Mathematical Programming manuscript No.
(will be inserted by the editor)

Stochastic Lipschitz Dynamic Programming

Shabbir Ahmed · Filipe Goulart Cabral · Bernardo Freitas Paulo da Costa

Received: date / Accepted: date

Abstract We propose a new algorithm for solving multistage stochastic mixed integer linear programming (MILP) problems with complete continuous recourse. In a similar way to cutting plane methods, we construct nonlinear Lipschitz cuts to build lower approximations for the non-convex cost-to-go functions. An example of such a class of cuts are those derived using Augmented Lagrangian Duality for MILPs. The family of Lipschitz cuts we use is MILP representable, so that the introduction of these cuts does not change the class of the original stochastic optimization problem.

We illustrate the application of this algorithm on two case studies, comparing our approach with the convex relaxation of the problems, for which we can apply SDDP, and for a discretized approximation, applying SDDiP.

Keywords Stochastic Optimization · Mixed-Integer Optimization · Lipschitz cuts · Augmented Lagrangian duality

The research of Shabbir Ahmed has been supported in part by the National Science Foundation grant 1633196 and the Office of Naval Research grant N00014-18-1-2075. The research of Bernardo Freitas Paulo da Costa has been supported in part by the COPPETec project IM-21780.

Shabbir Ahmed
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology, Atlanta, GA 30332
E-mail: shabbir.ahmed@isye.gatech.edu

Filipe Goulart Cabral
Operador Nacional do Sistema Elétrico
Rua Júlio do Carmo 251, Cidade Nova, Rio de Janeiro, RJ, Brazil
E-mail: filipegoulartcabral@poli.ufrj.br

Bernardo Freitas Paulo da Costa
Instituto de Matemática
Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brazil
E-mail: bernardofpc@im.ufrj.br

1 Introduction

Non-convex stochastic programming problems arise naturally in models that consider binary or integer variables, since such variables allow the representation of a wide variety of constraints at the cost of inducing non-convex feasible sets. Applications such as unit commitment [38], [11], [22], [1], optimal investment decisions [37], [8], and power system operational planning [7], [40], [19] have driven the development of new algorithms for problems that do not fit into the convex optimization framework.

For convex stochastic optimization problems, the theory and algorithms available are well-established, as can be seen for instance in reference books such as [4] and [35]. If there are not too many scenarios, one can consider the deterministic equivalent formulation, which includes all decision variables in a single convex optimization problem, that will be solved for optimality using a standard off-the-shelf solver. However, as the number of scenarios increases, the problem becomes too hard to solve in this fashion, and it is therefore necessary to develop special-purpose algorithms, usually based on decomposition, that leverage the special problem structure of stochastic optimization problems. Examples of such methods for multi-stage optimization are the Nested cutting plane algorithm [16], and Progressive Hedging [33].

These algorithms still require storing all parameters associated to each node of the scenario tree, which becomes prohibitive for larger planning horizons even with a modest number of scenarios per stage. The SDDP algorithm [31] and its variants are able to take advantage of structured stochastic processes, such as stagewise independent processes or Markovian processes, [36], [15]. This family of methods provides converging algorithms for multi-stage stochastic convex programs [14], including the case of risk-averse models [17].

Without convexity, the situation becomes more difficult. First, the deterministic equivalent becomes much harder to solve, shrinking the range of amenable problems for this approach. As for decomposition algorithms, they are less immediate, since several results rely on duality arguments which are not available in non-convex settings, or need much more complex objects such as subadditive dual functions for MILPs [2], [42], [20]. Algorithms such as Progressive Hedging [39], [24]; Lagrangian relaxation [6], [30], [5], [40]; and Branch and Cut [23] have been reported to produce good solutions for some classes of multistage stochastic MILP problems.

The development of the MIDAS [32] and the SDDiP [43] algorithms has pushed the limits of multi-stage stochastic optimization problems, proposing new methods for constructing lower bounds of value functions, which can be used in the nested form of dynamic programming. The MIDAS algorithm uses step functions to approximate monotone non-convex cost-to-go functions. The SDDiP algorithm relies on convex underapproximations, which are shown to be sufficient for convergence thanks to the binary discretization of the state variables and the tightness property of the Lagrangian cuts at binary states.

The aim of this paper is two-fold. First, we prove the convergence of a new algorithm for mixed integer multistage stochastic programming problems, which does not discretize the state variables, nor assumes monotonicity of the value functions. The second is to propose the use of non-linear, non-convex cuts in stochastic programming to capture non-convexities in the future cost function. Inspired by the exactness results from [13], we build non-convex underapproximations of the expected cost-to-go function, whose basic pieces are *augmented Lagrangian cuts*. They can be calculated from augmented Lagrangian duality, which has been shown in [13] to be exact for mixed-integer linear problems when the augmentation function is, for example, the L^1-norm. If the original problem already had pure binary state variables, then, as it was shown in [43], Lagrangian cuts are already tight, and we can use them in our algorithm by setting the non-linear term of the augmented Lagrangian to zero.

As we will see, even a countable number of augmented Lagrangian cuts cannot describe exactly the value function of a mixed-integer linear program, even when that function is continuous. However, the L^1 cuts have sufficient structure for our purposes, in two very important ways: first, they are Lipschitz functions, which still yield global estimates from local behaviour, although slightly weaker than linear bounds that are available when functions are convex. The Lipschitz estimates will be crucial for our convergence arguments, and the resulting algorithm is therefore called Stochastic Lipschitz Dynamic Programming (SLDP).

Second, it is possible to represent such L^1 cuts using binary variables and a system of linear equalities and inequalities. Therefore, we do not leave the class of mixed-integer linear problems when incorporating L^1-augmented Lagrangian cuts in each node/stage of the stochastic problem. Under some hypothesis for continuous recourse of each stage, it is possible to prove that the expected cost-to-go functions of stochastic mixed-integer linear problems are Lipschitz, and therefore our algorithm can be applied directly.

We have organized this paper as follows: the next section motivates the SLDP algorithm with a study of Lipschitz optimal value functions and a decomposition algorithm for deterministic Lipschitz optimization. Then, we present in section 3 the SLDP algorithm for both the full-tree and sampled scenario cases, and prove their convergence. Finally, we illustrate our results with a case study in section 4.

The authors would like to thank the anonymous reviewers for their suggestions and corrections to the manuscript.

2 Lipschitz value functions

Our SLDP algorithm uses Lipschitz cuts to approximate the cost-to-go function of a stochastic MILP in a similar fashion to the nested cutting planes algorithm for stochastic linear programs [34]. That is, we also compute lower approximations that iteratively improve the cost-to-go approximation in a neighborhood of the optimal solution.

The purpose of this section is to motivate the use of Lipschitz cuts for nonconvex functions, especially for optimal value functions of mixed integer problems. We start with the definition and basic properties of Lipschitz functions and introduce special Lipschitz cuts called *reverse norm cuts*. Then, we give in subsection 2.2 a convergence proof of an algorithm for Lipschitz optimization that employs reverse norm cuts. The idea of reverse norm cuts also appears in Global Lipschitz Optimization (GLO), see [27], [28] and more recently in [26]. However, our aim is not to develop a new algorithm for GLO but to explain the reverse norm cut algorithm and establish some results to extend it for stochastic multistage MILP programs.

In order to do so, we show in section 2.3 that MILP value functions can indeed be Lipschitz continuous. The essential requirement is that there is *continuous recourse* for any integer solution of the MILP problem, so that the value function does not become discontinuous.

Finally, we recall the definition of Augmented Lagrangian duality and the exactness results in [13] in subsection 2.4, and argue how they can be used to construct *augmented Lagrangian cuts* in place of the reverse norm cuts. This provides a unified framework, generalizing both nonlinear reverse norm cuts, from subsection 2.1, and the linear Lagrangian cuts, see [40] for the continuous and [43] for binary setting.

2.1 Lipschitz functions and reverse norm cuts

Let us recall the definition and some results for Lipschitz functions. We say that $f : \mathbb{R}^d \to \mathbb{R}$ is a Lipschitz function with constant $L > 0$ if for all $x, y \in \mathbb{R}^d$ we have $|f(x) - f(y)| \leq L\|x - y\|$. Note that the linear function $f(x) = a^\top x$ is Lipschitz with constant $L = \|a\|_*$, where $\|\cdot\|_*$ indicates the dual norm. Let f_1 and f_2 be Lipschitz functions with constants L_1 and L_2, respectively. It can be shown that

- the maximum and minimum of f_1 and f_2 are Lipschitz functions with constant $\max\{L_1, L_2\}$; and
- the sum of f_1 and f_2 is a Lipschitz function with constant $L_1 + L_2$.

Consider now an example of a non-convex Lipschitz function. Let f be the piecewise linear function defined on $[0, 3]$ by

$$f(x) = \begin{cases} x & : 0 \leq x \leq 1, \\ 1 & : 1 \leq x \leq 2, \\ 3 - x & : 2 \leq x \leq 3, \end{cases}$$

see figure 1 for an illustration. Note that f can be written as the minimum of linear functions, $f(x) = \min\{x, 1, 3 - x\}$, so f is a Lipschitz function with constant $L = 1$. It can also be seen from figure 1 that the tightest lower convex approximation g for f over $[0, 3]$ is the zero function. This implies that there will always be a gap between linear cuts and the original function f, since the

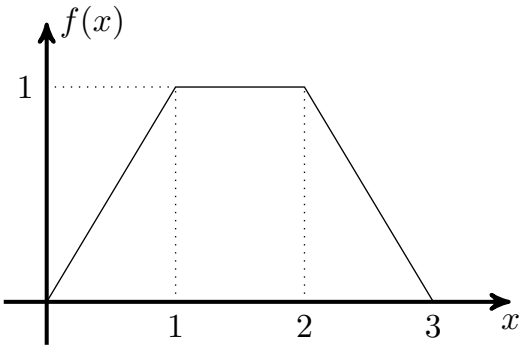


Fig. 1 Piecewise linear Lipschitz function.

best they could do is reproduce the convex relaxation g. In other words, there is no way to use lower linear cuts to close the gap with f, which motivates the introduction of reverse norm cuts.

Definition 1 A *reverse norm cut* for a function f, centered at $\overline{x}$ and with parameter ρ, is the function

$$C_{\rho,\overline{x}}(x) = f(\overline{x}) - \rho \cdot \|x - \overline{x}\|.$$

Note that $C_{\rho,\overline{x}}(x)$ is a Lipschitz function with constant ρ, and if $C_1(x)$ and $C_2(x)$ are reverse norm cuts with parameters ρ_1 and ρ_2 then $\max\{C_1, C_2\}(x)$ is a Lipschitz function with constant $\max\{\rho_1, \rho_2\}$.

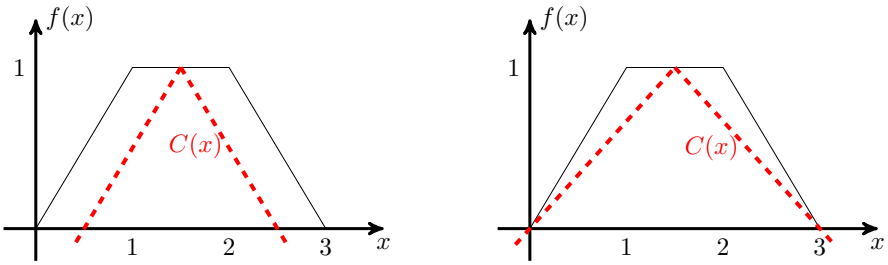
Suppose that f is a Lipschitz function with constant L. If ρ is greater than or equal to L, then all functions $C_{\rho,\overline{x}}(x)$ are valid reverse norm cuts, for any center point $\overline{x}$. Indeed,

$$\begin{aligned} C_{\rho,\overline{x}}(x) &= f(x) - f(x) + f(\overline{x}) - \rho \cdot \|x - \overline{x}\| \\ &\leq f(x) + L \cdot \|x - \overline{x}\| - \rho\|x - \overline{x}\| \\ &\leq f(x), \end{aligned} \tag{1}$$

for all $x \in \mathbb{R}^d$.

A fundamental difference between reverse norm cuts and cutting planes is that, in general, one cannot guarantee that a piecewise linear Lipschitz function f can always be represented as the maximum of a finite number of reverse norm cuts. Indeed, as can be seen in figure 2a, one would need all reverse norm cuts centered at every $\overline{x} \in [1, 2]$ to represent the non-convex function $f(x)$.

Sometimes, it is possible to obtain a reverse norm cut centered at a point $\overline{x}$ with Lipschitz constant ρ less than L, but such reverse norm cut may not be a lower approximation elsewhere if translated in the domain. In figure 2b, we show a tighter cut with $\rho = 2/3$ centered at $\overline{x} = 1.5$, but this lower ρ cannot be used for any other point in the domain of f. If the constant ρ is greater



(a) Reverse norm cut with $\rho = 1$. (b) Reverse norm cut with $\rho = 2/3$.

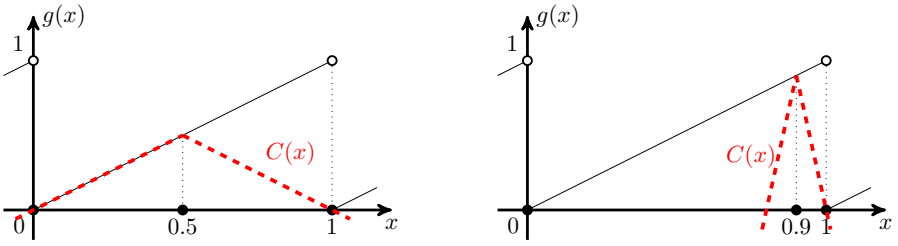
Fig. 2 Example of reverse norm cuts for a Lipschitz function.

than L, the corresponding reverse norm cut is a lower approximation of f if centered at any point $\overline{x}$ by the same deduction made in (1).

Finally, observe that if the function f is not Lipschitz then reverse norm cuts may need arbitrarily large parameters, depending on their center point. Indeed, let $g(x)$ be the fractional-part function:

$$g(x) = x - \lfloor x \rfloor = \min_{y \in \mathbb{Z}, y \leq x} x - y,$$

which is both piecewise linear and the optimal value function of a MILP, see figures 3a and 3b. As the point $\overline{x}$ approaches 1, the opening of the reverse norm



(a) Reverse norm cut with $\rho = 1$. (b) Reverse norm cut with $\rho = 9$.

Fig. 3 Example of reverse norm cuts for a non-Lipschitz function.

cuts centered at $\overline{x}$ goes to zero, so their corresponding Lipschitz constant L must go to infinity.

2.2 Optimization with reverse norm cuts

To develop the intuition for the SLDP method that will be presented in section 3, we introduce the reverse cut method for a deterministic problem and

its convergence proof. We hope that this simpler setting helps the reader understand some of the basic mechanisms of the proof without the notational burden of the general stochastic multistage setting.

The deterministic optimization problem that will be investigated is

$$\begin{aligned} \nu = \min_{x} \; & f(x) + g(x) \\ \text{s.t. } & x \in X, \end{aligned} \tag{2}$$

where $f(x)$ is a 'simple' function and $g(x)$ is a 'complex' one, so while it is relatively cheap to optimize f, we only assume that it is possible to evaluate g at given points. For example, this can be the stage problem in the nested decomposition of a multistage problem, where f is the immediate cost and g the expected cost-to-go function.

Assume that $g(x)$ is a Lipschitz function with constant L and X is a compact subset of $\mathbb{R}^d$. The reverse cut method, presented in Algorithm 1, similarly to decomposition methods, iteratively approaches $g(x)$ by reverse norm cuts centered at points obtained along the iterations of the algorithm. Those points are solutions of an approximated optimization problem obtained from (2), where $g(x)$ is replaced by the current Lipschitz approximation $g^k(x)$.

Algorithm 1 Reverse cut method

Input: lower bound $g^1 \equiv M$, cut constant $\rho \geq L$, and stopping tolerance $\varepsilon \geq 0$.
Output: optimal solution $x^k \in X$ and optimal value ν^k approximations, and Lipschitz lower estimate g^k.

Set initial iteration $k = 1$ and go to step 1.

1: Solve the following problem:

$$\begin{aligned} \nu^k = \min_{x} \; & f(x) + g^k(x) \\ \text{s.t. } & x \in X, \end{aligned} \tag{3}$$

and take an optimal solution $x^k \in \arg\min_{x \in X} f(x) + g^k(x)$.

2: Evaluate $g(x^k)$. Stop if $g(x^k) - g^k(x^k) \leq \varepsilon$. Otherwise, go to step 3.

3: Update the Lipschitz approximation, adding a reverse norm cut at x^k:

$$g^{k+1}(x) = \max\left\{g^k(x),\ g(x^k) - \rho \cdot \|x - x^k\|\right\}.$$

Set $k := k + 1$, and go to step 1.

The Lipschitz approximation $g^{k+1}(\cdot)$ is the maximum between the reverse norm cut centered at x^k and the previous approximation $g^k(\cdot)$:

$$g^{k+1}(x) = \max\left\{g^k(x),\ g(x^k) - \rho \cdot \|x - x^k\|\right\}.$$

By induction, all $g^k(\cdot)$ are Lipschitz functions with constant ρ, since it is the maximum of reverse norm cuts with that constant.

With this observation in mind, we will prove the convergence of Algorithm 1. The compactness assumption of X is used only to ensure the existence of a cluster point for the sequence of trial points $\{x^k\}_{k \in \mathbb{N}}$.

Lemma 1 *Suppose Algorithm 1 generates an infinite sequence of trial points* $\{x^k\}_{k\in\mathbb{N}}$. *Let* $x^* \in X$ *be any cluster point of this sequence, and let* $\mathcal{K}$ *be the indices of a subsequence that converges to* x^*. *Then* $\{g^k(x^k)\}_{k\in\mathcal{K}}$ *converges to* $g(x^*)$*:*

$$\lim_{k\in\mathcal{K}} g^k(x^k) = g(x^*).$$

Proof We will bound $|g(x^*)-g^k(x^k)|$ by the distance between the subsequence $\{x^k\}_{k\in\mathcal{K}}$ and the cluster point x^*. Using the triangular inequality and the Lipschitz definition, we get

$$\begin{aligned}\left|g(x^*) - g^k(x^k)\right| &\leq \left|g(x^*) - g(x^k)\right| + \left|g(x^k) - g^k(x^k)\right| \\ &\leq L \cdot \|x^* - x^k\| + \left|g(x^k) - g^k(x^k)\right|. \end{aligned} \tag{4}$$

Since x^k converges to x^* along $k \in \mathcal{K}$, we only need to prove that the lower bounds g^k will get arbitrarily close to g at the trial points, before it is updated. By construction of the reverse cut method, $g^k(x^k)$ is less than or equal to $g(x^k)$, so we only need upper bounds. Let j be the index just before k in the subsequence $\mathcal{K}$. Since g^k is ρ-Lipschitz,

$$g^k(x^k) \geq g^k(x^j) - \rho \cdot \|x^k - x^j\|.$$

But also by construction, $g^k(x^j) = g(x^j)$, so subtracting the equation above from $g(x^k)$ and applying once again the Lipschitz hypothesis on g we obtain:

$$\begin{aligned} g(x^k) - g^k(x^k) &\leq g(x^k) - g(x^j) + \rho \cdot \|x^k - x^j\| \\ &\leq L \cdot \|x^k - x^j\| + \rho \cdot \|x^k - x^j\|. \end{aligned} \tag{5}$$

By replacing (5) in (4) and taking the limit over $\mathcal{K}$, we conclude that the sequence $\{g^k(x^k)\}_{k\in\mathcal{K}}$ converges to $g(x^*)$. □

From the convergence result above, we now separate the analysis of the cases where ε is strictly positive and zero respectively in Corollary 1 and Theorem 1 below.

Corollary 1 *For any stopping tolerance* $\varepsilon > 0$, *Algorithm 1 stops in a finite number of iterations with an* ε*-optimal solution for* (2).

Proof From inequality (5) in the proof of Lemma 1 and compactness of X, Algorithm 1 will stop in a finite number of iterations if the stopping tolerance ε is strictly positive. Using the fact that $x^k \in X$ is a feasible solution to (2) and optimal solution to (3) such that $g(x^k) - g^k(x^k) \leq \varepsilon$, where that g^k is a lower estimate for g, we have the following inequalities:

$$\nu^k \leq \nu \leq f(x^k) + g(x^k) \leq f(x^k) + g^k(x^k) + [g(x^k) - g^k(x^k)] \leq \nu^k + \varepsilon,$$

which means that $f(x^k) + g(x^k)$ is bounded between ν and $\nu + \varepsilon$. □

Theorem 1 *Consider the stopping tolerance ε equal to zero. Then, Algorithm 1 stops with an optimal solution in a finite number of iterations or it generates a sequence of optimal value approximations $\{\nu^k\}_{k\in\mathbb{N}}$ that converges to the optimal value ν of problem* (2) *and every cluster point x^* of the sequence $\{x^k\}_{k\in\mathbb{N}}$ is a minimizer of* (2).

Proof Suppose that Algorithm 1 stops after a finite number of iterations. Using the same argument of Corollary 1, we obtain that the last trial point x^k is the optimal solution to (2).

Now, suppose that Algorithm 1 never reaches the stopping condition, so we know from Lemma 1 that the sequence $\{g^k(x^k)\}_{k\in\mathcal{K}}$ converges to $g(x^*)$. Moreover, x^k is a feasible solution to the main problem (2) and optimal solution to the approximate problem (3). Since g^k is a lower estimate for g, we obtain the following relationships:

$$f(x^k) + g(x^k) \geq \nu \geq \nu_k = f(x^k) + g^k(x^k). \tag{6}$$

Taking the limit over $\mathcal{K}$ on both sides of (6), and by continuity of f and g, we obtain:

$$f(x^*) + g(x^*) \geq \nu \geq \lim_{k\in\mathcal{K}} \nu_k = f(x^*) + g(x^*),$$

which shows that all inequalities above are equalities, and therefore x^* is an optimal solution to (2).

Going back to the full sequence, we recall that the sequence of objective functions $\{f(x) + g^k(x)\}_{k\in\mathbb{N}}$ is monotone nondecreasing, so the sequence of optimal values $\{\nu_k\}_{k\in\mathbb{N}}$ is also monotone and nondecreasing, which implies that $\{\nu_k\}_{k\in\mathbb{N}}$ also converges to ν. □

Observe that the proof of Lemma 1, Corollary 1 and Theorem 1 use only two properties of the reverse-norm cuts. Besides their Lipschitz character, they are exact at trial points, that is, $C_{\overline{x}}(\overline{x}) = g(\overline{x})$. Therefore, any other way of producing *uniformly Lipschitz tight cuts* for g yields a convergent Lipschitz cut method for optimizing $f + g$.

One such method for constructing tight Lipschitz cuts comes from augmented Lagrangian duality, which can be used for example if g is the value function of an optimization problem. Before explaining this in subsection 2.4, we characterize in the next section a subclass of MILPs whose value functions are Lipschitz.

2.3 MILPs with Lipschitz value functions

As we have seen, the Lipschitz property of the function g was an important piece in the proof of convergence of the reverse-norm method. It will also be fundamental in the analysis of the SLDP algorithm, which we employ to solve stochastic multi-stage optimization problems. In general, the optimal value function of a MILP need not be Lipschitz, already for simple mixed-integer

linear problems. Therefore, we present in this section a sufficient condition that guarantees Lipschitz continuity for the optimal value function of MILPs.

First, we show that the optimal value function for a Lipschitz objective over a family of polyhedra is still Lipschitz. Then, by enumerating the polyhedra over the realizations of integer variables, we will arrive at the *complete continuous recourse* (CCR) condition that ensures the optimal value function of a Lipschitz function over a family of mixed-integer domains is Lipschitz. These results are very similar to the cases of optimal value functions of linear problems and mixed-integer linear problems; in those cases, the objective function is linear, and not a general Lipschitz function, but the proofs are closely related.

Consider the parameterized optimization problem

$$\begin{aligned} \nu(b) = \min_x \ & f(x) \\ \text{s.t. } & (x,b) \in P \end{aligned} \tag{7}$$

for a Lipschitz function f and a polyhedron P. In order to analyze the function $\nu(b)$, we have to understand the effect of the variations on the feasible set $P(b) = \{\, x \mid (x,b) \in P \,\}$ with respect to the parameter b, and compound this effect with the Lipschitz function f. For the first part, the Hoffman Lemma below provides the answer, bounding the distance between two non-empty polyhedra $P(b)$ and $P(b^*)$ by a term proportional to the norm $\|b - b^*\|$. This result resembles a version for sets of the Lipschitz continuity definition.

Lemma 2 (Hoffman lemma [35]) *Let $S(b)$ be a polyhedron parameterized by the right-hand side vector $b \in \mathbb{R}^m$ of a given linear system, that is, $S(b) = \{x \in \mathbb{R}^d \mid Ax \leq b\}$. Let $b^* \in \mathbb{R}^m$ be a vector such that $S(b^*)$ is nonempty. Then, there exists $r > 0$ depending only on A such that*

$$S(b) \subseteq S(b^*) + r \cdot \|b - b^*\| \cdot \Delta,$$

where Δ is the unit ball, i.e., $\Delta = \{\varepsilon \in \mathbb{R}^d \mid \|\varepsilon\| \leq 1\}$.

With this, we can guarantee that the optimal value function of a minimization problem of a Lipschitz function over a given polyhedron is also Lipschitz in its essential domain.

Theorem 2 (Lipschitz cost-to-go functions) *Consider a polyhedron P in $\mathbb{R}^{d+m}$, and a Lipschitz continuous function f with constant L. Let ν be the optimal value function defined by*

$$\begin{aligned} \nu(b) = \min \ & f(x) \\ s.t. \ & (x,b) \in P, \end{aligned} \tag{8}$$

and assume that there is one value of b such that $\nu(b)$ is finite.

Then, the essential domain of ν, $\mathrm{dom}(\nu)$, is a polyhedron, and the function ν restricted to $\mathrm{dom}(\nu)$ is Lipschitz continuous with constant $L \cdot \widetilde{r}$, where $\widetilde{r}$ is a constant that depends only on P.

Proof First, we prove that $\mathrm{dom}(\nu)$ is a polyhedron. Recall that $\mathrm{dom}(\nu)$ is the set of vectors $b \in \mathbb{R}^m$ for which the problem (8) is feasible, that is, $\mathrm{dom}(\nu) = \left\{b \in \mathbb{R}^m \;\middle|\; \exists x \in \mathbb{R}^d; (x,b) \in P,\ f(x) < +\infty\right\}$. Since f is continuous, f does not assume $+\infty$ anywhere, so $\mathrm{dom}(\nu)$ is the projection of P over the component b:

$$\mathrm{dom}(\nu) = \mathrm{proj}_b(P).$$

As the image of a polyhedron by a linear map is also a polyhedron, we conclude that $\mathrm{dom}(\nu)$ is a polyhedron in $\mathbb{R}^m$.

Now, we need to prove that ν is Lipschitz continuous over $\mathrm{dom}(\nu)$. Denote by $Wx + Tb \leq h$ the linear constraint that defines P, that is, $P = \{(x,b) \in \mathbb{R}^{d+m} \mid Wx+Tb \leq h\}$, and let $S(u)$ be the set given by the linear system $Wx \leq u$. Now, let b_1 and b_2 be two points in the domain of ν. Taking a feasible point $(x_1, b_1) \in P$ for the problem defined by b_1, and applying the Hoffman Lemma for $u := h - Tb_2$, we get that there is a feasible point $(x_2, b_2) \in P$ such that

$$\|x_2 - x_1\| \leq r\|(h - Tb_2) - (h - Tb_1)\| \leq r\|T\| \cdot \|b_2 - b_1\|.$$

Therefore, by the Lipschitz hypothesis on f,

$$\nu(b_2) \leq f(x_2) \leq f(x_1) + Lr\|T\| \cdot \|b_2 - b_1\|.$$

Taking the infimum over $x_1 \in P(b_1)$, we see that $\nu(b_2) \leq \nu(b_1) + Lr\|T\| \cdot \|b_2 - b_1\|$. If $\nu(b_2)$ is a point where ν is finite, this shows that $\nu(b_1) > -\infty$, so ν never assumes the value $-\infty$.

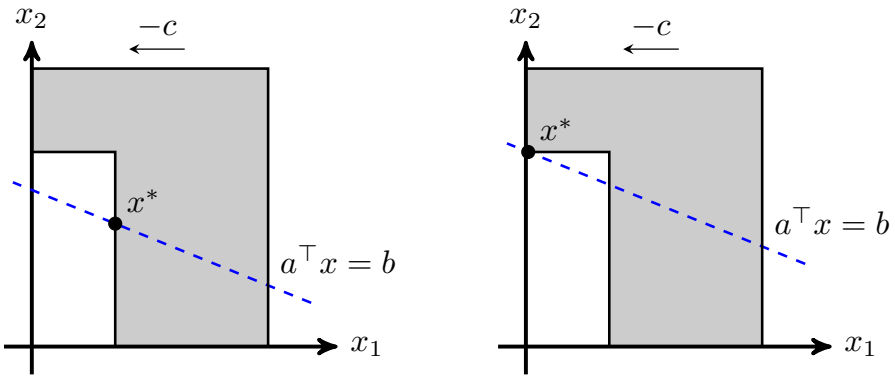
Finally, by symmetry, we obtain $|\nu(b_2) - \nu(b_1)| \leq Lr\|T\| \cdot \|b_2 - b_1\|$, which is the Lipschitz condition for ν. □

To handle the mixed-integer case, we split the optimization variable $x = (y, z)$ over the integer z and continuous variables y to obtain

$$\nu(b) = \min_z \min_{y \in P(b,z)} f(y,z). \tag{9}$$

Since each function $\nu_z(b) = \min_{y \in P(b,z)} f(y,z)$ is Lipschitz continuous on its domain by Theorem 2 above, we only need to understand their minimum. The main difficulty is that the domains of each ν_z may be different.

Indeed, we illustrate in figures 4a and 4b an example of a discontinuous optimal value function induced by the problem of minimizing a linear objective function over the union of two polyhedra. In this particular example, the feasible set is the intersection between the blue dashed line and the union of both vertical and horizontal rectangles, while the objective function is a linear function that decreases as the solution candidate moves to the left. We show in figures 4a and 4b the optimal solution of this problem for two different right-hand side parameters, which control the height of the dashed line. As that parameter changes, the dashed line moves up or down, and the optimal solution changes abruptly as soon as a point in the horizontal rectangle becomes feasible, as occurs in figure 4b. This shows that the optimal value function is discontinuous, so it cannot be Lipschitz continuous.



(a) Optimal solution with given b. (b) Discontinuous effect in b.

Fig. 4 Minimum over union of polyhedra can be discontinous.

On the other hand, we know that we can represent this MILP problem using only one binary variable, and if we fix the binary solution z to 0 or 1 the corresponding optimal value functions $\nu_0(b)$ and $\nu_1(b)$ of the continuous variable problem are Lipschitz continuous by Theorem 2. Since the optimal value function of the MILP problem $\nu(b)$ is the minimum of $\nu_0(b)$ and $\nu_1(b)$, our intuition says that $\nu(b)$ should be Lipschitz continuous as well. However, $\nu_0(b)$ and $\nu_1(b)$ are not Lipschitz continuous in the entire space, *only in their domains*, so we can only guarantee that $\nu(b)$ is Lipschitz in the intersection of the domains $\mathrm{dom}(\nu_0) \cap \mathrm{dom}(\nu_1)$, see figure 5 for an illustration. Without loss of generality, we can assume that the angular coefficient a_2 of the dashed line $a_1x_1 + a_2x_2 = b$ is equal to 1, and this allows us to interpret the domain of ν_i as the interval defined by all the intersections of the dashed line with the x_2-axis, for $i = 1, 2$.

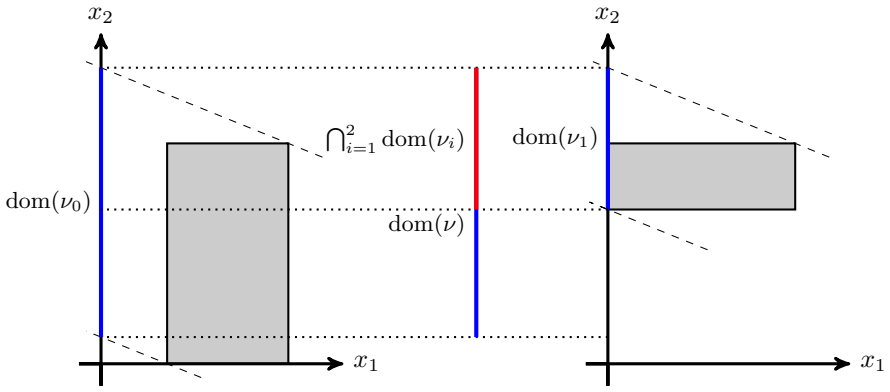


Fig. 5 Domain intersection.

Therefore, in order to guarantee the Lipschitz condition for the optimal value function, we assume that all functions $\nu_z(b)$ are defined for all $b \in \mathbb{R}^m$. This is called the *Complete Continuous Recourse* (CCR) condition since, for all integer z, there exists a feasible continuous y such that $(y, z, b) \in P$, compare [43, Definition 1].

If there are a finite number of feasible z's, say if the z's are binary variables, and the problem satisfies the CCR condition, then $\nu(b) = \min_z \nu_z(b)$ is a Lipschitz function *over the entire* $\mathbb{R}^m$, as desired.

If, however, the set of feasible z's is infinite, and the infimum over z does not degenerate to $-\infty$, the infimum remains a Lipschitz function in $\mathbb{R}^m$, but this infimum may not be attained. Indeed, if problem (9) satisfies the CCR condition and $\nu(b)$ is finite everywhere then each function $\nu_z(b)$ is Lipschitz continuous with the same constant L, by Theorem 2, and for any $b_1, b_2 \in \mathbb{R}^m$ we have that

$$\begin{aligned}
\nu(b_1) - \nu(b_2) &= \inf_{w,z \in \mathbb{Z}^k} [\nu_w(b_1) - \nu_z(b_2)] \\
&\leq \inf_{\substack{w=z \\ w,z \in \mathbb{Z}^k}} [\nu_w(b_1) - \nu_z(b_2)] \\
&\leq \nu_z(b_1) - \nu_z(b_2) \leq L \cdot \|b_1 - b_2\|, \quad \forall z \in \mathbb{Z}^k.
\end{aligned}$$

By exchanging the arguments b_1 and b_2 and repeating the same steps, we conclude that $\nu(\cdot)$ is Lipschitz continuous with constant L.

If the Lipschitz function f in (9) is MILP-representable with rational coefficients, such as the expected cost-to-go function of a stochastic MILP with rational data, and the polyhedron P is rational, then Meyer's theorem [29, 9] ensures that the infimum in z is attained.

When one cannot guarantee the CCR condition, it is possible to *approximate* the stochastic optimization problem by an appropriate relaxation. This can be done, for example, by adding continuous variables t and *penalizing them*, so that the problem becomes

$$\begin{aligned}
\nu_i(b) = \min_{y,t} \; & f(y) + M\|t\|_1 \\
\text{s.t. } & (y, b - t) \in P_i.
\end{aligned} \tag{10}$$

This is similar in spirit to the suggestion in [43]; it provides a Lipschitz relaxation of the original function ν_i, which is defined by construction over all of $\mathbb{R}^m$. The larger the constant M, the closer this relaxation will be to the original problem, but also the larger the Lipschitz constant L of the value function will become. Therefore, one must choose also a larger cut constant ρ, which might slow down the convergence of the reverse cut method, as suggested by the presence of the Lipschitz constants L and ρ in equation 5 for the convergence in the deterministic case.

2.4 Augmented Lagrangian cuts

We start recalling the definition of augmented Lagrangian duality for mixed-integer optimization problems. Let

$$\begin{aligned} g(b) = \min_{x \in X} \; & c^\top x \\ \text{s.t. } & Ax = b \end{aligned} \tag{11}$$

be a parameterized linear problem, where X describes the mixed-integer constraints.

Definition 2 Given an augmenting function ψ, which is non-negative and satisfies $\psi(0) = 0$, the *augmented Lagrangian* for problem (11) is given by

$$g^{AL}(b; \lambda, \rho) = \min_{x \in X} c^\top x - \lambda^\top (Ax - b) + \rho \cdot \psi(Ax - b). \tag{12}$$

Since any feasible solution to the original optimization problem (11) remains feasible with the same objective value for the augmented Lagrangian (12), we see that, for any b, λ and $\rho \geq 0$,

$$g^{AL}(b; \lambda, \rho) \leq g(b). \tag{13}$$

Moreover, on the MILP setting, we have exact duality [13, Theorem 4, p. 381] if the augmenting function ψ is a *norm*. More precisely:

Theorem 3 *If the set X is a rational polyhedron with integer constraints, if the problem data A, b_0 and c are rational and if for the given b_0 the problem is feasible with bounded value $g(b_0)$, then there is a finite ρ^* such that*

$$\sup_{\lambda} g^{AL}(b_0; \lambda, \rho^*) = g(b_0).$$

In addition, one can choose a finite *Lagrange multiplier λ^* that attains the supremum.*

This motivates the introduction of *augmented Lagrangian cuts* using norms as the augmentation function. Indeed, expanding the definition of g^{AL} in the weak duality equation (13), we get for any b:

$$\begin{aligned} g(b) &\geq g^{AL}(b; \lambda, \rho) \\ &= \min_{x \in X} \quad c^\top x - \lambda^\top (Ax - b) + \rho \|Ax - b\| \\ &= \min_{x \in X} \quad c^\top x - \lambda^\top (Ax - b_0 + b_0 - b) + \rho \|Ax - b_0 + b_0 - b\|. \end{aligned}$$

By the triangular inequality,

$$\|Ax - b\| \geq \|Ax - b_0\| - \|b - b_0\|,$$

so that

$$\begin{aligned} g(b) &\geq \min_{x \in X} \quad c^\top x - \lambda^\top (Ax - b_0) + \lambda^\top (b_0 - b) + \rho \|Ax - b_0\| - \rho \|b_0 - b\| \\ &= g^{AL}(b_0; \lambda, \rho) + \lambda^\top (b_0 - b) - \rho \|b_0 - b\|. \end{aligned}$$

Therefore, calculating the augmented Lagrangian at b_0 provides a lower Lipschitz estimate for $g(b)$:

Definition 3 If g is the optimal value function for a MILP, and given b_0, λ and $\rho > 0$, the *augmented Lagrangian cut* centered at b_0 is the function

$$ALC_{b_0,\lambda,\rho}(b) := g^{AL}(b_0;\lambda,\rho) + \lambda^\top(b_0 - b) - \rho\|b_0 - b\|.$$

Moreover, the exactness result above shows that there exists a sufficiently large ρ and appropriate λ for which $g^{AL}(b_0;\lambda,\rho) = g(b_0)$, so the cut is *tight* at b_0. However, this proof stills leaves open the question of whether the family of such cuts is *uniformly* Lipschitz, since as seen in the fractional-part example in figure 3 one might need arbitrarily large ρ near discontinuities of the value function.

The Lipschitz setting that we assume for the value function allows us to bypass this difficulty: choosing $\rho = Lip(g)$ and $\lambda = 0$ produces a valid and tight cut, so this provides an absolute upper bound on the needed ρ for exact augmented Lagrangian duality.

2.5 Representability of Augmented Lagrangian cuts

Using Augmented Lagrangian cuts instead of reverse-norm cuts as done in section 2.2 allows for additional flexibility. In both cases, each subproblem solved is, in principle, a nonlinear, non convex and non differentiable optimization problem, which can be computationally intractable even for a small number of iterations. However, if the norm $\|\cdot\|$ is the ℓ_1 or ℓ_∞ norm, and the domain of b is compact, for example a box $[-M, M]^n$, then the *reverse norm* function $-\|\cdot\|$ is mixed-integer linear representable. Indeed, the function $-\|b\|_1 = -\sum_{i=1}^n |b_i|$ is not convex, but it can described as the optimal value of the following MILP problem

$$\begin{aligned}
-\|b\|_1 = \min_{\gamma,y,z}\ & \gamma \\
\text{s.t. } & -\textstyle\sum_{i=1}^n (y_i^+ + y_i^-) \le \gamma, \\
& y_i^+ - y_i^- = b_i, && i = 1,\dots,n, \\
& 0 \le y_i^+ \le z_i \cdot M_i, && i = 1,\dots,n, \\
& 0 \le y_i^- \le (1 - z_i) \cdot M_i, && i = 1,\dots,n, \\
& y \in \mathbb{R}^n,\ \gamma \in \mathbb{R},\ z \in \{0,1\}^n.
\end{aligned} \tag{14}$$

We can represent $-\|b\|_\infty$ analogously if we replace the inequality

$$-\textstyle\sum_{i=1}^n (y_i^+ + y_i^-) \le \gamma$$

by the set of constraints

$$-(y_i^+ + y_i^-) \le \gamma \qquad \text{for all } i = 1,\dots,n.$$

Additionally, the epigraph of the minus ℓ_p-norm, $-\|b\|_p$, is strongly non-convex [25] if p is different from 1 or ∞. Moreover, according to the midpoint

lemma from [25], the reverse ℓ_p-norm function is not mixed-integer convex representable, that is, it is impossible to represent the epigraph of $-\|b\|_p$ using a finite number of binary variables and convex constraints if p is not equal to 1 or ∞. Therefore, these norms are the natural ones for using in Stochastic multistage mixed-integer optimization, since their use does not change the class of problems we are dealing with.

3 The Stochastic Lipschitz Dynamic Programming algorithm

In this section, we present the Stochastic Lipschitz Dynamic Programming (SLDP) algorithm for the multistage case in two different approaches: the full scenario and the sampled scenario case. In the full scenario approach, all the nodes are visited in the forward and backward steps, and Lipschitz cuts are added for every expected cost-to-go function. In contrast, in the sampled scenario approach, just the sampled scenarios are visited in the forward and backward steps, and Lipschitz cuts are added only for the expected cost-to-go functions of the sampled nodes. We prove convergence to an optimal policy for the full scenario case, and we introduce an additional procedure in the sampled scenario case to ensure convergence towards an ε-optimal policy.

3.1 Multistage setting and Lipschitz continuity of cost-to-go functions

For dealing with the stochastic multistage setting, we fix some notation for describing the scenario tree. Let $\mathcal{N}$ be the set of nodes, where 1 is the root node, and $a : \mathcal{N}\backslash\{1\} \to \mathcal{N}$ is the ancestor function that associates each node n except the root to its ancestor node $a(n) \in \mathcal{N}$. We also define the set $\mathcal{S}(n)$ of successor nodes as the set of nodes with ancestor n, that is, $\mathcal{S}(n) = \{m \in \mathbb{N} \mid a(m) = n\}$, and for each successor m there is a transition probability denoted by q_{nm} from node n to m. From 1 and $\mathcal{S}$, we define the *stage* $t(n)$ of a node $n \in \mathbb{N}$: the root node belongs to stage 1, and inductively the nodes in $S(n)$ belong to the stage $t(n) + 1$. The set of all nodes in stage τ is denoted by $\mathcal{N}_\tau$.

In the dynamic programming formulation of stochastic optimization, we have a state variable x_n and a mixed integer control variable y_n, ranging over a feasible set X_n and incurring an immediate cost $f_n(x_n, y_n)$. As it is standard, we introduce a copy variable z_n that carries the information from the previous state, so that the cost-to-go and expected cost-to-go functions $Q_n(\cdot)$ and $\overline{Q}_n(\cdot)$ of each node $n \in \mathcal{N}\backslash\{1\}$ satisfy the following recursive relationship:

$$\begin{aligned} Q_n(x_{a(n)}) = \min_{x_n, y_n, z_n} \; & f_n(x_n, y_n) + \overline{Q}_n(x_n) \\ \text{s.t. } & (x_n, y_n, z_n) \in X_n, \\ & z_n = x_{a(n)}, \end{aligned} \tag{15}$$

$$\overline{Q}_n(x_n) = \begin{cases} 0 & \text{if } \mathcal{S}(n) = \emptyset, \\ \sum_{m \in \mathcal{S}(n)} q_{nm} \cdot Q_m(x_n) & \text{otherwise.} \end{cases} \tag{16}$$

The nodes n without successor are called *leaf* nodes of the tree, and they correspond to the last decision to be taken in the planning horizon. Also, observe that the root node $1 \in \mathcal{N}$ does not have an ancestor, so we can still define $\overline{Q}_1(x_1)$ by (16), but its stage problem (15) should be written as

$$\begin{aligned} Q_1 = \min_{x_1, y_1} \; & f_1(x_1, y_1) + \overline{Q}_1(x_1) \\ \text{s.t. } & (x_1, y_1) \in X_1. \end{aligned}$$

However, in order to avoid having to single out this special case, we slightly abuse notation by fixing $0 = a(1)$, $x_0 = 0$, and extend X_1 to a further dimension z_1.

From our discussion in the previous section, we assume that f_n is Lipschitz continuous with constant L_n and that X_n satisfies the complete continuous recourse condition. Under those hypothesis, both the cost-to-go functions $Q_n(\cdot)$, and the expected cost-to-go functions $\overline{Q}_n(\cdot)$ are Lipschitz continuous.

Proposition 1 (Stochastic multistage mixed-integer programs) *Consider the stochastic multistage optimization program defined by* (15) *and suppose that for every node $n \in \mathcal{N}$ the cost-to-go function Q_n is not equal to $-\infty$ in any point, i.e., $Q_n(\cdot) > -\infty$, and the CCR condition holds for the feasible set X_n. Then, the expected cost-to-go function $\overline{Q}_n(\cdot)$ is Lipschitz continuous in $\mathbb{R}^{d_n}$ with Lipschitz constant at most*

$$\widetilde{L}_n = \begin{cases} 0 & \text{, if } \mathcal{S}(n) = \emptyset, \\ \sum_{m \in \mathcal{S}(n)} q_{nm} \cdot \left(L_m + \widetilde{L}_m\right) \cdot r_m & \text{, otherwise} \end{cases} \tag{17}$$

where r_m is a constant that depends only on X_m.

Proof We proceed by backward induction on the scenario tree. For the leaf nodes, the statement holds by definition, since $\overline{Q}_n$ is identically zero.

So, suppose this result holds for all successor nodes $m \in \mathcal{S}(n)$, and let's prove that it also holds for node n. By the induction hypothesis, the expected cost-to-go functions $\overline{Q}_m$ are Lipschitz with constant $\widetilde{L}_m$, and from Theorem 2 each Q_m is Lipschitz with constant $(L_m + \widetilde{L}_m) \cdot r_m$, where L_m is the Lipschitz constant of the objective function f_m and r_m is a constant from the Hoffman Lemma (Lemma 2, page 10) that only depends on X_m. Since the expected value of Lipschitz functions is also Lipschitz with constant equal to the expected value constant, the induction step is proved. □

Since problem (15) for each node admits the Lipschitz decomposable structure of (2), one could imagine using the reverse-norm method of Algorithm 1, or augmented Lagrangian cuts, to approximate its solution. However, in the multistage case we lack one fundamental property we used, namely that we can compute exactly the 'complex' function $g(x)$, which in this case is $\overline{Q}_n(x_n)$. Indeed, we are only able to produce lower approximations for it, and the next sections will deal with the necessary estimates to prove convergence under this weaker hypothesis.

3.2 Approximating the value functions

Before we present the SLDP algorithm, we need to introduce some notation for the approximations along the iterations of the algorithm. We start with a notation for Lipschitz cuts:

Definition 4 For any function g, we denote by LC a *Lipschitz cut* for g, that is, LC is a Lipschitz function that is a global lower bound for g.

They are a natural generalization of both reverse-norm cuts and augmented Lagrangian cuts introduced in the previous section, and our convergence results are valid for whichever family of Lipschitz cuts used.

As usual, we denote by $\overline{\Omega}_n^k(x_n)$ the expected cost-to-go approximation induced by Lipschitz cuts at iteration k. These will be the analogs of the approximations g^k of the Lipschitz function g from Lemma 1. For the purpose of convergence analysis, we consider the approximations $Q_n^k(x_{a(n)})$ and $\overline{Q}_n^k(x_{a(n)})$ of the cost-to-go and expected cost-to-go functions at iteration k defined below:

$$\begin{aligned} Q_n^k(x_{a(n)}) = \min_{x_n, y_n, z_n} \; & f_n(x_n, y_n) + \overline{\Omega}_n^k(x_n) \\ \text{s.t. } & (x_n, y_n, z_n) \in X_n, \\ & z_n = x_{a(n)}, \end{aligned} \tag{18}$$

$$\overline{Q}_n^k(x_n) = \begin{cases} 0 & \text{if } \mathcal{S}(n) = \emptyset, \\ \sum_{m \in \mathcal{S}(n)} q_{nm} \cdot Q_m^k(x_n) & \text{otherwise} \end{cases} \tag{19}$$

We assume we are given, for the first iteration, a Lipschitz lower approximation $\overline{\Omega}_n^1$ of the expected cost-to-go function $\overline{Q}_n$ for each node n in the tree. In practice, the first expected cost-to-go approximations are identically zero, since costs are usually non-negative.

To update the expected cost-to-go approximation $\overline{\Omega}_n^k$ of iteration k at a given point x_n^k, we calculate a Lipschitz cut $\overline{\mathsf{LC}}_n^k$ that satisfies

$$\begin{cases} \overline{\mathsf{LC}}_n^k(x_n^k) = \overline{Q}_n^{k+1}(x_n^k) \\ \overline{\mathsf{LC}}_n^k(x_n) \le \overline{Q}_n^{k+1}(x_n), \; \forall x_n, \end{cases} \tag{20}$$

where the Lipschitz constant of $\overline{\mathsf{LC}}_n^k$ is upper bounded by ρ_n, for all the iterations k. Then, we define the expected cost-to-go approximation of iteration $k+1$ as

$$\overline{\Omega}_n^{k+1}(x_n) := \max\left\{\overline{\Omega}_n^k(x_n), \; \overline{\mathsf{LC}}_n^k(x_n)\right\}. \tag{21}$$

Note that we have used $\overline{Q}_n^{k+1}(x_n^k)$ instead of $\overline{Q}_n^k(x_n^k)$ in the cut update (20) because the Lipschitz cuts of the SLDP algorithm are updated from the last to the first stage, so all expected cost-to-go approximations $\overline{\Omega}_m^k(x_m)$ of the successor nodes m of n are updated to $\overline{\Omega}_m^{k+1}(x_m)$ before the computation of the optimal value (18) with state $x_{a(m)} = x_n^k$.

Given the scenario decomposition for evaluating $\overline{Q}_n^{k+1}(x_n^k)$ indicated by equation (19), it is possible to obtain a Lipschitz cut for each Q_m^{k+1} at the same state x_n^k. Using Lipschitz cuts that satisfy conditions analogous to (20), that is,

$$\begin{cases} \mathsf{LC}_m^k(x_n^k) = Q_m^{k+1}(x_n^k) \\ \mathsf{LC}_m^k(x_n) \leq Q_m^{k+1}(x_n), \ \forall x_n, \end{cases} \tag{22}$$

it follows that their average $\overline{\mathsf{LC}}_n^k := \sum_{m \in \mathcal{S}(n)} q_{nm} \cdot \mathsf{LC}_m^k$ will satisfy (20).

Given this level of generality, the exact details of the algorithm will depend on the method used for calculating the Lipschitz cuts $\overline{\mathsf{LC}}_n$. For concreteness, the reader could keep in mind the reverse-norm setting, that is, where the Lipschitz cut $\overline{\mathsf{LC}}_n^k$ is given by evaluating each Q_m^{k+1} at the incoming state x_n^k, then forming the average

$$v_n^k := \overline{Q}_n^{k+1}(x_n^k) = \sum_{m \in \mathcal{S}(n)} q_{nm} \cdot Q_m^{k+1}(x_n^k),$$

and finally setting

$$\overline{\mathsf{LC}}_n^k(x_n) := v_n^k - \rho_n \|x_n - x_n^k\|$$

for an appropriate penalty constant ρ_n.

Another example of Lipschitz cuts that can be used are Augmented Lagrangian cuts, because the cost-to-go functions Q_m^k for each scenario are optimal value functions given by (18). We present a further discussion around these cuts in 4.1.

There are some important comments about the concepts introduced so far. First, we note that the sequence $\{\overline{\Omega}_n^k\}_{k \in \mathbb{Z}_+}$ is a non-decreasing sequence of functions, and since it belongs to the objective function of (18), we conclude that the sequence of cost-to-go function approximations $\{Q_n^k\}_{k \in \mathbb{Z}_+}$ is also non-decreasing. Therefore, the expected cost-to-go approximation $\overline{Q}_n^k(x_n)$ defined in (19) is a weighted average of non-decreasing functions Q_m^k, so the corresponding sequence $\{\overline{Q}_n^k\}_{k \in \mathbb{Z}_+}$ is also non-decreasing.

Second, the function $\overline{Q}_n^k$ plays an important role in the convergence analysis of the SLDP since the cuts of (21) are tight for $\overline{Q}_n^{k+1}(x_n)$ at the forward solution x_n^k. From equations (18) and (19), the quality of the expected cost-to-go approximation $\overline{Q}_n^k$ of a given node $n \in \mathcal{N}$ depends on the quality of those approximations at the successor nodes $m \in \mathcal{S}(n)$. This explains the reason of computing Lipschitz cuts from the last to first stage.

We will prove convergence of the full scenario (resp. sampled scenario) SLDP algorithm by proving that the sequence of feasible policies $\left(x_n^k\right)_{n \in \mathcal{N}}$ produced by the algorithm converges to an optimal (ε-optimal) policy $\left(x_n^*\right)_{n \in \mathcal{N}}$. As in Lemma 1, we show that $\overline{Q}_n^{k+1}(x_n^k)$ converges to (an ε-approximation

of) $\overline{Q}_n(x_n^*)$ where $\overline{Q}_n$ is the *true* expected cost-to-go function (16). In the examples of section 4, we will see that both expected cost-to-go approximations $\overline{Q}_n^k$ and $\overline{\mathfrak{Q}}_n^k$ approximate the true expected cost-to-go function $\overline{Q}_n$ in a neighborhood of the optimal (ε-optimal) policy solution x_n^*, however those approximations are usually poor elsewhere.

As we mentioned at the end of the reverse-norm method, it is an important requirement for the convergence that the Lipschitz cuts have a uniformly bounded Lipschitz constant. That's why we ask that all cuts $\overline{\mathsf{LC}}_n^k$ are constructed with Lipschitz constant at most ρ_n, so we can ensure, for all x_n:

$$\overline{\mathfrak{Q}}_n^{k+1}(x_n) \geq \overline{\mathfrak{Q}}_n^k(x_n) \tag{23}$$

$$\overline{\mathfrak{Q}}_n^{k+1}(x_n) \geq \overline{Q}_n^{k+1}(x_n^k) - \rho_n \cdot \|x_n - x_n^k\|, \tag{24}$$

where inequality (23) follows from the definition of the expected cost-to-go approximation $\overline{\mathfrak{Q}}_n^{k+1}(\cdot)$, and inequality (24) follows from $\overline{\mathfrak{Q}}_n^{k+1}$ being a Lipschitz function with constant ρ_n and the tightness from (20). Proposition 1 ensures that we can take ρ_n as any constant greater than or equal to the Lipschitz constant $\widetilde{L}_n$ defined on (17).

By the definition of cost-to-go and expected cost-to-go approximations, they form a monotone sequence of valid lower bounds for the true expected cost-to-go functions:

Lemma 3 *Consider the stochastic multistage MILP program* (15) *satisfying the CCR condition, and let* $\overline{\mathfrak{Q}}_n^1$ *be the initial lower bounds for the expected cost-to-go functions* $\overline{Q}_n$*. Then* $\overline{\mathfrak{Q}}_n^k$ *and* $\overline{Q}_n^k$*, the expected cost-to-go approximations of* (21) *and* (19)*, satisfy*

$$\overline{\mathfrak{Q}}_n^k \leq \overline{Q}_n \quad \textit{and} \quad \overline{Q}_n^k \leq \overline{Q}_n \tag{25}$$

for every node $n \in \mathcal{N}$ *and iteration* $k \in \mathbb{Z}_+$*.*

Moreover, if the initial lower bounds are compatible, that is, if $\overline{\mathfrak{Q}}_n^1 \leq \overline{Q}_n^1$ *for all nodes* n*, then it also holds that*

$$\overline{\mathfrak{Q}}_n^k \leq \overline{Q}_n^k \tag{26}$$

for all further iterations k*.*

Remark 1 Since $\overline{Q}_n^1$ depends on the approximations $\overline{\mathfrak{Q}}_m^1$ for all successor nodes $m \in \mathcal{S}(n)$, it could happen that a valid lower bound $\overline{\mathfrak{Q}}_n^1$ for the true value function $\overline{Q}_n$ were larger than $\overline{Q}_n^1$.

Even if the compatibility of the initial lower bounds is not a requirement, it is a common situation. For example, if all costs are non-negative, a typical initial lower bound $\overline{\mathfrak{Q}}_n^1$ is the zero function, and since therefore $\overline{Q}_n^1 \geq 0$ the condition is immediately satisfied.

Proof We proceed by induction on k and backward induction on the tree. The base case for $k = 1$ holds by hypothesis, since for every node $n \in \mathcal{N}$ we have $\overline{\mathfrak{Q}}_n^1 \leq \overline{Q}_n$. Moreover, the equations in (25) hold for the leaf nodes, because $\overline{\mathfrak{Q}}_n^k$, $\overline{Q}_n^k$ and $\overline{Q}_n$ are identically zero.

We now prove that, if $\overline{\mathfrak{Q}}_n^k \leq \overline{Q}_n$, then both $\overline{Q}_n^{k+1}$ and $\overline{\mathfrak{Q}}_n^{k+1}$ will also be valid lower bounds for $\overline{Q}_n$. By backwards induction on the tree, we can assume that at all successor nodes $m \in \mathcal{S}(n)$ it already holds that $\overline{\mathfrak{Q}}_m^{k+1} \leq \overline{Q}_m$. Plugging into equation (18) yields $Q_m^{k+1} \leq Q_m$, and averaging shows $\overline{Q}_n^{k+1} \leq \overline{Q}_n$, which is the second inequality in (25).

For the inequality concerning $\overline{\mathfrak{Q}}_n^{k+1}$, by the construction of $\overline{\mathfrak{Q}}_n^{k+1}$ in (21), it is enough to show that $\overline{\mathsf{LC}}_n^k \leq \overline{Q}_n$. Now, from equation (20), we obtain $\overline{\mathsf{LC}}_n^k \leq \overline{Q}_n^{k+1}$, which we have just shown to be a lower bound to $\overline{Q}_n$, so this finishes the induction step.

If moreover we have the compatibility property that $\overline{\mathfrak{Q}}_n^1 \leq \overline{Q}_n^1$, then it can be combined, again by induction, with the inequality $\overline{\mathsf{LC}}_n^k \leq \overline{Q}_n^{k+1}$ obtained above to show that $\overline{\mathfrak{Q}}_n^{k+1} \leq \overline{Q}_n^{k+1}$ as desired. □

Throughout this paper we assume the CCR condition for the true stochastic multistage mixed-integer program (15). Additionally, we also require the set of feasible policy's states $\mathrm{proj}_x X_n$ to be compact, and we name the resulting assumption as the *Compact State Complete Continuous Recourse* (CS-CCR) condition.

3.3 Full scenario approach

The SLDP algorithm for the full scenario approach is analogous to the Nested Cutting Plane algorithm [16], but with Lipschitz cuts instead of linear cuts. As described in Algorithm 2, starting from a valid lower bound $\overline{\mathfrak{Q}}_n^1$ for all cost-to-go functions, an upper bound ρ_n for their Lipschitz constants, and a method for calculating *uniformly Lipschitz tight cuts*, it improves the lower bounds near the candidate optimal solutions of each iteration k. Thus, in the forward step, the full scenario SLDP algorithm solves the optimization problems (18) from the root to the leaves, that is, in ascending order of stages, and obtains feasible state and control variables $(x_n^k, y_n^k, z_n^k) \in X_n$ for each node of the scenario tree. Then, in the backward step, it updates from the leaves to the root the expected cost-to-go approximation $\overline{\mathfrak{Q}}_n^k$, building Lipschitz cuts centered at the states obtained in the forward step and using formula (21).

We have not provided a stopping criterion for Algorithm 2. Although in the full scenario case we could have chosen a criterion equivalent to the one in Algorithm 1, in the sampled case one would need to compute the optimal solution at every node of the scenario tree to have a deterministic upper bound for the optimal policy, which is unrealistic. So, we preferred to emphasize the

Algorithm 2 Full scenario SLDP

Input: lower bound $\overline{\Omega}_n^1 \leq \overline{Q}_n$, cut constant $\rho_n \geq \widetilde{L}_n$, and a method for producing Lipschitz cuts $\overline{\mathsf{LC}}$ satisfying (20).

Output: policy $(x, y, z) \in \mathbb{X}$ and expected cost-to-go approximations $\{\overline{\Omega}_n^k\}_{n\in\mathcal{N}}$.

Set initial iteration $k = 1$ and go to step 1

1: Forward step: set $n = 1$, $x_0^k = 0$. For each t from 1 to $T-1$, and for each $n \in \mathcal{N}_t$ do:
   (a) Solve the problem (18) corresponding to $Q_n^k(x_{a(n)}^k)$ and obtain (x_n^k, y_n^k, z_n^k):

$$\begin{aligned} \min_{x_n, y_n, z_n} \quad & f_n(x_n, y_n) + \overline{\Omega}_n^k(x_n) \\ \text{s.t.} \quad & (x_n, y_n, z_n) \in X_n, \\ & z_n = x_{a(n)}^k; \end{aligned}$$

2: Backward step: For each t from $T-1$ to 1, and for each $n \in \mathcal{N}_t$ do:
   (a) Solve the children subproblems for $m \in \mathcal{S}(n)$:

$$\begin{aligned} Q_m^{k+1}(x_n^k) = \min_{x_m, y_m, z_m} \quad & f_m(x_m, y_m) + \overline{\Omega}_m^{k+1}(x_m) \\ \text{s.t.} \quad & (x_m, y_m, z_m) \in X_m, \\ & z_m = x_n^k; \end{aligned}$$

   (b) Calculate a ρ_n-Lipschitz cut LC_n^k for $\overline{Q}_n^{k+1} = \sum_{m\in\mathcal{S}(n)} Q_m^{k+1}$, centered at x_n^k and satisfying (20);
   (c) Update $\overline{\Omega}_n^{k+1} = \max\left\{\overline{\Omega}_n^k, \overline{\mathsf{LC}}_n^k\right\}$.

3: Increase k by 1, and go to step 1.

similarities between the full and sampled scenario cases. and present their convergence results only in asymptotic form.

In order to simplify the notation and improve readability, we will assume that a variable x or a vector (x, y, z) that do not have the subscript n is the vector composed by the corresponding variables or vectors for all nodes:

- $x := (x_n)_{n\in\mathcal{N}}$;
- $(x, y, z) := \Big((x_n, y_n, z_n)\Big)_{n\in\mathcal{N}}$.

We refer to (x, y, z) as a policy and x as the policy's states, and we denote by $\mathbb{X}$ the set of feasible policies and by $\mathrm{proj}_x \mathbb{X}$ the projection of $\mathbb{X}$ in the policy's states.

By analogy with the proof of the (deterministic) reverse-norm method in Lemma 1 and Theorem 1, we start proving that the expected cost-to-go approximation $\overline{\Omega}_n^k$ approximates the true expected cost-to-go function $\overline{Q}_n$ in a neighborhood of any cluster state induced by the forward step.

Lemma 4 *Let $x^* \in \mathrm{proj}_x \mathbb{X}$ be a cluster point of the sequence of policy states $\{x^k\}_{k\in\mathbb{N}}$ generated by the forward step of Algorithm 2, and let $\mathcal{K}$ be the indices of a subsequence that converges to x^*. Then $\{\overline{\Omega}_n^k(x_n^k)\}_{k\in\mathcal{K}}$ converges to $\overline{Q}_n(x_n^*)$,*

$$\lim_{k\in\mathcal{K}} \overline{\Omega}_n^k(x_n^k) = \overline{Q}_n(x_n^*), \tag{27}$$

for every node $n \in \mathcal{N}$.

Proof Let $\{(x^k, y^k, z^k)\}_{k\in\mathbb{N}}$ be the sequence of policies obtained in the forward step of algorithm 2. By the compactness assumption of $\text{proj}_x X_n$, the set of feasible policy states $\text{proj}_x \mathbb{X}$ is also compact, so there is a subsequence of $\{x^k\}_{k\in\mathbb{N}}$ that converges to a cluster point $x^* \in \text{proj}_x \mathbb{X}$. Denote by $\mathcal{K}$ the indices of this subsequence, that is, $\lim_{k\in\mathcal{K}} x^k = x^*$. We will show that equation (27) holds by backward induction on the tree. It trivially holds for the leaf nodes, since both functions $\overline{\mathfrak{Q}}_n$ and $\overline{Q}_n$ are identically zero, by hypothesis.

Now, suppose that equation (27) holds for every successor node $m \in \mathcal{S}(n)$. From Lemma 3, we have an upper bound:

$$\overline{Q}_n(x_n^k) \geq \overline{\mathfrak{Q}}_n^k(x_n^k),$$

which, by continuity of $\overline{Q}_n$, yields:

$$\overline{Q}_n(x_n^*) \geq \limsup \overline{\mathfrak{Q}}_n^k(x_n^k). \tag{28}$$

So we only need to prove that the lower approximations are large enough.

As in Lemma 1, we denote by j the index in $\mathcal{K}$ immediately before k. By monotonicity of the approximations, $\overline{\mathfrak{Q}}_n^k$ is larger than all of the Lipschitz cuts constructed, in particular the one at iteration j. Therefore, by inequality (24)

$$\overline{\mathfrak{Q}}_n^k(x_n^k) \geq \overline{\mathfrak{Q}}_n^{j+1}(x_n^k) \geq \overline{Q}_n^{j+1}(x_n^j) - \rho_n \cdot \left\|x_n^j - x_n^k\right\|.$$

Note that, differently from Lemma 1, we don't obtain the exact expected cost-to-go function $\overline{Q}_n$, but only its approximation $\overline{Q}_n^{j+1}$. That's why our proof splits in two parts: one bounding the difference between $\overline{Q}_n^j$ and $\overline{\mathfrak{Q}}_n^k$, and the other bounding the difference between $\overline{Q}_n$ and $\overline{Q}_n^j$. Let's complete the first one, which we already started. Since $\overline{Q}_n^j$ is an increasing sequence, $\overline{Q}_n^j \leq \overline{Q}_n^{j+1}$, and we obtain

$$\overline{\mathfrak{Q}}_n^k(x_n^k) \geq \overline{Q}_n^j(x_n^j) - \rho_n \cdot \left\|x_n^j - x_n^k\right\|. \tag{29}$$

To show that $\overline{Q}_n$ and $\overline{Q}_n^j$ are close, we use their definitions in (19) and (16):

$$\begin{aligned}
\overline{Q}_n^j(x_n^j) - \overline{Q}_n(x_n^j) &= \sum_{m\in\mathcal{S}(n)} q_{nm} \cdot \left[Q_m^j(x_n^j) - Q_m(x_n^j)\right] \\
&\geq \sum_{m\in\mathcal{S}(n)} q_{nm} \cdot \left[\overline{\mathfrak{Q}}_m^j(x_m^j) - \overline{Q}_m(x_m^j)\right], \qquad (30)
\end{aligned}$$

where the inequality follows because (x_m^j, y_m^j, z_m^j) are optimal solutions to (18) and feasible solutions to (15) for each $m \in \mathcal{S}(n)$. Taking (29) and (30) together and rearranging terms we get

$$\overline{\mathfrak{Q}}_n^k(x_n^k) \geq \overline{Q}_n(x_n^j) - \rho_n \cdot \left\|x_n^j - x_n^k\right\| - \sum_{m\in\mathcal{S}(n)} q_{nm} \cdot \left[\overline{Q}_m(x_m^j) - \overline{\mathfrak{Q}}_m^j(x_m^j)\right].$$

Now, take the limit as k goes to ∞, which also makes j grow to ∞, and both x^k and x^j converge to x^*. Since the expected cost-to-go function $\overline{Q}_n$ is continuous, we obtain:

$$\liminf \overline{\Omega}_n^k(x_n^k) \geq \overline{Q}_n(x_n^*)$$

because both residual terms vanish in the limit, the second one going to zero by our induction hypothesis. Together with the upper bound from equation (28), this shows that the limit exists and concludes our proof. □

As a consequence of Lemma 4, the expected cost-to-go approximation $\overline{Q}_n^k$ also approximates the true expected cost-to-go function $\overline{Q}_n$ in a neighborhood of any cluster point of the sequence of feasible policy's states induced by the forward step of the full scenario SLDP. Using the argument that leads to inequality (30) in the proof of Lemma 4 we get that the cost-to-go approximation Q_n^k also approximates the true cost-to-go function Q_n in a neighborhood of any cluster policy state. That is, the following limits also hold:

$$\lim_{k \in \mathcal{K}} \overline{Q}_n^k(x_n^k) = \overline{Q}_n(x_n^*),$$
$$\lim_{k \in \mathcal{K}} Q_n^k(x_{a(n)}^k) = Q_n(x_{a(n)}^*),$$

for any convergent subsequence $\{x^k\}_{k \in \mathcal{K}}$ of policy states induced by the forward step of the SLDP algorithm, and $x^* \in \text{proj}_x \mathbb{X}$ the corresponding limit point.

Theorem 4 *Suppose the stochastic optimization problem in* (15)-(16) *satisfies the Complete Continuous Recourse hypothesis, and fix a method for producing uniformly Lipschitz cuts satisfying* (20)*. Then, the sequence of lower bounds* $\{Q_1^k\}_{k \in \mathbb{N}}$ *induced by the SLDP algorithm 2 converges to the optimal value* Q_1 *of the true stochastic multistage problem, and every cluster point of the sequence of feasible policies* $\{(x^k, y^k, z^k)\}_{k \in \mathbb{N}}$ *generated by the forward step of Algorithm 2 is an optimal policy.*

In particular, the augmented Lagrangian cuts from section 2.4 are tight, and the CCR hypothesis ensures a uniform Lipschitz constant at each node, so Algorithm 2 converges.

Proof Let $\{(x^k, y^k, z^k)\}_{k \in \mathbb{N}}$ be the sequence of feasible policies generated by the full scenario SLDP algorithm 2. Let $\mathcal{K}$ be the set of indices of a convergent subsequence of policy states $\{x^k\}_{k \in \mathbb{N}}$, and let x^* be the corresponding limit point, which exists by compactness of $\text{proj}_x \mathbb{X}$. As for equation (30), at the root node we have

$$\begin{aligned} 0 \leq Q_1 - Q_1^k &\leq \Big[f_1(x_1^k, y_1^k) + \overline{Q}_1(x_1^k) \Big] - \Big[f_1(x_1^k, y_1^k) + \overline{\Omega}_1^k(x_1^k) \Big], \\ &= \overline{Q}_1(x_1^k) - \overline{\Omega}_1^k(x_1^k), \end{aligned}$$

because (x_1^k, y_1^k, z_1^k) is a feasible solution to the optimization problem whose optimal value is Q_1 and optimal solution to that whose optimal value is Q_1^k.

Using Lemma 4, we conclude the convergence of the subsequence $\{Q_1^k\}_{k\in\mathcal{K}}$ to Q_1. Since the whole sequence $\{Q_1^k\}_{k\in\mathbb{N}}$ is non-decreasing, we get convergence to Q_1.

Now, suppose that there is a cluster point (x, y, z) of the sequence of feasible policies $\{(x^k, y^k, z^k)\}_{k\in\mathbb{N}}$, and denote also by $\mathcal{K}$ the set of indices of the corresponding subsequence. In order to prove that (x, y, z) is an optimal policy, we need to show that its components are optimal solutions to the optimization problem of each node $n \in \mathcal{N}$ whose optimal value is $Q_n(x_{a(n)})$. We will proceed by forward induction on the tree. Indeed, we have just shown that (x_1, y_1, z_1) is an optimal solution at the root node. Now, assume that this result holds for the ancestor node $a(n)$. Using the same argument as before, we have the following inequalities:

$$f(x_n^k, y_n^k) + \overline{\Omega}_n^k(x_n^k) = Q_n^k(x_{a(n)}^k) \leq Q_n(x_{a(n)}^k) \leq f(x_n^k, y_n^k) + \overline{Q}_n(x_n^k).$$

So, taking the limit over $\mathcal{K}$ on both sides of the inequality and using Lemma 4, we conclude that $(x_n, y_n, x_{a(n)})$ is an optimal solution of the optimization problem whose optimal value is $Q_n(x_{a(n)})$. □

3.4 Sampled tree approach

In multistage stochastic programming problems with a reasonable number of stages, it is computationally intractable to visit every node of the scenario tree. So, one needs to sample paths on the scenario tree and iteratively approximate the expected cost-to-go functions at each stage to obtain a "reasonable" solution. In this paper, we focus on the sampling scheme of one random path per forward iteration, but its conversion to more general schemes is straightforward.

We emphasize that a path on the scenario tree is chosen at random, so a node n may not belong to the path of some forward step iterations. Let $\mathcal{J}_n$ be the set of iterations k of the Sampled-SLDP for which the path of the forward step contains the node n. Note that $\mathcal{J}_n$ is a random set, since it depends on each experiment $\omega \in \Omega$, and the probability of node n being drawn an infinite number of times equals one, i.e.,

$$\mathbb{P}(\{\omega \in \Omega \mid \#\mathcal{J}_n(\omega) = +\infty\}) = 1,$$

by the Borel-Cantelli Lemma. We will assume a realization of the sampling where this is the case, to avoid repeating "with probability one" in what follows.

Also, observe that the collection of sets $\{\mathcal{J}_m \mid m \in \mathcal{S}(n)\}$ induced by the successor nodes covers $\mathcal{J}_n$, that is

$$\mathcal{J}_n = \bigcup_{m\in\mathcal{S}(n)} \mathcal{J}_m,$$

since a path that contains a node n also contains some successor node m. In the deterministic case, the set of iterations $\mathcal{J}_n$ equals $\mathbb{Z}_+$ for every node n,

since all nodes are visited in the forward step. In the analysis of the Sampled-SLDP algorithm, we need to refer to optimal solutions of nodes that do not belong to a given forward path, even if in practice they are not computed. We still use the same notation (x_n^k, y_n^k, z_n^k) to refer to an optimal solution of node n and iteration k.

Following the same organization of the previous sections, we would like to prove that for each node n there is a subset $\mathcal{K}_n$ of $\mathcal{J}_n$ such that the following limit holds:

$$\lim_{k \in \mathcal{K}_n} \overline{\Omega}_n^k(x_n^k) = \overline{Q}_n(x_n^*), \tag{31}$$

where $\{x_n^k\}_{k \in \mathcal{K}_n}$ is a subsequence of policy states converging to a limit point x_n^*. However, the main obstacle of this lemma is the induction step, since we need to control the difference between $\overline{Q}_n(x_n^k)$ and $\overline{Q}_n^k(x_n^k)$ using inequality (30) or some variation, as in the proof of Lemma 4. Inequality (30) directly is not suitable for the proof, because there we implicitly used that $\mathcal{K}_m$ equals $\mathcal{K}_n$ for every successor node m to be able to use the induction hypothesis (31).

In order to ensure convergence of the Sampled-SLDP algorithm, we consider an additional step to stabilize the policy states obtained in the forward step. Instead of computing Lipschitz cuts at every *new* forward solution x_n^k, we check if the new feasible point is more than $\delta > 0$ away from all previous forward solutions $x_n^1, \ldots, x_n^{k-1}$. If this is the case, then we update the expected cost-to-go function $\overline{\Omega}_n^k(\cdot)$ with the Lipschitz cut centered at the new policy state x_n^k; otherwise we improve it at the closest previous forward solution x_n^j, see Algorithm 3. Note that after a finite number of iterations the forward incoming state x_n^k becomes trapped in a finite number of possibilities, since node n will be visited an infinite number of times and the set of feasible policy states $\mathrm{proj}_x X_n$ is compact.

We also assume that the Lipschitz cut $\overline{\mathsf{LC}}_n^k$ depends only on the expected cost-to-go function $\overline{Q}_n^k(\cdot)$ and the center point x_n^k, that is, the Lipschitz cut $\overline{\mathsf{LC}}_n^k$ is a function of $\overline{Q}_n^k(\cdot)$ and x_n^k only.

We show in Lemma 5 that the expected cost-to-go approximation $\overline{\Omega}_n^k$ converges in a finite number of iterations to a Lipschitz function $\overline{\Omega}_n$, which is an ε-approximation of the true expected cost-to-go function $\overline{Q}_n$ at any cluster point x_n^*.

Lemma 5 *With probability one, the sequence of expected cost-to-go approximations $\{\overline{\Omega}_n^k\}_{k \in \mathbb{N}}$ generated by Algorithm 3 converges to a Lipschitz function $\overline{\Omega}_n$ with constant $\widetilde{L}_n$ after a finite number of iterations. Moreover, the following relationships hold for every node n of the tree:*

$$\lim_{k \in \mathcal{K}_n} \overline{\Omega}_n^k(x_n^k) = \overline{\Omega}_n(x_n^*), \tag{32}$$

$$0 \leq \overline{Q}_n(x_n^*) - \overline{\Omega}_n(x_n^*) \leq (\widetilde{L} + \rho) \cdot \delta \cdot (T - t(n)), \tag{33}$$

Algorithm 3 Sampled SLDP

Input: lower bound $\overline{\Omega}_n^1 \leq \overline{Q}_n$, cut constant $\rho_n \geq \widetilde{L}_n$.

Output: policy $(x, y, z) \in \mathbb{X}$ and expected cost-to-go $\{\overline{\Omega}_n^k\}_{n\in\mathcal{N}}$ approximations.

Set initial iteration $k = 1$ and go to step 1

1: Forward step: set $n = 1$, $x_0^k = 0$. While $\mathcal{S}(n) \neq \emptyset$ do:
   (a) Solve the problem corresponding to $Q_n^k(x_{a(n)}^k)$ and obtain (u_n^*, y_n^*, z_n^*):

$$\begin{aligned} \min_{u_n, y_n, z_n} \; & f_n(u_n, y_n) + \overline{\Omega}_n^k(u_n) \\ \text{s.t.} \; & (u_n, y_n, z_n) \in X_n, \\ & z_n = x_{a(n)}^k; \end{aligned}$$

   (b) Compute the minimum $\Delta_n^k = \min_{j=1,\ldots,k-1} \|u_n^* - x_n^j\|$, and denote by i the minimizer index. If $\Delta_n^k < \delta$ then set $x_n^k = x_n^i$, else set $x_n^k = u_n^*$;
   (c) Sample $m \in \mathcal{S}(n)$, and set $n = m$;
2: Backward step: Start with the particular node n of the end of step 1. While n is not the root 1, do:
   (a) Solve the children subproblems for $m \in \mathcal{S}(n)$:

$$\begin{aligned} Q_m^{k+1}(x_n^k) = \min_{x_m, y_m, z_m} \; & f_m(x_m, y_m) + \overline{\Omega}_m^{k+1}(x_m) \\ \text{s.t.} \; & (x_m, y_m, z_m) \in X_m, \\ & z_m = x_n^k; \end{aligned}$$

   (b) Calculate a ρ_n-Lipschitz cut LC_n^k for $\overline{Q}_n^{k+1} = \sum_{m\in\mathcal{S}(n)} Q_m^{k+1}$, centered at x_n^k and satisfying (20);
   (c) Update $\overline{\Omega}_n^{k+1} = \max\left\{\overline{\Omega}_n^k, \overline{\mathsf{LC}}_n^k\right\}$.
   (d) Keep $\overline{\Omega}_{n'}^{k+1} = \overline{\Omega}_{n'}^k$ for nodes $n' \in \mathcal{N}_{t(n)}$ such that $n' \neq n$;
   (e) Set $n = a(n)$.
3: Increase k by 1 and go to step 1.

where $\mathcal{K}_n$ is a subset of indices from $\mathcal{J}_n$ such that the sequence of policy states $\{x_n^k\}_{k\in\mathcal{K}_n}$ converges, x_n^* is the corresponding limit point, $\widetilde{L}$ is the maximum Lipschitz constant $\widetilde{L}_n$ and ρ is the maximum penalty constant ρ_n over all nodes $n \in \mathcal{N}$.

Proof We start proving the finite convergence of $\overline{\Omega}_n^k$ to $\overline{\mathfrak{Q}}_n$, and the limit in (32) by backward induction on the tree. In the last stage this result is trivial since both functions $\overline{\Omega}_n^k$ and $\overline{Q}_n$ are identically zero.

Let n be a node such that the limit in (32) holds for every successor node $m \in \mathcal{S}(n)$. Recall that the updating rule of the Lipschitz cut has the form:

$$\overline{\Omega}_n^{k+1} = \max\left\{\overline{\Omega}_n^k, \overline{\mathsf{LC}}_n^k\right\}.$$

where the cut $\overline{\mathsf{LC}}_n^k$ is tight at x_n^k for the expected cost-to-go approximation $\overline{Q}_n^{k+1}$, which is the weighted average of the cost-to-go approximations Q_m^k over the successor nodes $m \in \mathcal{S}(n)$. By the induction hypothesis, after a finite

number of iterations we obtain

$$\begin{aligned} Q_m^k(x_n) = \min_{x_m, y_m, z_m} \; & f_m(x_m, y_m) + \overline{\mathfrak{U}}_m(x_m) \\ \text{s.t. } & (x_m, y_m, z_m) \in X_m, \\ & z_n = x_n. \end{aligned} \tag{34}$$

In other words, both the cost-to-go Q_m^k and the expected cost-to-go $\overline{Q}_n^k$ approximations stabilize after a finite number of iterations, so denote by U_m and $\overline{U}_n$ the corresponding limits, respectively. Since the number of different incoming states x_n^k is also finite and the Lipschitz cut $\overline{\mathsf{LC}}_n^k$ just depends on $\overline{U}_n$ and x_n^k, this implies that the number of different possible reverse norm cuts to update $\overline{\mathfrak{Q}}_n^k$ is also finite. Then, $\overline{\mathfrak{Q}}_n^k$ converges to a function $\overline{\mathfrak{U}}_n$ in a finite number of iterations.

Now, let's prove inequality (33) by backward induction on the tree. It is trivial at the leaf nodes, so suppose inequality (33) holds for every successor node $m \in \mathcal{S}(n)$. Since there is a finite number of different possible policy states at the node n, the sequence $\{x_n^k\}_{k \in \mathcal{K}_n}$ converges to x_n^* in a finite number of iterations, which means that a stabilized Lipschitz cut centered at x_n^* is also considered in the expected cost-to-go limit $\overline{\mathfrak{U}}_n$. In particular, we can take limits in (24) and get:

$$\overline{U}_n(x_n^*) - \rho_n \cdot \|x_n - x_n^*\| \leq \overline{\mathfrak{U}}_n(x_n),$$

which leads to the following bounds for the difference between $\overline{\mathfrak{U}}_n$ and the true cost-to-go function $\overline{Q}_n$:

$$\begin{aligned} 0 \leq \overline{Q}_n(x_n^*) - \overline{\mathfrak{U}}_n(x_n^*) &\leq \overline{Q}_n(x_n^*) - \overline{U}_n(x_n^*) \\ &= \sum_{m \in \mathcal{S}(n)} q_{nm} \cdot [Q_m(x_n^*) - U_m(x_n^*)]\,. \end{aligned} \tag{35}$$

Now, we have the crucial part of the argument. Because the expected cost-to-go approximations of all nodes stabilize, every incoming state of any successor node $m \in \mathcal{S}(n)$ equals x_n^* after a large number of iterations. Then, the optimal solution of node m with input state x_n^* is equal to u_m, which is less than δ away from the final state x_m^* of node m, by the design of the Sampled-SLDP algorithm. Then, we obtain the following inequalities:

$$\begin{aligned} Q_m(x_n^*) - U_m(x_n^*) &\leq \overline{Q}_m(u_m) - \overline{\mathfrak{U}}_m(u_m) \\ &\leq \widetilde{L} \cdot \|u_m - x_m^*\| + \overline{Q}_m(x_m^*) - \overline{\mathfrak{U}}_m(u_m) \\ &\leq \overline{Q}_m(x_m^*) - \overline{\mathfrak{U}}_m(x_m^*) + (\widetilde{L} + \rho) \cdot \|u_m - x_m^*\| \end{aligned}$$

where the first inequality results from u_m being the optimal policy's state of node m with input state x_n^*, and the following ones from the Lipschitz property of $\overline{Q}_m$ and $\overline{\mathfrak{U}}_m$, respectively. By our induction hypothesis,

$$\overline{Q}_m(x_m^*) - \overline{\mathfrak{U}}_m(x_m^*) \leq (\widetilde{L} + \rho) \cdot \delta \cdot (T - t(m)),$$

and since $t(m) = t(n) + 1$ we get

$$Q_m(x_n^*) - U_m(x_n^*) \leq (\widetilde{L} + \rho) \cdot \delta \cdot (T - t(n)). \tag{36}$$

because u_m and x_m^* are at most δ far away from each other. So, the upper bound (36) together with equation (35) concludes the induction step. □

Theorem 5 *With probability* 1*, the sequence of lower bounds* $\{Q_1^k\}_{k\in\mathbb{N}}$ *generated by Algorithm 3 converges in a finite number of iterations to an* ε*-approximation of the true optimal value* Q_1*, where* $\varepsilon = (\widetilde{L} + \rho) \cdot \delta \cdot (T - 1)$*, and every cluster point of the sequence of feasible policies* $\{(x^k, y^k, z^k)\}_{k\in\mathbb{N}}$ *generated by the forward step of Algorithm 3 is an* ε*-optimal policy.*

Proof This is a straightforward result of Lemma 5, using the same reasoning as in Theorem 4. □

4 Examples

In this section, we will present two applications of the SLDP algorithm for stochastic optimization. The first is a simple example of a 1-dimensional dynamics with discrete control. Due to its symmetry and relative simplicity, it is possible to evaluate the cost-to-go functions, so that we can understand the behavior of the algorithm in its different forms. The second one is the capacitated lot-sizing problem inspired from [41], which was suggested to the authors by Nils Löhndorf, who also provided us with the data for the instance we use in section 4.3.

4.1 Implementation details

The non-convex cuts used in our SLDP implementation are represented as inequalities of the form

$$\alpha \geq v + \lambda^\top (x - x^k) - \rho \cdot \|x - x^k\|,$$

where $\lambda = 0$ for the reverse-norm cuts (see Definition 1), but is needed for the augmented Lagrangian cuts described in Section 2.4. To incorporate them in the stage problems, this requires choosing a norm, and a MILP formulation of this constraint. As discussed in Section 2.5, this is possible for the L^1 norm, which we used for the experiments below.

Observe that this formulation includes a binary variable (and two continuous variables), per dimension of x, for each new non-convex cut we introduce, cf equation (14). This makes each iteration of the SLDP algorithm much more expensive than previous ones.

In the dynamic programming setting for SLDP, the Augmented Lagrangian cuts are given by

$$\overline{\mathsf{LC}}_n^k = \sum_{m \in \mathcal{S}(n)} q_{nm} \cdot \mathsf{LC}_m^k \tag{37}$$

where cuts LC_m^k are calculated for each scenario $m \in \mathcal{S}(n)$ by solving the corresponding Augmented Lagrangian problem for the functions Q_m^{k+1}:

$$v_m^k := \max_{\lambda_m^k, \rho_m^k} \mathrm{AL}(Q_m^{k+1})(x_n^k; \lambda_m^k, \rho_m^k) \tag{38}$$

$$\begin{aligned} \mathrm{AL}(Q_m^{k+1})(x_n^k; \lambda_m^k, \rho_m^k) := \min_{x_m, y_m, z_m} \; & f_m(x_m, y_m) + \overline{\Omega}_m^{k+1}(x_m) \\ & - {\lambda_m^k}^\top (z_m - x_n^k) + \rho_m^k \cdot \|z_m - x_n^k\| \\ \text{s.t.} \; & (x_m, y_m, z_m) \in X_m, \end{aligned} \tag{39}$$

and then setting

$$\mathsf{LC}_m^k(x_n) := v_m^k + {\lambda_m^k}^\top (x_n - x_n^k) - \rho_m^k \cdot \|x_n - x_n^k\|$$

with the optimal multipliers.

Observe that the coefficients for $\overline{\mathsf{LC}}_n^k$ can be obtained by simply taking the mean with respect to q_{nm}:

$$\overline{\mathsf{LC}}_n^k(x_n) = \overline{v}_n^k + {\overline{\lambda}_n^k}^\top (z_m - x_n) - \overline{\rho}_n^k \cdot \|z_m - x_n\|,$$

where

$$\overline{v}_n^k = \sum_{m \in \mathcal{S}(n)} q_{nm} \cdot v_m^k; \quad \overline{\lambda}_n^k = \sum_{m \in \mathcal{S}(n)} q_{nm} \cdot \lambda_m^k; \quad \overline{\rho}_n^k = \sum_{m \in \mathcal{S}(n)} q_{nm} \cdot \rho_m^k$$

One practical implementation of the SLDP method uses augmented Lagrangian cuts, and increases ρ_m^k progressively in (38). Since by construction the augmented Lagrangian cuts are valid, if ρ_m^k is not large enough then the cuts might not be tight, but they might fill faster the non-convex regions of the cost-to-go function. As ρ_m^k will eventually become larger than the (true) Lipschitz constant of the cost-to-go functions, the algorithm still converges.

Also, in analogy to Strengthened Benders cuts, it is possible to fix both the Lagrange multiplier and the augmenting term in (38), and solve the resulting augmented Lagrangian relaxation. This again yields a valid cut, which we call *strengthened augmented Benders cut* when the Lagrange multiplier is calculated via the linear relaxation of the nodal problem.

All results below were obtained using Julia-0.6.3 [3] and the Julia packages `SDDP.jl` [12] and `SDDiP.jl` [21], besides our own Julia implementation for both Lipschitz and strengthened augmented Benders cuts [10], extending `SDDP.jl`. The computations were performed on an computer with a Intel(R) Core(TM) i7-8550U CPU and 8Gb RAM memory.

4.2 A 1-dimensional control problem

We consider the following multistage control problem:

$$\begin{aligned} \min\ & \mathbb{E}\left[\sum_{t=1}^{T} \beta^{t-1}|x_t|\right] \\ \text{s.t.}\ & x_t = x_{t-1} + c_t + \xi_t \\ & c_t \in \{\pm 1\} \end{aligned}$$

The state variable x_t is 1-dimensional, as the discrete control $c_t = \pm 1$, and the uncertainty ξ_t. The objective is to minimize the expected displacement away from zero, subject to a decay factor β, over the planning horizon T. We fix $T = 8$, $\beta = 0.9$, $x_0 = 2$, and at each stage t we consider 10 independent scenarios symmetrically sampled around 0.

We will compare the performance and the policy generated by several methods: a convex approximation using Strengthened Benders cuts (shortened as SB), the original SLDP algorithm using reverse-norm cuts (SLDP RN), a modified SLDP algorithm using ALD cuts with increasing augmentation ρ and Benders Lagrange multipliers (SLDP ALD), and the SDDiP algorithm [43], using two discretization steps: 0.1 and 0.01. The resulting discretized problems for SDDiP don't have complete continuous recourse, since the state cannot absorb the noise below the discretization level, and we only have a discrete control, so we also add a slack variable and penalize it in the objective function. The original problem, with continuous state, doesn't need adjustments.

We present in Table 1 the lower bounds, the estimated upper bounds using policy simulations and the computation times after 100 iterations for each method.

	SB	SLDP RN	SLDP ALD	SDDiP 0.1	SDDiP 0.01
LB	1.167	3.073	3.085	3.420	2.370
UB	3.453	3.320	3.313	3.823	3.490
time (s)	12	558	605	1994	3317

Table 1 Results for an 8-stage non-convex problem

The convex approximations stall at a very low lower bound, while the non-convex methods all have better estimates, but they also need significantly more computation time. The SLDP approximations have a very similar performance — the ALD method requiring slightly more time. SDDiP has a relatively good performance with step 0.1, but not with 0.01. Observe that the higher lower bound for SDDiP with step 0.1 also comes with a higher upper bound, which is due to the addition of the penalization term and a loose state discretization. When the discretization step is the smaller 0.01, the upper bounds of the simulation agree more closely with the other cases, but we spend 66% more in computation time, and the lower bounds are much further away.

As we can see in figure 6, the future cost functions are nonconvex at all time stages, essentially driven by the discontinuous control c_t: the immediate

cost is $\min\{|u-1|, |u+1|\}$, where $u = x_{t-1} + \xi_t$. However, the future cost

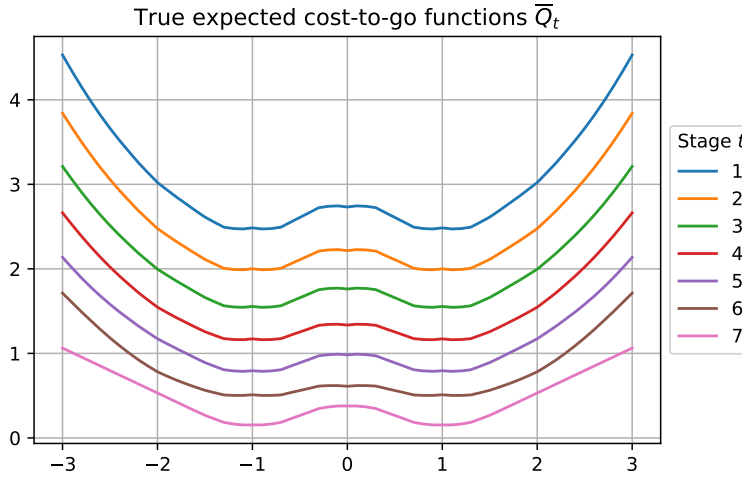


Fig. 6 Expected cost-to-go functions

functions built by the convex Strengthened Benders cuts can't pierce into the nonconvexities, and become flat over $[-1, 1]$, as depicted in figure 7. This explains why the convex approximations perform so poorly in this case.

The Lipschitz cuts, on the other hand, are indeed able to yield a better approximation of the problem, and approximate the expected cost-to-go functions inside their nonconvexities. The same happens with the value function obtained by SDDiP. In figure 8, we show a comparison of the expected cost-to-go functions thus constructed, using reverse-norm cuts, augmented Lagrangian cuts, SDDiP, and the actual future cost function.

One particular feature of this example is that the future cost functions are continuous in the original variables x_t, but since the control is $c_t \in \{-1, 1\}$, the stage problems lack the continuous recourse property. For this reason, when one takes the SDDiP discretization of the state variable, one must also include a slack term to the state dynamics. This increases the costs overall, and explains why the value functions estimated with the "coarse" discretization with $\varepsilon = 0.1$ are higher than the true expected cost-to-go functions $\overline{Q}_t$. The lower bounds $\overline{\Omega}_t$ obtained with the "fine" discretization with $\varepsilon = 0.01$, on the other hand, "detach" much faster from the respective $\overline{Q}_t$ as we move back in the stages of the tree.

Finally, we compare in figure 9 the evolution of the lower bounds, both as iterations and time increase. There, we see that both methods for SLDP were essentially equivalent, maybe except at the beginning, where the ALD's ρ was probably too small to yield good enough cuts. However, as iterations

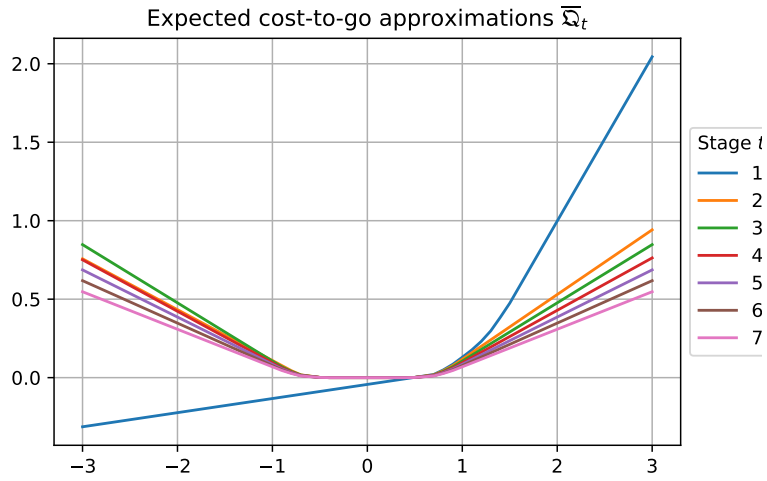


Fig. 7 Convex approximations of the expected cost-to-go functions, built with Strengthened Benders cuts

progressed, and ρ was large enough, the algorithm quickly reached a comparable lower bound. We also include a curve for the time taken per iteration, to highlight the rapid increase in time for the SLDP algorithm, even in a 1-dimensional problem. This same phenomenon happens for SDDiP, on a much smaller scale, but it already starts out with a significantly larger iteration time.

4.3 The Capacitated Lot Sizing Problem (CLSP)

The Capacitated Lot Sizing Problem [41] consists of determining the operation of a machine for the production of n types of products over T stages. If we decide to produce $P_{t,i}$ units of the i-th product at a given stage t, it is necessary to setup the machine for this purpose at that stage t. The binary vectors $S_t \in \{0,1\}^n$ represent whether the machine is setup or not for each product type at stage t. The constraint $P_t \leq M \cdot S_t$ models the necessity of setup for the production variables.

The overall production and setup time within stage t must remain below a maximum time limit, which is represented by the constraint $\sum_{i=1}^{M} a_i P_{t,i} + r_i S_{t,i} \leq K$, where a_i is the time per production for product number i, and r_i the time required to setup the machine for producing it (which we implicitly assume is independent of the previous configuration of the machine, and must be done for each stage).

We denote by L_t the lost sales at stage t, that is, the failure to meet the demand $\mathbf{d}_t \in \mathbb{R}^n_+$ for such products. The objective in this problem is to minimize the overall setup, inventory and lost sales costs within the planning horizon T, that is, we minimize the function $\sum_{t=1}^{T} \sum_{i=1}^{n} c_i \cdot S_{t,i} + u_i \cdot I_{t,i} + b_i \cdot L_{t,i}$.

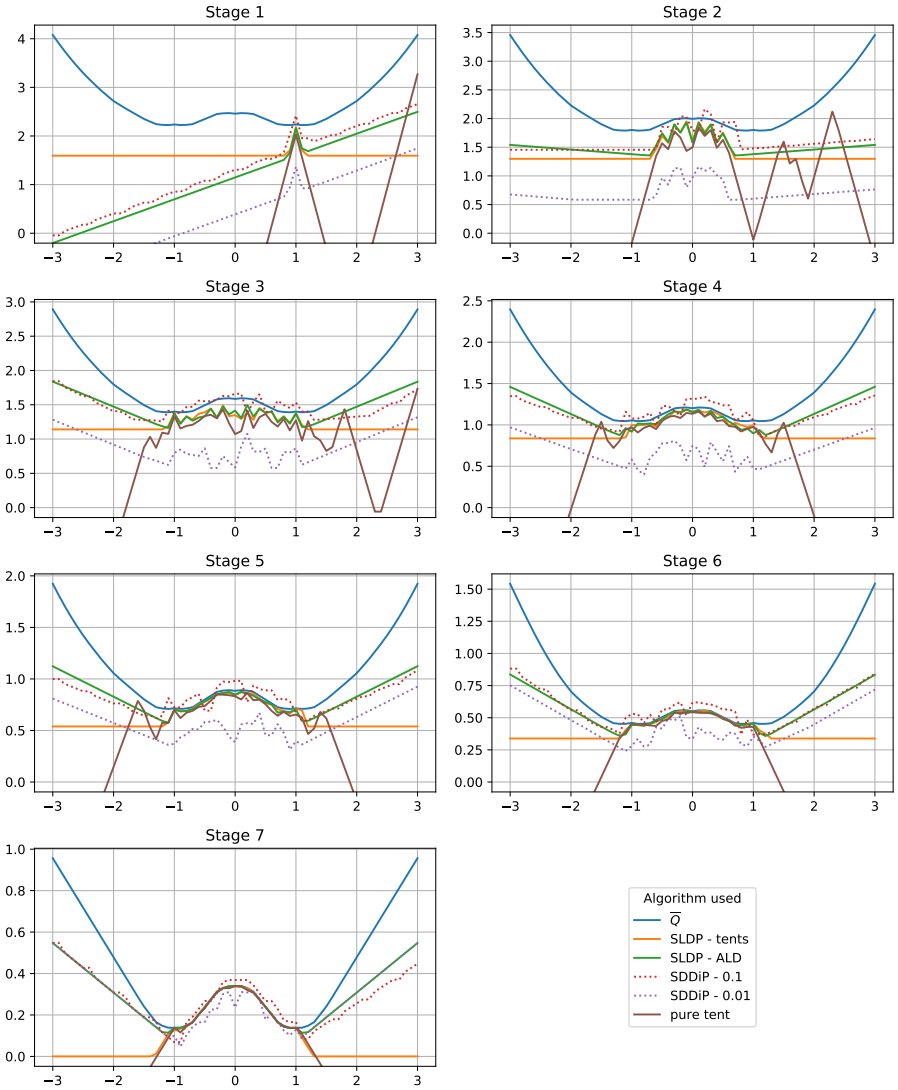


Fig. 8 Non-convex approximations of the expected cost-to-go functions

We denote by I_t the inventory at stage t, and assume that it has maximum capacity $\overline{I}$. The state equations are given by the inventory balance constraint $I_t - L_t = I_{t-1} + P_t - \mathbf{d}_t$, which is the only constraint that relates variables of a given stage t with the ones of stage $t-1$.

In this setting, the only uncertainty we consider is the demand $\mathbf{d}_t \in \mathbb{R}^n_+$ for such products. For a risk-neutral planning, the objective function becomes $\mathbb{E}_{\mathbf{d}}\left[\sum_{t=1}^{T}\sum_{i=1}^{n} c_i \cdot S_{t,i} + u_i \cdot I_{t,i} + b_i \cdot L_{t,i}\right]$. Below, we present the Stochastic

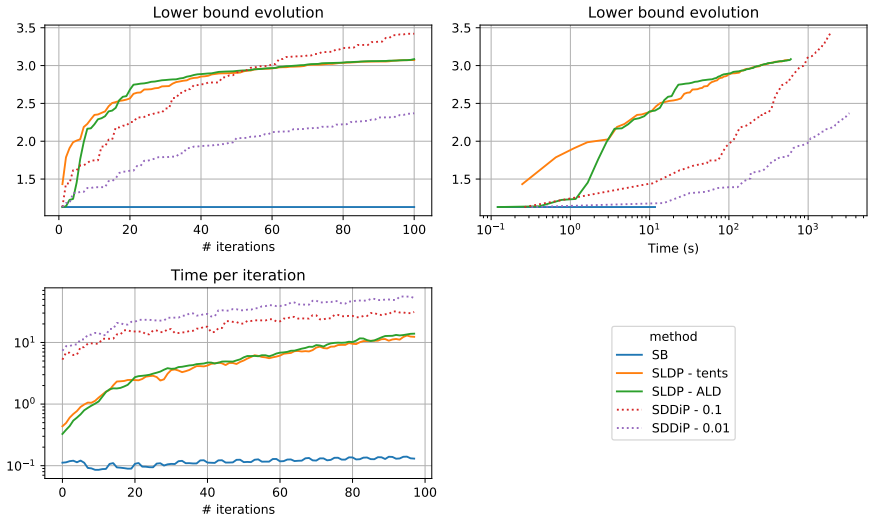


Fig. 9 Several timing comparisons: lower bound evolution and time per iteration

Capacitated Lot Sizing Problem in the dynamic programming form:

$$
\begin{aligned}
Q_t(I_{t-1}, \mathbf{d}_t) = \min \; & \textstyle\sum_{i=1}^{n} c_i \cdot S_{t,i} + u_i \cdot I_{t,i} + b_i \cdot L_{t,i} + \overline{Q}_{t+1}(I_t) \\
\text{s.t. } & I_t - L_t = I_{t-1} + P_t - \mathbf{d}_t, \\
& P_t \le M \cdot S_t, \\
& \textstyle\sum_{i=1}^{n} a_i P_{t,i} + r_i S_{t,i} \le K, \\
& 0 \le I_t \le \overline{I}, \quad P_t, L_t \ge 0, \quad S_t \in \{0,1\}^n.
\end{aligned}
$$

$$
\overline{Q}_{t+1}(I_t) = \begin{cases} \mathbb{E}_{\mathbf{d}}[Q_{t+1}(I_t, \mathbf{d}_{t+1})], & \text{if } t < T, \\ 0 & \text{if } t = T, \end{cases}
$$

for all $t = 1, \ldots, T$. We assume that the demand $\{\mathbf{d}_t\}_{t=1}^{T}$ follows a stagewise independent process.

For our example case, we consider the capacitated lot-sizing problem described in the appendix of [41], which consists of 3 products. Then, we modified it so that the demand for each product in each stage was multiplied by a log-normal random variable; the stagewise independent tree consisted then of 20 scenarios at each stage. For more than the original 4 stages, we extended the average demand periodically, but used different scenarios.

In Table 2, we present the run times and lower bounds of several methods employed to solve the Stochastic Capacitated Lot Sizing Problem. For two, three and four stages, the problem is sufficiently small that we are able to directly submit the Deterministic Equivalent formulation as a single MILP to Gurobi [18].

The decomposition algorithms we considered were the following: first, using Strengthened Benders cuts for SDDP; second, using Lagrangian and Strengthened Benders cuts for SDDiP, discretizing the state space; third, using Augmented Lagrangian Lipschitz cuts and Strengthened Benders cuts for SLDP. All decomposition algorithms used Gurobi as their internal LP/MILP solver, and were used from $T = 2$ up to $T = 7$ stages. For stopping criterion, we used 200 cuts for both SDDP, SDDiP and SLDP.

The discretization precision for the SDDiP algorithm was $\epsilon = 1.0$, which is a reasonable value given that the inventory state variable can in principle range from 0 to 600 units in our instance. The performance of the SLDP algorithm depends on the estimation of the Augmented Lagrangian penalty term ρ; for this experiment, we used a bisection procedure to find the smallest ρ for a given angular coefficient π (coming from the LP relaxation and the evaluation point x_0) that still yields a cut with zero duality gap. This is done to obtain the largest possible Lipschitz cut, so we get the greatest lower bound improvement at each iteration.

The Deterministic Equivalent formulation gives us a benchmark for the performance of the SDDiP and SLDP in small instances. As the size of the planning horizon increases, the size of the Deterministic Equivalent formulation increases exponentially, so this becomes impractical for planning horizons larger than $T = 4$. The Gurobi solver time limit is 2.4K seconds (40 minutes) and the only case it finds an optimal solution is for the two stage case.

Table 2 Results for the Lot Sizing problem

Planning Horizon (T)	2	3	4	5	6	7
Det. Equiv. LB	553.121*	1.077K	1.530K	NA	NA	NA
SB LB	382.264	838.273	1.105K	1.439K	2.015K	2.278K
SDDiP LB	488.475	918.183	1.242K	1.570K	2.070K	2.373K
SLDP LB	544.780	1.022K	1.420K	1.850K	2.401K	2.769K
ΔSDDiP LB (above SB)	106.2	79.9	137.0	131.0	55.0	95.0
ΔSLDP LB (above SB)	162.5	183.7	315.0	411.0	386.0	491.0
Det. Equiv. time (s)	0*	2.4K	2.4K	NA	NA	NA
SB time (s)	8.0	20.0	27.4	42.2	64.4	74.8
SDDiP time (s)	223.0	6.4K	12.2K	15.4K	17.1K	21.6K
SLDP time (s)	184.1	4.7K	8.5K	10.1K	13.6K	17.4K
SLDP/SDDiP time ratio	0.82	0.73	0.69	0.65	0.79	0.80

The first thing we note in Table 2 is that, although the pure Strengthened Benders algorithm is much faster than the SDDiP and SLDP algorithms, the lower bound it certifies is, as expected, always smaller the lower bounds obtained by both algorithms. We emphasize this difference in the rows ΔSDDiP LB and ΔSLDP LB, which show the difference between the corresponding algorithm lower bound and the lower bound given by the pure Strengthened Benders method. We see that the lower bound improvement of the SLDP algo-

rithm is consistently greater than the corresponding improvement of SDDiP, and achieved within a shorter time.

Also, as expected, the deterministic equivalent formulation has the best overall performance for few stages, which in this setting means no more than 5, but it is impractical for the multistage setting with a moderate number of stages due to the combinatorial explosion of the problem dimension as the planning horizon increases.

To investigate how the SB, SDDiP and SLDP algorithms perform for even larger planning horizons, we show in table 3 their results for 10, 13, 16 and 19 stages. As the planning horizon increases, the improvement of the lower bound obtained by the SDDiP algorithm over the pure Strengthened Benders method shrinks, although the costs are expected to increase. On the other hand, the SLDP algorithm is able to keep a reasonable improvement, while keeping its computational time smaller than that of the SDDiP algorithm.

Table 3 Results for 10, 13, 16 and 19 stages

Planning Horizon (T)	10	13	16	19
SB LB	3.346K	4.441K	5.704K	6.917K
SDDiP LB	3.367K	4.497K	5.714K	6.917K
SLDP LB	3.936K	5.301K	6.540K	7.882K
ΔSDDiP LB (above SB)	21.0	56.0	10.0	0.0
ΔSLDP LB (above SB)	590.0	860.0	836.0	965.0
SB time (s)	139.3	183.7	224.8	275.5
SDDiP time (s)	29.5K	39.5K	39.0K	39.3K
SLDP time (s)	14.0K	19.3K	27.7K	33.2K
SLDP/SDDiP time ratio	0.47	0.49	0.71	0.84

This case study shows that SLDP, through its use of non-convex cuts, is a practical alternative to SDDiP, providing better lower bounds, with comparable times, for multistage stochastic integer problems.

5 Conclusion

In this paper, we proposed a new algorithm for solving stochastic multistage MILP programming problems, called Stochastic Lipschitz Dynamic Programming (SLDP). Its major contribution is the inclusion of *nonlinear cuts* to iteratively underapproximate *nonconvex* Lipschitz cost-to-go functions. We explored two such families of cuts: (a) the ones induced by reverse-norm penalizations; and (b) augmented Lagrangian cuts, built from norm-augmented Lagrangian duality.

Assuming the Compact State Complete Continuous Recourse conditions, we proved convergence of the algorithm in the full scenario setting. In the sampled case, we provided an approximation method to reach ε-optimal policies

in finite time. Besides asymptotic convergence in the general Stochastic Multistage Lipschitz case, it would be interesting to prove a finite convergence result for Stochastic MILPs in the sampled case by further exploring the structure of the value functions in each stage.

Our experiments suggest that, at least for small-dimensional problems, the performance of the SLDP algorithm is reasonable.

Acknowledgements The second and third authors would like to express their sincere gratitude to professor Shabbir Ahmed for all his mentoring and contributions to this work. We hope that our effort in writing this paper could honor his memory.

The second author would like to express his gratitude to the Brazilian Independent System Operator (ONS) for its support for this research.

This research project was concluded while the third author was visiting Georgia Tech during a sabbatical leave from UFRJ. He would like to warmly thank the hospitality and the excellent environment of the ISyE institute.

References

1. Ackooij, W., Lopez, I.D., Frangioni, A., Lacalandra, F., Tahanan, M.: Large-scale unit commitment under uncertainty: an updated literature survey. Annals of Operations Research **271**(1), 11–85 (2018). DOI 10.1007/s10479-018-3003-z
2. Bertsimas, D., Weismantel, R.: Optimization over integers, vol. 13. Dynamic Ideas Belmont (2005)
3. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing. SIAM Review **59**(1), 65–98 (2017). DOI 10.1137/141000671. URL https://doi.org/10.1137/141000671
4. Birge, J.R., Louveaux, F.: Introduction to stochastic programming. Springer Science & Business Media (2011)
5. CarøE, C.C., Schultz, R.: Dual decomposition in stochastic integer programming. Operations Research Letters **24**(1-2), 37–45 (1999)
6. Carøe, C.C., Schultz, R.: Dual decomposition in stochastic integer programming. Operations Research Letters **24**(1-2), 37–45 (1999)
7. Cerisola, S., Latorre, J.M., Ramos, A.: Stochastic dual dynamic programming applied to nonconvex hydrothermal models. European Journal of Operational Research **218**(3), 687–697 (2012)
8. Conejo, A.J., Morales, L.B., Kazempour, S.J., Siddiqui, A.S.: Investment in Electricity Generation and Transmission: Decision Making Under Uncertainty, 1st edn. Springer Publishing Company, Incorporated (2016)
9. Conforti, M., Cornuéjols, G., Zambelli, G.: Integer programming, vol. 271. Springer (2014)
10. Freitas Paulo da Costa, B.: SLDP.jl: Stochastic Lipschitz Dynamic Programming extension for SDDP.jl (2019). URL https://github.com/bfpc/SLDP.jl
11. Costley, M., Feizollahi, M.J., Ahmed, S., Grijalva, S.: A rolling-horizon unit commitment framework with flexible periodicity. International Journal of Electrical Power & Energy Systems **90** (2017). DOI 10.1016/j.ijepes.2017.01.026
12. Dowson, O., Kapelevich, L.: SDDP.jl: a Julia package for Stochastic Dual Dynamic Programming. Optimization Online (2017). URL http://www.optimization-online.org/DB_HTML/2017/12/6388.html
13. Feizollahi, M.J., Ahmed, S., Sun, A.: Exact augmented lagrangian duality for mixed integer linear programming. Mathematical Programming **161**(1), 365–387 (2017). DOI 10.1007/s10107-016-1012-8. URL https://doi.org/10.1007/s10107-016-1012-8
14. Girardeau, P., Leclère, V., Philpott, A.B.: On the convergence of decomposition methods for multistage stochastic convex programs. Mathematics of Operations Research **40**, 130–145 (2015)

15. Gjelsvik, A., Belsnes, M., Haugstad, A.: Analgorithm for stochastic medium term hydrothermal scheduling under spot price uncertainty. In: Proceedings of the 13th Power Systems Computation Conference, pp. 1079–1085. Trondheim (1999)
16. Glassey, C.R.: Nested decomposition and multi-stage linear programs. Management Science **20**(3), 282–292 (1973). URL http://www.jstor.org/stable/2629718
17. Guigues, V.: Convergence analysis of sampling-based decomposition methods for risk-averse multistage stochastic convex programs. SIAM Journal on Optimization **26**, 2468–2494 (2016)
18. Gurobi Optimization, Inc.: Gurobi optimizer reference manual (2016). URL http://www.gurobi.com
19. Hjelmeland, M.N., Zou, J., Helseth, A., Ahmed, S.: Nonconvex medium-term hydropower scheduling by stochastic dual dynamic integer programming. IEEE Transactions on Sustainable Energy **10**(1), 481–490 (2019). DOI 10.1109/TSTE.2018.2805164
20. Hooker, J.N.: Integer programming duality, pp. 1657–1667. Springer US, Boston, MA (2009). DOI 10.1007/978-0-387-74759-0_289. URL https://doi.org/10.1007/978-0-387-74759-0_289
21. Kapelevich, L.: SDDiP.jl: SDDP extension for integer local or state variables (2018). URL https://github.com/lkapelevich/SDDiP.jl
22. Knueven, B., Ostrowski, J., Watson, J.P.: On mixed integer programming formulations for the unit commitment problem. Pre-print available at Optimization Online. (2018). URL http://www.optimization-online.org/DB_FILE/2018/11/6930.pdf
23. Laporte, G., Louveaux, F.V.: The integer l-shaped method for stochastic integer programs with complete recourse. Operations research letters **13**(3), 133–142 (1993)
24. Løkketangen, A., Woodruff, D.L.: Progressive hedging and tabu search applied to mixed integer (0, 1) multistage stochastic programming. Journal of Heuristics **2**(2), 111–128 (1996)
25. Lubin, M., Zadik, I., Vielma, J.P.: Mixed-integer convex representability. In: F. Eisenbrand, J. Könemann (eds.) Proceedings of the 19th Conference on Integer Programming and Combinatorial Optimization (IPCO 2017), *Lecture Notes in Computer Science*, vol. 10328, pp. 392–404 (2017)
26. Malherbe, C., Vayatis, N.: Global optimization of Lipschitz functions. In: D. Precup, Y.W. Teh (eds.) Proceedings of the 34th International Conference on Machine Learning, *Proceedings of Machine Learning Research*, vol. 70, pp. 2314–2323. PMLR, International Convention Centre, Sydney, Australia (2017). URL http://proceedings.mlr.press/v70/malherbe17a.html
27. Mayne, D.Q., Polak, E.: Outer approximation algorithm for nondifferentiable optimization problems. Journal of Optimization Theory and Applications **42**(1), 19–30 (1984)
28. Meewella, C., Mayne, D.: An algorithm for global optimization of lipschitz continuous functions. Journal of Optimization Theory and Applications **57**(2), 307–322 (1988)
29. Meyer, R.R.: On the existence of optimal solutions to integer and mixed-integer programming problems. Mathematical Programming **7**(1), 223–235 (1974)
30. Nowak, M.P., Römisch, W.: Stochastic lagrangian relaxation applied to power scheduling in a hydro-thermal system under uncertainty. Annals of Operations Research **100**(1-4), 251–272 (2000)
31. Pereira, M.V., Pinto, L.M.: Multi-stage stochastic optimization applied to energy planning. Mathematical programming **52**(1-3), 359–375 (1991)
32. Philpott, A., Wahid, F., Bonnans, F.: Midas: A mixed integer dynamic approximation scheme. Ph.D. thesis, Inria Saclay Ile de France (2016)
33. Rockafellar, R.T., Wets, R.J.B.: Scenarios and policy aggregation in optimization under uncertainty. Mathematics of Operations Research **16**(1), 119–147 (1991). DOI 10.1287/moor.16.1.119
34. Ruszczynski, A., Shapiro, A.: Stochastic programming, volume 10 of handbooks in operations research and management science (2003)
35. Shapiro, A., Dentcheva, D., Ruszczyński, A.: Lectures on stochastic programming: modeling and theory, second edition. SIAM (2014)
36. Shapiro, A., Tekaya, W., da Costa, J.P., Soares, M.P.: Risk neutral and risk averse stochastic dual dynamic programming method. European Journal of Operational Research **224**, 375–391 (2013)

37. Singh, K.J., Philpott, A.B., Wood, R.K.: Dantzig-wolfe decomposition for solving multistage stochastic capacity-planning problems. Operations Research **57**(5), 1271–1286 (2009)
38. Tahanan, M., van Ackooij, W., Frangioni, A., Lacalandra, F.: Large-scale unit commitment under uncertainty. 4OR **13**(2), 115–171 (2015). DOI 10.1007/s10288-014-0279-y. URL https://doi.org/10.1007/s10288-014-0279-y
39. Takriti, S., Birge, J.R., Long, E.: A stochastic model for the unit commitment problem. IEEE Transactions on Power Systems **11**(3), 1497–1508 (1996)
40. Thome, F., Pereira, M., Granville, S., Fampa, M.: Non-convexities representation on hydrothermal operation planning using sddp. URL: https://www.psr-inc.com/publications/scientific-production/papers/?current=t606, submitted (2013)
41. Trigeiro, W.W., Thomas, L.J., McClain, J.O.: Capacitated lot sizing with setup times. Management science **35**(3), 353–366 (1989)
42. Wolsey, L.A., Nemhauser, G.L.: Integer and combinatorial optimization, vol. 55. John Wiley & Sons (1999)
43. Zou, J., Ahmed, S., Sun, A.: Stochastic dual dynamic integer programming. Mathematical Programming (2018). URL https://doi.org/10.1007/s10107-018-1249-5