Computational Optimization and Applications (2019) 74:1–42
https://doi.org/10.1007/s10589-019-00104-x

On level regularization with normal solutions in decomposition methods for multistage stochastic programming problems

Wim van Ackooij[1] · **Welington de Oliveira**[2] · **Yongjia Song**[3]

Received: 6 December 2017 / Published online: 2 May 2019

Abstract
We consider well-known decomposition techniques for multistage stochastic programming and a new scheme based on normal solutions for stabilizing iterates during the solution process. The given algorithms combine ideas from finite perturbation of convex programs and level bundle methods to regularize the so-called *forward step* of these decomposition methods. Numerical experiments on a hydrothermal scheduling problem indicate that our algorithms are competitive with the state-of-the-art approaches such as *multistage regularized decomposition*, *nested decomposition* and *stochastic dual dynamic programming*.

Keywords Normal solution · SDDP algorithm · Stochastic optimization · Nonsmooth optimization

1 Introduction

Multistage stochastic programs with recourse are important modeling tools in real-life applications such as the ones coming from the areas of energy [7,22,27,39,44,49], transportation [20,23] and finance [16,17,41]. The typical approach to solve these

✉ Welington de Oliveira
welington.oliveira@mines-paristech.fr

Wim van Ackooij
wim.van-ackooij@edf.fr

Yongjia Song
yongjis@clemson.edu

1 EDF R&D. OSIRIS 7, Boulevard Gaspard Monge, F-91120 Palaiseau Cedex, France

2 MINES ParisTech, PSL - Research University, CMA - Centre de Mathématiques Appliquées, Sophia Antipolis, France

3 Clemson University, Clemson, SC, USA

problems is to approximate the underlying random process by using a scenario tree. This yields, in general, large-scale mathematical programming problems that can be handled using specialized algorithms that employ decomposition techniques (and very often sampling). Two very popular decomposition schemes for handling multistage stochastic programs are the *nested decomposition* (ND) proposed by [4] and the *stochastic dual dynamic programming* (SDDP) proposed by [40]. Both methods underestimate the cost-to-go functions (resulting from dynamic programming) using iteratively refined piecewise linear functions, defined by cutting planes computed by solving linear programs (LPs) in the so-called *backward step*.

Due to curse of dimensionality, ND is usually applied to multistage stochastic problems of moderate size (e.g., up to hundreds of scenarios). Under the assumption that the underlying stochastic process is stagewise independent, much larger scenario trees can be handled by combining decomposition and sampling, [5,15,25,40]. The optimization strategies proposed in these references mitigate the curse of dimensionality by sharing cuts among different nodes of the scenario tree. Among these methods, the most successful algorithm for solving large-scale multistage stochastic programs is SDDP [40]. Mathematical properties of SDDP have been extensively investigated. The first formal proof of almost-sure convergence of multistage sampling algorithms akin to SDDP is due to Chen and Powell [5]. This proof is extended by Linowsky and Philpott to cover the SDDP algorithm and other sampling-based methods in [35]. The convergence analysis of SDDP is revisited in [43], and more recently in [47].

Nested decomposition and SDDP techniques have been applied in a variety of applications, just to mention a few: [10,22,23,42,49]. An extensive computational study and new algorithmic techniques to accelerate the ND's solution process in a parallel enviroment are presented in [54]. Nested decomposition and SDDP have also been extended to cope with more general convex multistage stochastic programs, see e.g., [21].

The workhorse in the optimization techniques mentioned above is the Kelley's cutting-plane method [29], which is well-known for having slow convergence (in some cases, the number of trial points generated by the cutting-plane method grows exponentially with the problem dimension, [38, pp. 158-160]). As a result, ND and SDDP can also exhibit slow convergence, especially in high-dimensional problems. It has been largely demonstrated, in deterministic convex and nonsmooth optimization, that bundle methods [32–34] provide faster convergence than the Kelley's cutting-plane method, due to a regularization scheme that stabilizes iterates during the solution process. Such a class of methods has been successfully adapted and extended to two-stage stochastic programs in [9,11,18,45,51,53]. In the two-stage context, the stochastic decomposition of [24] combines sampling with a regularization scheme akin to proximal bundle methods. In [50] regularization is also combined with dynamic selection/aggregation of scenarios.

In the multistage setting, deploying these regularization ideas is more involved. Two regularization-based algorithms for multistage stochastic programming were recently proposed in [1] and [46]. Sen and Zhou [46] have extended the stochastic decomposition of [24] to the multistage setting. At every iteration of their algorithm, a sample path is randomly drawn and trial points are obtained by solving a quadratic program regularizing candidate solutions. Regularization is performed by adding a quadratic

term to the objective function of the LP subproblems, which is used to prevent next trial points to be far away from the ones used in the current iteration.

Asamov and Powell [1] propose the same kind of regularization as [46], but extend the regularized decomposition of [45] to the multistage stochastic programming setting with an algorithmic scheme closer to SDDP. Regularization is performed only in the forward step, and the multistage regularized decomposition employs the same backward step as the one in SDDP.

Both works [46] and [1] make use of stability centers, which strongly impact the iteration process by keeping iterates close to such centers. Stability centers can be understood as "anchors", the surroundings of which should be explored priority-wise by the algorithm. In general, a stability center is set to the best solution candidate obtained up to the current iteration of the algorithm. While such points are easily determined in the two-stage setting (because the objective function is known through an oracle), it turns out that choosing stability centers in the multistage setting is a tricky task. Choosing a stability center for every node of the underlying scenario tree would be impractical for large scenario trees commonly handled by SDDP.

In [1] the authors cope with this difficulty by defining stability centers as part of the state variables (denoted by post-decision state variables, see Definition 3 in [1]) of x_t for each stage t generated by the forward step in the previous iteration. When stability centers are improved due to better approximations of the cost-to-go functions, the algorithm of [1] diminishes the influence of these stability centers in the regularization scheme by decreasing the prox-parameter weighing the quadratic term. In other words, regularization is dismissed when it is still needed: it is well known in the nonsmooth optimization community that regularization plays a major role when iterates get closer to the set of optimal solutions.

To overcome this shortcoming, we propose a scheme for regularizing decomposition methods for multistage stochastic linear programs (MSLPs) that does not suffer from the effect of possibly bad quality stability centers. Our algorithms take inspiration in the level bundle methods for nonsmooth optimization, [31,34], well known for their efficiency [3,8], flexibility in defining stability centers and dealing with functions for which gradients and values are only computed approximately, [9].

Our approaches define trial points either as normal solutions of the LP subproblems solved by SDDP, or as specific points in the level sets of cutting planes approximating the cost-to-go functions. The given algorithms combine level bundle methods [34] with some results on finite perturbation of convex programs. As in [1] and [46], the proposed algorithms employ regularization only in the forward step. The backward step improving the cutting-plane approximations is exactly the same one employed by SDDP.

As a subproduct of the regularization scheme proposed in this paper, we show that after finitely many steps the multistage regularized decomposition of [1] boils down to a particular case of SDDP, which seeks specific policies during the forward step. As a conclusion, the convergence analysis of the multistage regularized decomposition follows directly from the analysis of SDDP, already studied in [15,43,47].

The remainder of this work is organized as follows: the dynamic equations and cost-to-go functions of a general MSLP, as well as the decomposition scheme of ND and SDDP, are presented in Sect. 2. Section 3 reviews some known results on finite

perturbation of convex programs. Such results are crucial for the normal-level ND and normal-level SDDP algorithms proposed in Sect. 4. Convergence analysis of these new algorithms are also given in this section. Section 5 reports some preliminary numerical results on a large-scale hydrothermal scheduling problem and on three smaller multistage stochastic problems found in the literature. We compare the new proposals with the multistage regularized decomposition of [1], the nested decomposition of [4] and an implementation of the standard SDDP, and show the effectiveness of our proposed regularization scheme. Finally, we conclude in Sect. 6 with some final remarks.

2 Preliminaries on multistage stochastic linear programs and decomposition schemes

Multistage stochastic programs explicitly model a series of decisions interplayed with partial observation of uncertainty. If the given set of possible realizations of the underlying stochastic process is discrete, uncertainty can be represented by a scenario tree. Multistage stochastic linear programs (MSLPs) are a special case of this class wherein the optimization problem solved in each stage is an LP. Hence, on a scenario tree the resulting model is a very large LP. In this section we will provide an overview of two popular decomposition approaches for MSLPs, *nested decomposition* (ND) and *stochastic dual dynamic programming* (SDDP). From an abstract viewpoint coming from non-linear non-smooth optimization, both methods are variants of Kelley's cutting plane method [29].

Consider the following MSLP:

$$\min_{\substack{A_1x_1=b_1\\ x_1\geq 0}} c_1^\top x_1 + \mathbb{E}_{|\xi_1}\left[\min_{\substack{B_2x_1+A_2x_2=b_2\\ x_2\geq 0}} c_2^\top x_2 + \mathbb{E}_{|\xi_{[2]}}\left[\cdots + \mathbb{E}_{|\xi_{[T-1]}}\left[\min_{\substack{B_Tx_{T-1}+A_Tx_T=b_T\\ x_T\geq 0}} c_T^\top x_T\right]\right]\right], \tag{1}$$

where some (or all) data $\xi_t = (c_t, B_t, A_t, b_t)$ can be subject to uncertainty for $t = 2, \ldots, T$. The expected value $\mathbb{E}_{|\xi_{[t]}}[\cdot]$ is taken with respect to the conditional probability measure of random vector $\xi_t \in \Xi_t$.

For numerical tractability, we assume that the number N of realizations (scenarios) of the data process is finite, i.e., support sets Ξ_t ($t = 1, \ldots, T$) have finite cardinality. This is, for instance, the case in which (1) is a sample average approximation (SAA) of a more general MSLP having a continuous probability distribution. For a discussion on the relationship between a SAA problem and the underlying true problem with a continuous distribution we refer to [47].

As in the classical work of [40] we also make the basic assumption that the random data process is stagewise independent, i.e., random vector ξ_{t+1} is independent of $\xi_{[t]}:=(\xi_1, \ldots, \xi_t)$, the history of the data process up to time t. In this case $\mathbb{E}_{|\xi_{[t]}}[\cdot]$ is simply $\mathbb{E}[\cdot]$. In some cases, problems with dependence between stages can either be reformulated to problems with stagewise independence by adding state variables to the model [47], or be dealt with by performing a cut-sharing strategy [14].

We highlight that stagewise independence is not required for either ND nor its regularized version proposed in this work. However, since we aim to present our approaches in a unified manner, we will adopt this assumption throughout this paper.

Under these assumptions, the dynamic programming equations for problem (1) take the form

$$Q_t(x_{t-1}, \xi_t) := \begin{cases} \min\limits_{x_t \geq 0} c_t^\top x_t + \mathcal{Q}_{t+1}(x_t) \\ \text{s.t.} \quad A_t x_t = b_t - B_t x_{t-1}\,, \end{cases} \tag{2}$$

where $\mathcal{Q}_{t+1}(x_t) := \mathbb{E}[Q_{t+1}(x_t, \xi_{t+1})]$ for $t = T-1, \ldots 1$, and $\mathcal{Q}_{T+1}(x_T) := 0$. The first-stage problem becomes

$$\begin{cases} \min\limits_{x_1 \geq 0} c_1^\top x_1 + \mathcal{Q}_2(x_1) \\ \text{s.t.} \quad A_1 x_1 = b_1\,. \end{cases} \tag{3}$$

An implementable policy for (1) is a collection of functions $\bar{x}_t = \bar{x}_t(\xi_{[t]})$, $t = 1, \ldots, T$. Such a policy gives a decision rule at every stage t of the problem based on a realization of the data process up to time t and based on previous decisions. A policy is feasible for problem (1) if it satisfies all the constraints under any realization in each stage t.

As in [47], we assume that the *cost-to-go* functions $\mathcal{Q}_t(\cdot)$'s are finite valued, in particular we assume *relatively complete recourse*. Since the number of scenarios is finite, the cost-to-go functions are convex piecewise linear functions [48, Chap. 3].

Decompositions. As already mentioned, two very important decomposition techniques for solving MSLPs are ND (see [4]), and SDDP (see [40]). Both methods consist of the following two main steps:

- *Forward step*, that goes from stage $t = 1$ up to $t = T$ solving subproblems to define feasible policies $\bar{x}_t(\xi_{[t]})$. After this step an (estimated) upper bound $\overline{z}$ for the optimal value is determined.
- *Backward step*, that comes from stage $t = T$ down to $t = 1$ solving subproblems to generate cuts that improve the cutting-plane approximations for the cost-to-go functions $\mathcal{Q}_t(\cdot)$. After this step a lower bound $\underline{z}$ is obtained.

Below we discuss these steps in more detail, starting with the backward one.

Backward step Let $\bar{x}_t = \bar{x}_t(\xi_{[t]})$ be a trial decision at stage $t = 1, \ldots, T-1$, and $\check{\mathcal{Q}}_t(\cdot)$ be the current approximation of the cost-to-go function $\mathcal{Q}_t(\cdot)$, $t = 2, \ldots, T$, given by the maximum of a collection of cutting planes. At stage $t = T$ the following problem is solved

$$\underline{Q}_T(\bar{x}_{T-1}, \xi_T) = \begin{cases} \min\limits_{x_T \geq 0} c_T^\top x_T \\ \text{s.t.} \quad A_T x_T = b_T - B_T \bar{x}_{T-1} \end{cases} \tag{4}$$

for all $\xi_T = (c_T, B_T, A_T, b_T) \in \Xi_T$. Let $\bar{\pi}_T = \bar{\pi}_T(\xi_T)$ be an optimal dual solution of problem (4). Then $\alpha_T := \mathbb{E}[b_T^\top \bar{\pi}_T]$ and $\beta_T := -\mathbb{E}[B_T^\top \bar{\pi}_T] \in \partial \mathcal{Q}_T(\bar{x}_{T-1})$ define the linearization

$$q_T(x_{T-1}) := \beta_T^\top x_{T-1} + \alpha_T \quad = \quad \mathcal{Q}_T(\bar{x}_{T-1}) + \langle \beta_T, x_{T-1} - \bar{x}_{T-1} \rangle\,,$$

satisfying

$$\mathcal{Q}_T(x_{T-1}) \geq q_T(x_{T-1}) \quad \forall\, x_{T-1}\,,$$

and $\mathcal{Q}_T(\bar{x}_{T-1}) = q_T(\bar{x}_{T-1})$, i.e., $q_T(\cdot)$ is a supporting plane for $\mathcal{Q}_T(\cdot)$. This linearization is added to the collection of supporting planes of $\mathcal{Q}_T(\cdot)$: $\check{\mathcal{Q}}_T(\cdot)$ is replaced by $\check{\mathcal{Q}}_T(x_{T-1}) := \max\{\check{\mathcal{Q}}_T(x_{T-1}), q_T(x_{T-1})\}$. In other words, the cutting-plane approximation $\check{\mathcal{Q}}_T(\cdot)$ is constructed from a collection J_T of linearizations:

$$\check{\mathcal{Q}}_T(x_{T-1}) = \max_{j\in J_T}\{\beta_T^{j\top} x_{T-1} + \alpha_T^j\}\,.$$

By starting with a model approximating the cost-to-go function from below, the cutting-plane updating strategy will ensure that $\check{\mathcal{Q}}_T(\cdot)$ is an underestimate of $\mathcal{Q}_T(\cdot)$. The updated model $\check{\mathcal{Q}}_T(\cdot)$ is then used at stage $T-1$, and the following problem needs to be solved for all $t = T-1, \ldots, 2$:

$$\begin{aligned}\underline{\mathcal{Q}}_t(\bar{x}_{t-1}, \xi_t) &= \begin{cases} \min\limits_{x_t \geq 0} c_t^\top x_t + \check{\mathcal{Q}}_{t+1}(x_t) \\ \text{s.t.} \quad A_t x_t = b_t - B_t \bar{x}_{t-1} \end{cases} \\ &\equiv \begin{cases} \min\limits_{(x_t, r_{t+1}) \in \mathbb{R}^{n_t}_+ \times \mathbb{R}} & c_t^\top x_t + r_{t+1} \\ \text{s.t.} & A_t x_t = b_t - B_t \bar{x}_{t-1} \\ & \beta_{t+1}^{j\top} x_t + \alpha_{t+1}^j \leq r_{t+1}, \quad j \in J_{t+1}\,. \end{cases} \end{aligned} \tag{5}$$

Let $\bar{\pi}_t = \bar{\pi}_t(\xi_t)$ and $\bar{\rho}_t^j = \bar{\rho}_t^j(\xi_t)$ be optimal dual multipliers associated with constraints $A_t x_t = b_t - B_t \bar{x}_{t-1}$ and $\beta_{t+1}^{j\top} x_t + \alpha_{t+1}^j \leq r_{t+1}$, respectively. Then the linearization

$$q_t(x_{t-1}) := \beta_t^\top x_{t-1} + \alpha_t \quad = \quad \mathbb{E}[\underline{\mathcal{Q}}_t(\bar{x}_{t-1}, \xi_t)] + \langle \beta_t, x_{t-1} - \bar{x}_{t-1} \rangle$$

of $\mathcal{Q}_t(\cdot)$ is constructed with

$$\alpha_t := \mathbb{E}[b_t^\top \bar{\pi}_t + \sum_{j\in J_{t+1}} \alpha_{t+1}^j \bar{\rho}_t^j] \quad \text{and} \quad \beta_t := -\,\mathbb{E}[B_t^\top \bar{\pi}_t] \in \mathbb{E}[\partial \underline{\mathcal{Q}}_t(\bar{x}_{t-1}, \xi_t)] \tag{6}$$

and satisfies $\mathcal{Q}_t(x_{t-1}) \geq q_t(x_{t-1}) \quad \forall\ x_{t-1}$. (To see why the inequality above holds, set $t = T-1$ and recall that $\check{\mathcal{Q}}_T(\cdot)$ approximates $\mathcal{Q}_T(\cdot)$ from below; then $\underline{\mathcal{Q}}_{T-1}(\bar{x}_{T-2}, \xi_{T-1}) \leq \mathcal{Q}_{T-1}(\bar{x}_{T-2}, \xi_{T-1})$ yielding that $q_{T-1}(\cdot)$ underestimates $\mathcal{Q}_{T-1}(\cdot)$. The result then follows from induction on T, $T-1, \ldots, t$.) Once the above linearization is computed, the cutting-plane model at stage t is updated: $\check{\mathcal{Q}}_t(x_{t-1}) = \max\{\check{\mathcal{Q}}_t(x_{t-1}), q_t(x_{t-1})\}$. Since $\check{\mathcal{Q}}_{t+1}(\cdot)$ can be a rough approximation (at early iterations) of $\mathcal{Q}_{t+1}(\cdot)$, the linearization q_t is not necessarily a supporting plane (but a cutting plane) for $\mathcal{Q}_{t+1}(\cdot)$: $q_t(\cdot)$ might be a strict underestimator for $\mathcal{Q}_t(\cdot)$ for all x_{t-1} feasible.

At the first stage, the following LP is solved

$$\underline{z} = \begin{cases} \min\limits_{x_1 \geq 0} c_1^\top x_1 + \check{\mathcal{Q}}_2(x_1) \\ \text{s.t.} \quad A_1 x_1 = b_1 \end{cases} \equiv \begin{cases} \min\limits_{(x_1, r_2) \in \mathbb{R}^{n_1}_+ \times \mathbb{R}} c_1^\top x_1 + r_2 \\ \text{s.t.} \quad A_1 x_1 = b_1 \\ \qquad\quad \beta_2^{j\top} x_1 + \alpha_2^j \leq r_2, \quad j \in J_2 \,. \end{cases} \tag{7}$$

The value $\underline{z}$ is a lower bound for the optimal value of (1). The computed cutting-plane models $\check{\mathcal{Q}}_t(\cdot)$, $t = 2, \ldots, T$, and a solution $\bar{x}_1$ of problem (7) can be used for constructing an implementable policy as follows.

Forward step Given a scenario $\xi = (\xi_1, \ldots, \xi_T) \in \Xi_1 \times \ldots \times \Xi_T$ (realization of the stochastic process), decisions $\bar{x}_t = \bar{x}_t(\xi_{[t]})$, $t = 1, \ldots, T$, are computed recursively going forward with $\bar{x}_1$ being a solution of (7), and $\bar{x}_t$ being an optimal solution of

$$\begin{cases} \min\limits_{x_t \geq 0} c_t^\top x_t + \check{\mathcal{Q}}_{t+1}(x_t) \\ \text{s.t.} \quad A_t x_t = b_t - B_t \bar{x}_{t-1} \end{cases} \equiv \begin{cases} \min\limits_{(x_t, r_{t+1}) \in \mathbb{R}^{n_t}_+ \times \mathbb{R}} c_t^\top x_t + r_{t+1} \\ \text{s.t.} \quad A_t x_t = b_t - B_t \bar{x}_{t-1} \\ \qquad\quad \beta_{t+1}^{j\top} x_t + \alpha_{t+1}^j \leq r_{t+1}, \quad j \in J_{t+1} \,. \end{cases} \tag{8}$$

for all $t = 2, \ldots, T$, with $\check{\mathcal{Q}}_{T+1}(\cdot) := 0$. Notice that $\bar{x}_t$ is a function of $\bar{x}_{t-1}$ and $\xi_t = (c_t, A_t, B_t, b_t)$, i.e., $\bar{x}(\xi_{[t]})$ is a feasible and implementable policy for problem (1) (up to stage t). As a result, the value

$$\bar{z} = \mathbb{E}\left[\sum_{t=1}^{T} c_t^\top \bar{x}_t(\xi_{[t]})\right] \tag{9}$$

is an upper bound for the optimal value of (1) as long as all scenarios $\xi \in \Xi$ are considered for computing the policy. This is the case for ND. However, the forward step of SDDP consists in taking a sample $\mathcal{J}$ with $|\mathcal{J}| \ll N$ scenarios $\{\xi^j\}_{j \in \mathcal{J}}$ of the data process and computing $\bar{x}_t(\xi^j_{[t]})$ and the respective values $z(\xi^j) = \sum_{t=1}^{T} c_t^\top \bar{x}_t(\xi^j_{[t]})$, $j \in \mathcal{J}$. The sample average $\tilde{z} = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} z(\xi^j)$ and the sample variance $\tilde{\sigma}^2 = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} (z(\xi^j) - \tilde{z})^2$ are easily computed. The sample average is an unbiased estimator of the expectation (9) (that is an upper bound for the optimal value of (1)). In the case of sampling, $\tilde{z} + 1.96\tilde{\sigma}/\sqrt{|\mathcal{J}|}$ gives an upper bound for the optimal value of (1) with confidence of approximately 95%. As a result, a possible stopping test for SDDP is $\tilde{z} + 1.96\tilde{\sigma}/\sqrt{|\mathcal{J}|} - \underline{z} \leq \epsilon$, for a given tolerance $\epsilon > 0$. We refer to [47, Sec.3] for a discussion on this topic.

3 Normal solutions and finite perturbation of convex programs

The regularization scheme of MSLPs to be proposed later in Sect. 4 relies on finite perturbation of convex programs. We put aside for a while the uncertainty and focus on deterministic convex programs, thus generalizing individual subproblems (8) solved during the forward step. In what follows we collect several well-known results on

finite perturbation of convex programs of the form:

$$\min_{x\in X} f(x)\,, \quad \text{where} X \subset \mathbb{R}^n \text{is a polyhedral set and } f:\mathbb{R}^n \to \mathbb{R} \text{ is a convex function.} \tag{10}$$

We assume that f is bounded from below on X and that (10) has a minimizer. We denote the set of optimal solutions of (10) as X^*, and the optimal value of (10) as f^*.

Let $\varphi : \mathbb{R}^n \to \mathbb{R}$ be another convex function. In this section we are concerned with the following problem:

$$\min_{x\in X^*} \varphi(x)\,, \tag{11}$$

which is equivalent to $\min_{x\in X} \varphi(x)$ s.t. $f(x) \leq f^*$. We make the assumption that problem (11) has a solution. When $\varphi(x) = x^\top x$, the unique solution of (11) is said to be the *normal solution* of problem (10).

As an attempt to solve (11), one might consider solving the following perturbed problem

$$\min_{x\in X} f(x) + \frac{1}{\tau}\varphi(x)\,, \quad \text{with } \tau > 0 \text{ a given parameter.} \tag{12}$$

The work [19] studies finite perturbation of (11), that is, assumptions on φ and (10) that ensure existence of a finite parameter $\bar{\tau} > 0$ such that for all $\tau \geq \bar{\tau}$, any solution of (12) is also a solution of (11). Finite perturbation is achieved, for instance, when f is polyhedral and φ is a strongly convex function; see Proposition 1 below. Prior to showing this result, we gather some results related to [6, Theorem 2.1] and [30, Lemma 2.1] in the following lemma, the proof of which is moved to the appendix.

Lemma 1 *Let $\tau > 0$ be a given parameter, $x(\tau)$ be an optimal solution of problem (12), and φ^* be the optimal value of (11), which is assumed to be finite. Then:*

(a) $\varphi(x(\tau)) \leq \varphi^*$ *for all* $\tau > 0$.

Let $\{\tau_k\}$ be an arbitrary sequence of positive numbers. If φ is a strongly convex function and $\tilde{x}$ is the unique solution of problem (11), then

(b) $\{x(\tau_k)\}$ *is a bounded sequence and there exists a constant $L < \infty$ bounding all subgradients of φ at $x(\tau_k)$ for all k.*

(c) $\varphi(\tilde{x}) = \varphi(x(\tau)) \quad \Leftrightarrow \quad f(\tilde{x}) = f(x(\tau)) \quad \Leftrightarrow \quad x(\tau) \in X^*$, *and if one of these three equalities holds for τ, then $x(\tau') = \tilde{x}$ for all $\tau' \geq \tau$.*

Finite perturbation of a class of optimization problems is established in the following proposition.

Proposition 1 *Let $X \subseteq \mathbb{R}^n$ be a polyhedral set and $f : \mathbb{R}^n \to \mathbb{R}$ be a convex piecewise linear function.*

1. *If φ is a strongly convex function, then there exists $\bar{\tau} > 0$ such that*

$$\operatorname*{argmin}_{x\in X^*} \varphi(x) = \operatorname*{argmin}_{x\in X} \; f(x) + \frac{1}{\tau}\varphi(x) \quad \forall \tau \geq \bar{\tau}\,.$$

2. *Suppose that φ is only convex (not necessarily strongly convex) and X is a bounded polyhedral set. Then there exists $\bar{\tau} > 0$ such that*

$$\operatorname*{argmin}_{x \in X^*} \varphi(x) \supseteq \operatorname*{argmin}_{x \in X} \; f(x) + \frac{1}{\tau}\varphi(x) \quad \forall \tau \geq \bar{\tau}\,.$$

Proof 1. Let i_X be the indicator function of the polyhedral set X and $F(x) := f(x) + i_X(x)$. Then (12) is equivalent (in terms of solution set and optimal value) with the unconstrained problem $\min_{x \in \mathbb{R}^n} F(x) + \varphi(x)/\tau$. Optimality conditions for this problem ensure that there exist vectors $v(\tau) \in \partial F(x(\tau))$ and $s(\tau) \in \partial\varphi(x(\tau))$ such that $0 = v(\tau) + \frac{1}{\tau}s(\tau)$. By employing this equation with $\tau = \tau_k$, passing to the limit as $\tau_k \to \infty$ and recalling Lemma 1 item (b) we obtain

$$\lim_{k\to\infty} \|v(\tau_k)\| = \lim_{k\to\infty} \left\| -\frac{1}{\tau_k} s(\tau_k) \right\| \leq \lim_{k\to\infty} \frac{L}{\tau_k} = 0\,.$$

As f and X are polyhedral, so is function F. This ensures that there exist only finitely many different subdifferential sets ∂F. Hence, there exists a constant $\eta > 0$ such that if $\partial F(\bar{x}) \cap B(0, \eta) \neq \emptyset$ for some $\bar{x} \in \mathbb{R}^n$, then $\bar{x} \in \operatorname{argmin}_{x \in X} F(x)$. (Here $B(0, \eta)$ stands for the closed Euclidean ball centered at zero and with radius η.) Since $v(\tau_k) \in \partial F(x(\tau_k))$ and $\lim_{k\to\infty} v(\tau_k) = 0$, there exists an index $\bar{k}$ such that $\|v(\tau_{\bar{k}})\| < \eta$, implying that $x(\tau_{\bar{k}}) \in X^*$. The result thus follows by taking $\bar{\tau} = \tau_{\bar{k}}$ and by using Lemma 1 item (c).

2. Without the strong convexity assumption, problems (11) and (12) can have more than one solution for every given $\tau_k > 0$. Let us assume that $x(\tau_k)$ is an arbitrary solution of (12). By noting that $\{x(\tau_k)\}$ is a bounded sequence and the subdifferential $\partial\varphi(x)$ is uniformly bounded on the compact set X, we can use the same reasoning employed in the previous item to conclude that there exists a $\bar{k}$ such that $\bar{x} = x(\tau_{\bar{k}})$ satisfies $\bar{x} \in X^*$ and, by Lemma 1 item (a) $\varphi(\bar{x}) \leq \varphi^*$. As a result, $\bar{x}$ also solves (11).

□

Under the assumption of Proposition 1 item 1) and $\varphi(x) = \frac{1}{2}x^\top x$, the work [30] proposes an algorithm that solves problem (11) after finitely many steps by successively solving (12) with an appropriate rule for defining τ. In the next section we rely on Proposition 1 with $x = x_t$ and $f(x_t) = c_t^\top x_t + \breve{\mathcal{Q}}_{t+1}(x_t)$ to regularize subproblem (8) for each stage $t = 2, 3, \ldots, T-1$ during the forward step of ND and SDDP.

4 Regularized decompositions for multistage stochastic linear programs

Since both ND and SDDP employ a cutting-plane method in all stages $t = 1, \ldots, T-1$, its convergence can be slow. Due to its proximity with cutting plane methods and the success of bundle-type stabilization (e.g., [13]), an attempt to stabilize ND and SDDP similarly is a natural idea. There is however an essential difficulty hindering the direct

application of bundle methods to problem (2)–(3): feasible sets change along the iterative process due to the dynamic equation $A_t x_t = b_t - B_t \bar{x}_{t-1}$ (with $\bar{x}_{t-1} = x_{t-1}^k$ being the current iterate k at the previous stage $t-1$). Convergence analysis of bundle methods rely on the fact that the feasible set is fixed. In particular, proximal and trust-region bundle methods require the stability center to be feasible at the current iteration (a stability center $\hat{x}_t$ can be seen as a landmark that assists, together with a stability function φ, the search for trial points; see (13) below where φ is a quadratic function). As argued, this might not be the case for a direct application of the bundle methods to MSLPs.

A more flexible bundle method is of the level family, see, e.g., [31,34]. This class of bundle methods does not require stability centers to be feasible, but relies on valid lower bounds that are iteratively updated to approach the optimal value of the convex program; see for instance [9,18,31,34,53]. This property cannot be ensured for subproblems (8) because, once again, feasible sets are not fixed and thus a current lower bound might not be valid in subsequent iterations (except at the stage $t = 1$, in which the feasible set does not change). As a result, bundle methods cannot be applied to MSLPs without considerable modifications. A first attempt in this direction is the main goal of this paper.

Remark 1 Regularizing only the first stage of MSLPs can be done without much difficulties by following the general lines of recent bundle methods [9] and [13], which can deal with inexact computations of the objective function and subgradients. Notice that vector β_2 computed according to (6) is a subgradient of $\mathbb{E}[\underline{Q}_2(x_1, \xi_2)]$ but not necessarily of $\mathcal{Q}_2(x_1)$ (at least at the beginning of the iterative process). Nevertheless, we could take $c_1 + \beta_2$ as an inexact subgradient of $c_1^\top x_1 + \mathcal{Q}_2(x_1)$; and $c_1^\top x_1 + \mathbb{E}[\underline{Q}_2(x_1, \xi_2)]$ (respec. $\tilde{z}$) as a lower approximation (respec. upper approximation) of $c_1^\top x_1 + \mathcal{Q}_2(x_1)$. Since the feasible set does not change in the first-stage problem, the theory of [13] is applicable to the context by seeing the forward and backward steps as an oracle procedure returning (inexact) information on $\mathcal{Q}_2(\cdot)$. However, stabilizing only the first-stage subproblem in MSLPs may be unsatisfactory, even if the benefits in the two-stage case can be spectacular (e.g., [52]).

4.1 A connection between regularized decomposition and finite perturbation of convex programs

An attempt to accelerate SDDP through regularization has been proposed in [1]. The regularized decomposition of [1] employs the backward step of SDDP but modifies the forward step by replacing subproblem (8) with the quadratic program:

$$\begin{cases} \min\limits_{x_t \geq 0} c_t^\top x_t + \check{\mathcal{Q}}_{t+1}(x_t) + \frac{1}{2\tau}\langle x_t - \hat{x}_t, G_t(x_t - \hat{x}_t)\rangle \\ \text{s.t.} \quad A_t x_t = b_t - B_t \bar{x}_{t-1}\,, \end{cases} \tag{13}$$

where G_t is a positive semi-definite matrix and $\hat{x}_t$ is a given stability center. In [1] the authors define G_t from the identity matrix and set some elements to zero in the diagonal, to stabilize only the (post-decision) state variable R_t^x in the vector x_t. Such

a post-decision state variable R_t^x (see Definition 3 in [1]) is a part of x_t that does not depend directly on the sample path history nor the current node being visited. The stability center for iteration $k+1$ at time stage t is chosen to be the state visited in iteration k at stage t.

The quadratic term is weighted by the prox-parameter $\tau > 0$, which is required to satisfy $\tau \to \infty$ to ensure convergence of the regularized decomposition according to [1, Theorem 6].

Under the light shed by Proposition 1 item 2) with $\varphi(\cdot) = \langle \cdot - \hat{x}_t, G_t(\cdot - \hat{x}_t)\rangle$ (provided the feasible set is bounded), there exists a constant $\bar{\tau} < \infty$ such that the solution of problem (13) also solves

$$\begin{cases} \min_{x_t \in \mathbb{R}^{n_t}} & \langle x_t - \hat{x}_t, G_t(x_t - \hat{x}_t)\rangle \\ \text{s.t.} & x_t \in S_t^*(x_{t-1}), \end{cases} \quad \text{where } S_t^*(x_{t-1}) \text{ is the set of optimal solutions to problem (8),} \tag{14}$$

for all $\tau \geq \bar{\tau}$. Hence the regularized decomposition of [1] boils down to a variant of SDDP (that seeks particular solutions of (8) in the forward step). Consequently, the results of [1, Theorem 6] can be simplified to the analysis of SDDP; see for instance [15,43] and [47, Proposition 3.1].

Although the method of [1] becomes a special case of SDDP when τ is large enough, the former is an enhancement of the latter when problem (8) has multiple solutions: the quadratic term $\langle x_t - \hat{x}_t, G_t(x_t - \hat{x}_t)\rangle$ in (13) allows for picking specific points (post-decision state variables) in the set of optimal solutions to (8). However, if the subproblems do not have multiple solutions then the method of [1] does not provide any regularization when τ is large enough. One can argue that this will happen only late in the iteration process when trial points are approaching the solution set. However, regularization plays, in general, a significant role when iterates get closer to the set of optimal solutions.

In what follows we rely on level bundle methods for proposing another type of regularization to multistage stochastic programs. Depending on a level parameter ℓ_t estimating the optimal value of (2), the regularization by level sets (of cutting-plane models) can be active till the end of the iterative process.

4.2 Normal and level iterates

We next provide a scheme akin to level bundle methods for regularizing the decomposition methods for MSLPs that does not rely on specific choices of stability centers $\hat{x}_t$, which, e.g., can be simply chosen as the vector of zeros. As in the method of [1], our algorithm performs the backward step exactly the same way as that of SDDP. We change the forward step as follows. Let $\ell_t \in \mathbb{R}$ be a given parameter. Given a trial point $\bar{x}_{t-1}$ we define by $\mathbb{X}_t(\bar{x}_{t-1}; \ell_t)$ the set of optimal solutions to the following LP:

$$\mathbb{X}_t(\bar{x}_{t-1}; \ell_t) := \begin{cases} \underset{x_t \geq 0}{\text{argmin}} \max \left\{ c_t^\top x_t + \check{\mathcal{Q}}_{t+1}(x_t),\ \ell_t \right\} \\ \text{s.t.} \quad A_t x_t = b_t - B_t \bar{x}_{t-1}\,. \end{cases} \tag{15}$$

Instead of solving LP (8) as SDDP does, or solving the QP (13) as the multistage regularized decomposition of [1] (and [46]) does, we define the trial point $\bar{x}_t = \bar{x}_t(\xi_{[t]})$, at stage t and scenario ξ_t, as an optimal solution of

$$\begin{cases} \min\limits_{x_t \in \mathbb{R}^{n_t}} & \varphi_t(x_t) \\ \text{s.t.} & x_t \in \mathbb{X}_t(\bar{x}_{t-1}; \ell_t)\,, \end{cases} \tag{16}$$

where $\varphi_t : \mathbb{R}^{n_t} \to \mathbb{R}$ is a given convex function. It follows from Proposition 1 that (16) can be solved by dealing with the perturbed problem

$$\begin{aligned} &\begin{cases} \min\limits_{x_t \geq 0} \left\{ \max\left\{ c_t^\top x_t + \check{\mathcal{Q}}_{t+1}(x_t),\ \ell_t \right\} + \frac{1}{2\bar{\tau}}\varphi_t(x_t) \right\} \\ \text{s.t.} \quad A_t x_t = b_t - B_t \bar{x}_{t-1}\,, \end{cases} \\ \equiv &\begin{cases} \min\limits_{(x_t, w_t) \in \mathbb{R}^{n_t}_+ \times \mathbb{R}} & w_t + \frac{1}{2\bar{\tau}}\varphi_t(x_t) \\ \text{s.t.} & A_t x_t = b_t - B_t \bar{x}_{t-1} \\ & c_t^\top x_t + \check{\mathcal{Q}}_{t+1}(x_t) \leq w_t \\ & \ell_t \leq w_t\,, \end{cases} \end{aligned} \tag{17}$$

for a large enough prox-parameter $\bar{\tau} > 0$. For the choice $\varphi_t(x_t) := x_t^\top x_t$, the specialized QP algorithm proposed in [30] solves (16) in finitely many steps by determining an appropriate parameter $\bar{\tau}$ for the perturbed problem above. Still for this choice, the solution $\bar{x}_t$ of (16), is called the normal solution of problem (15), justifying thus the name of the algorithms below.

Remark 2 We now distinguish two cases regarding problem (16):

- *Case 1: normal iterate* Parameter ℓ_t is less than or equal to the optimal value of (8).
  In this case, the objective function of problem (15) becomes $c_t^\top x_t + \check{\mathcal{Q}}_{t+1}(x_t)$, and the set $\mathbb{X}_t(\bar{x}_{t-1}; \ell_t)$ in (15) coincides with the set of optimal solutions $S_t^*(\bar{x}_{t-1})$ of problem (8). As a result, the point $\bar{x}_t$ obtained by solving (16) is also a solution of (8), the subproblem solved by the forward step of ND/SDDP. In this case we say that $\bar{x}_t$ is a *normal iterate*.
- *Case 2: level iterate* Parameter ℓ_t is strictly greater than the optimal value of (8). In this case, the optimal value of (15) is just ℓ_t, and the set of optimal solutions to (15) is

$$\mathbb{X}_t(\bar{x}_{t-1}; \ell_t) = \left\{ x_t \in \mathbb{R}^{n_t}_+ \;\middle|\; \begin{array}{l} A_t x_t = b_t - B_t \bar{x}_{t-1} \\ c_t^\top x_t + \check{\mathcal{Q}}_{t+1}(x_t) \leq \ell_t\,. \end{array} \right\}$$

  As a result, the point $\bar{x}_t$ obtained by solving (16) is an optimal solution of

$$\begin{array}{ll} \min\limits_{x_t \geq 0} & \varphi_t(x_t) \\ \text{s.t.} & A_t x_t = b_t - B_t \bar{x}_{t-1} \\ & c_t^\top x_t + \check{\mathcal{Q}}_{t+1}(x_t) \leq \ell_t\,, \end{array} \tag{18}$$

which is a typical subproblem of level bundle methods. In this case we say that $\bar{x}_t$ is a *level iterate*.

We highlight that the optimal value of (8) depends on the previous decisions $\bar{x}_1, \ldots \bar{x}_{t-1}$. Therefore, the algorithms presented below will have to adjust the level parameter ℓ_t in every iteration depending on the values of the current iterate $\bar{x}_t = x_t^k(\xi_{[t]})$.

4.3 The normal-level nested decomposition algorithm

In this section we do not require the assumption that the stochastic process $\{\xi_t\}_{t=1}^T$ is stagewise independent, although we employ for convenience the same notation introduced in Sect. 2.

Consider an algorithm that takes all scenarios in the forward step. In this case, $\overline{z}^k$ in (9) is indeed an upper bound for the optimal value of problem (1). We can thus define the optimality gap (observed at iteration k) by

$$\texttt{Gap}^k := \overline{z}^k - \underline{z}^k \geq 0 \quad \forall\, k = 1, 2, \ldots$$

where $\underline{z}^k$ is given in (7). If $\texttt{Gap}^k = 0$ at iteration k, the policy issuing $\overline{z}^k$ is optimal for (1).

The *Normal-level Nested Decomposition* algorithm presented below is very similar in spirit to ND, with the sole difference being that the forward step solves subproblems (16) rather than (8). We leave unspecified a rule to define the level parameters. We will discuss this topic in Sect. 4.3.1 below.

Algorithm 1 Normal-level Nested Decomposition.

Step 0: initialization Let $k = 0$ and $\check{\mathcal{Q}}_t(\cdot) := -\infty$ for $t = 2, \ldots, T$. Choose a tolerance $\epsilon \geq 0$ and parameters $\gamma_t \in (0, 1)$.
Step 1: forward For all $t = 1, \ldots, T-1$ and all $\xi_t^j \in \Xi_t$, get $(c_t, B_t, A_t, b_t) = \xi_t^j$, choose level parameters $\ell_{t,j}^k$ according to some rule, and solve subproblem (16) with $\ell_t = \ell_{t,j}^k$ to obtain $\bar{x}_t^k := \bar{x}_t(\xi_{[t]}^j)$.
Solve (4) for all scenarios and compute an upper bound $\overline{z}^k$ for (1).
Step 2: backward Compute α_T and β_T and update the model $\check{\mathcal{Q}}_T(\cdot)$. For $t = T-1, \ldots, 2$ and all $\xi_t^j \in \Xi_t$ solve LP (5), compute α_t and β_t and update model $\check{\mathcal{Q}}_t(\cdot)$. Solve LP (7) to compute $\underline{z}^k$.
Step 3: loop Define $\texttt{Gap}^k = \overline{z}^k - \underline{z}^k$ and stop if $\texttt{Gap}^k \leq \epsilon$. Otherwise set $k \leftarrow k+1$ and go back to Step 1.

Depending on the rule for defining level parameters it may happen that $\ell_{t,j}^k$ is less than the optimal value of (8). In this case, the computed trial point $\bar{x}_t = \bar{x}_t(\xi_{[t]}^j)$ is a normal iterate, meaning that such a point not only lies in the set of optimal solutions to (8), but also minimizes the function φ_t (if $\varphi_t(\cdot) = \|\cdot\|_2^2$, then $\bar{x}_t$ is indeed the normal solution of (8). If $\ell_{t,j}^k$ is greater than the optimal value of (8), then by Case

2 of Remark 2 we have that $c_t^\top \bar{x}_t^k + \check{\mathcal{Q}}_{t+1}(\bar{x}_t^k) \leq \ell_{t,j}^k$, which is equivalent, (by the development between (5)–(6)) to

$$c_t^\top \bar{x}_t^i + \mathbb{E}_{|\xi_{[t]}^j}[\underline{Q}_{t+1}(\bar{x}_t^i, \xi_{t+1})] + (c_t + \beta_{t+1}^i)^\top (\bar{x}_t^k - \bar{x}_t^i) \leq \ell_{t,j}^k \quad \forall\, i < k\,. \tag{19}$$

In this case, the computed trial point $\bar{x}_t^k = \bar{x}_t(\xi_{[t]}^j)$ is a *level iterate*, meaning that $\bar{x}_t^k$ solves (18).

Hence, while normal iterates are particular ND iterates, level iterates have a different nature and depend strongly on the given level parameter $\ell_{t,j}^k$. In order to have an effective regularization/stabilization of the optimization process, the level parameter should be properly chosen. In what follows we discuss some possible rules for defining $\ell_{t,j}^k$ in Algorithm 1.

4.3.1 Heuristic rules for setting level parameters

Level bundle methods for deterministic convex programs define the level parameter ℓ^k (estimating the optimal value of the problem) as a value between known lower and upper bounds. In the first stage of an MSLP, choosing $\ell_1^k \in [\underline{z}^{k-1}, \bar{z}^{k-1}]$ as in standard level bundle methods makes sense. However, the choice $\ell_t^k \in [\underline{z}^{k-1}, \bar{z}^{k-1}]$ for stages $t > 1$ does not seem appropriate. The reason is that the lower bound $\underline{z}^{k-1}$ (obtained at stage $t = 1$) might be greater than the value $Q_t(\bar{x}_{t-1}^k, \xi_t^j)$ given in (2) because the former estimates the cost associated with the entire time horizon, while the latter takes into account only a part of it, from stage t up to stage T.

Discount rule A more consistent manner for setting level parameters for $t > 1$ and scenario node (t, j) (at stage t and scenario ξ^j) would discount (subtract) from the lower and upper bounds the costs issued by the history of trial points $\bar{x}_\tau^k := \bar{x}_\tau^k(\xi_{[\tau]}^j)$ (for $\tau = 1, \ldots, t-1$) generated in the current forward step and scenario ξ^j. More precisely, one possible rule to define level parameters in Algorithm 1 is the following one

$$\ell_{t,j}^k \in \left[\underline{z}^{k-1} - \sum_{\tau=1}^{t-1} c_\tau^\top \bar{x}_\tau^k,\ \bar{z}^{k-1} - \sum_{\tau=1}^{t-1} c_\tau^\top \bar{x}_\tau^k\right],$$

$$\text{e.g.,}\ \ell_{t,j}^k = \gamma_t \underline{z}^{k-1} + (1-\gamma_t)\bar{z}^{k-1} - \sum_{\tau=1}^{t-1} c_\tau^\top \bar{x}_\tau^k, \tag{20}$$

for some given $\gamma_t \in (0, 1)$. We call this inexpensive and easily implementable approach the *discount rule*.

Minimum upper bound rule In this rule, we also use an estimated upper bound for $Q_t(\bar{x}_{t-1}^k, \xi_t^j)$, but we dismiss its lower bound estimation. The idea is to set level parameters to be less than the lowest (present plus future) cost associated with node (t, j). More specifically, given the node (t, j) of the scenario tree, we denote by

$\overline{z}_{t,j}^{k-1}$ (respectively $\underline{z}_{t,j}^{k-1}$) an estimated upper bound (respectively lower bound) on $Q_t(\bar{x}_{t-1}^k, \xi_t^j)$:

$$\begin{aligned}\overline{z}_{t,j}^{k-1} &:= c_t^\top \bar{x}_t^{k-1} + \sum_{\tau=t+1}^{T} \mathbb{E}_{|\xi_{[\tau-1]}^j}\left[c_\tau^\top \bar{x}_\tau^{k-1}(\xi_{[\tau]})\right] \qquad (21)\\ &\geq c_t^\top \bar{x}_t^{k-1} + \mathcal{Q}_{t+1}(\bar{x}_t^{k-1})\\ &\geq c_t^\top \bar{x}_t^{k-1} + \check{\mathcal{Q}}_{t+1}(\bar{x}_t^{k-1}) =: \underline{z}_{t,j}^{k-1}. \qquad (22)\end{aligned}$$

To this end, we keep track of the lowest value of $\overline{z}_{t,j}^{k-1}$ computed in (21):

$$\overline{z}_{\min,t,j}^{k-1} := \min_{k'\leq k-1} \overline{z}_{t,j}^{k'} = \min_{k'\leq k-1}\left\{c_t^\top \bar{x}_t^{k'} + \sum_{\tau=t+1}^{T} \mathbb{E}_{|\xi_{[\tau-1]}^j}\left[c_\tau^\top \bar{x}_\tau^{k'}(\xi_{[\tau]})\right]\right\}. \qquad (23)$$

Notice the minimum above is taken with respect to all previous computed policies $\bar{x}_t^i$ for the descendant nodes of $\xi_{[t]}^j$. As a result, for stage $t = 1$, $\bar{z}_1 = \overline{z}_1^k(\xi_{[1]})$ is an upper bound for the optimal value of problem (1). Given this bound on node (t, j), we can define the level parameter for every stage, iteration, and node of the scenario tree by

$$\ell_{t,j}^k = \overline{z}_{\min,t,j}^{k-1} - \gamma_t \,\mathtt{Gap}^{k-1}, \quad \text{with a given} \gamma_t > 0. \qquad (24)$$

This defines a simple and cheap rule for setting up level parameters.

4.3.2 Convergence analysis

It follows from the observations right after Algorithm 1 that if one periodically defines the level parameter sufficiently low so that only normal iterates are produced in such iterations, the convergence analysis of Algorithm 1 follows from the analysis of ND. Indeed, in this situation, the forward steps with level iterates in Algorithm 1 can be seen as an extra procedure for improving the cutting-plane models: it generates trial points at which the backward step constructs new (and valid) cuts that can enrich the models. Convergence of ND follows directly if one relies only on the normal iterates. As a result, any of the above heuristic rules can be applied with Algorithm 1 without hindering convergence as long as periodically one performs a *normal forward step*, i.e., only normal iterates are generated during such a step. This can be easily accomplished, for instance, by setting $\ell_{t,j}^k = -\infty$ in all nodes (t, j) of the scenario tree from time to time.

In what follows we show that when the level parameters are updated according to the minimum upper bound rule, i.e., rule (24), then Algorithm 1 is convergent without resorting to the above artifice. For all $t = 1, \ldots, T-1$ and $\xi_t^j \in \Xi_t$ we denote by $\mathcal{L}_{t,j} \subset \{1, 2, \ldots, \}$ the set of indices of iterations that correspond to level iterates. Since the backward step of the algorithm is exactly the one of standard ND, and every trial point defined in the forward step by solving (16) is either a normal or level iterate,

convergence analysis of Algorithm 1 relies on the analysis of standard ND and level bundle methods, by studying the following cases:

- if the algorithm stops generating level steps, then convergence results follow from the analysis of standard ND. In this case, the algorithm stops after finitely many steps.
- if the algorithm generates infinitely many level iterates, then convergence results are shown by following the general lines of analysis for level bundle methods. We note that, as discussed in [12], level bundle methods do not have finite convergence. Therefore, we cannot ensure that Algorithm 1 will terminate after finitely many steps when the requested accuracy ϵ is set to zero. However, this is not really an issue since a user usually will set a tolerance $\epsilon > 0$ anyway.

We start with the following lemma.

Lemma 2 *Let $t = 1, ..., T-1$ be given and assume that there exists an iteration index $\bar{k} \geq 1$ such that the lower-approximating function $x \mapsto \mathbb{E}_{\xi^j_{[t]}}[\underline{Q}_{t+1}(x, \xi_{t+1})]$ coincides with the cost-to-go function $\mathcal{Q}_{t+1}(\cdot)$ at $\bar{x}^k_t$ for all $k \geq \bar{k}$. Furthermore, suppose that the level parameter is defined by the minimum upper bound rule given by (24), and assume that for at least a fixed node (t, j), the inequality $\overline{z}^{k-1}_{\min,t,j} \leq c_t^\top x^{k'}_t + \mathcal{Q}_{t+1}(\bar{x}^{k'}_t)$ holds for all $k \geq k' \geq \bar{k}$, with $\overline{z}^{k-1}_{\min,t,j}$ given in (23). If $k \in \mathcal{L}_{t,j}$, then*

$$\|c_t + \beta^{k'}_{t+1}\| \|\bar{x}^k_t - \bar{x}^{k'}_t\| \geq \gamma_t \, \mathtt{Gap}^{k-1} \quad \textit{for all } k \geq k' \geq \bar{k}.$$

Proof If $k \in \mathcal{L}_{t,j}$, then inequality (19) becomes, under the given assumptions,

$$\begin{aligned} c_t^\top \bar{x}^{k'}_t + \mathcal{Q}_{t+1}(\bar{x}^{k'}_t) + (c_t + \beta^{k'}_{t+1})^\top (\bar{x}^k_t - \bar{x}^{k'}_t) &\leq \overline{z}^{k-1}_{\min,t,j} - \gamma_t \, \mathtt{Gap}^{k-1} \\ &\leq c_t^\top \bar{x}^{k'}_t + \mathcal{Q}_{t+1}(\bar{x}^{k'}_t) - \gamma_t \, \mathtt{Gap}^{k-1}, \end{aligned}$$

for all $k \geq k' \geq \bar{k}$. The stated result thus follows from the Cauchy-Schwarz inequality. □

Notice that the above result is always true for $t = T - 1$, because

- the definition of $\underline{Q}_T(\cdot, \cdot)$ in (4) yields $\mathbb{E}_{\xi^j_{[T-1]}}[\underline{Q}_T(\cdot, \xi_T)] = \mathcal{Q}_T(\cdot)$;
- $\overline{z}^{k-1}_{\min,T-1,j}$ is by definition the lowest known bound on the objective $c^\top_{T-1} x_{T-1} + \mathcal{Q}_T(x_{T-1})$ defined at scenario node $(T-1, j)$.

The inequality in Lemma 2 shows that a fraction of the optimality gap bounds the distances among all level iterates from below. Since the feasible set is compact, generating infinitely many level iterates will thus force the gap to vanish (because bounded sequences possess convergent subsequences in finite-dimensional spaces). This reasoning is employed by the following proposition to establish convergence of Algorithm 1.

Proposition 2 *Consider Algorithm 1 with level parameter given by (24). Assume relatively complete recourse, the feasible sets are compact, and $\varphi_t : \mathbb{R}^{n_t} \to \mathbb{R}$ are convex*

functions for all t. Moreover, assume that in the backward steps basic optimal solutions are employed. Then Algorithm 1 asymptotically computes an optimal policy for problem (1).

Proof Under the assumption of relatively complete recourse, all cost-to-go functions $\mathcal{Q}_t(\cdot)$'s are finitely valued. Furthermore, feasible sets are compact and all these assumptions imply that vectors $\beta_t, t = 2, \ldots, T$, (inexact subgradients of $\mathcal{Q}_t(\cdot)$'s) are bounded as well. Therefore, there exists a constant $\Lambda_t > 0$ such that $\|c_t - \beta_{t+1}\| \leq \Lambda_t$ for all $t = 1, \ldots, T-1$. We proceed by induction on $t = T-1,\ T-2, \ldots, 1$. Let us first fix $t = T-1$ and an arbitrary scenario node (t, j). For this stage, all assumptions of Lemma 2 are satisfied and thus $k \in \mathcal{L}_{T-1,j}$ implies $\|c_{T-1} + \beta_T^j\| \|\bar{x}_{T-1}^k - \bar{x}_{T-1}^i\| \geq \gamma_{T-1}\, \mathtt{Gap}^{k-1}$ for all $i < k$. Lemma 2 thus yields

$$\|\bar{x}_{T-1}^k - \bar{x}_{T-1}^i\| \geq \frac{\gamma_{T-1}}{\Lambda_{T-1}}\, \mathtt{Gap}^{k-1} \quad \text{for all} i < k.$$

In what follows we distinguish two cases, regarding infinitely and finitely many level iterates.

Suppose that the sequence of level iterates at some scenario node $(T-1, j)$ is infinite, i.e., $|\mathcal{L}_{T-1,j}| = \infty$, then compactness of the feasible set (see also Remark 3 below) ensures that there exists a convergent subsequence $\{\bar{x}_{T-1}^{k'}\}_{k'}$. By plugging this subsequence into the above inequality (with $k = k'$ and $i = k'-1$) and passing to the limit as $k' \to \infty$ we conclude that $\mathtt{Gap}^k \to 0$, proving convergence of the algorithm in this case (because $\{\mathtt{Gap}^k\}_k$ is monotone and $\mathtt{Gap}^k$ is the global optimality gap at iteration k).

Suppose now that $|\mathcal{L}_{T-1,j}|$ is finite for all scenario nodes $(T-1, j)$. Then for all k large enough the algorithm will generate only normal iterates, i.e., $\bar{x}_{T-1}^k = \bar{x}_{T-1}^k(\xi_{[T-1]}^j)$ also solves (8) (and minimizes the stability function φ_t as well). Since the feasible set is compact and cut coefficients β_T and α_T are constructed via basic optimal solutions, which are finitely many, we conclude that after finitely many steps $\bar{k}$, model $\check{\mathcal{Q}}_T(\cdot)$ will remain fixed (only repeated cuts will be generated). Hence, for all $k \geq \bar{k}$ the optimal value of subproblem (8) will coincide with the optimal value of (2) at stage $T-1$,

$$\begin{aligned}\underline{Q}_{T-1}(\bar{x}_{T-2}, \xi_{T-1}) &= \begin{cases} \min\limits_{x_{T-1} \geq 0} & c_{T-1}^\top x_{T-1} + \check{\mathcal{Q}}_T(x_{T-1}) \\ \text{s.t.} & A_{T-1}x_{T-1} = b_{T-1} - B_{T-1}\bar{x}_{T-2} \end{cases} \\ &= \begin{cases} \min\limits_{x_{T-1} \geq 0} & c_{T-1}^\top x_{T-1} + \mathcal{Q}_T(x_{T-1}) \\ \text{s.t.} & A_{T-1}x_{T-1} = b_{T-1} - B_{T-1}\bar{x}_{T-2}\,, \end{cases}\end{aligned}$$

because otherwise a new cut improving model $\check{\mathcal{Q}}_T(\cdot)$ would be generated. Then, $\underline{Q}_{T-1}(\bar{x}_{T-2}^k, \xi_{T-1}) = c_{T-1}^\top \bar{x}_{T-1}^k + \mathcal{Q}_T(\bar{x}_{T-1}^k)$ for all given $\bar{x}_{T-2}^k$ with $k \geq \bar{k}$ and, moreover,

$$\begin{aligned}&\mathbb{E}_{|\xi_{[T-2]}^j}[\underline{Q}_{T-1}(\bar{x}_{T-2}^k, \xi_{T-1})] = \mathcal{Q}_{T-1}(\bar{x}_{T-2}^k) \\ &= \mathbb{E}_{|\xi_{[T-2]}^j}\left[c_{T-1}^\top \bar{x}_{T-1}^k + \mathbb{E}_{|\xi_{[T-1]}}[c_T^\top \bar{x}_T^k]\right] \quad \forall\, k \geq \bar{k}\,, \end{aligned} \tag{25}$$

where the last equality is due to the fact that $\check{\mathcal{Q}}_T(x_{T-1}^k) = \mathcal{Q}_T(x_{T-1}^k)$ for $k \geq \bar{k}$. Notice that (25) satisfies the first assumption of Lemma 2 for $t = T-2$. We now proceed to show that the second assumption also holds. Indeed, definition (23) and identity (25) (with k replaced by $k' \geq \bar{k}$) provide

$$\begin{aligned}\bar{z}^{k-1}_{\min,T-2,j} &\leq c_{T-2}^\top x_{T-2}^{k'} + \mathbb{E}_{|\xi^j_{[T-2]}}\left[c_{T-1}^\top \bar{x}_{T-1}^{k'} + \mathbb{E}_{|\xi_{[T-1]}}[c_T^\top \bar{x}_T^{k'}]\right] \\ &= c_{T-2}^\top x_{T-2}^{k'} + \mathcal{Q}_{T-1}(\bar{x}_{T-2}^{k'}) \quad \forall\, k \geq k' \geq \bar{k}\,.\end{aligned}$$

Therefore, the assumptions of Lemma 2 are both satisfied. We can thus repeat the same reasoning for $t = T-2$ and conclude that:

- either $\mathcal{L}_{T-2,j}$ is infinite for at least one node $(T-2, j)$, but then $\texttt{Gap}^k \to 0$;
- or $\mathcal{L}_{T-2,j}$ is finite for all nodes $(T-2, j)$, and therefore after finitely many steps of normal iterates, $\check{\mathcal{Q}}_{T-2}(\cdot)$ coincides with $\mathcal{Q}_{T-2}(\cdot)$ after finitely many steps.

By continuing recursively down to $t = 1$ we conclude that either $\mathcal{L}_{1,j}$ is infinite and thus $\texttt{Gap}^k \to 0$ or $\check{\mathcal{Q}}_2(\cdot)$ coincides with $\mathcal{Q}_2(\cdot)$ after finitely many steps. In the latter case, when solving (16) for $t = 1$ its optimal value will be an upper bound on the optimal value of (1). Moreover, such a value is by definition $\underline{z}^k$. Hence, $\texttt{Gap}^k = 0$ and the corresponding $\bar{x}_1^k$ is a first-stage optimal solution of (1). This concludes the proof. □

Remark 3 (*On compactness*) In proposition 2 one may wonder how to interpret "compactness of the feasible sets". Does this mean, at stage t, that the feasible solutions x_t all belong to a compact set independent of the current node and solution at the previous stage, or simply that for the latter items fixed the feasible set (now thus dependent on the "scenario" and solution at the previous stage) is compact? Let us argue that the second option implies the first. Indeed observe that at stage t for a fixed scenario and fixed decision x_{t-1}, the set of feasible solutions at stage t belongs to $\{y \;:\; A_t y = b_t - B_t x_{t-1}\}$. For a fixed scenario we may thus study the set-valued map: $x' \mapsto M_t(x') := \{y \;:\; A_t y = b_t - B_t x'\}$. This is a special case of the study of polyhedral valued set-valued maps under modifications in the right-hand side, i.e., $b \mapsto \{y \;:\; Ay \leq b\}$.

First, as a result of relatively complete recourse at any x', $M_t(x') \neq \emptyset$ and thus as a result of [2, Theorem 3.4.1], M_t is Hausdorff-continuous. Moreover, at a fixed $\bar{x}'$, by assumption, $M_t(\bar{x}')$ is compact. For some given $\varepsilon > 0$, we may thus identify a compact set $K(\bar{x}')$, such that $M_t(\bar{x}') + \varepsilon\mathbb{B} \subseteq K(\bar{x}')$. Hence by Hausdorff upper semi-continuity, we may find a neighbourhood U of $\bar{x}'$ such that $M_t(x') \subseteq M_t(\bar{x}') + \varepsilon\mathbb{B} \subseteq K(\bar{x}')$ for all $x' \in U$. Here $\mathbb{B}$ denotes the unit ball in an appropriate space.

Next observe that at stage $t = 1$ the set of all feasible solutions X_1 is independent of the scenario (and of any past decisions) and is moreover compact. Hence for any possible fixed scenario at $t = 2$ and for any possible $x_1 \in X_1$ we may find a compact set $K(x_1)$ and neighbourhood $U(x_1)$ of x_1 such that $M_2(x_1') \subseteq K(x_1)$ for all $x_1' \in U(x_1)$. We may extract a finite subcover of $U(x_1)$ to cover X_1, say $U(x_1^1), \ldots, U(x_1^p)$ and define the compact set $K_2 := \bigcup_{i=1}^p K(x_1^i)$. Then for all $x_1 \in X_1$ it holds that $M_2(x_1) \subseteq K_2$. Since the total number of scenarios at stage 2 is finite, we may take a further union over this finite number. Next we may proceed by induction to conclude the argument.

4.4 The normal-level SDDP algorithm

This section incorporates the proposed regularization approach into SDDP, which is applicable when the stochastic process $\{\xi_t\}_{t=1}^T$ is stagewise independent. The main difference between ND and SDDP is that the latter employs sampling in the forward step: instead of considering all N scenarios, the algorithm samples a subset $\mathcal{J}$ with $|\mathcal{J}| \ll N$ and solves subproblems (8) ($t = 1, \ldots, T-1$) only for these scenarios. In this case, $\tilde{z}^k = \frac{1}{|\mathcal{J}|}\sum_{r\in\mathcal{J}}[\sum_{t=1}^T c_t^\top \bar{x}_t^k(\xi_{[t]}^r)]$ is an estimated upper bound for the optimal value of problem (1). We can thus define an estimate for the optimality gap (observed at iteration k) by

$$\widetilde{\texttt{Gap}}^k := \max\{0\,,\, \tilde{z}^k - \underline{z}^k\}, \quad \forall\, k = 1, 2, \ldots,$$

where $\underline{z}^k$ is given in (7). Notice that the difference $\tilde{z}^k - \underline{z}^k$ can be negative in some iterations, which does not necessarily mean that problem (1) is solved (see the discussion in [47]).

Algorithm 2 Normal-level regularized SDDP algorithm

Step 0: initialization Same as Step 0 of Algorithm 1.
Step 1: forward Randomly draw (with replacement) a sample $\mathcal{J}^k$ with $1 \le |\mathcal{J}^k| \le N$ scenarios. For each $t = 1, \ldots, T-1$ and each $\xi^{k,j} \in \mathcal{J}^k$, get $(c_t, B_t, A_t, b_t) = \xi_t^{k,j}$, choose level parameter $\ell_{t,j}^k$ according to some rule, and solve subproblem (16) (with $\ell_t = \ell_{t,j}^k$) to obtain $\bar{x}_t^k := \bar{x}_t(\xi_{[t]}^{k,j})$. Solve (4) for all $|\mathcal{J}^k|$ scenarios and compute an upper bound estimate $\tilde{z}^k$ for (1).
Step 2: backward Same as Step 2 of Algorithm 1, evaluate the polices generated in the forward step.
Step 3: loop Define $\widetilde{\texttt{Gap}}^k = \max\{0,\ \tilde{z}^k - \underline{z}^k\}$, set $k \leftarrow k+1$ and go back to Step 1.

Algorithm 2 is similar to standard SDDP, with the difference being that the forward step above solves subproblems (16) rather than (8). A stopping test for the algorithm should be placed in Step 3 of Algorithm 2. We skip this matter since any stopping-test rule used in SDDP can be employed here as well. We refer the interested reader to [27,37,47].

4.4.1 A heuristic for setting level parameters

Under the stagewise independence assumption, every node (t, j) of the scenario tree is not only shared by the descendant scenarios of node (t, j) in the scenario tree, but also by scenarios that do not share the same history $\xi_{[t]}^j$. For instance, the gray node in the third stage of the tree represented in Fig. 1 is traversed by 12 scenarios, instead of only 4 scenarios having the same history up to time $t = 3$. For this reason we employ the following alternative to the minimum upper bound rule given by (23), when estimating an upper bound for each node:

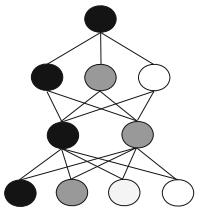

Fig. 1 Stagewise independent scenario tree (lattice) with 4 stages and 24 scenarios. Notice the black node in stage 3 composes 12 scenarios. Whenever one of these 12 scenarios is chosen in the forward step, such node will be visited and therefore its upper bound estimate $\tilde{z}_t(\xi_t)$ might be updated

$$\tilde{z}^{k-1}_{\min,t,j} := \min_{k' \le k-1} \left\{ \min_{\xi \in \mathcal{J}^{k'}} \texttt{cost}^{k'}_{|\xi_{[t]}} \ \text{s.t.}(t,j) \text{ is a node traversed by sample path (scenario) } \xi \right\}, \tag{26a}$$

where

$$\texttt{cost}^{k'}_{|\xi_{[t]}} := c_t^\top \bar{x}_t^{k'}(\xi_{[t]}) + \mathbb{E}\left[\sum_{\tau=t+1}^{T} c_\tau^\top \bar{x}_\tau^{k'}(\xi_{[\tau]}) \right]. \tag{26b}$$

The inner minimum above is taken with respect to all drawn scenarios that share tree node (t, j), and the expected value is taken with respect to all scenarios in $\mathcal{J}^{k'}$ that traverse node (t, j) and have the same history $\xi_{[t]}$. In other words, the value $\tilde{z}^{k-1}_{\min,t,j}$ is an estimation of present (at stage t) plus expected future costs of the "cheapest" scenario history $\xi_{[t]}$ containing node (t, j). Although this value might not be a valid upper bound for $c_t^\top x_t + Q_{t+1}(x_t)$ defined at the node (t, j) of an "expensive" scenario history, it might still be useful in guiding the algorithm (through level parameter ℓ) to search for trial points yielding lower costs. We illustrate the definitions in (26) in the following example.

Example 1 Consider Fig. 1 and denote black nodes by (B), gray nodes by (G), light gray by (LG) and white nodes by (W). Suppose that at iteration k' the following 5 scenarios were considered in the forward step: $\mathcal{J}^i = \{\xi^a, \xi^b, \xi^c, \xi^d, \xi^e\}$, with $\xi^a = (\xi_1^B\ \xi_2^B\ \xi_3^B\ \xi_4^B)$, $\xi^b = (\xi_1^B\ \xi_2^B\ \xi_3^B\ \xi_4^{LG})$, $\xi^c = (\xi_1^B\ \xi_2^B\ \xi_3^B\ \xi_4^W)$, $\xi^d = (\xi_1^B\ \xi_2^W\ \xi_3^B\ \xi_4^G)$, and $\xi^e = (\xi_1^B\ \xi_2^W\ \xi_3^B\ \xi_4^W)$. Let's use the notation $\bar{x}_t(\xi^d_{[t]})$ to represent the decision made at stage t of scenario ξ^d after the forward step.[1] In order to compute $\tilde{z}^k_3(\xi_3^B)$, we first compute the present and expected future cost with respect to the first 3 scenarios sharing the same history $\xi_{[3]} = (\xi_1^B\ \xi_2^B\ \xi_3^B)$:

$$\texttt{cost}^{k'}_{|\xi_{[3]}} = \texttt{cost}^{k'}_{|(\xi_1^B\ \xi_2^B\ \xi_3^B)} = c_3^\top \bar{x}_3(\xi^a_{[3]}) + [c_4^\top \bar{x}_4(\xi^a_{[4]}) + c_4^\top \bar{x}_4(\xi^b_{[4]}) + c_4^\top \bar{x}_4(\xi^c_{[4]})]/3\,.$$

Then we compute the present and expected future cost with respect to the remaining scenarios sharing the history $\xi_{[3]} = (\xi_1^B\ \xi_2^W\ \xi_3^B)$: $\texttt{cost}^{k'}_{|\xi_{[3]}} = \texttt{cost}^{i}_{|(\xi_1^B\ \xi_2^W\ \xi_3^B)} =$

[1] Note that $\bar{x}_3(\xi^a_{[3]}) = \bar{x}_3(\xi^b_{[3]}) = \bar{x}_3(\xi^c_{[3]})$ and $\bar{x}_3(\xi^d_{[3]}) = \bar{x}_3(\xi^e_{[3]})$.

$c_3^\top \bar{x}_3(\xi^d_{[3]}) + [c_4^\top \bar{x}_4(\xi^d_{[4]}) + c_4^\top \bar{x}_4(\xi^e_{[4]})]/2$. The inner minimization problem in (26a) is thus:

$$\min_{\xi \in \mathcal{J}^{k'}} \left\{ \texttt{cost}^{k'}_{|\xi_{[t]}} \text{ s.t. } \xi_t^j \text{ be a node of} \xi \right\} \quad = \quad \min\{\texttt{cost}^{k'}_{|(\xi_1^B\, \xi_2^B\, \xi_3^B)},\ \texttt{cost}^{k'}_{|(\xi_1^B\, \xi_2^W\, \xi_3^B)}\},$$

estimating the cost of the cheapest scenario history up to node ξ_3^B. (The other visited nodes ξ_2^B, ξ_2^W and ξ_1^B are updated accordingly.) We then take the minimum of the value above over all $k' \leq k$ to define $\tilde{z}_3^k(\xi_3^B)$. □

Given this estimated bound on node (t, j), we can define the level parameter for every stage, iteration, and node of the scenario tree by

$$\tilde{\ell}^k_{t,j} := \tilde{z}^{k-1}_{\min,t,j} - \gamma_t \widetilde{\texttt{Gap}}^{k-1}, \quad \text{with } \gamma_t > 0 \text{ a given parameter.} \tag{27}$$

As argued above, the value $\tilde{z}^{k-1}_{\min,t,j}$ by itself can be an underestimation of the present and future costs. Therefore, a small value for γ_t is advisable (in our numerical experiments we take $\gamma_t = \kappa/t$, with $\kappa = 0.5$). When the upper bound estimation obtained from the forward step is not reliable, e.g., when only a single sample path is employed in the forward step at each iteration, we perform an upper bound estimation step periodically by employing a larger number of samples using our most updated approximate cost-to-go functions. Other rules that require a reasonable estimation on the upper bound, such as the discount rule (20), can also be used in this setting.

4.4.2 Convergence analysis

Again, if one occasionally picks the level parameter sufficiently low so that only normal iterates are produced in these iterations, the convergence analysis of Algorithm 2 follows from the analysis of standard SDDP. In order to resort to the convergence results of [47], we assume that at every forward step Algorithm 2 arbitrarily selects a scenario $\xi^{j^*} \in \mathcal{J}^k$ for which the level parameter is set to $-\infty$. In this manner, all iterates $\bar{x}_t(\xi^{j^*}_{[t]})$ issued by (16) (with scenario ξ^{j^*}) will be of the normal type, and the analysis of standard SDDP applies.

Proposition 3 *Consider Algorithm 2 and assume that the problem satisfies relatively complete recourse. Assume moreover, that the feasible sets are compact and* $\varphi_t : \mathbb{R}^{n_t} \to \mathbb{R}$ *are convex functions, for all* t. *Finally, assume that*

(i) *in every forward step* k, *a scenario* $\xi^{j^*} \in \mathcal{J}^k$ *is randomly chosen and the level parameters are set as* $\ell^k_{t,j^*} = -\infty$ *for all* $t = 1, 2, \ldots$ *on sample path (scenario)* $\xi_t^{j^*}$.

(ii) *in the backward steps basic optimal solutions are employed.*

Then the algorithm asymptotically computes, with probability one, an optimal policy for problem (1).

Proof It follows from the assumptions on the feasible set, regularizing function φ, and hypothesis (i) that all iterates $\bar{x}_t^k(\xi_{[t]}^{j^*})$ issued by scenario ξ^{j^*} and subproblem (16) are of the normal type. We recall that normal iterates also solve (8), the subproblem handled by SDDP in forward step. As result, at least for such a randomly selected scenario ξ^{j^*}, Algorithm 2 behaves like[2] a variant of SDDP that selects only one scenario in the forward step. Convergence analysis of such variant follows from assumption (ii) and [47, Proposition 3.1]. Hence, Algorithm 2 defines with probability one an optimal policy for problem (1). □

4.5 General comments on the proposed algorithms

Many variants of Algorithms 1 and 2 are possible, since there is considerable freedom in choosing the functions φ_t (these functions are only required to be convex). A possible choice is $\varphi_t(x_t) := \|x_t - \hat{x}_t\|$, where $\|\cdot\|$ is a given norm (for instance the Euclidean one) and $\hat{x}_t \in \mathbb{R}^{n_t}$ a given stability center, not necessary belonging to the feasible set. One could take $\hat{x}_t = 0$, $\hat{x}_t = \bar{x}_t^{k-1}(\xi_{[t]})$ (the trial point used in the previous iteration), $\hat{x}_t = \frac{1}{|\mathcal{J}^k|}\sum_{r\in\mathcal{J}^k}[\bar{x}_t^{k-1}(\xi_{[t]}^r)]$ (the average of all trial points used in the previous iteration), etc. We recall that: if $\varphi_t(x_t) = x_t^\top x_t$, then the QP solver of [30] is an appropriate choice for solving (16) directly without resorting to the perturbed problem (17); if $\varphi_t(x_t) = \|x_t - \hat{x}_t\|_1$, then subproblem (17) is an LP.

We highlight that even if the total number of level iterates is finite, the resulting algorithms would still differ from the standard ND/SDDP because (8) can have multiple solutions and $\bar{x}_t$ from (16) is the normal one (when $\varphi_t(x_t) = x_t^\top x_t$). A situation wherein problem (8) has multiple solutions can be found, for instance, in hydro-thermal planning problems of a predominantly hydraulic power system, whose hydropower plants can be planned in several ways to provide the same amount of power with the same (present) cost. Many problems arising in the energy application exhibit this "symmetry" structure.

Finally we would like to highlight that the level bundle method's strength is achieved when level parameters estimate the optimal values of the nonlinear subproblems (2) (with a given optimal policy $\bar{x}_{t-1} = x_{t-1}^*$). Notice, however, that if the level parameter is set to be too low, then more normal iterates are likely to be performed by the algorithms. Although such iterates regularize the forward step in the case of multiple solutions, such regularization may not be as effective as the one provided by level iterates. This is observed in our numerical experiments, which we present in the next section.

5 Numerical experiments

We conduct numerical experiments to test the performance of the proposed normal-level ND and normal-level SDDP algorithms for MSLPs. We first present in Sect. 5.1 a set of instances that are used in our experiments for both the normal-level SDDP

[2] Although the regularization effect still presents if (8) has multiple solutions.

algorithm and normal-level ND algorithm. In Sect. 5.2, we focus on the normal-level SDDP algorithm proposed in Sect. 4.4, and compare it with standard SDDP as well as the multistage regularized decomposition of [1]. In Sect. 5.3, we focus on the normal-level ND algorithm proposed in Sect. 4.3, and compare it with standard ND algorithm. We implemented all algorithms in C++ using commercial solver CPLEX, version 12.5.1 to solve the subproblems. All tests are conducted on an iMac desktop with four 4.00GHz processors and 16Gb memory. The number of threads is set to be one.

Although not exhaustive, the numerical experiments illustrate well the effectiveness and the potential interest of the normal-level ND and SDDP techniques. A thorough numerical assessment of the interests and limits of the algorithms would deserve a whole study of itself to take into account various rules for setting level parameters, stability centers and stability functions. Here we consider three basic implementation of the algorithms in a serial setting and four test problems. Extensive numerical experiments and more effective implementations of the algorithms in a parallel setting (investigating different sequencing protocols as suggested in [54]) are beyond the scope of this paper.

5.1 Test problem description

We consider a multistage hydro-thermal power generation planning problem in our numerical experiments. The problem instance is provided by E. Finardi and F. Beltrán, and models the Brazilian hydro-thermal power system. The objective of the model is to minimize the (expected) total cost over a certain number of time stages, including the power generation cost and the penalty of insufficient power to satisfy the demand, under the uncertainty on the amount of rainfall in the future. Power can be generated by a set H of hydro power plants ($|H| = 30$) and a set F of thermal plants ($|F| = 38$) that are interconnected with each other (see the hydro plant network structure in the Appendix). Among the 30 hydro power plants, 16 of them have reservoirs (denoted by H_R) so that we could control the state of reservoir level, while the other 14 of them are the so-called "run-of-river" plants (denoted by H_I), which do not have a reservoir. Hydro power generation has no cost, but there are upper and lower limits (denoted by $\bar{v}_h$ and $\underline{v}_h$, respectively) on the water level in each reservoir $h \in H_R$. The inflow of water in each reservoir $\tilde{b}_h^t$ is random, and a finite set of scenarios for each time stage (monthly by default) in the planning horizon is available from prediction. In each stage, we need to make decisions for each hydro power plant $h \in H$ on the amount of water used for hydro power generation (denoted by h_h^t), the amount of water to spill in order to keep the water level of the reservoir within the limits (denoted by s_h^t), and the amount of power generated by each thermal plant $f \in F$ (denoted by g_f^t). The original data set contains monthly data for demand, power generation capacity for each plant, and generation cost for the thermal plants, over a planning horizon of a total of 120 months. It is assumed that the water inflows follow an autoregressive stochastic process of the form $\tilde{b}_h^{t+1} = \Phi_t^h \tilde{b}_h^t + \epsilon_{t+1}^h$, where the white noise ϵ_{t+1}^h is stagewise independent. The problem thus presents a dependence between stages, but can be made stagewise independent by applying the well-known trick recalled in [47,

Table 1 Notation for the stage-t problem of the multistage hydro-thermal power generation planning problem

Notation	Description
c_f	Unit cost of thermal power generated from plant f
c_p	Unit penalty cost for unsatisfied demand
$\tilde{b}_h^t$	Random inflows to hydro plant h during stage t
r_h	Amount of power generated by releasing one unit of water flow in hydro plant h
d^t	Demand in stage t
$\bar{y}_h^t$	Maximum allowed amount of turbined flow in hydro plant h in stage t
$\underline{g}_f^t, \bar{g}_f^t$	Minimum/maximum allowed amount of power generated by thermal plant f in stage t
$\underline{v}_h, \bar{v}_h$	Minimum/maximum level of water allowed in hydro plant h
$U(h)$	Set of immediate upper stream hydro plants of h in the network
x_h^t	Water level of hydro plant h in stage t
y_h^t	Amount of turbined water flow released by plant h in stage t
s_h^t	Amount of spilled water (without generating power) by plant h in stage t
g_f^t	Amount of thermal power generated by plant f in stage t
p^t	Amount of unsatisfied demand in stage t

Eq. 1.2]: enlarge the decision variables with $\tilde{b}_h^t$ (i.e., $(x_t, \tilde{b}_h^t)$), set $\xi_t = \epsilon_t$ and enlarge the constraint matrices with Φ_t^h.

With notation as in Table 1, let $\mathcal{Q}_{t+1}(x^t)$ be the expected cost-to-go function at stage t. Then the problem at stage t can be written as:

$$
\begin{aligned}
\min_{g_f^t, p^t, x_h^t, y_h^t, s_h^t} \ & \sum_{f \in F} c_f g_f^t + c_p p^t + \mathcal{Q}_{t+1}(x^t) \\
\text{s.t. } & x_h^t = x_h^{t-1} + \tilde{b}_h^t + \left[\sum_{m \in U(h)} (y_m^t + s_m^t) \right] - (y_h^t + s_h^t), \ \forall h \in H_R \\
& y_h^t + s_h^t = \tilde{b}_h^t + \left[\sum_{m \in U(h)} (y_m^t + s_m^t) \right], \ \forall h \in H_I \\
& \sum_{h \in H} y_h^t r_h + \sum_{f \in F} g_f^t + p^t \geq d^t \\
& y_h^t \leq \bar{y}_h^t, \ \forall h \in H \\
& \underline{g}_f^t \leq g_f^t \leq \bar{g}_f^t, \ \forall f \in F \\
& \underline{v}_h \leq x_h^t \leq \bar{v}_h, \ \forall h \in H_R. \qquad (28a)
\end{aligned}
$$

We customize this data set and create a variety of instances with different number of time stages $T \in \{25, 61, 97\}$. We then sample from a given distribution of random amount of inflows for each hydro plant in each time stage with sample sizes: $|\Xi_t| \in$

$\{20, 50, 80\}$, $t = 2, \ldots, T$. For simplicity, we let the number of realizations be the same for each stage. Therefore, if the planning horizon of the problem is $T = 97$ and the sample size is 80, then the resulting scenario tree has approximately 4.9732×10^{182} scenarios. We remark that SDDP type algorithms are more appropriate for these considered problem instances.

5.2 Computational experiments on the normal-level SDDP algorithm

5.2.1 Implementation details

In our experiments, we consider the following variants of SDDP for comparison:

- The standard SDDP algorithm.
- The proposed normal-level SDDP algorithm (Algorithm 2).
- The normal-solution SDDP algorithm, which disables the level regularization of Algorithm 2 by setting level parameters to be $\ell_t = -\infty$ (so that only normal solutions of (8) are considered at each forward step).
- The multistage regularized decomposition algorithm proposed by [1].

Implementation of the standard SDDP algorithm We follow the description in Sect. 2 for the implementation of the standard SDDP algorithm. To get an initial lower bound for the cost-to-go function $\check{\mathcal{Q}}_t(\cdot)$, $t = 1, 2, \ldots, T-1$, we solve the mean value problem with respect to the $(t+1)$-th stage problem by taking the expectations of the random amount of inflows $\tilde{b}^t$ and treating $\bar{x}_t$ as decision variables. We consider two variants of this algorithm according to the number of scenarios employed in the forward pass:

- `SDDP-1`: the algorithm uses only one scenario per forward step;
- `SDDP`: the algorithm employs ten scenarios per forward step.

Implementation of the normal-level SDDP algorithm (*Algorithm 2*) The backward pass of the normal-level SDDP algorithm is identical to that of SDDP. The difference between the two algorithms is in the forward pass. In each forward pass, the trial points are obtained by solving (16). Instead of using a specialized QP algorithm proposed in [30] to solve (16), we solve the equivalent problem (17) by setting $\bar{\tau} = 10^8$ for the convenience of implementation. Exactly solving these QPs with high precision could be time consuming, and thus we set the optimality tolerance parameter to be 10^{-4}. We implement the minimum upper bound rule (27) for choosing the level parameters. We have tried other heuristic rules such as the simple discount rule (20) in our implementation, and concluded that the minimum upper bound rule yielded the best performance according to our numerical study. We set the level parameter as $\gamma_t = 0.5/t$ for $t = 2, \ldots, T-1$. As rule (27) makes use of estimated upper bounds $\tilde{z}_k$, our implementation of the normal-level SDDP considers ten scenarios per forward step: $\tilde{z}_k$ is then iteratively estimated based upon these scenarios. To compare with `SDDP-1` we have also tested a variant taking only one scenario per forward step. However, the results were not competitive in terms of CPU time: the extra computational effort to obtain reasonable estimates of upper bounds $\tilde{z}_k$ makes this variant noncompetitive with `SDDP-1`.

We follow [1] by regularizing the iterates only on the state variables (post-decision state variables). This algorithm is denoted by `normal-level` in what follows. In addition, we consider the following two variants of `normal-level`:

- `normal-level-LP`: this algorithm uses the L-1 norm for regularization, motivated by the fact that the regularized formulation with L-1 norm can be formulated as an LP, which is easier to solve and is more robust to numerical issues. Every forward step employs ten scenarios (the same ones as `SDDP`).
- `normal-level-prev`: different from `normal-level` and `normal-level-LP` that set the vector of zeros as the stability center during all the stages and the entire iterative process, this algorithm sets stability centers as the previous trial points. This is akin to [1] but with an important difference: as every forward step employs ten scenarios (therefore ten different trial points are produced per iteration), we randomly choose a stability center among the ten previous trial points.

Implementation of the normal-solution SDDP algorithm The implementation of the normal-solution SDDP algorithm is identical to that of the algorithm `normal-level`, except that we set the level parameter to $-\infty$, so that only normal iterates will be performed. This algorithm is denoted by `normal` throughout this section. Again, every forward step considers ten scenarios.

Implementation of the multistage regularized decomposition proposed by [1] We follow [1] to implement the multistage regularized decomposition algorithm: in each iteration $k > 1$, one single scenario is employed per forward step and we use the trial point produced at the previous iteration $k - 1$ as the stabilization center. Moreover, we set the regularization parameter (coefficient of the quadratic term) to be 0.95^k as suggested by [1]. This algorithm is denoted by `multi-reg` in the remainder of this section. We adopt the setting used in [1], which is to use only one single scenario per forward step. We have also investigated a variant of `multi-reg` considering ten scenarios per forward step. Similar to the algorithm `normal-level-prev` we randomly chose the stability center among the ten previous trial points. However, this variant was not found to be competitive and therefore we do not report its results.

5.2.2 Performance on lower bounds with fixed CPU time

In this section, we report the quality of lower bounds obtained by each variant of the SDDP algorithm described above given a fixed time limit of three hours (10800 seconds). To make the comparison fair, we set the number of threads to be one. Except for `SDDP`-1 and `multi-reg` algorithms that employ one single scenario per forward step, all the other investigated algorithms use ten scenarios in each forward step, so that reasonable upper bound estimates can be obtained from these scenarios for variants "normal-level". (We also performed experiments by varying the number of scenarios in the forward step, but did not observe significant differences, and therefore chose to report results using 10 as the sample size.)

Table 2 reports the lower bounds yielded by each one of the seven considered algorithms on nine different instances of the multistage hydro-thermal power generation

planning problem described in Sect. 5.1. We compare the progress on lower bounds with the ones provided by algorithm SDDP-1 after one, two and three hours of processing. The last column of Table 2 reports the number of iterations performed by every algorithm in each one of the nine problem instances. The time horizon of the considered instances are $T = 25$, $T = 61$ and $T = 97$ months, which represent 1 deterministic month (the first-stage) plus an uncertain future of respectively 2, 5 and 8 years split into months.

We care to mention that the lower bound progress is the feature of major interest when dealing with the considered application. More importantly, the main goal in solving the hydro-thermal power generation planning problem of Sect. 5.1 is to obtain a good cutting-plane approximation $\check{\mathcal{Q}}_2(\cdot)$ of the cost-to-go function $\mathcal{Q}_2$. Indeed, the cutting-plane function $\check{\mathcal{Q}}_2(\cdot)$ is given as an input to a more detailed optimization model. Such a model (with a shorter planning horizon but with hourly, daily or weekly time steps) appends $\check{\mathcal{Q}}_2(\cdot)$ in its last stage to represent "future costs" in decision making on power generation (sometimes for every turbine of all considered plants) and operation of transmission lines [36]. The importance of lower bound progress is then directly connected to the quality of the cutting-plane model $\check{\mathcal{Q}}_2(\cdot)$ by subproblem (7). This justifies our interest in a fast progress of lower bounds.

From Table 2, we see that the performance of algorithms varies depending on the time limit of processing: after one hour of processing all the algorithms (except normal) performed better (in most of the instances) than the algorithm SDDP-1 in terms of lower bounds. After three hours of processing the improvement on lower bounds yielded by the regularized algorithms are, as expected, less expressive: the reason is that the sequence of lower bounds stabilizes near the optimal value of the underlying problem. Nonetheless, the proposed algorithms normal-level and normal-level-LP still provide better bounds for all instances with more dense scenario trees, i.e., $|\Xi_t|$ equal to 50 and 80.

As normal-level-LP uses the L1-norm (applied to the post-decision state variables) as the stability function φ_t in (17), the subproblems solved in the forward steps are LPs, which are much more efficient to solve than the QPs. This explains why the algorithm normal-level-LP performs more iterations than normal-level. In fact, the total number of iteration of normal-level-LP is comparable to the number of iterations performed by SDDP. One can thus conclude that the regularization yielded by the Euclidean norm as stability function is more effective than the regularization issued by the L1 norm: the algorithm normal-level performed fewer iterations than normal-level-LP but provided as good as (sometimes better) lower bounds. From this perspective, we see that the SDDP-1 performed better than the regularized algorithms and SDDP on our test instances when the number of realizations in each stage is relatively small ($|\Xi_t| = 20$). Moreover, the SDDP algorithm and the proposed normal-level SDDP algorithms yield similar performance on these smaller instances. On the other hand, we see that the proposed normal-level SDDP algorithms are more competitive when the number of realizations per stage is larger. Indeed, in this case all the algorithms will take more time in the backward step, leading to fewer iterations within the time limit overall (see for example the instance $T = 97$ and $|\Xi_t| = 80$). Therefore, the impact of computational inefficiency on solving a QP (17) compared to solving an LP (8) is less significant on the overall computational time.

Table 2 Lower bounds obtained by various algorithms with a fixed time limit of one hour, two hours and three hours.

Instances			1 h		2 h		3 h		
Algorithm	T	$\lvert\Xi_t\rvert$	LB	%LB	LB	%LB	LB	%LB	# iter
SDDP-1	25	20	689.251	–	725.832	–	737.698	–	6547
multi-reg	25	20	690.426	0.170	700.683	− 3.649	709.969	− 3.759	2133
SDDP	25	20	697.968	1.265	718.273	− 1.097	732.345	− 0.726	592
normal-level	25	20	729.338	5.816	735.079	1.342	739.530	0.248	195
normal-level-LP	25	20	697.475	1.193	718.694	− 1.036	736.049	− 0.224	583
normal-level-prev	25	20	686.591	− 0.386	694.994	− 4.474	703.153	− 4.683	204
normal	25	20	702.572	1.933	716.211	− 1.396	728.639	− 1.228	222
SDDP-1	25	50	1229.010	–	1279.910	–	1311.130	–	3978
multi-reg	25	50	1271.930	3.492	1289.640	0.792	1302.430	− 0.664	1728
SDDP	25	50	1276.000	3.823	1333.240	4.339	1355.200	3.361	369
normal-level	25	50	1368.610	11.359	1379.580	8.110	1386.460	5.745	178
normal-level-LP	25	50	1323.100	7.656	1355.400	6.142	1373.930	4.790	360
normal-level-prev	25	50	1307.960	6.424	1319.180	3.195	1327.020	1.212	181
normal	25	50	1271.970	3.495	1297.870	1.461	1314.030	0.221	177
SDDP-1	25	80	1177.820	–	1217.920	–	1237.450	–	3131
multi-reg	25	80	1208.850	2.635	1226.550	0.733	1236.260	− 0.096	1656
SDDP	25	80	1145.960	− 2.705	1183.470	− 2.925	1205.920	− 2.548	295
normal-level	25	80	1261.630	7.116	1277.980	5.099	1287.250	4.024	170
normal-level-LP	25	80	1220.960	3.663	1253.180	2.994	1270.230	2.649	285
normal-level-prev	25	80	1209.500	2.690	1233.050	1.285	1238.970	0.123	173
normal	25	80	1191.390	1.152	1209.630	− 0.704	1219.090	− 1.484	171
SDDP-1	61	20	9292.500	–	9885.080	–	10219.400	–	3444
multi-reg	61	20	8396.890	− 9.638	8777.360	− 11.921	9042.660	− 11.515	1218

Table 2 continued

Instances			1 h		2 h		3 h		
Algorithm	T	$\lvert\Xi_t\rvert$	LB	%LB	LB	%LB	LB	%LB	# iter
SDDP	61	20	9387.320	1.020	9831.140	−0.580	10092.200	−1.245	313
normal-level	61	20	8904.890	−4.171	9223.560	−7.119	9355.760	−8.451	124
normal-level-LP	61	20	9300.700	0.088	9801.330	−0.901	10052.300	−1.635	314
normal-level-prev	61	20	8649.000	−6.925	8968.450	−9.864	9119.950	−10.758	125
normal	61	20	8387.480	−9.739	8858.340	−11.049	9045.840	−11.484	125
SDDP-1	61	50	12418.000	–	13233.000	–	13663.100	–	2154
multi-reg	61	50	12002.900	−3.343	12770.000	−3.728	13017.900	−4.722	1118
SDDP	61	50	12603.500	1.494	13393.700	1.294	13809.000	1.068	207
normal-level	61	50	12979.500	4.522	13435.000	1.627	13736.800	0.539	117
normal-level-LP	61	50	12881.100	3.729	13539.700	2.470	13885.400	1.627	207
normal-level-prev	61	50	12323.400	−0.762	12808.500	−3.418	13047.300	−4.507	118
normal	61	50	10735.600	−13.548	11775.600	−11.736	12261.900	−10.255	116
SDDP-1	61	80	9533.480	–	10550.700	–	11028.700	–	1712
multi-reg	61	80	9946.880	4.336	10297.900	−2.652	10649.600	−3.437	1064
SDDP	61	80	9750.050	2.272	10583.200	0.341	10998.300	−0.276	167
normal-level	61	80	10728.800	12.538	11325.900	8.131	11541.300	4.648	111
normal-level-LP	61	80	10577.100	10.947	11471.900	9.663	11743.300	6.479	167
normal-level-prev	61	80	10048.600	5.403	10560.300	0.101	10781.100	−2.245	112
normal	61	80	8661.090	−9.151	9557.010	−10.423	9960.510	−9.686	110
SDDP-1	97	20	17252.900	–	18613.200	–	19280.900	–	2619
multi-reg	97	20	15640.700	−9.345	16767.500	−10.698	17538.100	−9.039	1020
SDDP	97	20	16644.100	−3.529	18264.000	−2.024	19126.600	−0.800	246

Table 2 continued

Instances			1 h		2 h		3 h		
Algorithm	T	$\lvert\Xi_t\rvert$	LB	%LB	LB	%LB	LB	%LB	# iter
`normal-level`	97	20	16562.500	−4.002	17646.400	−5.604	18112.100	−6.062	106
`normal-level-LP`	97	20	17131.200	−0.705	18508.000	−0.610	19361.200	0.416	244
`normal-level-prev`	97	20	15637.500	−9.363	16835.200	−10.306	17269.800	−10.431	105
`normal`	97	20	13756.600	−20.265	15187.000	−19.859	15735.700	−18.387	105
`SDDP-1`	97	50	21052.500	–	23979.800	–	25274.400	–	1690
`multi-reg`	97	50	21783.600	3.473	23316.400	−3.151	24279.900	−3.935	947
`SDDP`	97	50	21756.000	3.342	24395.600	1.975	25261.100	−0.053	163
`normal-level`	97	50	22401.000	6.405	24706.300	3.451	25695.000	1.664	99
`normal-level-LP`	97	50	22458.500	6.679	24799.300	3.893	25761.300	1.926	164
`normal-level-prev`	97	50	23333.100	10.833	24763.300	3.722	25339.600	0.258	99
`normal`	97	50	19892.800	−5.509	22330.000	−7.837	22990.300	−9.037	98
`SDDP-1`	97	80	14674.700	–	17527.400	–	19485.400	–	1365
`multi-reg`	97	80	16991.600	15.788	18176.800	4.425	18629.900	−4.390	884
`SDDP`	97	80	15466.700	5.397	18140.500	4.178	19271.000	−1.100	132
`normal-level`	97	80	16533.100	12.664	19165.100	11.160	20024.600	2.767	94
`normal-level-LP`	97	80	16168.700	10.181	19205.100	11.433	20631.300	5.881	133
`normal-level-prev`	97	80	16793.100	14.436	18752.100	8.346	19459.100	−0.135	94
`normal`	97	80	14571.500	−0.703	16702.300	−5.623	17632.700	−9.508	92

We take `SDDP-1` as a basis of comparison. % LB stands for $100\left(\frac{LB-LB_{\texttt{SDDP-1}}}{LB_{\texttt{SDDP-1}}}\right)$. Therefore, a positive (negative) value means that the examined algorithm performed better (worse) than SDDP-1 in terms of the cost-to-go function approximation (lower bound)

Comparing `normal-level`, `normal-level-LP`, `normal-level-prev` and `multi-reg` across all cases, we conclude that algorithms `normal-level` and `normal-level-LP` perform better on the test instances (with `normal-level-LP` providing the best performance on the large instances).

Regarding the algorithm `multi-reg`, we notice that its improvements on lower bounds are significant within the first minutes of processing (see Figs. 2, 3 and 4 below) but becomes less evident after a few hours of processing: at the end of three hours of processing the lower bounds yielded by `multi-reg` are poorer than the ones issued by the unregularized solvers `SDDP`-1 and `SDDP`. An explanation for such a performance is the fact that `multi-reg` eventually dismisses regularization: as already argued, the algorithm `multi-reg` eventually computes normal solutions of (8), which for the considered problem do not yield an effective regularization. To see this, note from Table 2 that the algorithm `normal` was often less competitive than the unregularized algorithm `SDDP`-1. Such a poor performance of `normal` then explains why `multi-reg` was not competitive at the end of three hours of processing (in contrast to its good performance in the first hour).

We care to mention that algorithm `normal-level-prev` was not competitive at the end of three hours of processing. This shows that the naive strategy of randomly choosing stability centers out of ten previous trial points is inappropriate for the considered test problems. Further rules to update stability centers in Algorithm 2 should be investigated in a future computational study.

Figure 2 presents the lower bound progress yielded by the seven considered algorithms on the instance of the problem with $T = 25$ and $|\Xi_t| = 20$. The top-left image presents the solution progress "CPU time (s) $\times$ LB" in the first 20 minutes of processing whereas the top-right image displays the whole iterative process of three hours. The same data is presented again in the bottom-left image but with the abscissa in the logarithmic scale. The abscissa of bottom-right image is the logarithm of iterations, and not CPU time. We can see from Fig. 2 that `multi-reg` was the most effective algorithm up to nearly 200 seconds of processing, and then it was overtaken by `normal-level` up to the end of the three hours of processing. Moreover, the lower bound 1311.130 computed by `SDDP`-1 in three hours was obtained by the algorithm `normal-level` in only eleven minutes. As the figure shows, `multi-reg` was not able to reach such a bound (see also Table 2).

The results for instance with $T = 61$ stages ($1 + 5 \times 12$ months) and $|\Xi_t| = 80$ are illustrated in Fig. 3. Once again, algorithm `multi-reg` starts very well but then is overtaken by the normal-level algorithms. As in theory `multi-reg` eventually becomes algorithm `normal`, the bad performance (on this test problem) of the latter explains the limiting behaviour of `multi-reg`.

We repeat the same analysis for the instance of the hydro-thermal power generation planning problem with $T = 97$ stages ($1 + 8 \times 12$ months) and $|\Xi_t| = 80$. Comparing among options `normal-level`, `normal-level-LP` and `normal-level-prev`, we see that option `normal-level-LP` finishes a much larger number of iterations within the time limit, due to the fact that LPs as opposed to QPs are solved in the forward pass. The performances of the first two algorithms are more competitive than option `normal-level-prev`.

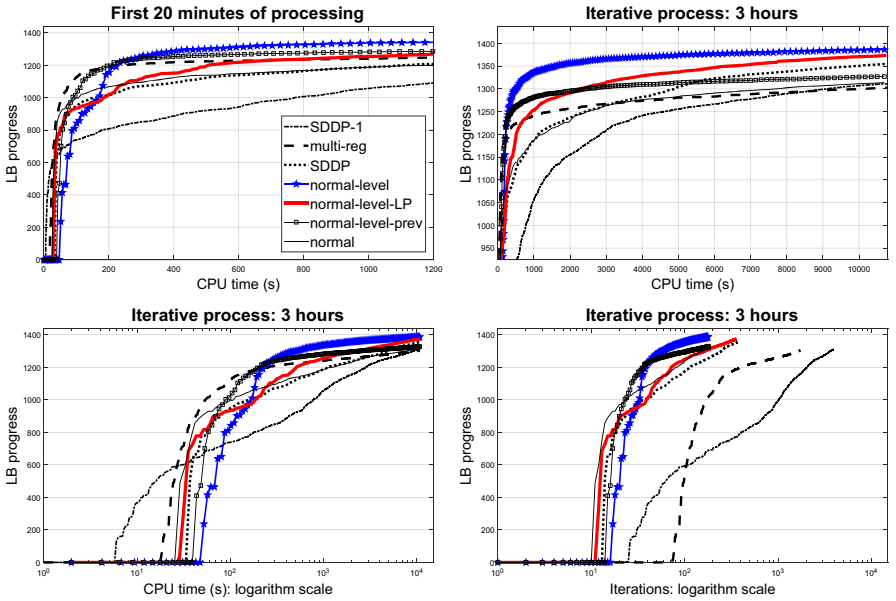

Fig. 2 Solution progress for instance $T = 25$ stages and $|\Xi_t| = 50$ scenarios per stage

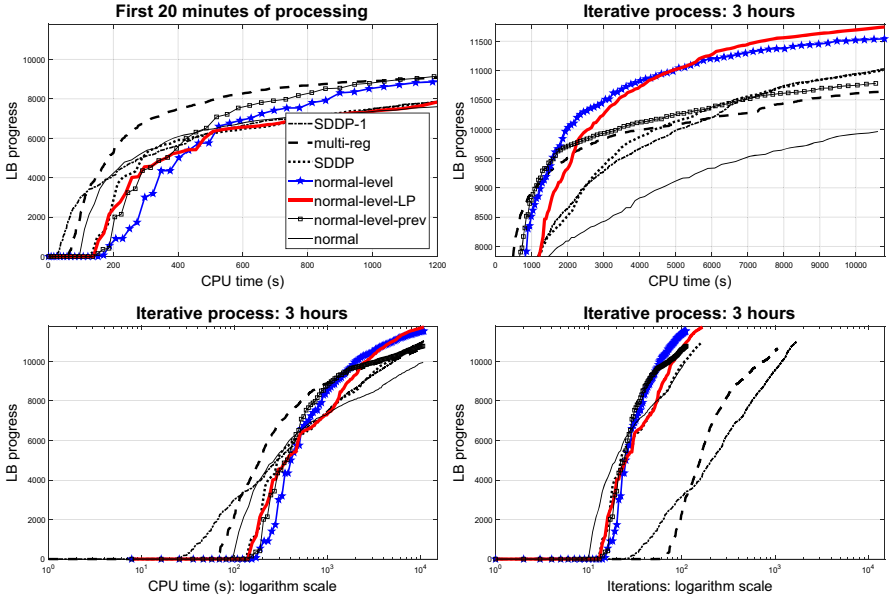

Fig. 3 Solution progress for instance $T = 61$ stages and $|\Xi_t| = 80$ scenarios per stage

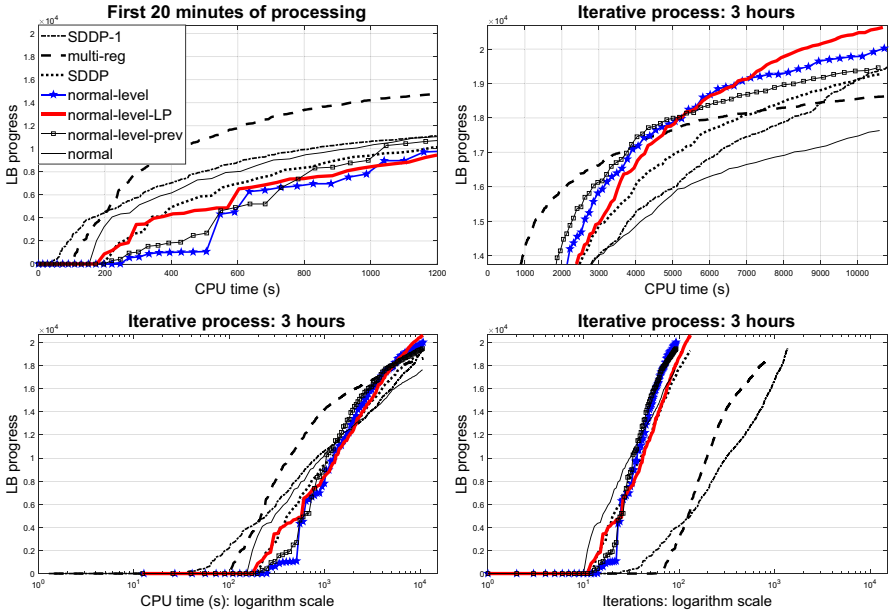

Fig. 4 Solution progress for instance $T = 97$ stages and $|\Xi_t| = 80$ scenarios per stage

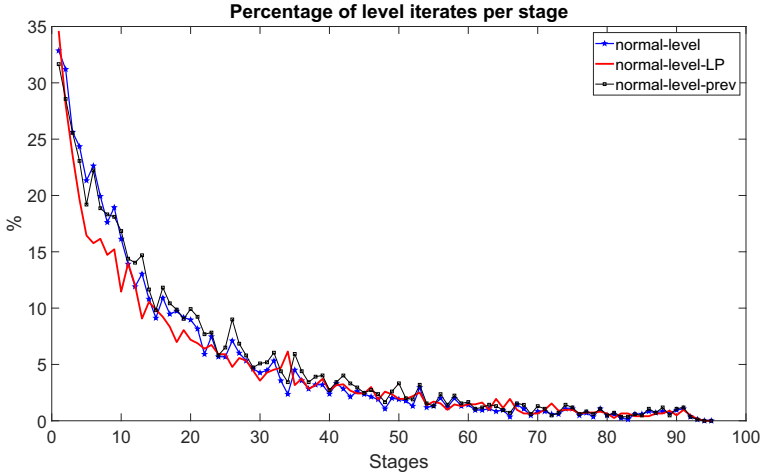

Fig. 5 Instance $T = 97$ stages and $|\Xi_t| = 80$ scenarios per stage. Percentage of level iterates per stage

Finally, Figure 5 presents the percentage of level iterates performed by these three solvers in each stage. Since each iterate generated by our approach is either a level or a normal iterate, the percentage of normal iterates is complementary to the data of Fig. 5. Notice that although the percentage of level iterates is at least 15% in the first ten stages, this percentage decreases significantly at the last stages (more normal

iterates are generated in the last few stages). From this observation we can conclude the following:

- rule (27) is very often unable to estimate useful level parameters at the last few stages;
- the level regularization is compelling because the iterative process has been improved (with respect to the unregularized SDDP algorithm) even with a reasonably small percentage of level iterates. We recall that although the normal iterates are the majority in the iterative process of algorithms `normal-level`, `normal-level-LP` and `normal-level-prev`, the algorithm `normal` that performs only normal iterates presented a poorer performance (see for instance Fig. 4).

All in all, these results indicate that the proposed level-like regularization can be very effective provided an appropriate heuristic for determining the level parameters is employed: even the simple rule (27), which presented its weakness in the last stages (c.f. Fig. 5), could significantly improve the numerical performance of the SDDP-like algorithms.

5.3 Computational experiments on the normal-level ND algorithm

Our implementations of the standard ND algorithm and normal-level ND algorithm are similar to their SDDP counterparts, respectively. The key difference is that the cost-to-go functions $\mathcal{Q}_t(\cdot)$ are defined not only for each non-terminal stage t, but for each non-leaf node of the scenario tree (although the same notation was used in Sect. 4.3 for notational convenience). The standard ND implementation follows the description in Sect. 2. The normal-level ND implementation follows Algorithm 1, using the minimum upper bound rule for setting level parameters. The normal-solution ND implementation is identical to that of normal-level ND except that the level parameter is set to $-\infty$ always.

The test instances for our ND algorithm come from three different sources: (i) instances directly available from the literature [54]; (ii) instances constructed by modifying those in the literature; and (iii) instances constructed from the hydro-thermal power generation planning problem described above in Sect. 5.1. For (ii), we take the pltexpA instances with 3 and 4 stages, but generate more scenarios than the ones provided by [26], according to their scenario data. For (iii), we consider instances with 4 and 5 stages (these stages are taken from the 1st, 7th, 13th, 19th, and 25th stage from the full data, due to the cyclic rainfall pattern), and use different samples from the full set of scenarios with various sizes.

We see from Table 3 that, `normal-level ND` works better in terms of both computational time and number of iterations performed by the algorithm than `standard ND` on the set of hydropower planning instances. However, this is not the case for instances pltexpA and sgpf, where the help from regularization brought by the `normal-level ND` is less significant, resulting in more computational time in many cases. In spite of the somewhat mixed results, the `normal-level ND` tends to be more effective on instances that require more iterations to converge. This is in accordance to what is known in the literature for the two-stage setting: see for instance

Table 3 Nested decomposition: a comparison of with two regularization strategies

Instances	T	$\lvert\Xi_t\rvert$	`standard ND`		`normal-level ND`		`normal ND`	
			Time (s)	# iter	Time (s)	# iter	Time (s)	# iter
Hydro	4	10	2984.7	126	899.3	43	1414.2	69
		20	7377.8	51	5173.3	42	6527.3	46
	5	5	121.4	109	287.0	59	215.4	56
		10	3784.1	138	2379.7	74	2482.9	76
pltexpA	3	100	23.0	8	24.8	8	24.3	8
		200	154.9	9	159.7	9	159.9	9
	4	30	169.2	10	242.1	14	172.1	10
		40	649.0	11	584.1	9	674.7	11
sgpf	4	5	0.1	3	0.1	3	0.1	3
	5	5	0.6	5	0.8	6	0.7	5
	6	6	4.3	8	4.6	7	4.8	8

the analysis on Benders decomposition versus level bundle method for two-stage stochastic programming presented in [53, Table 14].

6 Conclusions

In this paper, we have proposed a regularization scheme based on normal solutions and level sets to well-known decomposition methods for MSLPs. We have discussed convergence of the suggested algorithms and shown their interest in a set of instances coming from a large scale hydrothermal scheduling problem. In our implementation of Algorithm 2, we have considered the minimum upper bound rule (27) for choosing the level parameters for all the scenarios in the forward step. However, the convergence of the normal-level SDDP algorithm has only been shown for setting $\ell_t = -\infty$ for one arbitrary scenario in the forward step, but not for other rules for setting level parameters. Showing convergence without this assumption, as was done for the regularized nested decomposition, should be considered for the future.

As perspectives are concerned we can think of extending the given algorithms to deal with risk-averse MSLPs [28], as well as furnishing the backward step with an adaptive partition-based scheme akin to [50] to reduce the computational burden. Furthermore, other alternatives to define level parameters ℓ_t shall be addressed in future research, in addition to the possibility of extending the complexity analysis of level bundle methods to the proposed normal-level ND and SDDP algorithms.

From the numerical point of view, an extensive analysis comparing specialized algorithms for solving the perturbed subproblems, different stability functions and different rules for defining stability centers shall be investigated. Moreover, the given algorithms will benefit from paralellization techniques and an analysis of different sequencing protocols as investigated in [54].

Acknowledgements We would like to acknowledge the coordinating editor and two anonymous referees for their constructive suggestions that considerably improved the original version of this article. We also thank C. Wolf and A. Koberstein for providing us with some test problems and E. Finardi and F. Beltrán for the instances of the multistage hydro-thermal power generation planning problem. Finally, the first and the second authors would like to acknowledge the partial financial support of PGMO (Gaspard Monge Program for Optimization and operations research) of the Hadamard Mathematics Foundation, through the project "Models for planning energy investment under uncertainty". The third author acknowledges partial support by the National Science Foundation (NSF) under grant CMMI 1854960. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

Appendix A: Proof of Lemma 1

Item (a) Let $\tilde{x}$ be a solution of (11), then $f(\tilde{x}) \leq f(x(\tau))$ (recall that both $\tilde{x}$ and $x(\tau)$ belong to X). Moreover, $f(x(\tau)) + \frac{1}{\tau}\varphi(x(\tau)) \leq f(\tilde{x}) + \frac{1}{\tau}\varphi(\tilde{x})$. These two inequalities show that $\varphi(x(\tau)) \leq \varphi(\tilde{x}) = \varphi^* < \infty$.

Item (b) The previous item ensures that $\varphi(x(\tau_k)) \leq \varphi(\tilde{x}) < \infty$ for all k. Since φ is a strong convex function, its level sets are compact. As a result, $\{x(\tau_k)\}$ is a bounded sequence and, therefore, there exists a constant $L < \infty$ bounding all subgradients of φ at $x(\tau_k)$ for all k.

Item (c) We first prove that

$$\varphi(x(\tau')) < \varphi(x(\tau'')) \text{ and } f(x(\tau')) < f(x(\tau'')) \text{ for all } 0 < \tau' < \tau''. \tag{29}$$

It follows from optimality of $x(\tau')$ and $x(\tau'')$ and strongly convexity of φ that

$$\begin{aligned} f(x(\tau')) + \varphi(x(\tau'))/\tau' &< f(x(\tau'')) + \varphi(x(\tau''))/\tau' \\ f(x(\tau'')) + \varphi(x(\tau''))/\tau'' &< f(x(\tau')) + \varphi(x(\tau'))/\tau''. \end{aligned} \tag{30}$$

By summing these inequalities we obtain $\varphi(x(\tau'))\left(\frac{1}{\tau'} - \frac{1}{\tau''}\right) < \varphi(x(\tau''))\left(\frac{1}{\tau'} - \frac{1}{\tau''}\right)$, showing that $\varphi(x(\tau')) < \varphi(x(\tau''))$. Inequality (30) implies $f(x(\tau'')) - f(x(\tau')) < \frac{1}{\tau''}[\varphi(x(\tau')) - \varphi(x(\tau''))] < 0$, yielding $f(x(\tau'')) < f(x(\tau'))$.

Next we show that

$$\begin{aligned} x(\tau) &= \operatorname{argmin}\{f(x) \text{ s.t. } x \in X,\ \varphi(x) \leq \varphi(x(\tau))\} \\ &= \operatorname{argmin}\{\varphi(x) \text{ s.t. } x \in X,\ f(x) \leq f(x(\tau))\}, \quad \forall\, \tau > 0. \end{aligned} \tag{31}$$

To this end, let $x \in X$ such that $x \neq x(\tau)$. Then $\frac{1}{\tau}[\varphi(x(\tau)) - \varphi(x)] < f(x) - f(x(\tau))$ from the definition of $x(\tau)$. If $\varphi(x) \leq \varphi(x(\tau))$ then $f(x(\tau)) < f(x)$, showing that $x(\tau)$ is the unique solution of $\min\{f(x) \text{ s.t. } x \in X,\ \varphi(x) \leq \varphi(x(\tau))\}$. Furthermore, if $f(x) \leq f(x(\tau))$, then $\varphi(x(\tau)) < \varphi(x)$, showing that $x(\tau)$ is the unique solution of $\min\{\varphi(x) \text{ s.t. } x \in X,\ f(x) \leq f(x(\tau))\}$.

We are now in position to proof item (c). Recall that $\varphi(x(\tau)) \leq \varphi(\tilde{x})$ from item (a). The equivalences follow from (31) $\varphi(\tilde{x}) = \varphi(x(\tau)) \Leftrightarrow f(\tilde{x}) = f(x(\tau)) \Leftrightarrow x(\tau) \in X^*$, and the identity $x(\tau') = \tilde{x}$ for all $\tau' \geq \tau$ follows from (29). □

Appendix B: Detailed descriptions of the test instances

We describe the test instances used in our numerical experiments in more detail. The interconnected network structure of the hydrothermal power plants in Brazil is depicted in Fig. 6, and the detailed data on hydro and thermo plants are given in Table 4 and Table 5, respectively. In addition, we set the demand to be a constant 14,500 in each stage, and we set the unit penalty cost of not satisfying the demand to be 4500. The scenario data file is too big to be displayed here, but will be available upon request.

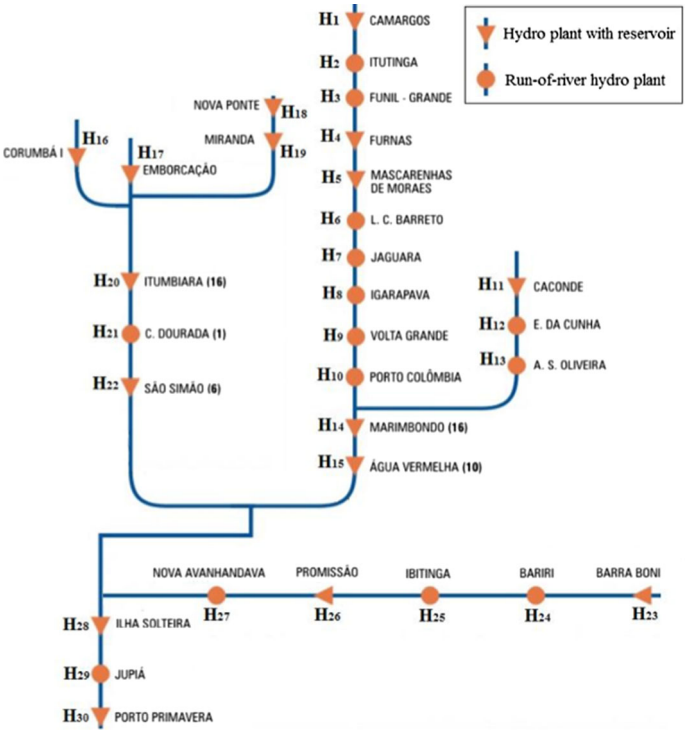

Fig. 6 The interconnected network structure of the hydrothermal power plants in Brazil

Table 4 Hydro plant data: the maximum allowed amount of turbined flow, the minimum/maximum level of water allowed, the initial water level, and the amount of power generated by releasing one unit of water flow in each hydro plant $h = 1, 2, \ldots, 30$

Plant	$\bar{y}_h^t (m^{3/s})$	$\underline{v}_h (hm^3)$	$\bar{v}_h (hm^3)$	$x_h^0 (\%)$	$r_h (MWa/(m^{3/s}))$
H_1	220	120	792	50	0.178645
H_2	236	11	11	–	0.244686
H_3	585	304	304	–	0.350193
H_4	1692	5733	22950	50	0.748479
H_5	1328	1540	4040	50	0.315981
H_6	2028	1423	1423	–	0.562731
H_7	1076	450	450	–	0.404259
H_8	1480	480	480	–	0.154781
H_9	1584	2244	2244	–	0.247221
H_{10}	1988	1524	1524	–	0.203826
H_{11}	94	51	555	50	0.775442
H_{12}	148	14	14	–	0.746079
H_{13}	178	25	25	–	0.206450
H_{14}	2944	890	6150	50	0.466867
H_{15}	2958	5856	11025	50	0.456953
H_{16}	570	470	1500	50	0.576865
H_{17}	1048	4669	17725	50	1.041320
H_{18}	576	2412	12792	50	0.940192
H_{19}	675	974	1120	50	0.617276
H_{20}	2940	4573	17027	50	0.645664
H_{21}	2513	460	460	–	0.282615
H_{22}	2670	7000	12540	50	0.609431
H_{23}	759	569	3135	50	0.156818
H_{24}	771	544	544	–	0.188124
H_{25}	702	985	985	–	0.187166
H_{26}	1293	5280	7408	50	0.199675
H_{27}	1431	2720	2720	–	0.260129
H_{28}	11604	25467	34432	50	0.383595
H_{29}	8344	3354	3354	–	0.198022
H_{30}	8904	14400	20000	50	0.171769

Table 5 Thermo plant data: maximum allowed amount of power generated and the unit cost of thermal power generated from each thermo plant f

Plant	$\bar{g}^{tf}$ (MWa)	c_f (R$/MWa)
T_1	640	23.21
T_2	1350	20.12
T_3	530	89.33
T_4	36	937
T_5	157	262.56
T_6	59	273.52
T_7	28	189.41
T_8	529	511.77
T_9	44	838.15
T_{10}	235	151.2
T_{11}	321	230.31
T_{12}	65	275.53
T_{13}	572	399.02
T_{14}	140	895.17
T_{15}	226	274.79
T_{16}	131	653.43
T_{17}	87	213.84
T_{18}	204	166.44
T_{19}	400	37.8
T_{20}	100	58.89
T_{21}	200	102.84
T_{22}	127	282.05
T_{23}	176	731.91
T_{24}	200	470.34
T_{25}	30	421.52
T_{26}	436	310.41
T_{27}	500	111.1
T_{28}	31	90
T_{29}	134	155.61
T_{30}	216	274.03
T_{31}	340	678.04
T_{32}	929	421.02
T_{33}	770	182.87
T_{34}	266	275.03
T_{35}	10	1047.38
T_{36}	175	440.24
T_{37}	206	197.85
T_{38}	54	1171.37

References

1. Asamov, T., Powell, W.B.: Regularized decomposition of high-dimensional multistage stochastic programs with markov uncertainty. SIAM J. Optim. **28**(1), 575–595 (2018)
2. Bank, B., Guddat, J., Klatte, D., Kummer, B., Tammer, K.: Non-linear Parametric Optimization. Birkhäuser, Basel (1982)
3. Ben-Tal, A., Nemirovski, A.: Non-euclidean restricted memory level method for large-scale convex optimization. Math. Program. **102**(3), 407–456 (2005)
4. Birge, J.R.: Decomposition and partitioning methods for multistage stochastic linear programs. Oper. Res. **33**(5), 989–1007 (1985)
5. Chen, Z.L., Powell, W.B.: Convergent cutting-plane and partial-sampling algorithm for multistage stochastic linear programs with recourse. J. Optim. Theory Appl. **102**(3), 497–524 (1999)
6. Clark, D.I., Osborne, M.R.: A descent algorithm for minimizing polyhedral convex functions. SIAM J. Sci. Stat. Comput. **4**(4), 757–786 (1983)
7. de Matos, V.L., Morton, D.P., Finardi, E.C.: Assessing policy quality in a multistage stochastic program for long-term hydrothermal scheduling. Ann. Oper. Res. **253**, 713–731 (2016)
8. de Oliveira, W.: Target radius methods for nonsmooth convex optimization. Oper. Res. Lett. **45**(6), 659–664 (2017)
9. de Oliveira, W., Sagastizábal, C.: Level bundle methods for oracles with on demand accuracy. Optim. Methods Softw. **29**(6), 1180–1209 (2014)
10. de Oliveira, W., Sagastizábal, C., Jardim Penna, D.D., Maceira, M.E.P., Damázio, J .M.: Optimal scenario tree reduction for stochastic streamflows in power generation planning problems. Optim. Methods Softw. **25**(6), 917–936 (2010)
11. de Oliveira, W., Sagastizábal, C.A., Scheimberg, S.: Inexact bundle methods for two-stage stochastic programming. SIAM J. Optim. **21**(2), 517–544 (2011)
12. de Oliveira, W., Solodov, M.: A doubly stabilized bundle method for nonsmooth convex optimization. Math. Program. **156**(1), 125–159 (2016)
13. de Oliveira, W., Sagastizábal, C., Lemaréchal, C.: Convex proximal bundle methods in depth: a unified analysis for inexact oracles. Math. Prog. Ser. B **148**, 241–277 (2014)
14. de Queiroz, Anderson Rodrigo, Morton, David P: Sharing cuts under aggregated forecasts when decomposing multi-stage stochastic programs. Oper. Res. Lett. **41**(3), 311–316 (2013)
15. Donohue, C., Birge, J.R.: The abridged nested decomposition method for multistage stochastic linear programs with relatively complete recourse. Algorithmic Oper. Res. **1**(1), 20–30 (2006)
16. Dupačová, J., Polívka, J.: Asset-liability management for Czech pension funds using stochastic programming. Ann. Oper. Res. **165**(1), 5–28 (2009)
17. Dupačová, J.: Portfolio Optimization and Risk Management via Stochastic Programming. Osaka University Press, Osaka (2009)
18. Fábián, C.I.: Bundle-type methods for inexact data. In: Proceedings of the XXIV Hungarian Operations Researc Conference (Veszprém, 1999). Special issue, T. Csendes and T. Rapcsák (eds.), vol. 8, pp. 35–55, (2000)
19. Ferris, M.C., Mangasarian, O.L.: Finite perturbation of convex programs. Appl. Math. Optim. **23**(1), 263–273 (1991)
20. Fhoula, B., Hajji, A., Rekik, M.: Stochastic dual dynamic programming for transportation planning under demand uncertainty. In: 2013 International Conference on Advanced Logistics and Transport, pp. 550–555, May (2013)
21. Girardeau, P., Leclere, V., Philpott, A.B.: On the convergence of decomposition methods for multistage stochastic convex programs. Math. Oper. Res. **40**(1), 130–145 (2014)
22. Goel, V., Grossmann, I.E.: A stochastic programming approach to planning of offshore gas field developments under uncertainty in reserves. Comput. Chem. Eng. **28**(8), 1409–1429 (2004)
23. Herer, Y.T., Tzur, M., Yücesan, E.: The multilocation transshipment problem. IIE Trans. **38**(3), 185–200 (2006)
24. Higle, J.L., Sen, S.: Stochastic decomposition: an algorithm for two-stage linear programs with recourse. Math. Oper. Res. **16**(3), 650–669 (1991)
25. Hindsberger, M., Philpott, A.B.: Resa: A method for solving multi-stage stochastic linear programs. In: SPIX Stochastic Programming Symposium, Berlin (2001)
26. Holmes, D.: A (po)rtable (s)tochastic programming (t)est (s)et (posts). http://users.iems.northwestern.edu/~jrbirge/html/dholmes/post.html (1995)

27. Homem de Mello, T., de Matos, V.L., Finardi, E.C.: Sampling strategies and stopping criteria for stochastic dual dynamic programming: a case study in long-term hydrothermal scheduling. Energy Syst. **2**(1), 1–31 (2011)
28. Homem de Mello, T., Pagnoncelli, B.: Risk aversion in multistage stochastic programming: a modeling and algorithmic perspective. Eur. J. Oper. Res. **249**, 188–199 (2016)
29. Kelley, J.E.: The cutting-plane method for solving convex programs. J. Soc. Ind. Appl. Math. **8**(4), 703–712 (1960)
30. Kiwiel, K.C.: Finding normal solutions in piecewise linear programming. Appl. Math. Optim. **32**(3), 235–254 (1995)
31. Kiwiel, K.C.: Proximal level bundle methods for convex nondifferentiable optimization, saddle-point problems and variational inequalities. Math. Program. **69**(1), 89–109 (1995)
32. Lemaréchal, C.: An extension of davidon methods to nondifferentiable problems. Math. Program. Study **3**, 95–109 (1975)
33. Lemaréchal, C.: Constructing bundle methods for convex optimization. In: Hiriart-Urruty, J. B. (ed.) Fermat Days 85: Mathematics for Optimization. North-Holland Mathematics Studies, vol. 129, pp. 201–240. North-Holland (1986)
34. Lemaréchal, C., Nemirovskii, A., Nesterov, Y.: New variants of bundle methods. Math. Program. **69**(1), 111–147 (1995)
35. Linowsky, K., Philpott, A.B.: On the convergence of sampling-based decomposition algorithms for multistage stochastic programs. J. Optim. Theory Appl. **125**(2), 349–366 (2005)
36. Maceira, M.E.P., Terry, L.A., Costa, F.S., Damázio, J.M., Melo, A.C.G.: Chain of optimization models for setting the energy dispatch and spot price in the Brazilian system. In: Proceedings of the 14th Power Systems Computation Conference—PSCC, pp. 1–7. Servilla, Spain (2002)
37. Morton, D.P.: Stopping rules for a class of sampling-based stochastic programming algorithms. Oper. Res. **46**(5), 710–718 (1998)
38. Nesterov, Y.: Introductory Lectures on Convex Optimization. A Basic Course. Applied Optimization, vol. 87. Springer, Berlin (2004)
39. Pereira, M.V., Granville, S., Fampa, M.H.C., Dix, R., Barroso, L.A.: Strategic bidding under uncertainty: a binary expansion approach. IEEE Trans. Power Syst. **11**(1), 180–188 (2005)
40. Pereira, M.V.F., Pinto, L.M.V.G.: Multi-stage stochastic optimization applied to energy planning. Math. Program. **52**(2), 359–375 (1991)
41. Ch Pflug, G., Römisch, W.: Modeling. Measuring and Managing Risk. World Scientific, Singapore (2007). https://www.worldscientific.com/worldscibooks/10.1142/6478
42. Philpott, A.B., de Matos, V.L.: Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion. Eur. J. Oper. Res. **218**(2), 470–483 (2012)
43. Philpott, A.B., Guan, Z.: On the convergence of stochastic dual dynamic programming and related methods. Oper. Res. Lett. **36**(4), 450–455 (2008)
44. Rebennack, S.: Combining sampling-based and scenario-based nested benders decomposition methods: application to stochastic dual dynamic programming. Math. Program. **156**(1), 343–389 (2016)
45. Ruszczyński, A.: On the regularized decomposition method for stochastic programming problems. In: Marti, K., Kall, P. (eds.) Stochastic Programming: Numerical Techniques and Engineering Applications, pp. 93–108. Springer, Berlin (1995)
46. Sen, S., Zhou, Z.: Multistage stochastic decomposition: a bridge between stochastic programming and approximate dynamic programming. SIAM J. Optim. **24**(1), 127–153 (2014)
47. Shapiro, A.: Analysis of stochastic dual dynamic programming method. Eur. J. Oper. Res. **209**, 63–72 (2011)
48. Shapiro, A., Dentcheva, D., Ruszczyński, A.: Lectures on stochastic programming. Modeling and Theory. MPS-SIAM Series on Optimization. SIAM and MPS, vol. 9. Philadelphia, (2009)
49. Shapiro, A., Tekaya, W., da Costa, J.P., Soares, M.P.: Risk neutral and risk averse stochastic dual dynamic programming method. Eur. J. Oper. Res. **224**(2), 375–391 (2013)
50. van Ackooij, W., de Oliveira, W., Song, Y.: An adaptive partition-based level decomposition for solving two-stage stochastic programs with fixed recourse. Inf. J. Comput. **30**(1), 57–70 (2018)
51. van Ackooij, W., Frangioni, A., de Oliveira, W.: Inexact stabilized Benders' decomposition approaches: with application to chance-constrained problems with finite support. Comput. Optim. Appl. **65**(3), 637–669 (2016)
52. van Ackooij, W., Lebbe, N., Malick, J.: Regularized decomposition of large-scale block-structured robust optimization problems. Comput. Manag. Sci. **14**(3), 393–421 (2017)

53. Wolf, C., Fábián, C .I., Koberstein, A., Stuhl, L.: Applying oracles of on-demand accuracy in two-stage stochastic programming. A computational study. J. Oper. Res. **239**(2), 437–448 (2014)
54. Wolf, C., Koberstein, A.: Dynamic sequencing and cut consolidation for the parallel hybrid-cut nested L-shaped method. Eur. J. Oper. Res. **230**(1), 143–156 (2013)