Manuscript

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

Wim van Ackooij, Welington de Oliveira and Yongjia Song

December 6, 2017

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* 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 [6,20,26,37,42,47], transportation [18,22] and finance [14,15,39]. The typical approach to solve these 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 [3] and the *stochastic dual dynamic programming* (SDDP) proposed by [38]. 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, [4,13,24, 38]. 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 [38]. 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 [4]. This proof is extended by Linowsky and Philpott to cover the SDDP algorithm and other sampling-based methods in [33]. The convergence analysis of SDDP is revisited in [41], and more recently in [45]. ND and SDDP techniques have been applied in a variety of

W. van Ackooij
EDF R&D. OSIRIS 7, Boulevard Gaspard Monge, F-91120 Palaiseau Cedex France
E-mail: wim.van-ackooij@edf.fr

W. de Oliveira
MINES ParisTech, PSL – Research University, CMA – Centre de Mathématiques Appliquées, France.
E-mail: welington.oliveira@mines-paristech.fr

Y. Song
Virginia Commonwealth University, USA E-mail: ysong3@vcu.edu

applications, just to mention a few: [9, 20, 22, 40, 47]. ND and SDDP have also been extended to cope with more general convex multistage stochastic programs, see, e.g., [19, 21].

The workhorse in the optimization techniques mentioned above is Kelley's cutting-plane method [27], 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, [35, 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 [30–32] 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 [8, 10, 16, 43, 49]. In the two-stage context, the stochastic decomposition of [23] combines sampling with a regularization scheme akin to proximal bundle methods. In [48] 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 [44]. Sen and Zhou [44] have extended the stochastic decomposition of [23] 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 [44], but extend the regularized decomposition of [43] 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 [44] and [1] make use of stability centers, which strongly impact the iteration process by keeping iterates close to such centers. In general, a stability center is 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 do not suffer from the effect of possibly bad quality stability centers. Our algorithms take inspiration in the level bundle methods for nonsmooth optimization, [29, 32], well known for their efficiency [2, 7], flexibility in defining stability centers and dealing with functions for which gradients and values are only computed approximately, [8].

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 [32] with some results on finite perturbation of convex programs. As in [1] and [44], 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 [13, 41, 45].

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 Section 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 Section 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. We compare the new proposals with the multistage regularized decomposition of [1] and an implementation of the classical SDDP, and show the effectiveness of our proposed regularization scheme. Finally, we conclude in Section 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 [27].

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 $t = 2, \dots, T$. 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 $(t = 1, \dots, T)$ 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 an SAA problem and the underlying true problem with a continuous distribution we refer to [45].

As in the classical work of [38] we also make the basic assumption that the random data process is stagewise independent, i.e., random vector ξ_{t+1} is independent of $\xi_{[t]} := (\xi_1, \dots, \xi_t)$, 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 [45], or be dealt with by performing a cut-sharing strategy [12].

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

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

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

$$\begin{cases} \min\limits_{x_1\geq 0} c_1^\top x_1 + \mathcal{Q}_2(x_1) \\ \text{s.t.} \quad A_1x_1 = b_1\,. \end{cases} \tag{3}$$

An implementable policy for (1) is a collection of functions $\bar{x}_t = \bar{x}_t(\xi_{[t]})$, $t = 1, \dots, T$. 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. A policy is feasible for problem (1) if it satisfies all the constraints under any realization in each stage t.

As in [45], 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 [46, Chap. 3].

Decompositions. As already mentioned, two very important decomposition techniques for solving MSLPs are ND (see [3]), and SDDP (see [38]). 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

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

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:

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

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 $t = T-1, \ldots, 2$:

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

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

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

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

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

and satisfies $\mathcal{Q}_t(x_{t-1}) \geq q_t(x_{t-1}) \;\; \forall\; x_{t-1}$. (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 $\underline{Q}_{T-1}(\bar{x}_{T-2}, \xi_{T-1}) \leq Q_{T-1}(\bar{x}_{T-2}, \xi_{T-1})$ 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

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

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

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

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

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

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 [45, Sec.3] for a discussion on this subject.

3 Normal solutions and finite perturbation of convex programs

The regularization scheme of MSLPs to be proposed later in Section 4 relies on finite perturbation of convex programs. We put aside for a while the uncertainty and focus on deterministic convex programs, 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 [17] 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 [5, Theorem 2.1] and [28, 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:*

(i) $\varphi(x(\tau)) \leq \varphi^*$ for all $\tau > 0$.
(ii) For $0 < \tau' < \tau''$, $\varphi(x(\tau')) \leq \varphi(x(\tau''))$ and $f(x(\tau')) \geq f(x(\tau''))$.

Moreover, if φ is a strongly convex function and $\tilde{x}$ is the unique solution of problem (11), then

(iii) Item (ii) holds with strict inequalities if $x(\tau') \neq x(\tau'')$.
(iv) Let $\{\tau_k\}$ be an arbitrary sequence of positive numbers. Then $\{x(\tau_k)\}$ is a bounded sequence and there exists a constant $L < \infty$ bounding all subgradients of φ at $x(\tau_k)$ for all k.
(v) The following equality holds

$$x(\tau) = \operatorname{argmin}\{f(x) \;\; s.t.\; x \in X,\; \varphi(x) \leq \varphi(x(\tau))\} = \operatorname{argmin}\{\varphi(x) \;\; s.t.\; x \in X,\; f(x) \leq f(x(\tau))\},$$

for all $\tau > 0$.
(vi) As a function of τ, the maps $x(\tau)$, $\varphi(x(\tau))$ and $f(x(\tau))$ are continuous for all $\tau > 0$.
(vii) We have that $\varphi(\tilde{x}) = \varphi(x(\tau)) \quad \Leftrightarrow \quad f(\tilde{x}) = f(x(\tau)) \quad \Leftrightarrow \quad x(\tau) \in X^*$, and if one of these three equalities holds for τ, then $x(\tau') = \tilde{x}$ for all $\tau' \geq \tau$.
(viii) $\lim_{\tau\to\infty} x(\tau) = \tilde{x}$ and $\lim_{\tau\to\infty} f(x(\tau)) = f(\tilde{x})$.

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 iv) we obtain

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

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 $\bar{x} \in \operatorname{argmin}_{x\in X} F(x)$. 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 vii).

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 i) $\varphi(\bar{x}) \leq \varphi^*$. 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 [28] proposes an algorithm that solves problem (11) after finitely many steps by successively solving (12) with an appropriate rule for defining τ.

Remark 1 Suppose that f is a convex piecewise linear function, X is a polyhedral set, φ a strongly convex function and $\ell \in \mathbb{R}$ is a given parameter. Let $\check{f}(x) := \max\{f(x), \ell\}$ and $X^*(\ell) \neq \emptyset$ be the solution set of $\min_{x\in X} \check{f}(x)$. Consider the following optimization problem $\tilde{x} = \operatorname{argmin}_{x\in X^*(\ell)} \varphi(x)$. Then

$$\tilde{x} \in \begin{cases} \operatorname{argmin}_{x\in X} \varphi(x) \;\text{ s.t. }\; f(x) \leq \ell & \text{if } \ell \geq f^* \\ \operatorname{argmin}_{x\in X} f(x) & \text{if } \ell < f^* \,. \end{cases}$$

Moreover, $\tilde{x}$ can be obtained by solving the *perturbed problem* $\min_{x\in X} \check{f}(x) + \varphi(x)/\bar{\tau}$ with a large enough parameter $\bar{\tau}$; see Proposition 1 item 1). If X is a bounded polyhedral set, then we can employ, for instance, $\varphi(x) = \|x\|_1$, making the perturbed problem a LP.

In the next section we rely on Remark 1 with $x = x_t$ and $f(x_t) = c_t^\top x_t + \check{\mathcal{Q}}_{t+1}(x_t)$ 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., [36]), 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. 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., [29, 32]. 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 [8, 16, 29, 32]. 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 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 2 Regularizing only the first stage of MSLPs can be done without much difficulties by following the general lines of recent bundle methods [8] and [36], 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 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 [36] 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., [50]).

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) (provided the feasible set is bounded), there exists a constant $\bar{\tau} < \infty$ such that the solution of problem (13) also solves

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

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 [13, 41] and [45, 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).

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, 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:

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

Instead of solving LP (8) as SDDP does, or solving the QP (13) as the multistage regularized decomposition of [1] (and [44]) 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

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

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

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

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 [28] 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 3 We now distinguish two cases regarding problem (16):

– **Case 1**. *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**. *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*.

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 Section 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 the define the level parameters. We will discuss this topic in subsection 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 3 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 SDDP 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 $\ell_1^k \in [\underline{z}^{k-1}, \overline{z}^{k-1}]$ as in standard level bundle methods makes sense. However, the choice $\ell_t^k \in [\underline{z}^{k-1}, \overline{z}^{k-1}]$ 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

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

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 $Q_t(\bar{x}_{t-1}^k, \xi_t^j)$:

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

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

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

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

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

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, the forward steps with level iterates in Algorithm 1 can be seen as an extra procedure for improving the cutting-plane models. 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 classical 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 classical 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 classical 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 [11], 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 $x \mapsto \mathbb{E}_{\xi_{[t]}^j}[\underline{Q}_{t+1}(x, \xi_{t+1})]$ coincides with the cost-to-go function $\mathcal{Q}_{t+1}(\cdot)$ at $\bar{x}_t^k$ 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 $\overline{z}_{\min,t,j}^{k-1} \leq c_t^\top x_t^{k'} + \mathcal{Q}_{t+1}(\bar{x}_t^{k'})$ holds for all $k \geq k' \geq \bar{k}$, with $\overline{z}_{\min,t,j}^{k-1}$ given in (23). If $k \in \mathcal{L}_{t,j}$, then*

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

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

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

for all $k \geq k' \geq \bar{k}$. The stated result thus follows from the Cauchy-Schwartz 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 $\mathbb{E}_{\xi_{[T-1]}^j}[\underline{Q}_T(\cdot, \xi_T)] = \mathcal{Q}_T(\cdot)$;
- $\overline{z}_{\min,T-1,j}^{k-1}$ is by definition the lowest known bound on the objective $c_{T-1}^\top x_{T-1} + \mathcal{Q}_T(x_{T-1})$ defined at scenario node $(T-1, j)$.

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 β_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

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

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, j) is infinite, i.e., $|\mathcal{L}_{T-1,j}| = \infty$, then compactness of the feasible set ensures that there exists a convergent subsequence $\{\bar{x}_{T-1}^{k'}\}_{k'}$. By plugging this subsequence into the above inequality 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., $\bar{x}^k_{T-1} = \bar{x}^k_{T-1}(\xi^j_{[T-1]})$ 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, $\underline{Q}_{T-1}(\bar{x}^k_{T-2}, \xi_{T-1}) = c_{T-1}^\top \bar{x}^k_{T-1} + \mathcal{Q}_T(\bar{x}^k_{T-1})$ for all given $\bar{x}^k_{T-2}$ with $k \geq \bar{k}$ and, moreover,

$$\mathbb{E}_{|\xi^j_{[T-2]}}[\underline{Q}_{T-1}(\bar{x}^k_{T-2}, \xi_{T-1})] = \mathcal{Q}_{T-1}(\bar{x}^k_{T-2}) = \mathbb{E}_{|\xi^j_{[T-2]}}\left[c_{T-1}^\top \bar{x}^k_{T-1} + \mathbb{E}_{|\xi_{[T-1]}}[c_T^\top \bar{x}^k_T]\right] \quad \forall\, k \geq \bar{k}\,, \tag{25}$$

where the last equality is due to the fact that $\check{\mathcal{Q}}_T(x^k_{T-1}) = \mathcal{Q}_T(x^k_{T-1})$ 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

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

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 $\bar{x}^k_1$ is a first-stage optimal solution of (1). This concludes the proof. □

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 $|\mathcal{J}| \ll N$ scenarios and solve subproblems (8) ($t = 1, \ldots, T-1$) only for these scenarios. In this case, $\tilde{z}^k = \frac{1}{|\mathcal{J}|}\sum_{r\in\mathcal{J}}[\sum_{t=1}^T c_t^\top \bar{x}^k_t(\xi^r_{[t]})]$ estimates an 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 [45]).

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 $1 \leq |\mathcal{J}^k| \leq N$ scenarios. For each $t = 1, \ldots, T-1$ and each $\xi^{k,r} \in \mathcal{J}^k$, get $(c_t, B_t, A_t, b_t) = \xi_t^{k,r}$, choose level parameter $\ell_{t,r}^k$ according to some rule, and solve subproblem (16) (with $\ell_t = \ell_{t,r}^k$) to obtain $\bar{x}_t^k := \bar{x}_t(\xi_{[t]}^{k,r})$. 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 classical 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 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 [34, 45].

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 Figure 1 is traversed by 12 scenarios, instead of only 4 scenarios having the same history up to time $t = 3$. For this reason

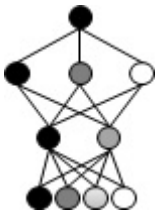


Fig. 1 Stagewise independent scenario tree (lattice) with 4 stages and 24 scenarios. Notice the black node in stage 3 composes 12 scenarios. Whenever one of these 12 scenarios is chosen in the forward step, such node will be visited and therefore its upper bound estimate $\tilde{z}_t(\xi_t)$ might be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

we employ the following alternative to the minimum upper bound rule given by (23), when estimating an upper bound for each node:

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

where

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

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 $\tilde{z}_{\min,t,j}^{k-1}$ 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 Figure 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 $\bar{x}_t(\xi_{[t]}^d)$ to represent the decision made at stage t of scenario ξ^d after the forward step[1]. In order to compute $\tilde{z}_3^k(\xi_3^B)$, we first compute the present and expected future cost with respect to the first 3 scenarios sharing the same history $\xi_{[3]} = (\xi_1^B\, \xi_2^B\, \xi_3^B)$:

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

[1] Note that $\bar{x}_3(\xi_{[3]}^a) = \bar{x}_3(\xi_{[3]}^b) = \bar{x}_3(\xi_{[3]}^c)$ and $\bar{x}_3(\xi_{[3]}^d) = \bar{x}_3(\xi_{[3]}^e)$.

Then we compute the present and expected future cost with respect to the remaining scenarios sharing the history $\xi_{[3]} = (\xi_1^B\ \xi_2^W\ \xi_3^B)$*:* $\texttt{cost}^{k'}_{|\xi_{[3]}} = \texttt{cost}^{i}_{|(\xi_1^B\ \xi_2^W\ \xi_3^B)} = c_3^\top \bar{x}_3(\xi^d_{[3]}) + [c_4^\top \bar{x}_4(\xi^d_{[4]}) + c_4^\top \bar{x}_4(\xi^e_{[4]})]/2$ *. The inner minimization problem in* (26a) *is thus:*

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

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 classical SDDP. In order to resort to the convergence results of [45], 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 classical 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 $\bar{x}^k_t(\xi^{j^*}_{[t]})$ 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 [45, 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 $\hat{x}_t = 0$, $\hat{x}_t = \bar{x}^{k-1}_t(\xi_{[t]})$ (the trial point used in

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

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 [28] 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 classical 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 arise 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 experiment, 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. In particular, we focus on the normal-level SDDP algorithm proposed in Section 4.4, and compare it with classical SDDP as well as the multistage regularized decomposition of [1]. SDDP type approaches are more appropriate than ND for solving the test problems that we choose in our numerical experiments, which we describe in detail later. We implement all algorithms in C++ using commercial solver CPLEX, version 12.5.1. 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.

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, used to model 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 of 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 into 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 stochastic water inflows are stage-wise independent. Let $\mathcal{Q}_{t+1}(x^t)$ be the expected cost-to-go function at stage t, the stage t problem can be written as:

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

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

We customize this data set and create a variety of instances with different number of time stages. We then sample from a given distribution of random amount of inflows for each hydro plant in each time stage with various sample sizes: $|\Xi_t| \in \{10, 20, 50, 80\}$, $t = 2, \ldots, T$. For simplicity, we let the number of realizations be the same for each stage. We remark that SDDP type algorithms are more appropriate for these problem instances that we consider.

5.2 Implementation details

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

- The classical SDDP.
- 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 Section 2 for the implementation of the classical SDDP. 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. This algorithm is denoted by "SDDP" in the following tables.

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 [28] 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) (denoted by "normal-level-1" in the following tables) and the discount rule (20) (denoted by "normal-level-2" in the following tables) for choosing the level parameters. For both cases, we set the level parameter as $\gamma_t = 0.5/t$ for $t = 2, \ldots, T-1$. We follow [1] by regularizing the iterates only on the state variables (post-decision state variables).

Implementation of the normal-solution SDDP algorithm. The implementation of the normal-solution SDDP algorithm is identical to that of the normal-level SDDP algorithm, except that we set the level parameter as $-\infty$, so that only normal iterates will be performed. This algorithm is denoted by "normal" in the following tables.

Implementation of the multistage regularized decomposition proposed by [1]. We follow [1] to implement the multistage regularized decomposition algorithm: in each iteration $k > 1$, we use the trial points obtained in the forward pass from the previous iteration $k-1$ as the stabilization centers, and 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-regularization" in the following tables.

5.3 Performance on improving lower bounds with a fixed number of iterations using a single sample path per forward step

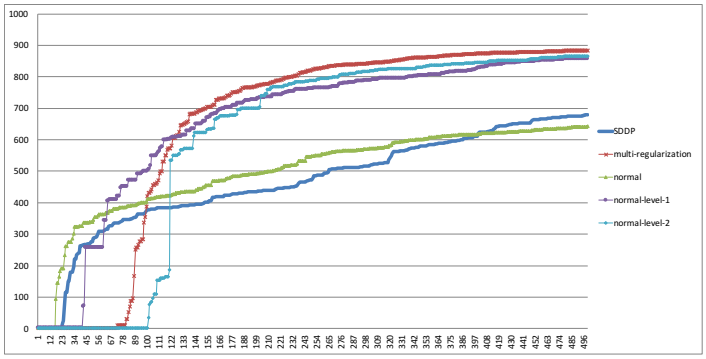
We first show our experiment results by setting the number of sample paths to be one in each forward step. We note that this is the setting used in multistage regularized decomposition [1], and suggested in other settings as well [51]. Since we only use one sample path in each forward pass, we may not obtain a reliable upper bound estimation $\tilde{z}^k$. To address this issue, we obtain a more reasonable estimation of $\tilde{z}^k$ using 200 sample paths every 20 iterations for the first 200 iterations, and every 50 iterations after that. Since the goal of this section is to compare the iteration-by-iteration lower bound improvement progress by different algorithms, we do not impose any time limit, and instead report the lower bound progress up to 500 iterations.

Table 2 Lower bound improvements obtained by various algorithms compared to the one obtained by the standard SDDP at iteration = 500 (one sample path per forward step, use extra upper bound estimation periodically for variant "normal-level-1" and "normal-level-2"). Positive values mean improvement and negative ones mean deterioration.

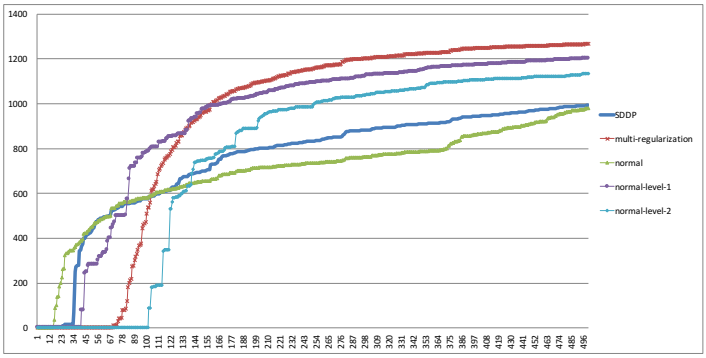
T	$\lvert\Xi_t\rvert$	multi-regularization	normal	normal-level-1	normal-level-2
25	10	30.0%	-5.4%	26.2%	27.5%
	50	27.4%	-1.4%	21.0%	15.7%
61	10	15.9%	6.1%	22.0%	14.1%
	50	13.0%	-0.4%	29.5%	6.6%

Table 2 and Figure 2 show the lower bound quality and lower bound progress comparison among various algorithms, respectively. Comparing variants "normal" and "normal-level-1" or "normal-level-2", we see that the majority of regularization effect in the proposed normal-level SDDP algorithms comes from the level iterates. In addition, we see that the regularization effect of the proposed normal-level SDDP (especially if implemented with the minimum upper bound rule (27)) is competitive with that of "multi-regularization", and is more significant for instances with a larger number of stages. Finally, we see from Figure 2 that the proposed normal-level SDDP algorithms yield more progress in terms of improving the lower bounds early on during the process.

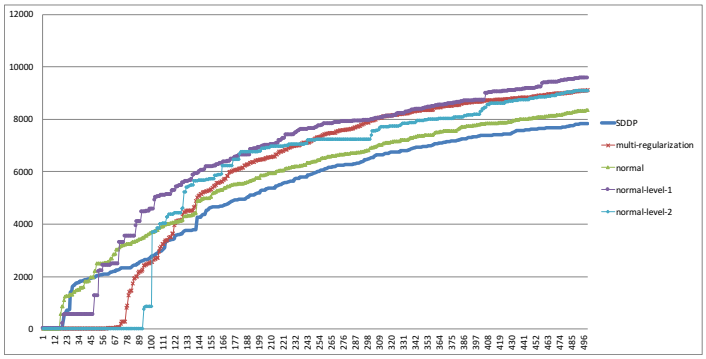
We note that compared to alternative approaches, the proposed normal-level SDDP algorithm relies on a reasonable estimation of $\tilde{z}^k$, which requires extra computational effort as described above. However, we



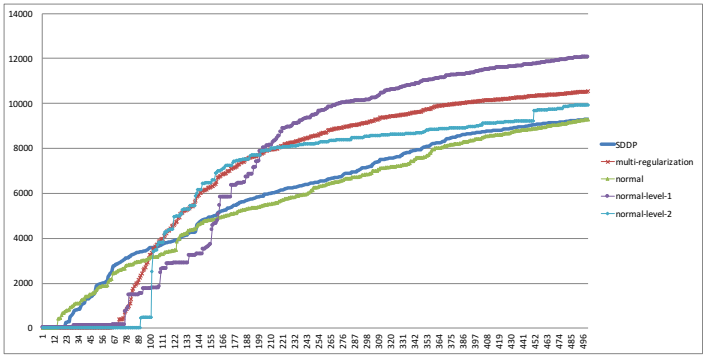
(a) 25 stages, 10 scenarios per stage



(b) 25 stages, 50 scenarios per stage



(c) 61 stages, 10 scenarios per stage



(d) 61 stages, 50 scenarios per stage

Fig. 2 Solution progress for various instances depending on the number of considered stages.

argue that this extra effort can be allocated to separate CPUs or computers, e.g., in a parallel computing environment. In spite of this argument, we consider using multiple sample paths per forward step in the experiment result reported in the next section on performance comparison with a fixed CPU time, to make it a fair comparison between different algorithms.

5.4 Performance on lower bounds with fixed CPU time using multiple sample paths per forward step

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 one hour (3600 seconds). To make the comparison fair, we set the number of threads to be one, and we employ 10 sample paths in each forward step, so that reasonable upper bound estimates can be obtained from these sample paths for variant "Normal-level SDDP". (We also performed experiments by varying the number of sample paths in the forward step, but did not observe significant differences, and therefore chose to report results using 10 as the sample size.) An exception is that for the multistage regularized decomposition, we adopt the setting used in [1], which is to use only one sample path per iteration (we also observed in our numerical experiments that this variant yielded better results).

Table 3 Lower bounds obtained by various algorithms with a fixed time limit of one hour compared to the lower bound obtained by "SDDP". Use multiple sample paths per iteration in the forward step for "SDDP", "normal-level-1" and "normal-level-2", and use a single sample path per iteration in the forward step for "multi-regularization".

T	$\lvert\Xi_t\rvert$	multi-regularization		normal-level-1		normal-level-2	
		LB	# of iterations	LB	# of iterations	LB	# of iterations
25	20	0.2%	1222	3.3%	130	-0.7%	126
	50	2.8%	1125	5.9%	122	-3.3%	114
	80	2.9%	1067	8.0%	115	1.5%	108
61	20	-10.8%	817	-2.2%	87	-10.4%	86
	50	-8.4%	744	2.5%	79	-9.1%	78
	80	-4.3%	692	8.8%	74	-5.9%	73
97	20	-12.2%	685	-6.9%	73	-14.0%	72
	50	-0.3%	617	3.3%	67	-10.3%	65
	80	-1.1%	567	2.0%	61	-6.9%	59

From Table 3, we see that although the standard SDDP has a slower progress of lower bound improvement, as (8) is an LP, which is much more efficient to solve than the QP (17), much more iterations are finished within the fixed time limit by SDDP than other algorithms. From this perspective, we see that the standard SDDP and the proposed normal-level SDDP algorithms yield similar performance on our test instances when the number of realizations in each stage is relatively small. 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. 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. Comparing "normal-level-1", "normal-level-2" and "multi-regularization" across all cases, we conclude that the proposed normal-level SDDP algorithm implemented with the minimum upper bound rule gives the best performance on the test instances.

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.

As perspectives are concerned we can think of extending the given algorithms to deal with risk-averse MSLPs [25], as well as furnishing the backward step with an adaptive partition-based scheme akin to [48]

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.

Acknowledgement

The authors would like to acknowledge E. Finardi and F. Beltrán (Federal University of Santa Catarina, Brazil) for their helpful comments and for providing us with the problem instance considered in this work.

References

1. T. Asamov and W. B. Powell. Regularized decomposition of high-dimensional multistage stochastic programs with markov uncertainty. Technical report, Princeton, ArXiv report number 1505.02227, 2015.
2. A. Ben-Tal and A. Nemirovski. Non-euclidean restricted memory level method for large-scale convex optimization. *Mathematical Programming*, 102(3):407–456, 2005.
3. J. R. Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research*, 33(5):989–1007, 1985.
4. Z. L. Chen and W. B. Powell. Convergent cutting-plane and partial-sampling algorithm for multistage stochastic linear programs with recourse. *Journal of Optimization Theory and Applications*, 102(3):497–524, 1999.
5. D. I. Clark and M. R. Osborne. A descent algorithm for minimizing polyhedral convex functions. *SIAM Journal on Scientific and Statistical Computing*, 4(4):757–786, 1983.
6. V. L. de Matos, D. P. Morton, and E. C. Finardi. Assessing policy quality in a multistage stochastic program for long-term hydrothermal scheduling. *Annals of Operations Research*, pages 1–19, 2016.
7. W. de Oliveira. Target radius methods for nonsmooth convex optimization. *Operations Research Letters*, 45(6):659 – 664, 2017.
8. W. de Oliveira and C. Sagastizábal. Level bundle methods for oracles with on demand accuracy. *Optimization Methods and Software*, 29(6):1180–1209, 2014.
9. W. de Oliveira, C. Sagastizábal, D. D. Jardim Penna, M. E. P. Maceira, and J. M. Damázio. Optimal scenario tree reduction for stochastic streamflows in power generation planning problems. *Optimization Methods and Software*, 25(6):917–936, 2010.
10. W. de Oliveira, C.A. Sagastizábal, and S. Scheimberg. Inexact bundle methods for two-stage stochastic programming. *SIAM Journal on Optimization*, 21(2):517–544, 2011.
11. W. de Oliveira and M. Solodov. A doubly stabilized bundle method for nonsmooth convex optimization. *Mathematical Programming*, 156(1):125–159, 2016.
12. Anderson Rodrigo de Queiroz and David P. Morton. Sharing cuts under aggregated forecasts when decomposing multi-stage stochastic programs. *Operations Research Letters*, 41(3):311 – 316, 2013.
13. C. Donohue and J. R. Birge. The abridged nested decomposition method for multistage stochastic linear programs with relatively complete recourse. *Algorithmic Operations Research*, 1(1):20–30, 2006.
14. J. Dupačová and J. Polívka. Asset-liability management for czech pension funds using stochastic programming. *Annals of Operations Research*, 165(1):5–28, 2009.
15. J. Dupačová. *Portfolio Optimization and Risk Management via Stochastic Programming*. Osaka University Press, 2009.
16. C.I. Fábián. Bundle-type methods for inexact data. In *Proceedings of the XXIV Hungarian Operations Researc Conference (Veszprém, 1999)*, volume 8 (special issue, T. Csendes and T. Rapcsk, eds.), pages 35–55, 2000.
17. M. C. Ferris and O. L. Mangasarian. Finite perturbation of convex programs. *Applied Mathematics and Optimization*, 23(1):263–273, 1991.
18. B. Fhoula, A. Hajji, and M. Rekik. Stochastic dual dynamic programming for transportation planning under demand uncertainty. In *2013 International Conference on Advanced Logistics and Transport*, pages 550–555, May 2013.
19. P. Girardeau, V. Leclere, and A. B. Philpott. On the Convergence of Decomposition Methods for Multistage Stochastic Convex Programs. *Mathematics of Operations Research*, 40(1), February 2015.
20. V. Goel and I. E. Grossmann. A stochastic programming approach to planning of offshore gas field developments under uncertainty in reserves. *Computers & Chemical Engineering*, 28(8):1409 – 1429, 2004.
21. V. Guigues, M. Lejeune, and W. Tekaya. Regularized decomposition methods for deterministic and stochastic convex optimization and application to portfolio selection with direct transaction and market impact costs. *Submitted for publication, preprint available at* `http://www.optimization-online.org/DB_FILE/2017/01/5815.pdf`, 2017.
22. Y. T. Herer, M. Tzur, and E. Yücesan. The multilocation transshipment problem. *IIE Transactions*, 38(3):185–200, 2006.
23. J. L. Higle and S. Sen. Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics Of Operations Research*, 16(3):650–669, 1991.
24. M. Hindsberger and A.B. Philpott. Resa: A method for solving multi-stage stochastic linear programs. In *SPIX Stochastic Programming Symposium, Berlin*, 2001.

25. T. Homem de Mello and B. Pagnoncelli. Risk aversion in multistage stochastic programming: A modeling and algorithmic perspective. *European Journal of Operations Research*, 249:188–199, 2016.
26. Tito Homem-de Mello, Vitor L. de Matos, and Erlon C. Finardi. Sampling strategies and stopping criteria for stochastic dual dynamic programming: a case study in long-term hydrothermal scheduling. *Energy Systems*, 2(1):1–31, Mar 2011.
27. J.E. Kelley. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.
28. K.C. Kiwiel. Finding normal solutions in piecewise linear programming. *Applied Mathematics and Optimization*, 32(3):235–254, 1995.
29. K.C. Kiwiel. Proximal level bundle methods for convex nondifferentiable optimization, saddle-point problems and variational inequalities. *Math. Programming*, 69(1):89–109, 1995.
30. C. Lemaréchal. An extension of davidon methods to nondifferentiable problems. *Mathematical programming study*, 3:95–109, 1975.
31. C. Lemaréchal. Constructing bundle methods for convex optimization. In *Fermat Days 85: Mathematics for Optimization*, volume 129 of *North-Holland Mathematics Studies*, pages 201–240. North-Holland, 1986.
32. C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. New variants of bundle methods. *Math. Programming*, 69(1):111–147, 1995.
33. K. Linowsky and A.B. Philpott. On the convergence of sampling-based decomposition algorithms for multistage stochastic programs. *Journal of Optimization Theory and Applications*, 125(2):349–366, 2005.
34. David P. Morton. Stopping rules for a class of sampling-based stochastic programming algorithms. *Operations Research*, 46(5):710–718, 1998.
35. Y. Nesterov. *Introductory Lectures on Convex Optimization. A Basic Course*, volume 87 of *Applied Optimization*. Springer Science, 2004.
36. W. de Oliveira, C. Sagastizábal, and C. Lemaréchal. Convex proximal bundle methods in depth: a unified analysis for inexact oracles. *Math. Prog. Series B*, 148:241–277, 2014.
37. M.V. Pereira, S. Granville, M.H.C. Fampa, R. Dix, and L.A. Barroso. Strategic bidding under uncertainty: A binary expansion approach. *IEEE Transactions on Power Systems*, 11(1):180–188, February 2005.
38. M.V.F. Pereira and L.M.V.G. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52(2):359–375, 1991.
39. G. Ch. Pflug and W. Römisch. *Modeling, Measuring and Managing Risk*. World Scientific Publishing Company, 2007.
40. A.B. Philpott and V. L. de Matos. Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion. *European Journal of Operational Research*, 218(2):470 – 483, 2012.
41. A.B. Philpott and Z. Guan. On the convergence of stochastic dual dynamic programming and related methods. *Operations Research Letters*, 36(4):450–455, 2008.
42. S. Rebennack. Combining sampling-based and scenario-based nested benders decomposition methods: application to stochastic dual dynamic programming. *Mathematical Programming*, 156(1):343–389, 2016.
43. A. Ruszczyński. On the regularized decomposition method for stochastic programming problems. In K. Marti and P. Kall, editors, *Stochastic Programming: Numerical Techniques and Engineering Applications*, pages 93–108. Springer Berlin Heidelberg, 1995.
44. S. Sen and Zhihong Zhou. Multistage stochastic decomposition: A bridge between stochastic programming and approximate dynamic programming. *SIAM Journal on Optimization*, 24(1):127–153, 2014.
45. A. Shapiro. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209:63–72, 2011.
46. A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on Stochastic Programming. Modeling and Theory*, volume 9 of *MPS-SIAM series on optimization*. SIAM and MPS, Philadelphia, 2009.
47. A. Shapiro, W. Tekaya, J. P. da Costa, and M. P. Soares. Risk neutral and risk averse stochastic dual dynamic programming method. *European Journal of Operational Research*, 224(2):375 – 391, 2013.
48. W. van Ackooij, W. de Oliveira, and Y. Song. An adaptive partition-based level decomposition for solving two-stage stochastic programs with fixed recourse. *Informs Journal on Computing*, 30(1):57–70, 2018.
49. W. van Ackooij, A. Frangioni, and W. de Oliveira. Inexact stabilized Benders' decomposition approaches: with application to chance-constrained problems with finite support. *Computational Optimization and Applications*, 65(3):637–669, 2016.
50. W. van Ackooij, N. Lebbe, and J. Malick. Regularized decomposition of large-scale block-structured robust optimization problems. *Computational Management Science*, 14(3):393–421, 2017.
51. J. Zou, S. Ahmed, and X.A. Sun. Nested decomposition of multistage stochastic integer programs with binary state variables. *Submitted for publication, preprint available at* `http://www.optimization-online.org/DB_FILE/2016/05/5436.html`, 2016.

A Proof of Lemma 1

(i) 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 $\varphi(x(\tau)) \leq \varphi(\tilde{x}) = \varphi^* < \infty$.
(ii) It follows from optimality of $x(\tau')$ and $x(\tau'')$ that

$$f(x(\tau')) + \varphi(x(\tau'))/\tau' \leq f(x(\tau'')) + \varphi(x(\tau''))/\tau' \tag{29}$$
$$f(x(\tau'')) + \varphi(x(\tau''))/\tau'' \leq f(x(\tau')) + \varphi(x(\tau'))/\tau''\,. \tag{30}$$

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

(iii) Notice that inequalities (29) and (30) hold strictly if φ is strongly convex and $x(\tau') \neq x(\tau'')$. The result thus follows from the same reasoning as before, but with strict inequalities instead.
(iv) Item i) 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.
(v) Let $x \in X$ such that $x \neq x(\tau)$. Then $\frac{1}{\tau}[\varphi(x(\tau)) - \varphi(x)] < f(x) - f(x(\tau))$. 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 $\min\{\varphi(x) \text{ s.t. } x \in X,\ f(x) \leq f(x(\tau))\}$.
(vi) Let $\tau_k \to \bar{\tau} > 0$ as $k \to \infty$. It follows from item iv) that $\{x(\tau_k)\}$ is a bounded sequence. Let $\bar{x} \in X$ be one of its cluster points, i.e., $\lim_K x(\tau_k) = \bar{x}$ for $K \subset \{1, 2, \cdots\}$. It follows from optimality of $x(\tau_k)$ that $f(x(\tau_k)) + \frac{1}{\tau_k}\varphi(x(\tau_k)) \leq f(x(\bar{\tau})) + \frac{1}{\tau_k}\varphi(x(\bar{\tau}))$ for all $k \in K$. By passing the limit superior with $k \in K$ we obtain $\limsup_K f(x(\tau_k)) + \frac{1}{\bar{\tau}} \limsup_K \varphi(x(\tau_k)) \leq f(x(\bar{\tau})) + \frac{1}{\bar{\tau}}\varphi(x(\bar{\tau}))$. By closeness of functions f and φ, it follows that $f(\bar{x}) \leq \liminf_K f(x(\tau_k))$ and $\varphi(\bar{x}) \leq \liminf_K \varphi(x(\tau_k))$, then

$$f(\bar{x}) + \frac{1}{\bar{\tau}}\varphi(\bar{x}) \leq \limsup_K f(x(\tau_k)) + \frac{1}{\bar{\tau}} \limsup_K \varphi(x(\tau_k)) \leq f(x(\bar{\tau})) + \frac{1}{\bar{\tau}}\varphi(x(\bar{\tau}))\,, \tag{31}$$

showing that $\bar{x} \in \operatorname{argmin}_{x \in X} f(x) + \frac{1}{\bar{\tau}}\varphi(x)$. As φ is strongly convex, the perturbed problem has a unique solution and, therefore, $\bar{x} = x(\bar{\tau})$, $\varphi(\bar{x}) = \lim_K \varphi(x(\tau_k))$ and $f(\bar{x}) = \lim_K f(x(\tau_k))$. We have shown that every subsequence of $\{x(\tau_k)\}$ tends to $x(\bar{\tau})$. As a result $x(\bar{\tau}) = \lim_{k\to\infty} x(\tau_k)$, $f(x(\bar{\tau})) = \lim_{k\to\infty} f(x(\tau_k))$ and $\varphi(x(\bar{\tau})) = \lim_{k\to\infty} \varphi(x(\tau_k))$.
(vii) Recall that $\varphi(x(\tau)) \leq \varphi(\tilde{x})$ from item i). The equivalences follow from item v) $\varphi(\tilde{x}) = \varphi(x(\tau)) \Leftrightarrow f(\tilde{x}) = f(x(\tau)) \Leftrightarrow x(\tau) \in X^*$, and the identity $x(\tau') = \tilde{x}$ for all $\tau' \geq \tau$ follows from item iii).
(viii) Let $\bar{x} \in X$ be a cluster point of the bounded sequence $\{x(\tau)\}$ (item iv)). Replace $x(\bar{\tau})$ in (31) by an arbitrary $x \in X$ and conclude (by taking $\bar{\tau} = \infty$) that $f(\bar{x}) \leq f(x)$, showing that $\bar{x}$ solves (10). By employing the same reasoning in the proof of item vi), we conclude that $f(\bar{x}) = \lim_{\tau\to\infty} f(x(\tau))$ and $\varphi(\bar{x}) = \lim_{\tau\to\infty} \varphi(x(\tau)) \leq \varphi(\tilde{x})$. Since $\bar{x} \in X^*$, the relation $\lim_{\tau\to\infty} x(\tau) = \lim_{\tau\to\infty} x(\tau) = \tilde{x}$ follows from the uniqueness of $\tilde{x}$ and continuity of $x(\tau)$.

□

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 Figure 3, 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 14500 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.

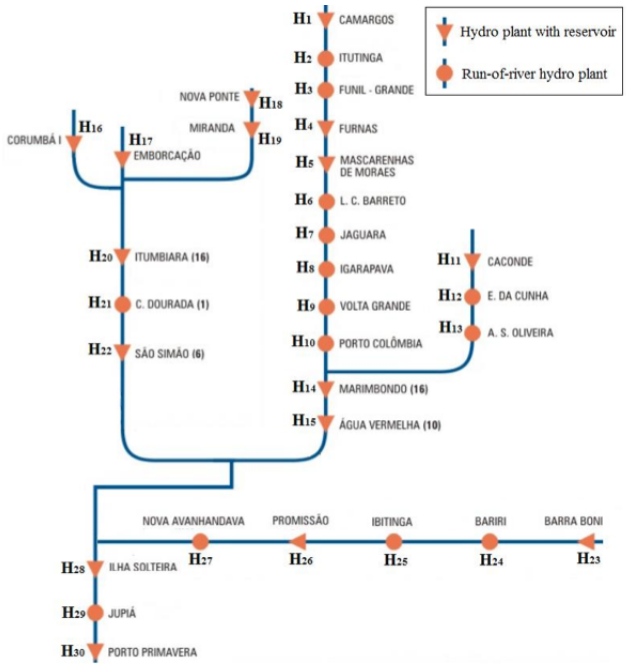


Fig. 3 The interconnected network structure of the hydrothermal power plants in Brazil.

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	$\bar{g}_f^t$ (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