Recently, a change of probability measure has been also discussed in detail in [48]. Instead of determining an ordering and \( j^* \) based on the approximate value functions considered in one specific iteration, all previous iterations are taken into account as well. More precisely, the number of iterations in which an index \( j \) exceeds VaR\( \alpha \) is counted. This is considered as a good proxy for the ordering of the actual value functions. The ordering and, thus, the probability measure, can either be updated dynamically within SDDP or be determined by running risk-averse SDDP once in advance to identify the outcomes contributing to CVaR\( \alpha \). The latter approach has the advantage that the changed probability measure can be fixed for the following actual run, which yields a risk-neutral problem and allows for application of ordinary SDDP.

As an overview, the different forms of dynamic programming equations for \((P_R)\) using nested convex combinations of expectations and conditional CVaR are summarized in Table 2.

### 7.3.2. Forward and Backward Pass.

All approaches in Table 2 to formulate the dynamic programming equations allow for a solution of a risk-averse problem \((P_R)\) using SDDP. Some approaches are more efficient, since the state space, the decision space or the number of constraints are not augmented. Others are advantageous in the sense that \( Q_{R,t}(\cdot) \) is expressed by a neat formula and, thus, cut formulae can be derived more easily. With some epigraph reformulation, for all the approaches all subproblems can be formulated as LPs.

The forward pass of SDDP basically remains the same as for risk-neutral SDDP from Sect. 1.3. That is, \( k \in K \) scenarios are sampled and considered, with \( K \subset S \) and \( |K| \ll |S| \). However, the subproblems and the associated value functions \( Q_{R,t}(x_{t-1}^k, \xi_t^k) \) differ from the risk-neutral case. Instead of subproblem (1.15), one of the dynamic programming equations from Table 2 is chosen and the occurring risk value functions \( Q_{R,t+1}(\cdot) \) are replaced by an approximating function \( \mathcal{Q}_{R,t+1}(\cdot) \) or an epigraph variable \( \theta_{t+1} \), respectively.

In the backward pass, as in risk-neutral SDDP, those subproblems are solved for each \( t = T, \ldots, 1 \), trial solution \( x_{t-1}^k, k \in K \), and possible stage-\( t \) realization.
STOCHASTIC DUAL DYNAMIC PROGRAMMING

- general 
  - augmented state, 
    sophisticated formula for $Q_t(\cdot)$ [76] (7.18), (7.19), (7.20) 
    additional constraints and variables [60] (7.21), (7.22), (7.23) 
    VaR $\alpha_t$ explicitly determined, 
    sophisticated formula for $Q_t(\cdot)$ [79] (7.18), (7.24), (7.20) 
  - modified probability measure [61] (7.7), (7.8) with (7.25), (7.9)

Table 2: Dynamic programming equation formulations for $(P)$ using nested convex combinations of expectations and conditional CVaR.

\[ \xi^k_{ij} \equiv \xi_{ij}, j = 1, \ldots, q_t, \] using an updated cut approximation $Q_{R,t+1}(\cdot)$. Using dual information, on stage $t$, a new cut for $Q_{R,t}(\cdot)$ is derived and handed back to stage $t - 1$. The main difference to risk-neutral SDDP is again the definition of $Q_{R,t}(\cdot)$. Therefore, the formulae for the cut intercepts and gradients have to be adapted to the individual approach chosen from Table 2.

For example, using the dynamic programming equations with modified probability measure (7.7)-(7.9) with (7.25), we obtain the cut

(7.27) \[ Q_{R,t}(x_{t-1}) \geq \phi_t(x_{t-1}) = \sum_{j=1}^{q_t} p_{ij} Q_{R,t}^{+1}(x_{t-1}, \xi_{ij}) + \beta_t^T (x_{t-1} - x_{t-1}) \]

where $\beta_t$ is defined as

\[ \beta_t = -\sum_{j=1}^{q_t} p_{ij} (\pi_i^{ij})^T T_{t-1}(\xi_{ij}). \]

For the derivations of the subgradients in the other cases, we refer to [60, 76, 79].

7.3.3. Upper Bound Determination and Stopping. The main challenge in SDDP extensions using nested conditional risk measures is to determine upper bounds for $v^*$ and, thus, allowing for a reasonable stopping criterion. The reason is that the upper bound construction method from the risk-neutral case, see Sect. 1.3, cannot be efficiently extended to the risk-averse case.

Recall that in the risk-neutral case, a feasible policy $(x_t(\xi_{[t]}))_{t=1}^T$ is determined in the backward pass and evaluated in the forward pass for different scenarios $k \in K$, yielding trial points $x^k_t$. Then, a statistical upper bound $\bar{\pi}_K$ for $v^*$ is determined as the sample average of the objective values of all these sample paths $\xi^k$, see (1.16). Analogously, a true upper bound $\pi$ can be obtained by taking the expectation of such objective value for all scenarios $\xi^s, s \in S$. However, this is possible only due to the tower property of expected values. It induces the equivalence of taking the expectation over the objective values of all scenarios, cf. (7.1), and the nested objective function assumed for $(P)$. Most nested conditional risk measures, e.g. (7.15), do not possess the tower property. Therefore, the sample average cannot directly be applied to obtain a (statistical) upper bound.

As determining reasonable upper bounds is a crucial ingredient of SDDP, developing appropriate upper bound estimators has been an active research field in the...
last decade. In the following, we discuss different approaches that have been proposed. We mostly follow the presentation of Kozmík and Morton [44], who provide a comprehensive study within their own work on upper bound estimators.

A Sample Average Estimator. In Subsection 7.3.1, we managed to formulate each $\rho^{\xi k}_t[\cdot]$ only by means of expectations in (7.19). Still, this does not imply the tower property, since the risk value functions $Q_{\mathcal{R}, t}(\cdot)$ contain a nested nonlinearity due to the $[\cdot]_+$-function. However, we can derive an estimator similar to (1.16) [44]. To this end, we remove the expectation in (7.19) to obtain

$$\hat{v}_t(\xi^k_t) := (1 - \lambda_t) \left( c^T_t (\xi^k_t)x^k_t + \hat{v}_{t+1}(\xi^k_t) \right)$$

$$+ \lambda_t u^k_{t-1} + \frac{\lambda_t}{\alpha_t} \left[ c^T_t (\xi^k_t)x^k_t + \hat{v}_{t+1}(\xi^k_t) - u^k_{t-1} \right]_+,$$

where we replace the value functions $Q_{\mathcal{R}, t+1}(\cdot)$ by the estimator of the following stage. For stage $T$ it follows $\hat{v}_{T+1}(\xi^k_T) \equiv 0$ and for the first stage

$$\hat{v}(\xi^k) := c^T_1 x_1 + \hat{v}_2(\xi^k_1).$$

Equation (7.29) provides a recursive estimator for the cost associated with sample path $\xi^k$. This estimator has to be evaluated by backward recursion starting with stage $T$. Determining this estimator for all scenarios $\xi^k, k \in \mathcal{K}$, sampled in the forward pass of SDDP, we can form an upper bound estimator

$$U^n := \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \hat{v}(\xi^k),$$

which resembles the sample average estimator (1.16).

Estimator (7.30) is an unbiased and consistent estimator of $\bar{v}_T$, but has a large variance. Kozmík and Morton point out several reasons for this behaviour. Only a small portion of the sampled scenarios contributes to estimating the CVaR, while most solely contribute to the expectation. Therefore, a very large number of scenarios would be required to obtain an appropriate CVaR-estimate. Additionally, since expectations are not taken conditionally on each stage as in (7.19), and due to to division by $\alpha_t \in (0, 1)$, small or large values are very likely to propagate from late to earlier stages in the recursion to determine $\hat{v}(\xi^k)$.

Conditional Sampling Estimator. For the above reasons, estimator $U^n$ in (7.30) is rarely considered in the literature on risk-averse SDDP. Instead, Shapiro proposes a conditional sampling estimator [76]. Here, the idea is to estimate the expectations (7.19) in the nested structure conditionally by sampling on each stage. Since in principle, the upper bound estimator can be determined independently of the scenarios sampled in the forward pass, we denote the set of samples by $\mathcal{M}$ instead of $\mathcal{K}$. $\mathcal{M}_t$ denotes the corresponding scenario set for stage $t$.

For each stage, $t = 2, \ldots, T$, this yields [44]

$$\hat{v}_c^c(\xi^k_t) := \frac{1}{|\mathcal{M}_t|} \sum_{m_i \in \mathcal{M}_t} \left( 1 - \lambda_t \right) \left( c^T_t (\xi^m_i)_t x^m_i_t + \hat{v}_{t+1}^c(\xi^m_i) \right)$$

$$+ \lambda_t u^m_{t-1} + \frac{\lambda_t}{\alpha_t} \left[ c^T_t (\xi^m_i)_t x^m_i_t + \hat{v}_{t+1}^c(\xi^m_i) - u^m_{t-1} \right]_+, \quad \text{(7.31)}$$
and for the first stage the estimator

\begin{equation}
U^c := c_1^T x_1 + \hat{\psi}(\xi_1).
\end{equation}

As Shapiro himself points out, this estimator has two significant drawbacks. It requires \( \prod_{t=2}^{T} |\mathcal{M}_t| + 1 \) subproblems to be solved, which is exponentially growing in the number of stages. Moreover, the obtained upper bounds are typically not very tight. Therefore, estimator \( U^c \) might not be useful for practical problems [44].

Importance Sampling Estimators. Applying conditional sampling appears computationally intractable, but the drawbacks of estimator \( U^n \) may also be addressed by importance sampling, see Sect. 3. With this technique it is possible to better represent CVaR by sampling the associated scenarios with higher importance. Based on this idea, Kozmík and Morton put forward different importance sampling upper bound estimators [44].

Using importance sampling with respect to CVaR, we face a considerable challenge, though. In order to determine the importance sampling distribution for some stage \( t \), it has to be identified which scenarios are associated with CVaR on that stage, i.e., which of them provide a value \( Q_{R,t}(x_{t-1}, \xi_{tj}) \) beyond the \( (1 - \alpha) \)-quantile. If we estimate this by solving subproblems for several \( \xi_{tj} \) and determining \( Q_{R,t}(x_{t-1}, \xi_{tj}) \), we face a similar computational burden as for conditional sampling.

Kozmík and Morton propose the following approach: They use an approximation function \( d_t(x_{t-1}, \xi_t) \), which estimates the recourse value of the decisions \( x_{t-1} \) after the random parameters \( \xi_t \) have been observed [44]. Instead of solving the subproblems for several \( \xi_{tj} \), they simply evaluate \( d_t(x_{t-1}, \xi_{tk}) \) and sort these values. Based on the obtained order, it can be decided then which scenarios are used to estimate CVaR, i.e., \( u_d := \text{VaR}_{\alpha_t}[d_t(x_{t-1}, \xi_t)] \) is determined.

This allows defining an importance sampling distribution depending on \( x_{t-1} \) [44]. For simplicity, we assume that all scenarios are equally likely in the original distribution, that is, \( f_t(\xi_t) = \frac{1}{q_t} \), and it follows:

\begin{equation}
g_t(\xi_t|x_{t-1}) := \begin{cases}
\frac{1}{2\lceil \alpha_t q_t \rceil}, & d_t(x_{t-1}, \xi_t) \geq u_t, \\
\frac{1}{2(q_t - \lceil \alpha_t q_t \rceil)}, & d_t(x_{t-1}, \xi_t) < u_t.
\end{cases}
\end{equation}

Basically, this distribution ensures that it is equally likely to draw sample observations above and below \( u_t \). Note that the formula presented in [44] looks a bit different, since it is presented in context of SAA and uses \( \lfloor \cdot \rfloor \) instead of \( \lceil \cdot \rceil \). In that case, however, a division by zero has to be ruled out.

Defining weights

\[ \Lambda_t(\xi_t|x_{t-1}) := \frac{f_t(\xi_t)}{g_t(\xi_t|x_{t-1})} \]

and multiplying them along the sample paths

\[ \Lambda(\xi^k) := \prod_{t=2}^{T} \Lambda_t(\xi_t^k|x_{t-1}) \]

we can derive the estimator

\begin{equation}
U^i := \frac{1}{\sum_{k \in \mathcal{K}} \Lambda(\xi^k)} \sum_{k \in \mathcal{K}} \Lambda(\xi^k) \hat{\psi}(\xi^k).
\end{equation}
This estimator is similar to (7.30), as the same recursive term \( \hat{v}(\xi^k) \) is used, but combined with importance instead of standard MC sampling.

With our assumptions of relatively complete recourse (Assumption 5) and stage-wise independence (Assumption 9), estimator (7.34) is asymptotically valid, i.e., for \(|K| \to \infty\), \( U^i \) converges to \( E_f[\hat{v}(\xi)] \) with probability 1 and \( E_f[\hat{v}(\xi)] \geq v^*_R \). Moreover, for sufficiently good choice of \( d_t(\cdot) \), it can be expected that the variance is lower than for \( U^n \) [44].

Based on the importance sampling idea for \( U^i \), even better estimators can be developed, provided that \( d_t(\cdot) \) satisfies some goodness assumptions. For example, assuming that

\[
Q_{R,t}(x_{t-1}, \xi_t) \geq \text{VaR}_{\alpha_t}[Q_{R,t}(x_{t-1}, \xi_t)] \iff d_t \geq \text{VaR}_{\alpha_t}(d_t(x_{t-1}, \xi_t)),
\]

i.e., if \( d_t(\cdot) \) correctly classifies whether a realization is in the upper \( \alpha \) tail of the recourse value distribution, \( U^i \) can be improved by using a modified recursion formula for \( \hat{v}_t(\cdot) \).

The idea is not only to sample with higher importance scenarios associated with CVaR, but also to use only such scenarios to estimate CVaR and contribute to the \([\cdot]^+\)-term [44]:

\[
\hat{v}^d(\xi_t) := (1 - \lambda_t) \left( (c_t(\xi_t)^T x_t^k + \hat{v}^d_{t+1}(\xi_t^k)) + \lambda_t u_{t-1} + I[d_t(x_{t-1}, \xi_t) \geq u_d] \frac{\lambda_t}{\alpha_{t-1}} \left[ (c_t(\xi_t)^T x_t^k) + \hat{v}^d_{t+1}(\xi_t^k) - u_{t-1} \right]^+ \right).
\]

Here \( I[\cdot] \) denotes an indicator function. For the first stage it follows

\[
\hat{v}^d(\xi) := c_1^T x_1 + \hat{v}^d_{2}(\xi_1^k).
\]

Combining this with (6.6), we obtain

\[
U^d := \frac{1}{\sum_{k \in K} \Lambda(\xi_k)} \sum_{k \in K} \Lambda(\xi_k) \hat{v}^d(\xi_k).
\]

It is proven that this estimator is asymptotically valid as well, but also provides tighter upper bounds than \( U^i \) in expectation, as long as the previously stated assumption is satisfied. Moreover, a smaller variance should be expected [44], as this estimator mitigates the multiplicative effects, which lead to high variance of \( U^n \). The practical applicability relies heavily on the goodness of the approximation function \( d_t(\cdot) \), see Assumption (7.35).

Numerical results in [44] illustrate that even for a medium number of stages \( U^d \) provides significantly lower upper bounds than \( U^n, U^c \) and \( U^i \) and that also the variance of the estimators is reduced significantly.

Apart from the above sampling estimators, some completely different strategies may be used to obtain upper bounds for \( v^*_R(t) \) or define stopping criteria for SDDP in the risk-averse case.

**Using deterministic upper bounds.** As already discussed in Sect. 5, we may circumvent the determination of sampling-based upper bound estimators completely if we resort to deterministic upper bounding procedures.

To this end, Philpott et al. [61] extend their inner approximation based upper bounding procedure to the risk-averse case with nested conditional coherent risk
measures. The main downside of this procedure, to require prohibitively large computational effort for a large number of state variables and an increasing number of cuts, also holds in this case, though.

The alternative deterministic upper bound procedure by Leclèr et al. [46] using dual SDDP has not been extended to a risk-averse setting so far.

**Determining bad outcomes beforehand.** As discussed in Sect. 7.3.1, following the approach of a change of probability measure, see (7.7), (7.8) with (7.25), (7.9), it is also possible to run SDDP once in advance to identify the probability measure and then a second time for the actual solution - this time fixing the probability measure. This is referred to as solving the change-of-measure risk neutral problem in [48]. The advantage of this approach is that indeed a risk neutral problem can be solved by risk neutral SDDP and therefore, also the classical upper bounding and stopping procedures can be applied.

**Fixing the number of iterations.** This approach is proposed by Philpott and de Matos [60]. They run a risk-neutral variant of SDDP first and then fix the number of iterations required until termination. The same number of iterations is then used in the risk-averse case, avoiding the challenge of upper bound evaluation.

In some practical applications, in which it is computationally intractable to determine a sophisticated upper bound estimator, this approach may be useful. For some numerical experiments promising results are reported in [60]. However, there is no theoretical guarantee to find a sufficiently good solution for a risk-averse version of (P_R) in the same number of iterations as for a risk-neutral version. Additionally, for large problems it may already take considerably long to run SDDP one time. Running it a second time for risk-neutral (P_R) may partially annihilate the computational advantage of avoiding upper bound estimation.

**Lower bound stabilization.** As for risk neutral SDDP, instead of using upper bounds at all, the algorithm can be terminated, once the lower bounds \( v^{R}_{i,t} \) stabilize. This provides no convergence guarantee but may be worthwhile in large-scale practical applications where other approaches become computationally prohibitive.

**Using benefit factors.** Instead of the lower bounds \( v^{R}_{i,t} \), it is also possible to condition termination of SDDP on the improvements of the cut approximations \( \Omega^{R}_{i,t}(\cdot) \), \( t = 2, \ldots, T \). For that purpose, Brandi et al. define a benefit factor

\[
B_{i,t,k} = \min \left\{ 1, \frac{\delta(x^{ik}_{t-1})}{\delta^{i}_{t,\max}} \right\},
\]

which determines how much a new cut improves the current cut approximation \( \Omega^{R}_{i,t} \) at \( x^{ik}_{t-1} \) [12]. \( \delta(x^{ik}_{t-1}) \) is the absolute increase, while \( \delta^{i}_{t,\max} \) is a proxy for the maximum improvement possible. For each sample path \( k \in K \), a total benefit factor can be determined by

\[
B^{i}_{k} = \max \left\{ B^{i}_{2,k}, B^{i}_{3,k}, \ldots, B^{i}_{T,k} \right\}.
\]

The risk-averse SDDP method is then stopped if the values \( B^{i}_{k} \) for all \( k \in K \) are below a predefined tolerance, either for one iteration or, alternatively and more robustly, for a specific number of iterations.

**7.4. SDDP with expected conditional CVaR.** We already discussed some other classes of risk measures, which also satisfy some notion of time-consistency. For instance, this is true for expected conditional risk measures, which are displayed in (7.10). It may be worthwhile to consider SDDP combined with such risk measures, considering the challenges for nested conditional risk measures.
Using $\rho_t = \text{CVaR}_\alpha$ in (7.10) yields the so called $E - \text{CVaR}$ or multi-period average value-at-risk [41]. This risk measure goes back to Pflug and Ruszczyński [59].

As stated in [41], by some lengthy reformulations, the objective function of $(P_R)$ using (7.10) can be expressed in a nested way. Thus, some dynamic programming equations can be deduced. Moreover, the $[\cdot]_+$-function can be reformulated by an epigraph approach. Then, for $t = 2, \ldots, T$, the dynamic programming equations read

$$Q_{R,t}(x_{t-1}, u_t, \xi_t) = \left\{ \begin{array}{l}
\min_{x_t, u_{t+1}, u_t} \dfrac{1}{\alpha_t} u_t + u_{t+1} + Q_{R, t+1}(x_t, u_{t+1}) \\
\text{s.t.}
W_t(\xi_t)x_t = h_t(\xi_t) - T_{t-1}(\xi_t)x_{t-1} \\
u_t - c^T_t(\xi_t)x_t(\xi_t) \geq -u_t \\
u_t \geq 0 \\
x_t \geq 0
\end{array} \right.$$

with

$$Q_{R, t+1}(x_t, u_{t+1}) = \mathbb{E}_{\xi_{t+1}} \left[ Q_{R, t+1}(x_t, u_{t+1}, \xi_{t+1}) \right],$$

$Q_{R, T+1}(\cdot) \equiv 0$ and first stage

$$v^*_R = \left\{ \begin{array}{l}
\min_{x_1, u_2} c_1^T x_1 + u_2 + Q_{R, 2}(x_1, u_2) \\
\text{s.t.} \\
W_1 x_1 = h_1 \\
x_1 \geq 0.
\end{array} \right.$$ 

Contrary to using nested conditional risk measures, the dynamic programming equations here only depend on nested sums of conditional expectations, i.e., have the same structure as in the risk-neutral case. As a result, the standard SDDP method for risk-neutral problems $(P_R)$ can be applied. With this comes the big advantage to resort also to the upper bounding techniques developed for risk-neutral SDDP.

8. SDDP for convex programs [relaxing Assumption 1]. A natural extension of SDDP can be achieved by relaxing the assumption of linearity of the objective function and constraints, i.e., Assumption 1, but assuming a convex program. Such program can be formulated as

$$(P_c) \quad v^*_c := \left\{ \begin{array}{l}
\min_{x_1} \quad f_1(x_1) + \mathbb{E}_{\xi_{t_1}[\xi_{1}]} \left[ \min_{x_2} \quad f_2(x_2, \xi_2) + \mathbb{E}_{\xi_{t_2}[\xi_{2}]} \left[ \ldots \\
+ \mathbb{E}_{\xi_{t}[\xi_{t-1}]} \left[ \min_{x_T} \quad f_T(x_T, \xi_T) \right] \right] \right]
\text{s.t.} \\
g_1(x_1) = 0 \\
g_t(x_{t-1}, x_t, \xi_t) = 0 \quad \forall t = 2, \ldots, T, \\
\forall \xi_{t-1} \in \xi_{t-1}, \xi_t \in \Xi_t[\xi_{t-1}] \text{ a.s.}
\end{array} \right.$$

with convex lower semicontinuous proper functions $f_t(\cdot)$, linear functions $g_t(\cdot)$ and nonempty convex compact sets $\mathcal{X}_t$ for $t = 1, \ldots, T$.

Problem $(P_c)$ can be equivalently expressed by dynamic programming equations, which for $t = 2, \ldots, T$ read

$$Q_{t,c}(x_{t-1}, \xi_{[t-1]}, \xi_t) := \left\{ \begin{array}{l}
\min_{x_t} \quad f_t(x_t, \xi_t) + Q_{t+1,c}(x_t, \xi_{[t]}) \\
\text{s.t.} \\
g_t(x_{t-1}, x_t, \xi_t) = 0 \\
x_t \in \mathcal{X}_t
\end{array} \right.$$
with expected value functions defined as

\begin{equation}
Q_{t+1,c}(x_t, \xi_t) := E_{\xi_{t+1}|\xi_t} \left[ Q_{t+1,c}(x_t, \xi_t, \xi_{t+1}) \right]
\end{equation}

and \( Q_{T+1,c}(x_T) \equiv 0 \). For the first stage, this yields

\begin{equation}
v^*_c = \begin{cases} 
\min_{x_1} f_1(x_1) + Q_{2,c}(x_1, \xi_1) \\
\text{s.t.} \quad g_1(x_1) = 0 \\
x_1 \in X_1.
\end{cases}
\end{equation}

Assuming that \( Q_{T,c}(\cdot) \) is Lipschitz continuous, all expected value functions \( Q_{t,c}(\cdot) \) are convex and Lipschitz continuous on \( X_{t-1} \), as can be shown by backward recursion [25]. Therefore, they can be approximated by linear cutting planes. Such cuts can be generated using SDDP.

Contrary to the linear case, however, the expected value functions \( Q_{t,c}(\cdot) \) are no longer polyhedral. Thus, they cannot be represented exactly by a finite number of cuts. Still, it can be shown under some assumptions that finitely many cuts are sufficient to ensure convergence of SDDP. More precisely, we assume the following constraint qualification, which generalizes relatively complete recourse [25]:

**Assumption 10.** Let \( B_t(\delta) := \{ y \in Aff(X_t) : \|y\| < \delta \} \) where \( Aff(X_t) \) denotes the affine hull of \( X_t \). Problem \((P_c)\) has extended relatively complete recourse if there exists some \( \delta > 0 \) such that for all \( t = 1, \ldots, T-1 \),

(a) \( \forall x_{t-1} \in X_{t-1} + B_{t-1}(\delta) \) and \( \forall \xi_t \in \Xi_t ; \)

\( \{ x_t \in \mathbb{R}^n : g_t(x_{t-1}, x_t, \xi_t) = 0, x_t \in X_t \} \neq \emptyset. \)

(b) \( \forall x_{t-1} \in X_{t-1} + B_{t-1}(\delta) \) and \( \forall \xi_t \in \Xi_t ; \)

\( f_t(x_{t-1}, \xi_t) < \infty. \)

Under Assumption 10, the norm of the subgradients obtained in the course of SDDP can be shown to be bounded. Using this result and assuming i.i.d. sampling with positive probabilities for all possible scenarios in the SDDP forward pass, Girardeau et al. provide a proof of finite almost sure convergence of SDDP applied to problem \((P_c)\) [25].

While most research on SDDP deals with multistage stochastic linear programming problems, some of the extensions presented in Sect. 2-7 have also been enhanced to the convex case, e.g., risk-aversion [26], inexact cuts [28], regularization [30] or exact upper bounding procedures [4, 46]. [26] contains an extension of the convergence proof from [25] to the risk-averse case.
REFERENCES


[52] D. P. Morton. An enhanced decomposition algorithm for multistage stochastic hydroelectric


