# Stochastic Dynamic Linear Programming: A Sequential Sampling Algorithm for Multistage Stochastic Linear Programming

Harsha Gangammanavar*[1] and Suvrajeet Sen†[2]

[1]Department of Engineering Management, Information, and Systems, Southern Methodist University, Dallas TX

[2]Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA

First submission: October, 2019. Current version: May 3, 2021.

**Abstract**

Multistage stochastic programming deals with operational and planning problems that involve a sequence of decisions over time while responding to an uncertain future. Algorithms designed to address multistage stochastic linear programming (MSLP) problems often rely upon scenario trees to represent the underlying stochastic process. When this process exhibits stagewise independence, sampling-based techniques, particularly the stochastic dual dynamic programming (SDDP) algorithm, have received wide acceptance. However, these sampling-based methods still operate with a deterministic representation of the problem which uses the so-called sample average approximation. In this work, we present a sequential sampling approach for MSLP problems that allows the decision process to assimilate newly sampled data recursively. We refer to this method as the stochastic dynamic linear programming (SDLP) algorithm. Since we use sequential sampling, the algorithm does not necessitate a priori representation of uncertainty, either through a scenario tree or sample average approximation, both of which require a knowledge/estimation of the underlying distribution. This method constitutes a generalization of the Stochastic Decomposition (SD) algorithm for two-stage SLP models. The approximations used within SDLP may be viewed either through the lens of proximal methods or via regularization. Furthermore, we introduce the notion of basic feasible policies which provide a piecewise-affine solution discovery scheme, which is embedded within the optimization algorithm to identify incumbent solutions used in the context of proximal iterations. Finally, we show that the SDLP algorithm provides a sequence of decisions and corresponding value function estimates along a sequence of state trajectories that asymptotically converge to their optimal counterparts, with probability one.

*harsha@smu.edu

†s.sen@usc.edu

# 1 Introduction

Many practical applications require sequences of decisions to be made under evolving and often uncertain conditions. Multistage stochastic programming (SP) is an effective approach used to guide decision-making. A variety of fields ranging from traditional production systems, hydro-electric reservoir scheduling, and financial planning, to emerging applications in electricity grids with renewable generation and revenue management have successfully used multistage SP.

Multistage stochastic linear programming (MSLP) models with recourse were used to formulate the early applications of multistage SP. These MSLP models were solved using extensions of the L-shaped method [40] such as the Nested Benders Decomposition (NBD) method [3], the scenario decomposition method [28], and the specialization of a primal-dual proximal point method in the form of progressive hedging (PH) algorithm [33]. A common feature across all these algorithms is the use of an approximate deterministic representation of uncertainty through scenario trees (i.e., precedence relations) built using scenario generation techniques (e.g., [8]). When the underlying stochastic process becomes complicated, their deterministic representation may result in large, unwieldy scenario trees. To handle such scenario trees in a computationally viable manner, one may have to resort to scenario reduction methods (e.g., [9]). For models in which uncertainty is represented using stagewise independent random variables, [29] proposed the stochastic dual dynamic programming (SDDP) algorithm. All these multistage algorithms are designed to solve a base model with an uncertainty representation involving a finite sample space and known probability distribution (a scenario tree or a sample average approximation (SAA)). The model resulting from such a representation is deterministic in nature. In this regard, we refer to these methods as deterministic decomposition (DD)-based methods.

In problems where reliable knowledge of uncertainty is not available, an approach that does not rely on exact probabilistic information is desirable. For MSLP models, the first inexact bundle method proposed in [36] called the multistage stochastic decomposition (MSD) achieves this objective. This algorithm is a dynamic extension of the regularized version of the two-stage stochastic decomposition (2-SD) algorithm [19]. It accommodates very general stochastic processes, possibly with time correlations, through a nodal formulation which requires only a "layout" of a scenario tree, and an oracle that provides transitions between nodes. A standard scenario tree formulation is a special case of such a mechanism. When the stochastic process exhibits interstage independence, a time-staged formulation (as opposed to nodal scenario-tree formulation) is more convenient. With this in mind, we present a sequential sampling-based algorithm that addresses decision-making under stagewise independent stochastic processes.

## 1.1 Contributions

We refer to our sequential sampling-based approach for MSLP with interstage independence as the *stochastic dynamic linear programming* (SDLP) algorithm. In light of the existing deterministic and stochastic decomposition-based methods, the contributions of this work are as follows.

**An Algorithm for Stagewise Independent MSLP Models**

SDLP harnesses the advantages offered by both the interstage independence of stochastic processes (like SDDP) as well as the sequential sampling methods (like 2-SD) to build an algorithm which achieves asymptotic convergence while sampling only a small number of scenarios (e.g., one) in any iteration. The algorithm is designed for a state variable formulation of MSLP models. Despite the focus on MSLP, SDLP integrates some features of dynamic programming, such as the generation of policies (instead of simply one implementable decision). In this sense, it can support out-of-sample scenarios for decisions. Obversely unlike dynamic programming, where descent properties of algorithms are not a common feature, SDLP is designed as a recursive stagewise proximal algorithm, which in the long-run, produces a type of stochastic descent, much like many standard optimization algorithms [22]. There are many other distinguishing features of SDLP when compared with DD-based methods. The principal differences are highlighted below.

- *Static v. Dynamic Instances:* DD-based methods, including SDDP, can be classified as external sampling methods where the uncertainty representation step precedes the optimization step. In such methods, one begins by first identifying the nodes (observations) and the probability of observing the nodes at each stage, which is then used to set up the MSLP instance. DD-based methods aims to optimize the resulting MSLP instance. When sampling is used to obtain the uncertainty representation, the decisions provided DD-based methods are justified using the mathematical theory of SAA. The uncertainty representation (observations and probabilities) is explicitly used in computing the cost-to-go value function approximations. In contrast to that, SDLP accommodates the possibility of observing new scenarios during the course of the algorithm. As a result, the uncertainty representation, and therefore, the MSLP instance evolves dynamically with the introduction of new scenarios.

- *Implications of Sampling:* SDLP completes forward and backward recursion computations along a single sample-path that is generated independently in every iteration. Although this feature is reminiscent of SDDP variants that incorporate sampling in the forward and backward passes, there are two main differences: (a) Since SDDP operates with a fixed uncertainty representation, the sampled paths selected for forward and backward pass calculations are a subset of sample-paths used in the uncertainty representation. On the other hand, the sample-path used in the forward and backward recursions of SDLP may include observations that have not been encountered before. (b) Since the number of observations increases, the piecewise affine approximations need to be updated to ensure that they continue to provide a lower bound for the dynamically MSLP instance.

- *Asymptotic Behavior:* Unlike DD-based methods which can recover the cost-to-go value functions in finitely many steps, we show asymptotic optimality of SDLP using primal-dual relationships that are fundamental to mathematical programming. In other words, SDLP approximations are not finitely convergent. Finally, the sequential sampling approach of SDLP permits a modeler to pause optimization computations so as to perform auxiliary tests (e.g., out-of-sample simulations and confidence interval predictions for a given policy)), and then resume the computations from where it was stopped (if necessary), without

having to restart from scratch.

The distinguishing features identified above are all consequences of sequential sampling. In the two-stage setting (as in the regularized 2-SD algorithm of [19]), the recourse function is a deterministic optimization problem, a linear program to be specific. On the other hand, in the multistage setting, the recourse functions in non-terminal stages involve dynamically updating nested SAAs. Therefore, MSD as well as SDLP include provisions to address the stochasticity in value function approximations. Since MSD works with a layout of a scenario tree, it uses a node-specific approximation. With stagewise independent stochastic processes, the future value function approximations are shared by all observations at a stage. Therefore, updates along the current sample-path perturb the future approximations for all observations. This marks a subtle but significant difference in the way the approximations are constructed and updated in SDLP when compared to MSD. This also has an impact on the convergence analysis.

**A Policy to Identify Incumbent Solutions**

Due to the use of quadratic regularization in non-terminal stages of SDLP, we can view the SDLP algorithm design as a recursive stagewise proximal algorithm. In addition to producing stochastic descent that is critical for convergence analysis, the inclusion of the proximal term allows one to limit the size of the stage optimization problem to at most $n_t + 3$ "cuts", where $n_t$ is the number of decision variables in stage $t$. The quadratic proximal terms at all non-terminal stages are centered around the "incumbent" decisions[1] that are maintained for all sample-paths discovered during the algorithm. Maintaining and updating these incumbent solutions becomes cumbersome as the number of sample-paths increases. To address this critical issue we develop the notion of a piecewise-affine policy which is used to identify incumbent solutions for out-of-sample scenarios (new sample-paths) generated sequentially within the algorithm. Such a policy is referred to as a basic feasible policy (BFP). A BFP is based on the optimal bases of the approximate stage problems that are solved during the course of the algorithm. While the BFP designed in this paper is used to identify incumbent solutions for SDLP, the general idea underlying a BFP can also be adopted for other multistage SP algorithms, including SDDP.

This paper also serves as a companion to our earlier work [12] by providing the theoretical corroboration of the empirical evidence presented there. In [12], SDLP was used for controlling distributed storage devices in power systems with significant renewable resources. Computational experiments conducted on large-scale instances showed that such an approach provides solutions that are statistically indistinguishable from solutions obtained using SDDP, while significantly reducing the computational time. These improvements (in comparison to SDDP) can be attributed to two key features of the SDLP algorithm. Firstly, the forward and backward recursion calculations are carried out only along one sample-path. This significantly reduces the total number of optimization problems solved in any iteration. Secondly, the fixed-sized optimization problems at each stage due to the use of quadratic proximal term implies that the computational effort per iteration (necessary to solve stagewise optimization problems) does not

[1] In SP algorithms, especially methods based on 2-SD, an "incumbent decision" is one for which the predicted objective value appears to be the best (at the current iteration). When predictions change, the incumbent decision must also be updated.

increase with iterations. Moreover, it has been recently established that 2-SD provides a sequence of incumbent solutions that converges to the optimal solution at a sublinear convergence rate [27]. It is important to emphasize that this result pertains to a solution sequence, rather than the objective function sequence which was already known for first-order methods such as stochastic approximation. Because the design and analysis of this paper mirror that of 2-SD, we suspect that a similar rate of convergence may be possible for SDLP as well. However, a detailed convergence rate analysis is beyond the scope of the current paper.

### Organization

The remainder of the paper is organized as follows. In §2 we present the MSLP formulation used in this paper. We begin §3 with an overview of the DD-based MSLP methods and then provide a detailed description of the SDLP algorithm. We present the convergence analysis of SDLP in §4. Our presentation will emphasize the differences in approximations used within deterministic and stochastic decomposition-based methods.

## 2 Notation and Formulation

We consider a system where sequential decisions are made at discrete decision epochs denoted by the set $\mathcal{T} := \{0, \ldots, T\}$ over a finite horizon ($T < \infty$). In the interest of brevity (especially because there are many subscripted elements) we denote by $t+$ and $t-$ the succeeding and preceding time periods $(t+1)$ and $(t-1)$, respectively. We use a subscript $[t]$ to denote the history of the stochastic process $\{v_t\}_{t=0}^{T}$ until (and including) stage $t$, i.e., $v_{[t]} = v_0, v_1, \ldots, v_t$. Likewise, we use $v_{(t+)}$ to denote the process starting from stage $t+1$ until the end of horizon (stage $T$), i.e., $v_{(t+)} = v_{t+1}, \ldots, v_T$. We use $\langle \cdot, \cdot \rangle$ to denote the inner product of vectors (e.g., $\langle v, w \rangle = v^\top w$) and the product of a matrix transpose and a vector, i.e., $\langle M, v \rangle = M^\top v$.

Commonly in SP, MSLP models are formulated without state variables, focusing only on decisions in each stage. However in many applications, especially those involving dynamic systems, it is common to use the state variable description of system evolution. Because we expect SDLP to be able to provide decision support for such systems, it is advisable to use a state variable formulation. This approach is also common in the dynamic programming community. In this regard, we use a state variable $s_t := (x_t, \omega_t) \in \mathcal{S}_t$ to describe the system at stage $t$. This state variable is comprised of two components: $x_t \in \mathcal{X}_t$ is the endogenous state of the system and $\omega_t \in \Omega_t$ captures the exogenous information revealed in interval $(t-1, t]$. A stochastic process over which the decision-maker cannot exert any control drives the exogenous state evolution. For example, the exogenous state variable may represent a weather phenomenon like wind speed, or a market phenomenon like the price of gasoline. The evolution of the endogenous state, on the other hand, can be controlled by an algorithm through decisions $u_t$ and is captured by stochastic linear dynamics:

$$x_{t+} = \mathcal{D}_{t+}(x_t, \omega_{t+}, u_t) = a_{t+} + A_{t+}x_t + B_{t+}u_t. \tag{1}$$

Here, $(a_{t+}, A_{t+}, B_{t+})$ are components of the exogenous information vector $\omega_{t+}$ corresponding to the next time period.

To characterize the exogenous process $\{\tilde{\omega}_t\}_{t=1}^T$, we use $(\Omega, \mathcal{F}, \mathbb{P})$ to denote a filtered probability space. Here, $\Omega = \Omega_1 \times \ldots \times \Omega_T$ denotes the set of outcomes and $\omega_t$ denotes an observation of the random variable $\tilde{\omega}_t$. The $\sigma$-algebras $\mathcal{F}_t \subseteq \mathcal{F}$ represent the data available to the decision-maker at time $t$, which satisfy $\mathcal{F}_t \subseteq \mathcal{F}_{t'}$ for $t < t'$. The exogenous data $\omega_t$ includes components of $(a_t, A_t, B_t)$ that appear in (1) and parameters $(b_t, C_t)$ in the right-hand side of the constraints at stage $t$.

With these notations the multistage program is stated in the following recursive form for all $t \in \mathcal{T}$:

$$\begin{aligned} h_t(s_t) = \langle c_t, x_t \rangle + \min \ & \langle d_t, u_t \rangle + \mathbb{E}[h_{t+}(\tilde{s}_{t+})] \\ & s.t.\ u_t \in \mathcal{U}_t(s_t) := \{u_t | D_t u_t \leq b_t - C_t x_t, u_t \geq 0\}, \end{aligned} \tag{2}$$

where $\tilde{x}_{t+} = \mathcal{D}_{t+}(x_t, \tilde{\omega}_{t+}, u_t)$. The set of feasible decisions $\mathcal{U}_t(s_t)$ is a convex polyhedron which depends on the input state. The initial state $x_0$ is assumed to be given and therefore, the stage-0 (henceforth known as the root-stage) problem has deterministic input. In our finite horizon framework, we assume that the terminal cost $h_{T+}(s_{T+})$ is known for all $s_{T+}$ (or negligible enough to be set to 0). The expectation is taken with respect to the exogenous stochastic process $\tilde{\omega}_{(t)}$ over the remainder of the horizon.

In general, MSLP problems are PSPACE-hard [10, 17] and require exponential effort in horizon $T$ for provably tight approximations with high probability. To keep our presentation consistent with our algorithmic goals, we make the following assumptions:

(A1) The set of root-stage decisions $\mathcal{U}_0$ is compact.

(A2) The sets $\mathcal{X}_t$ are compact and the complete-recourse assumption is satisfied at all non-root stages. That is, the feasible set $\mathcal{U}_t(s_t)$ is non-empty for all state trajectories $s_t$ with $x_t$ satisfying (1) for all $t \in \mathcal{T} \setminus \{0\}$.

(A3) The constraint matrices $D_t$ are fixed and have full row rank.

(A4) Zero provides the lower bound on all cost-to-go value functions.

(A5) The stochastic process for exogenous information is stagewise independent and its support is finite.

These assumptions provide a special structure and are fairly standard in the SP literature ([31, 36]). The fixed recourse assumption (A3) implies that the recourse matrix $D_t$ does not depend on exogenous information. As for assumption (A4), note that most loss functions used in engineering applications and statistical learning obey this property. For situations in which this assumption is not satisfied, one can perform a pre-processing step as follows: first estimate a lower bound on the optimal objective function value for each stage, and then, add the absolute value of the most negative stagewise lower bound to all stages. Introducing such a constant into the objective function does not alter the optimal decisions while rendering the validity of (A4). The finite support assumption (A5) on exogenous information ensures that $\mathcal{F}_t$ is finite. We note that the algorithms presented here can be extended, after some refinement, to settings where some of the above assumptions can be relaxed. For instance, certain extensions to Markovian stochastic

processes can be envisioned. However, a detailed treatment of these extensions is beyond the scope of this paper.

## 3 Stochastic Dynamic Linear Programming

The fundamental difficulty of solving (2) is associated with the *nested multidimensional* integral for computing the expectation. The most direct approach involves incorporating simulation to estimate the expected recourse function. Doing so results in the so-called SAA problem. In this case, we can view the support of $\Omega_{t+}$ as consisting of a simulated sample $\Omega_{t+}^N := \{\omega_{t+}^1, \omega_{t+}^2, \ldots, \omega_{t+}^N\}$, where each observation vector $\omega_{t+}^n$ has the same probability $p(\omega_{t+}^n) = (1/N)\ \forall n = 1, \ldots, N$.

When the underlying stochastic process has a finite support or when the SAA is employed, DD-based solution methods have been successfully applied. DD-based algorithms originally developed for 2-SLP (the L-shaped method [40]) have been extended to successive stages of dynamic linear programs. One of the early successes was reported in [3], where the classical two-stage Benders decomposition algorithm was extended to multiple stages. This procedure has subsequently come to be known as the NBD algorithm. The starting point of this algorithm is the scenario tree representation of the underlying uncertainty where all possible outcomes and their interdependence are represented as nodes on a tree.

It is well known that the number of nodes in the scenario tree grows exponentially with the number of stages, and therefore, the need to visit all the nodes in the scenario tree significantly increases the computational requirements of the NBD algorithm. Pereira and Pinto [29] provided a sampling-based approach to address this issue in the SDDP algorithm. Like NBD, SDDP creates an outer approximation of the stage value function using subgradient information by performing a forward and a backward pass in each iteration. However, it avoids the intractability of scenario trees by assuming that the stochastic process is stagewise independent. While the algorithm traverses forward using sampling, the approximations are created on the backward pass similar to deterministic Benders type cuts. The interstage independence assumption allows these cuts to be shared across different states within a stage. A variety of features such as, Cut sharing under special stagewise dependency [23], the use of sampling on the backward pass [26, 31], regularization [1, 15], and the inclusion of risk measures [16, 30] have extended the capabilities of the original algorithm [29] and have contributed to the success of SDDP. The abridged nested decomposition algorithm in [7] and the cutting plane and partial sampling algorithm proposed in [5] are other sampling-based methods which are similar in flavor to SDDP. A SAA-based SDDP algorithm was analyzed in [38]. Relationships between various algorithmic approaches are summarized in Figure 1.

In contrast to the SDDP algorithm, where a fixed sample is used at each stage, the SDLP algorithm will generate approximations that are based on sample average functions constructed using a sample whose size increases with iterations. The sequential nature of introducing new observations into the sample requires additional care within the algorithm design, particularly in the backward pass when approximations are generated. We present these details next.

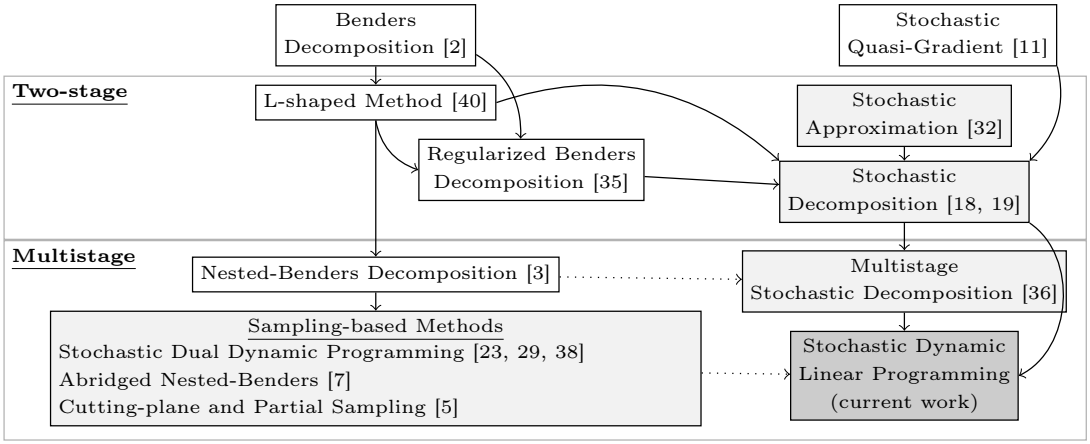


Figure 1: Multistage Stochastic Linear Programming Algorithms

## Algorithm Design

An iteration of SDLP involves two principal steps: forward and backward recursion. The use of forward and backward recursions is common to almost all the multistage SP algorithms (except those based on progressive hedging [33]). The forward-backward recursion approach to solving dynamic optimization problems can be traced back to the differential dynamic programming (DDP) algorithm [24]. The SDLP algorithm is closely related to the DDP algorithm, in the sense that we create locally accurate approximations of the subdifferential, whereas DDP works with quadratic approximations of smooth deterministic dynamic control problems. The algorithmic constructs of SDLP are designed to accommodate the inherent non-smoothness of MSLP models, and of course, stochasticity. We present details of these in iteration $k$ of the algorithm. Note that except for the manner in which the stochastic phenomenon is represented, we make essentially the same assumptions as the SDDP algorithm.

### 3.1 Forward Recursion

The forward recursion begins by solving the following quadratic regularized optimization problem:

$$\min_{u_0 \in \mathcal{U}_0} \left\{ f_0^{k-1}(s_0, u_0) + \frac{\sigma}{2} \|u_0 - \hat{u}_0^{k-1}\|^2 \right\}. \tag{3}$$

Here, the proximal parameter $\sigma \geq 1$ is assumed to be given. We denote the optimal solution of the above problem as $u_0^k$ and refer to it as the candidate solution. The incumbent solution $\hat{u}_0^k$ used in the proximal term is similar to that used in the regularized 2-SD [19] algorithm. This is followed by simulating a sample-path $\omega_{(0)}^k$ that is generated independently of previously observed sample-paths. The remainder of the forward recursion is carried out only along this simulated sample-path in two passes - a prediction pass and an optimization pass.

#### Prediction Pass

At all non-terminal stages we use a regularized stage optimization problem which is centered around the incumbent solution. The goal of the prediction pass is to ensure that the incumbent

solutions, and the corresponding incumbent states, satisfy the underlying model dynamics in (1) along the current sample-path $\omega^k_{(0)}$. Given the initial state $x_0$, the prediction pass starts by using the root-stage incumbent solution $\hat{u}^k_0$ and computing the incumbent state for stage-1 as: $\hat{x}^k_1 = \mathcal{D}_1(x_0, \omega^k_1, \hat{u}^k_0)$. At subsequent stages, we use the BFP to identify the incumbent solutions as $\hat{u}^k_t(\hat{s}^k_t) = \mathcal{M}_t(\hat{s}^k_t)$. Here, $\mathcal{M}_t : \mathcal{S}_t \to \mathcal{U}_t$ is a vector valued mapping that takes the state vector $s_t$ as an input and maps it on to a solution in $\mathcal{U}_t(s_t)$. We postpone the details of specifying BFP to §3.3.2 and continue with the algorithm description here. We proceed by computing the incumbent state using (1) and identifying the incumbent solution using the BFP for the remainder of the horizon. At the end of the prediction pass, we have an incumbent state $\{\hat{x}^k_t\}$ and solution $\{\hat{u}^k_t\}$ trajectories[2] that satisfy state dynamics in (1) over the entire horizon.

**Optimization Pass**

After completing the prediction pass, the optimization pass is carried out to simulate candidate solutions along the current sample-path $\omega^k_{(0)}$ for all $t \in \mathcal{T} \setminus \{T\}$:

$$u^k_t \in \operatorname{argmin}\{f^{k-1}_t(s^k_t, u_t) + \frac{\sigma}{2}\|u_t - \hat{u}^k_t(\hat{s}^k_t)\|^2 \mid u_t \in \mathcal{U}_t(s^k_t)\}. \tag{4}$$

Here $f^{k-1}_t(s_t, u_t)$ is the current approximation of the cost-to-go value function. Structurally, $f^{k-1}_t$ is a piecewise affine and convex function and is similar to the approximations used in the SDDP algorithm. However, each individual piece is a minorant[3] generated using certain sample average functions. The candidate decision for a particular stage is used to set up the subsequent endogenous state $x^k_{t+} = \mathcal{D}_{t+}(x^k_t, \omega^k_{t+}, u^k_t)$ and thus the input state $s^k_{t+}$. We refer to the decision problem in (4) as *Time-staged Decision Simulation* at stage $t$ ($\text{TDS}_t$). This completes the optimization pass, and hence the forward recursion, for the current iteration. At the end of forward recursion, we have the incumbent trajectory $\{\hat{x}^k_t\}$ and the candidate trajectory $\{x^k_t\}$ which will be used for updates during the backward recursion.

## 3.2 Backward Recursion

The primary goal in the backward recursion procedure is to update the cost-to-go value function approximations $f^{k-1}_t$ at all non-terminal stages. As the name suggests these calculations are carried out backward in time, starting from the terminal stage to the root-stage, along the same sample-path that was observed during the forward recursion. These calculations are carried out for both the candidate as well as the incumbent trajectories.

In both the DD and SD-based approaches, the value function is approximated by the point-wise maximum of affine functions. However, the principal difference between these approaches lies in how the expected value function is approximated. In DD-based methods, it is the true

[2]We will use the more explicit notation $\hat{u}^k_t(\hat{s}^k_t)$ that shows the dependence of the incumbent solution on the input incumbent state only when it does not add undue notational burden. In most cases, we will simply use $\hat{u}^k_t$ for incumbent solution.

[3]Since the approximations generated in sequential sampling-based methods are based on statistical estimates which are updated iteratively, we use the term "minorant" to refer the lower bounding affine functions. This usage follows its introduction in [36] and is intended to distinguish them from the more traditional terminology of "cuts" in DD-based methods.

expected value function which requires the knowledge of the probability distribution or a SAA with a fixed sample. On the other hand, SD-based methods create successive approximations $\{f_t^k\}$ (for $t < T$) that provide a lower bound on a SAA using only $k$ observations in iteration $k$, and therefore, satisfies:

$$f_t^k(s_t, u_t) - \langle c_t, x_t\rangle - \langle d_t, u_t\rangle \leq \widehat{H}_{t+}^k(s_{t+}) := \sum_{\omega_{t+}\in\Omega_{t+}^k} p^k(\omega_{t+}) h_{t+}(x_{t+}, \omega_{t+}), \tag{5}$$

where $x_{t+}$ is the endogenous state obtained from (1) with $(x_t, \omega_{t+}, u_t)$ as input, for all $\omega_{t+} \in \Omega_{t+}^k$ and $u_t \in \mathcal{U}_t(s_t)$. The quantity $p^k(\omega_{t+})$ in (5) measures the relative frequency of an observation which is defined as the number of times the exogenous state $\omega_{t+}$ is observed ($\kappa^k(\omega_{t+})$) over the number of iterations ($k$). This quantity approximates the unconditional probability of exogenous information at stage $t+$, and is updated as follows. Given the current sample-path $\omega_{(0)}^k$, a collection of observations at a non-root stage $t$ ($t > 0$) is updated to include the latest observation $\omega_t^k$ as: $\Omega_t^k = \Omega_t^{k-1} \cup \omega_t^k$. The observation count is also updated as: $\kappa^k(\omega_t) = \kappa^{k-1}(\omega_t) + \mathbb{1}_{\omega_t=\omega_t^k}$, for all $\omega_t \in \Omega_t^k$. Using these counts, the observation frequency for $\omega_t \in \Omega_t^k$ is given by: $p^k(\omega_t) = \frac{\kappa^k(\omega_t)}{k}$. Notice the superscript $k$ (iteration count) that is used in our notation of the SAA function $\hat{H}_{t+}^k$, the collection of observations $\Omega_{t+}^k$, and the observation frequency $p^k$. This is intended to convey the sequential nature of sampling in SDLP.

### 3.2.1 Terminal Stage Approximation

At the terminal stage, recall that $\mathbb{E}[h_{T+}(s_{T+})] = 0$, and the value function $h_T$ is the value of a deterministic linear program for a given state input $s_T$. The sample average $\widehat{H}_T^k(s_T) = \sum_{\omega_T\in\Omega_T^k} p^k(\omega_T) h_T(s_T)$ provides an unbiased estimate of $\mathbb{E}[h_T(\tilde{s}_T)]$. Hence, the value function at the penultimate stage ($t = T-1$) can be approximated using a procedure similar to the one employed in the 2-SD algorithm.

In this procedure, a subproblem corresponding to the current observation $\omega_T^k$ is setup and solved. This subproblem uses $s_T^k = (x_T^k, \omega_T^k)$ as input, where $x_T^k = \mathcal{D}_T(x_{T-1}, \omega_T^k, u_{T-1}^k)$. Let the optimal dual solution obtained be denoted as $\pi_T^k(\omega_T^k)$, and is used to augment the collection of previously discovered dual vertices: $\Pi_T^k = \Pi_T^{k-1} \cup \pi_T^k(\omega_T^k)$. For other observations in $\Omega_T^k$, i.e., $\omega_T \in \Omega_T^k$ and $\omega_T \neq \omega_T^k$, we identify the best dual vertex in $\Pi_T^k$ using the "argmax" operation as in the case of 2-SD algorithm [18]. This operation is as follows:

$$\pi_T^k(\omega_T) \in \text{argmax}\{\langle \pi_T, (b_T - C_T x_T) \mid \pi_T \in \Pi_T^k\rangle\}. \tag{6}$$

Using the dual vertices $\{\pi_T^k(\omega_T)\}_{\omega_T\in\Omega_T^k}$, we compute the lower bounding affine function $\ell_T^k(s_T) := \alpha_T^k(\omega_T) + \langle \beta_T^k(\omega_T), x_T\rangle$, where

$$\alpha_T^k(\omega_T) = \langle b_T, \pi_T^k(\omega_T)\rangle; \quad \beta_T^k(\omega_T) = c_T - \langle C_T, \pi_T^k(\omega_T)\rangle. \tag{7}$$

The above calculations are also carried out for the incumbent state $\hat{s}_t^k$, resulting in the affine function $\hat{\ell}_T^k(s_T) = \hat{\alpha}_T^k(\omega_T) + \langle \hat{\beta}_T^k(\omega_T), x_T\rangle$. The set of affine functions thus obtained ($\mathcal{J}_T^k = \mathcal{J}_T^{k-1} \cup \{\ell_T^k(s_T), \hat{\ell}_T^k(s_T)\}$) provides the piecewise affine lower bounding function to the value function $h_T(s_T)$ that is given by:

$$h_T^k(s_T) = \max_{j\in\mathcal{J}_T^k(\omega_T)} \{\ell_T^j(s_T) = \alpha_T^j(\omega_T) + \langle \beta_T^j(\omega_T), x_T\rangle\}. \tag{8}$$

The above function provides an outer linearization of the terminal value function.

### 3.2.2 Non-terminal Stage Approximation

When updating the approximations at a non-terminal stage $t$, we have access to the minorants at stage $t+$ (recall that the value functions are being updated recursively backwards from the terminal stage). Using these we can define:

$$H_t^k(s_t) := \langle c_t, x_t\rangle + \min_{u_t \in \mathcal{U}_t(s_t)} \langle d_t, u_t\rangle + \sum_{\omega_{t+} \in \Omega_{t+}^k} p^k(\omega_{t+}) h_{t+}^k(s_{t+}), \tag{9}$$

where $s_{t+} = (\mathcal{D}_{t+}(x_t, \omega_{t+}, u_t), \omega_{t+})$ for all $\omega_{t+} \in \Omega_{t+}^k$. The expression in (9) represents a sample average computed over the current set of observations $\Omega_{t+}^k$ at stage $t+$ at an arbitrary input state $s_t$. Since we use lower bounding approximations $h_{t+}^k$ in building this sample average, this sampled estimate is biased. The stage approximation is updated using a lower bound to the above sample average function, and hence, is biased as well.

In order to compute this lower bound, notice that we can obtain the subgradient, i.e., $\beta_{t+}^k(\omega_{t+}) \in \partial h_{t+}^k(\mathcal{D}_{t+}(x_t^k, \omega_{t+}, u_t^k), \omega_{t+})$ using the collection of affine functions $\mathcal{J}_{t+}^k(\omega)$ for all observations $\omega_{t+} \in \Omega_{t+}$ (see §3.3.1 for details). Let $\alpha_{t+}^k(\omega_{t+})$ be the corresponding intercept term. Using these, a valid lower bound to the sample average function in (9) can be written as:

$$H_t^k(s_t) \geq \langle c_t, x_t\rangle + \min_{u_t \in \mathcal{U}_t(s_t)} \langle d_t, u_t\rangle + \sum_{\omega_{t+} \in \Omega_{t+}^k} p^k(\omega_{t+}) \Big[\alpha_{t+}^k(\omega_{t+}) + \langle \beta_{t+}^k(\omega_{t+}), x_{t+}\rangle\Big]. \tag{10}$$

Substituting the state dynamics equation in (1), and dualizing the linear program on the right-hand side of the above inequality, we obtain:

$$\begin{aligned} H_t^k(s_t) \geq \langle c_t, x_t\rangle + \bar{\alpha}_{t+}^k(\omega_t^k) + \langle \bar{\beta}_{t+}^k(\omega_t^k), x_t\rangle + \\ \max\ \{\langle \pi_t, (b_t - C_t x_t)\rangle \mid \langle D_t, \pi_t\rangle \leq \bar{\rho}_{t+}^k(\omega_t^k),\ \pi_t \leq 0\}, \end{aligned} \tag{11}$$

where,

$$\bar{\alpha}_{t+}^k(\omega_t^k) = \sum_{\omega_{t+} \in \Omega_{t+}^k} p^k(\omega_{t+}) [\alpha_{t+}^k(\omega_{t+}) + \langle \beta_{t+}^k(\omega_{t+}), a_{t+}\rangle], \tag{12a}$$

$$\bar{\beta}_{t+}^k(\omega_t^k) = \sum_{\omega_{t+} \in \Omega_{t+}^k} p^k(\omega_{t+}) \langle \beta_{t+}^k(\omega_{t+}), A_{t+}\rangle, \quad \text{and} \tag{12b}$$

$$\bar{\rho}_{t+}^k(\omega_t^k) = d_t + \sum_{\omega_{t+} \in \Omega_{t+}^k} p^k(\omega_{t+}) \langle \beta_{t+}^k(\omega_{t+}), B_{t+}\rangle. \tag{12c}$$

We refer to the linear program on the right-hand side of inequality in (11) as the *stagewise-dual approximation* at stage $t$ and denote it as ($\text{SDA}_t^k$). Let $\pi_t^k(\omega_t^k)$ denote the optimal dual solution obtained by solving ($\text{SDA}_t^k$) with $s_t^k$ as input. Using this multiplier we obtain a lower bounding affine function $\ell_t^k(s_t) = \alpha_t^k(\omega_t^k) + \langle \beta_t^k(\omega_t^k), x_t\rangle$ with the following coefficients:

$$\alpha_t^k(\omega_t^k) = \langle \pi_t^k(\omega_t^k), b_t\rangle + \bar{\alpha}_{t+}^k(\omega_t^k)\ ; \quad \beta_t^k(\omega_t^k) = c_t - \langle C_t, \pi_t^k(\omega_t^k)\rangle + \bar{\beta}_{t+}^k(\omega_t^k). \tag{13}$$

Similar calculations using $\hat{\pi}_t^k(\omega_t^k)$, an optimal solution to the (SDA$_t^k$) with $\hat{s}_t^k$ as input, yields an incumbent affine function $\hat{\ell}_t^k(s_t)$. As before, these functions are included in a collection of affine functions to obtain the updated set $\mathcal{J}_t^k(\omega_t^k)$.

While it is true that the latest affine functions satisfy $H_t^k(s_t) \geq \ell_t^k(s_t)$, the same may not hold for affine functions generated at earlier iterations. Hence, it is possible that there exists a $j \in \mathcal{J}_t^k(\omega_t)$ such that the affine function $\ell_t^j(s_t)$ may not provide a lower bound to the current sample average $H_t^k(s_t)$. In keeping with the updates of 2-SD [18], the old minorants need to be updated as the sample average estimate changes during the course of the algorithm. Under assumption (A4), this is achieved by scaling down the previously generated affine functions. In the two-stage case, 2-SD minorants are updated by multiplying the coefficients by $(k-1)/k$. In the multistage case, the minorants are updated[4] as follows

$$h_t^k(s_t) = \max \left\{ \left\{ \left( \frac{k-1}{k} \right)^{T-t} \ell_t^j(s_t) \right\}_{j \in \mathcal{J}_t^{k-1}(\omega_t)}, \ \ell_t^k(s_t), \ \hat{\ell}_t^k(s_t) \right\}. \tag{14}$$

Notice that both the candidate and incumbent affine functions generated in previous iterations are treated similarly while scaling down.

We use these updated minorants to obtain the stage objective function as follows:

$$f_t^k(s_t, u_t) = \langle c_t, x_t \rangle + \langle d_t, u_t \rangle + \sum_{\omega_{t+} \in \Omega_{t+}^k} p^k(\omega_{t+}) h_{t+}^k(s_{t+}), \tag{15}$$

where $s_{t+} = (\mathcal{D}_{t+}(x_t, \omega_{t+}, u_t), \omega_{t+})$, for all $\omega_{t+} \in \Omega_{t+}^k$. Similar updates are carried out at all the non-terminal stages by progressing backwards to the root-stage along the same sample-path that was used in the forward recursion. The backward recursion for iteration $k$ is said to be complete once the root-stage objective function is updated. The sequentially ordered steps of SDLP algorithm are presented in Algorithm 1.

### 3.2.3 Comparison of DD and SD-based approximations

The complete recourse assumption ensures that the dual feasible set is non-empty and the optimal dual solution $\pi_T$ is an extreme point of $\{\pi_T \mid \langle D_t, \pi_t \rangle \leq d_T, \pi_T \leq 0\}$. There are finitely many of these extreme points, and hence, coefficients for the terminal stage approximations for both the DD-based and SD-based algorithms take finitely many values.

In DD-based multistage algorithms, including SDDP, the cut coefficients belong to a finite set at stage $t+$, and therefore, there exists an iteration $k'$ such that the set of coefficients $\mathcal{J}_{t+}^k(\omega) = \mathcal{J}_{t+}^{k'}(\omega)\ \forall \omega \in \Omega_{t+}$ for $k > k'$. Consequently, the dual feasible set of the problem solved in the backward pass is a convex polytope that does not change for iterations $k > k'$. Since there are finite number of extreme points to the dual feasible set, the coefficients computed using these extreme point solutions result in at most a finite number of distinct values at stage $t$. Moreover, given that the scenario tree is discrete, and the stagewise distributions are assumed to be known, there can only be a finite number of cuts that are generated in DD-based multistage algorithms. This property also guarantees the convergence of these algorithms in finitely many of iterations.

[4]The exponent $(T-t)$ results from the fact that minorants in the future $T-t$ stages are also updated in a similar manner. Theorem 3.2 provides the formal argument.

**Algorithm 1** Stochastic Dynamic Linear Programming

1: **Initialization**:
2: Choose a proximal parameter $\sigma \in [\sigma^{min}, \sigma^{max}]$ with $1 \leq \sigma^{min} < \sigma^{max}$.
3: Set observations $\Omega_t^0 = \emptyset$; a trivial affine functions $\ell_t^0 = 0$ in the set $\mathcal{J}_t^0$ for all $t \in \mathcal{T}$; iteration counter $k \leftarrow 1$.
4: **Forward recursion**: Decision simulation along simulated sample-path
5: Solve the root-stage optimization problem of the form (3) to identify $u_0^k$.
6: Simulate a sample-path $\omega_{(0)}^k$.
7: *Prediction pass*:
8: **for** $t = 1, \ldots, T-1$ **do**
9: Setup the incumbent state $\hat{x}_t^k = \mathcal{D}_t(\hat{x}_{t-}^k, \omega_t^k, \hat{u}_{t-}^k(\hat{s}_{t-}^k))$.
10: Identify an incumbent solution $\hat{u}_t^k(\hat{s}_t^k) = \mathcal{M}_t^k(\hat{s}_t^k)$.
11: **end for**
12: *Optimization pass*:
13: **for** $t = 1, \ldots, T$ **do**
14: Setup the candidate state $x_t^k = \mathcal{D}_t(x_{t-}^k, \omega_t^k, u_{t-}^k)$.
15: Solve the stage optimization problem (4) using $s_t^k$ as input, and obtain the candidate primal solution $u_t^k$.
16: **end for**
17: **Backward recursion**: Update value function approximations.
18: **for** $t = T, \ldots, 1$ **do**
19: Setup the stagewise-dual approximation (11).
20: Solve the dual approximation using the candidate and the incumbent states, and compute the coefficients for affine functions using (13).
21: Obtain the updated value function approximation as in (15).
22: **end for**
23: Increment the iteration count $k \leftarrow k+1$, and go to Line-4.

In contrast, SDLP updates the coefficients associated with affine minorants of (14) at stage $t+$. Because of the assumption that the stagewise value functions are bounded below by zero, the updated coefficients of the minorants can be viewed as a convex combination of the constructed coefficient vector $(\alpha_{t+}^j, \beta_{t+}^j)$ and a zero vector. Due to these updates, the dual feasible set depends on updated coefficients (particularly $\beta_{t+}^k(\omega)$) as well as frequencies $p^k(\omega)$ as follows:

$$\Pi_t^{k,SD} = \{\pi_t \mid \ \langle D_t, \pi_t \rangle \leq d_t + \textstyle\sum_{\omega_{t+} \in \Omega_{t+}^k} p^k(\omega_{t+}) \langle \beta_{t+}^k(\omega_{t+}), B_{t+} \rangle, \pi_t \leq 0 \ \}. \tag{16}$$

This implies that dual solutions used to compute the coefficients no longer belong to a finite set, and in fact, for this reason, we refer to these iteratively evolving affine functions as minorants, instead of "cuts". However, following assumption (A2) the dual feasible set in ($\text{SDA}_t^k$) is bounded. Therefore, the coefficients computed in (13) for a non-terminal stage are only guaranteed to belong to a compact set. Proceeding backwards, we can conclude that this is the case for coefficients at all non-terminal stages. These observations are summarized in the following lemma.

**Lemma 3.1.** *Suppose the algorithm runs for infinitely many iterations under Assumption (A1) and (A2). Then, for all $k \geq 1$, we have the following properties.*

*(i) The coefficients of minorants generated for the terminal stage within SD-based methods in* (7) *belong to finite sets.*

*(ii) The coefficients of minorants generated within SD-based methods for non-terminal stages in* (13) *belong to compact sets.*

One final observation regarding the nature of minorants (subgradients) computed in SD-based methods: due to sampling, these approximations are stochastic. In addition, they satisfy the lower bounding property for the current SAA, but not necessarily for the true value function. The previous affine functions are updated using the scheme described in (14). This scheme ensures that the minorant $h_t^k$, obtained after computing the current affine function and updating all previous affine functions, provides a lower bound to the sample average function $H_t^k$ at all non-terminal stages. The outer linearization property of the minorants is formalized in the following theorem.

**Theorem 3.2.** *Suppose assumption (A1)-(A5) hold.*

*(i) The minorant computed in* (8) *for terminal stage satisfies:*

$$h_T(s_T) \geq h_T^k(s_T) \geq h_T^{k-1}(s_T) \geq \ldots \geq h_T^j(s_T), \tag{17a}$$

*for all $1 \leq j \leq k$, $s_T \in \mathcal{S}_T$.*

*(ii) At non-terminal stages, the minorant computed in* (14) *satisfies for $s_t \in \mathcal{S}_t$:*

$$H_t^k(s_t) \geq h_t^k(s_t) \geq \left(\frac{k-1}{k}\right)^{T-t} h_t^{k-1}(s_t). \tag{17b}$$

*Proof.* The first part of the theorem follows directly from the linear programming duality and the construction of the affine functions $\ell_T^k$ in (6) and (7). For proof of the second part, we use $m = \omega_{t+}^k$ which is the observation encountered at stage $t+$ in iteration-$k$ and $n$ to index the set $\Omega_{t+}$. Following this notation, we denote $x_{t+} = \mathcal{D}_{t+}(x_t, \omega_{t+}, u_t)$ as $x_{t+}(n)$ and $s_{t+} = (x_{t+}, \omega_{t+})$ as $s_{t+}(n)$. Consider the stage sample average problem in (9):

$$H_t^k(s_t) - \langle c_t, x_t \rangle = \min_{u_t \in \mathcal{U}_t(s_t)} \langle d_t, u_t \rangle + \sum_{n \in \Omega_{t+}^k} p^k(n) h_{t+}^k(s_{t+}(n)). \tag{18}$$

Recall that the affine function $\ell_t^k$ is computed using a dual solution of the problem on the right-hand side of the above equation. Using (11) and linear programming duality, we obtain that $H_t^k(s_t) \geq \ell_t^k(s_t)$. We distribute the summation in (18) over observations encountered in the first $i < k$ iterations (i.e., $\Omega_{t+}^i$) and those encountered after iteration $i$.

$$\begin{aligned} &H_t^k(s_t) - \langle c_t, x_t \rangle \\ &\quad = \min_{u_t \in \mathcal{U}_t(s_t)} \langle d_t, u_t \rangle + \sum_{n \in \Omega_{t+}^j} p^k(n) h_{t+}^k(s_{t+}(n)) + \sum_{n \in \Omega_{t+}^k \setminus \Omega_{t+}^j} p^k(n) h_{t+}^k(s_{t+}(n)). \end{aligned}$$

Since $h^k_{t+} \geq 0$, we have

$$\begin{aligned} H^k_t(s_t) - \langle c_t, x_t\rangle \geq & \min_{u_t \in \mathcal{U}_t(s_t)} \langle d_t, u_t\rangle + \sum_{n\in\Omega^i_{t+}} p^k(n) h^k_{t+}(s_{t+}(n)) \\ = & \min_{u_t \in \mathcal{U}_t(s_t)} \langle d_t, u_t\rangle + \sum_{n\in\Omega^j_{t+}} \frac{\kappa^i(n) + \kappa^{[i,k]}(n)}{k} \cdot h^k_{t+}(s_{t+}(n)). \end{aligned}$$

For observations in $\Omega^i_{t+}$, we distribute the computation of their relative frequency by setting $\kappa^k(n) = \kappa^i(n) + \kappa^{[i,k]}(n)$, where $\kappa^{[i,k]}(n)$ is the number of times observation $n$ was encountered after iteration $i$. Once again invoking $h^k_{t+} \geq 0$ we obtain:

$$H^k_t(s_t) - \langle c_t, x_t\rangle \geq \min_{u_t \in \mathcal{U}_t(s_t)} \langle d_t, u_t\rangle + \sum_{n\in\Omega^i_{t+}} \frac{i}{k} \times \frac{\kappa^i(n)}{i} \cdot h^k_{t+}(s_{t+}(n)).$$

Recall that the minorants at stage $t+$ are updated in (14) by including a new affine function into the collection while multiplying the previously generated affine functions by a factor of $(\frac{i}{k})^{T-t-1} < 1$. By replacing the current minorant $h^k_{t+}$ by the scaled version of the one available in iteration $j$, we have:

$$\begin{aligned} H^k_t(s_t) \geq & \ \langle c_t, x_t\rangle + \min_{u_t \in \mathcal{U}_t(s_t)} \langle d_t, u_t\rangle + \frac{i}{k} \sum_{n\in\Omega^i_{t+}} p^i(n) \left[ \left(\frac{i}{k}\right)^{T-t-1} h^i_{t+}(s_{t+}(n)) \right] \\ \geq & \left(\frac{i}{k}\right)^{T-t} \left[ \langle c_t, x_t\rangle + \min_{u_t \in \mathcal{U}_t(s_t)} \langle d_t, u_t\rangle + \sum_{n\in\Omega^j_{t+}} p^i(n) h^i_{t+}(s_{t+}(n)) \right]. \end{aligned}$$

The second inequality follows from assumption (A4). Notice that the scaling factor used when $t+ = T$ reduces to one. In this case, the future cost corresponds to the terminal stage, and the affine functions satisfy $\ell^j_T(s_T) \leq h_T(s_T)$ for all $j \in \mathcal{J}^k_T(\omega_T)$. Therefore, $h^k_T(s_T) \leq h_T(s_T)$. At other stages, an affine function generated in iteration $i < k$, viz. $\ell^j_t(s_t)$ with $j \in \mathcal{J}^i_t$ provides a lower bound to the sample average in the same iteration $H^i_t(s_t)$. This leads us to conclude that

$$H^k_t(s_t) \geq \left(\frac{i}{k}\right)^{T-t} H^i_t(s_t) \geq \left(\frac{i}{k}\right)^{T-t} \ell^j_t(s_t). \tag{19}$$

Applying the same arguments for all $i < k$, and using the definition of minorant in (14) we obtain $H^k_t(s_t) \geq h^k_t(s_t)$.

Since,

$$\begin{aligned} H^k_t(s_t) \geq \left(\frac{i}{k}\right)^{T-t} \ell^i_t(s_t) = & \left(\frac{k-1}{k}\right)^{T-t} \times \left(\frac{k-2}{k-1}\right)^{T-t} \times \ldots \times \left(\frac{i}{i+1}\right)^{T-t} \ell^i_t(s_t) \\ = & \left(\frac{k-1}{k}\right)^{T-t} \left(\frac{i}{k-1}\right)^{T-t} \ell^i_t(s_t) = \left(\frac{k-1}{k}\right)^{T-t} h^{k-1}_t(s_t). \end{aligned}$$

This completes the proof. □

As noted in the above proof, the scaling factor $(\frac{i}{k})^{T-t}$ used in (19) is applied to affine functions in $\mathcal{J}^i_t$ that were generated in iteration $i < k$. Since these affine functions are updated in every iteration, computational efficiency can be achieved by using recursive updates. In iteration

$k$, the affine functions in $\mathcal{J}_t^{k-1}$ are updated by multiplying them by the factor $(\frac{k-1}{k})^{T-t}$ and storing the updated minorants in $\mathcal{J}_t^k$. We refer the reader to [20] and [14] for details regarding efficient implementation of these updates. In the next result we capture the asymptotic behavior of the sequence of minorants $\{h_t^k\}$.

**Theorem 3.3.** *Under assumption (A2), (A3) and (A5), the sequence of functions $\{h_t^k\}_k$ is equicontinuous and uniformly convergent at all non-root stages.*

*Proof.* Recall that the coefficients of the minorants belong to a compact set at all the non-root stages (LEMMA 3.1). Therefore, $\{h_t^k\}$ is a sequence of bounded continuous functions with a uniform Lipschitz constant, say $M$. Further, the sequence $\{h_t^k\}$ converges pointwise on $s_t \in \mathcal{S}_t$. Let $s_t^{k_1}$ and $s_t^{k_2}$ be input states such that $\|s_t^{k_1} - s_t^{k_2}\| < \epsilon/M$, for a positive constant $\epsilon$. From Lipschitz continuity, we have

$$|h_t^k(s_t^{n_1}) - h_t^k(s_t^{n_2})| \leq M\|s_t^{n_1} - s_t^{n_2}\| < \epsilon.$$

for any $k \geq 1$. Hence, the sequence $\{h_t^k\}$ is equicontinuous. Equicontinuity and pointwise convergence together imply uniform convergence [34]. □

In contrast to the above results, the approximations created in the DD-based methods provide outer linearization for a fixed cost function $H_t^N(\cdot)$. Since, the probability distribution is explicitly used (as constants) in computing the DD-based cuts, the approximations improve monotonically over iterations. That is, $H_t^N(s_t) \geq h_t^k(s_t) \geq h_t^{k-1}(s_t)$ for all $s_t$, without any need for updates.

We close this section with the following remarks. The first contrasts the incorporation of sampling during backward recursion of SDDP with the role of sampling adopted in SDLP. The second identifies the online sampling feature of SDLP that has many advantages in practical settings. Finally the third remark pertains to the differences in the manner in which the proximal term in SDLP differs from the notion of regularization in SDDP.

*Remark* 3.1. Sampling during backward recursion has also been explored in SDDP (e.g., [5] [6], and [31]). However, there are important factors that distinguish value function updates undertaken during the backward recursion of SDLP when compared to SDDP calculations. The inclusion of new sample-paths can results in new nodes at non-root stages and an update in the weights (that are synonymous with estimated probability) used in calculating the SDLP minorants (13). This is different from SDDP where the set of sample-paths is fixed and the backward pass calculations are carried out over all scenarios in every iteration. Even when sampling is employed in the backward pass of SDDP, calculations are carried out along all sample-paths by either solving a subproblem or using the "argmax" procedure in (6) (similar to the cut formation first suggested in [18]). Sampling on the backward pass does not impede the finite convergence of SDDP as the number of cuts that can be generated is still finite.

*Remark* 3.2. Since the SDLP algorithm works with data discovered through sequential sampling, it does not rely on any a priori knowledge of an exogenous probability distribution. This feature makes this algorithm suitable to work with external simulators or statistical models that can better capture the nature of exogenous uncertainty. In each iteration, the algorithm can invoke a simulator to provide a new sample-path. This feature is particularly appealing when

a priori representation of uncertainty using scenario trees is either cumbersome or inadequate due to computational and/or timeliness constraints. Such optimization problems are commonly encountered in the operations of power systems with significant renewable penetration. Due to the intermittent nature of renewable resources, such as wind and solar, a scenario tree representation may be difficult (perhaps even impossible) to create within the timeliness constraints. State-of-the-art numerical weather prediction and other time series models are known to be more accurate descriptors of such uncertainty. However, output from such models are unlikely to satisfy the stagewise independence assumption, therefore suitable modifications to SDLP will be necessary. In the case of dependent stochastic processes, one may adopt the MSD algorithm [37]. In any case, optimization algorithms which use simulated scenarios yield more reliable plans and cost estimates [12, 13], and therefore, more desirable for practical applications.

*Remark* 3.3. The use of quadratic regularization in multistage DD-based methods has been limited to the SDDP setting [1, 15]. The quadratic term in SDDP is employed for accelerating convergence and is viewed as a penalty to reduce drastic oscillation of solutions. Moreover, convergence of regularized SDDP requires that the penalty coefficients, akin to $\sigma$ in (3), vanish as the algorithm progresses (see Theorem 6 in [1]). Contrary to this requirement, the role of the quadratic term in SDLP is more aligned with its use in proximal methods. In this sense, SDLP can be viewed as a stagewise proximal algorithm. The proximal term may be credited with at least two significant computational features of SDLP. Firstly, the proximal term results in a finite-sized stage optimization problems with only $n_t + 3$ minorants in each stage (where $n_t$ is the number of stage decision variables). Therefore, the computational burden does not grow with iterations. Secondly, the proximal parameter $\sigma$ can be updated based on the progress made by the algorithm using a procedure similar to that presented in [37]. The dynamic adaption of proximal parameter is unique to SD-based methods. Moreover, we do not require the proximal term to vanish as the algorithm progresses. Most importantly, the proximal term guarantees the iterative descent property [22] as we will see in §4 (Theorem 4.2).

## 3.3 Subgradient and Incumbent Selection

In this section we address two important components of the SDLP algorithm: the "argmax" procedure to identify the subgradient of a SDLP approximation at non-root stage that is used during the backward recursion, and the selection of an incumbent solution for the proximal term used during timestaged decision simulation.

### 3.3.1 Subgradient Selection

During the backward recursion, we build a lower bound to the sample average function $H^k_{t-}$ using the best lower bounding affine functions from the collection $\mathcal{J}^k_t$ for all $\omega_t \in \Omega^k_t$. This procedure is accomplished differently based on whether the observation belongs to the current sample-path $\omega^k_{(0)}$, or not. We utilize the collection of dual vertices $\Pi^k_t$ identified during the course of the algorithm for this purpose. We denote by $i(\pi_t)$ the iteration in which the dual vertex $\pi_t \in \Pi^k_t$ was generated. As seen in (16), the dual vertex $\pi_t \in \Pi^k_t$ depends on $H_t^{i(\pi_t)}$, the sample average function in iteration $i(\pi_t)$. This dependence is reflected in the calculation of coefficients

$(\bar{\alpha}_{t+}^{i(\pi_t)}, \bar{\beta}_{t+}^{i(\pi_t)})$ and the term $\bar{\rho}_{t+}^{i(\pi_t)}$ that defines the feasible set associated with $\pi_t$ (see (11)).

**For observation $\omega_t^k$:** This observation is encountered at stage $t$ along the current sample-path. Consequently in the current backward recursion, we built and solved a $\text{SDA}_t^k$ to optimality using $s_t^k$ as input. Using the optimal dual solution thus obtained, we compute the coefficients in (13) for the hyperplanes $\ell_t^k(s_t)$ to $\text{SDA}_t^k$ at the candidate state. Similar calculations with $\hat{s}_t^k$ as input yield the hyperplane $\hat{\ell}_t^k(s_t)$ to $\text{SDA}_t^k$ at incumbent state $\hat{s}_t^k$.

**For observations $\omega_t \in \Omega_t^k \setminus \{\omega_t^k\}$:** These are the observations not included in the current sample-path, and therefore, no backward recursion optimization is carried out for these observations. Instead, we use an "argmax" procedure to identify the subgradient approximations. These subgradients correspond to the best lower bounding affine functions of $\text{SDA}_t^k$ for these observations. In order to accomplish this, we maintain a set of dual solutions $\Pi_t^k$ obtained by solving the $\text{SDA}_t^i$ in iterations $i \leq k$ as in the case of 2-SD. For each $\omega_t \in \Omega_t^k \setminus \{\omega_t^k\}$, we setup $s_t = (x_t, \omega_t)$, where $x_t$ is computed with $(x_{t-1}^k, \omega_t, u_{t-1}^k)$ as input in (1), and identify a dual solution:

$$\pi_t^k(\omega_t) \in \text{argmax}\left\{ \left(\frac{i(\pi_t)}{k}\right)^{T-t} \langle \pi_t, (b_t - C_t x_t)\rangle \mid \pi_t \in \Pi_t^k \right\}.$$

The scaling factor used in the above calculation reflects the scaling of affine functions discussed in Theorem 3.3. Notice from the structure of dual LPs above, that while the set of dual vertices $\Pi_t^k$ changes with iterations, the collection of dual variables remains constant (see also (16)). Further discussion of this issue is provided in section §3.3.2. Using the dual solution obtained by the above procedure, we can compute the coefficients:

$$\alpha_t^k(\omega_t) = \left(\frac{i(\pi_t^k(\omega_t))}{k}\right)^{T-t} [\langle \pi_t^k(\omega_t), b_t\rangle + \bar{\alpha}_{t+}^{i(\pi_t^k(\omega_t))}],$$
$$\beta_t^k(\omega_t) = \left(\frac{i(\pi_t^k(\omega_t))}{k}\right)^{T-t} [\langle -C_t, \pi_t^k(\omega_t)\rangle + \bar{\beta}_{t+}^{i(\pi_t^k(\omega_t))}].$$

In essence, the above procedure identifies a dual solution $\pi_t^k(\omega_t)$ which was obtained using a $\text{SDA}_t^{i(\pi_t^k(\omega_t))}$, and scales it appropriately to provide the best lower bounding approximation to the current $\text{SDA}_t^k$.

### 3.3.2 Incumbent Selection

The procedure described here identifies an incumbent solution at all non-root, non-terminal stages is motivated by the optimal basis propagation policy presented in [4]. This identification, which is performed during the prediction pass, relies on the basis of the stage dual approximation ($\text{SDA}_t^k$) that appears on the right-hand side of (11). To facilitate the discussion here, we have restated $\text{SDA}_t^k$ below:

$$\max\ \langle \pi_t, (b_t - C_t x_t)\rangle \text{ subject to } \langle D_t, \pi_t\rangle \leq \bar{\rho}_t^k,\ \pi_t \leq 0, \tag{20}$$

where $\bar{\rho}_t^k$ is defined in the expressions following (11). In each iteration, the above linear program is solved to optimality along the iteration sample-path and potentially a new basis is discovered. Let $\mathbb{B}_t^k$ denote the index set whose elements are the rows which are active in (20). Denote by

$D_{t,\mathbb{B}_t^k}$ the submatrix of $D_t$ formed by columns indexed by $\mathbb{B}_t^k$ (the basis matrix). From standard linear programming results we have that a feasible point is an extreme point of the feasible set if and only if there exists an index set that satisfies $\langle D_{t,\mathbb{B}_t^k}, \pi_t^k \rangle = \bar{\rho}^k_{t,\mathbb{B}_t^j}$. This index set is added to the collection of previously discovered index sets, that is: $\mathcal{B}_t^k \leftarrow \mathcal{B}_t^{k-1} \cup \mathbb{B}_t^k$. We use this collection of index sets to construct dual solutions of the linear program in (20). Assumption (A2) ensures that the optimal set of the dual linear program is non-empty which implies that there exists an index set $\mathbb{B}_t^j \in \mathcal{B}_t^k$ such that for any arbitrary input state $s_t$ we can write:

$$\hat{u}_{t,i} = D^{-1}_{t,\mathbb{B}_t^j}(b_t - C_t x_t),\ i \in \mathbb{B}_t^j; \qquad \hat{u}^j_{t,i} = 0,\ i \notin \mathbb{B}_t^j. \tag{21}$$

Note that, if $\hat{u}_t^j$ satisfies the constraints of dual of (20) then it is a suboptimal basic feasible solution to the dual problem (and if complementarity conditions are also satisfied then it is an optimal solution). We use $\widehat{\mathcal{U}}_t^k(s_t) \subseteq \mathcal{U}_t(s_t)$ to denote the set of basic feasible solutions generated using (21) for all index sets in $\mathcal{B}_t^k$. Since (20) corresponds to $\text{SDA}_t^k$, its dual feasible solutions are feasible to the stage optimization problem (2). Using these index sets we define the mapping used for incumbent selection at non-root stages as follows:

$$\mathcal{M}_t^k(s_t) = \text{argmin}\{f_t^{k-1}(s_t, \hat{u}_t^j) \mid \hat{u}_t^j \in \widehat{\mathcal{U}}_t^k(s_t)\} \qquad \forall t \in \mathcal{T} \setminus \{0\}. \tag{22}$$

We refer to the above mapping as the *basic feasible policy* (BFP) of the MSLP problem. In case the argument that minimizes the right-hand of (22) is not unique, we choose an index set with the smaller iteration index $k$. Notice that the dual LP of (20) has cost coefficients that vary over iterations, akin to 2-SD with random cost coefficients in the second-stage [14]. The steps involved in identifying the BFP, particularly computation of dual solutions in (21) and establishing their feasibility, can be implemented in a computationally efficient manner using a sparsity preserving representation of dual solutions. We refer the reader to [14] for a detailed discussion of this representation and its implementation.

At the root-stage it suffices to maintain a single incumbent solution. This incumbent solution is updated based on predicted objective value reduction at the root-stage:

$$f_0^k(s_0, u_0^k) - f_0^k(s_0, \hat{u}_0^{k-1}) \ \leq\ q\ [f_0^{k-1}(s_0, u_0^k) - f_0^{k-1}(s_0, \hat{u}_0^{k-1})], \tag{23}$$

where $q \in (0, 1)$ is a given parameter. If the above inequality is satisfied, then the candidate solution at the root node will replace the incumbent solution $\hat{u}_0^{k-1}$ and will serve as the next incumbent solution; that is, $\hat{u}_0^k \leftarrow u_0^k$. On the other hand, if the inequality is not satisfied, then the current incumbent solution for the root stage is retained ($\hat{u}_0^k \leftarrow \hat{u}_0^{k-1}$). This update rule is similar to incumbent updates carried out in non-smooth optimization methods including regularized 2-SD [19, 21].

## 4 Convergence Analysis

In this section, we present the convergence results for SDLP. We begin by discussing the behavior of the sequence of states and decisions generated by the SDLP algorithm, then proceed to show the convergence of value function estimates. Finally, we show that the incumbent solution

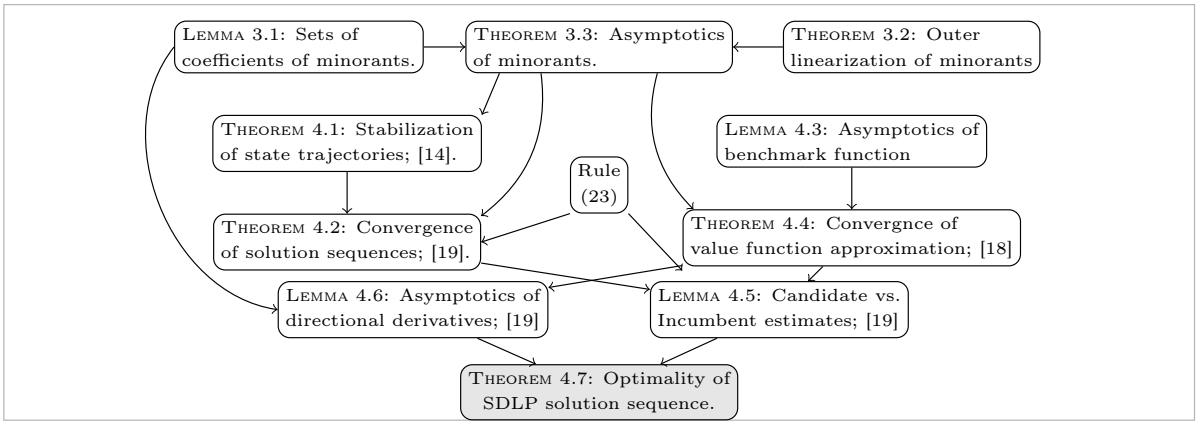


Figure 2: Sketch of SDLP Analysis

sequence at the root-stage $\{\hat{u}_0^k\}$ converges and establish the optimality of the accumulation point. The SDLP convergence analysis is built upon the results of the 2-SD algorithm [18], its regularized variant [19], and 2-SD for 2-SLPs with random cost coefficients [14]. Figure 2 illustrates the development of the SDLP convergence analysis. The cited references serve as pointers to related results in the two-stage setting.

### State and decision accumulation points

Under assumption (A5), we have a finite number possible sample-paths over the horizon. We use $\mathcal{P}_t$ to denote the set of all sample-paths until stage $t$. We focus on the evolution of states and decisions along these sample-paths.

*Theorem* 4.1. *Suppose assumptions (A1)-(A5) hold. Let $\{\hat{u}_0^k\} \subseteq \mathcal{U}_0$ denote any infinite sequence of root-stage incumbent solutions. Then,*

*(a) The BFP defined in §3.3.2 is a piecewise linear mapping.*

*(b) There exists a subsequence $\mathcal{K}_0$ of iterations such that $\{\hat{u}_0^k\}_{\mathcal{K}_0}$ has an accumulation point. Moreover, in subsequent stages, for all possible paths $\omega_{[t]} \in \mathcal{P}_t$ there exists a subsequence of iterations indexed by $\mathcal{K}_t(\omega_{[t]})$ such that the sequence $\{\hat{u}_t^k(\hat{s}_t^k)\}_{k\in\mathcal{K}_t(\omega_{[t]})}$ has an accumulation point.*

*Proof.* (a) To show the continuity of the BFP used to identify the incumbent solutions, consider the optimization problem on the right-hand side of (11) for a given $t$ in its dual form:

$$\min \ \{\langle \bar{\rho}_{t+}^k(\omega_t^k), u_t\rangle \mid D_t u_t \leq b_t - C_t x_t, u_t \geq 0\}.$$

Recall that $\bar{\rho}_{t+}^k(\omega_t^k)$ is the sample mean of subgradients of minorants corresponding to stage $t+$. Since these subgradients belong to a compact set (Lemma 3.1 (i)), the coefficients $\{\bar{\rho}_{t+}^k(\omega_t^k)\}$ also belong to a compact set. Let $\mathfrak{U}(x_t, \bar{\rho}_{t+}^k(\omega_t^k))$ denote the set of optimal solutions to the above problem with cost coefficient as $\bar{\rho}_{t+}^k(\omega_t^k)$ and endogenous state $x_t$ as input. Because of (A2), the mapping $\mathfrak{U}$ is non-empty for all input pairs $(x_t, \bar{\rho}_{t+})$. A slight variant of Hoffman's lemma

(Lemma A.1 in the appendix) leads us to conclude that for any $(x'_t, \bar{\rho}'_{t+})$ and $u'_t \in \mathfrak{U}_t(x'_t, \bar{\rho}'_{t+})$ we have

$$\text{dist}(u'_t, \mathfrak{U}(x_t, \bar{\rho}^k_{t+}(\omega^k_t))) \leq \chi_1 \|x'_t - x_t\| + \chi_2 \ \|\bar{\rho}'_{t+} - \bar{\rho}^k_{t+}(\omega^k_t)\|.$$

Here, $\chi_1, \chi_2 > 0$ are constants of the mapping $\text{dist}(\cdot)$ that depend only on the transfer matrix $C_t$ and the recourse matrix $D_t$. In other words, the set of optimal solutions $\mathfrak{U}_t(\cdot)$ is Lipschitz continuous in the above sense. The elements of the set $\mathfrak{U}_t(x_t, \bar{\rho}^k_{t+}(\omega^k_t))$ are basic feasible solutions associated with bases of $D_t$. Since there are finitely many bases, the BFP outlined in §3.3.2 is a continuous piecewise linear mapping.

(b) For the root-node the feasible set $\mathcal{U}_0$ is compact by (A1), hence there exists a subsequence of iterations indexed by $\overline{\mathcal{K}}_0$ such that $\{\hat{u}^k_0\}_{k\in\overline{\mathcal{K}}_0} \to \bar{u}_0$. Following (A5), there exists an infinite subsequence $\mathcal{K}_1(s_{[1]}) \subseteq \overline{\mathcal{K}}_0$ such that the algorithm selects sample-path $\omega_{[1]} \in \mathcal{P}_1$. Since $\{\hat{u}^k_0\}_{k\in\overline{\mathcal{K}}_0}$ converges and $x_0$ is fixed, the sequence of endogenous state $\{x^k_1\}_{k\in\mathcal{K}_1(s_{[1]})}$ converges to $\bar{x}_1(s_{[1]})$. For the sample-path $\omega_{[1]}$, since the sequence of input states $\{x^k_1\}_{k\in\mathcal{K}_1(s_{[1]})}$ converges and $\{\bar{\rho}^k_2(\omega^k_1)\}$ belong to a compact set, the continuity of BFP implies that the corresponding sequence of incumbent solutions $\{\hat{u}^k_1(\hat{s}^k_1)\}_{k\in\mathcal{K}_1(s_{[1]})}$ has a converging subsequence. Let $\overline{\mathcal{K}}_1(s_{[1]})$ denote this subsequence. Therefore, we have $\{\hat{u}^k_1(\hat{s}^k_1)\}_{k\in\overline{\mathcal{K}}_1(s_{[1]})} \to \bar{u}_t(s_{[1]})$.

Now consider an arbitrary stage $t > 1$. For any sample-path $\omega_{[t]} \in \mathcal{P}_t$, once again assumption (A5) guarantees that there exists an infinite subsequence of $\mathcal{K}_t(s_{[t]}) \subseteq \mathcal{K}_{t-}(s_{[t-]})$ when sample-path $\omega_{[t]}$ is encountered. Here $\omega_{[t]} = (\omega_{[t-]}, \omega_t)$, i.e., sample-path $\omega_{[t]}$ shares the same observations with $\omega_{[t-]}$ until stage $t-$. Over this subsequence, the convergence of endogenous state sequence $\{\hat{x}^k_t = \mathcal{D}_t(\hat{x}^k_{t-}, \omega_t, \hat{u}^k_{t-})\}_{\mathcal{K}_t(s_{[t]})} \to \bar{x}_t(s_{[t]})$ ensures the convergence of the incumbent states $\{\hat{s}^k_t\}_{\mathcal{K}_t(s_{[t]})}$. Further since $\{\bar{\rho}^k_{t+}(\omega^k_t)\}$ belong to a compact set, the continuity of BFP applied at stage $t$ ensures that the corresponding sequence of incumbent solutions $\{\hat{u}^k_t(\hat{s}^k_t)\}$ have a converging subsequence. That is, there exists $\overline{\mathcal{K}}_t(s_{[t]}) \subset \mathcal{K}_t(s_{[t]})$ such that $\{\hat{u}^k_t(\hat{s}^k_t)\}_{\overline{\mathcal{K}}_t(s_{[t]})} \to \bar{u}_t(s_{[t]})$. Proceeding recursively to the rest of the stages, we conclude the validity of the theorem. □

The above result captures the impact of using the argmin mapping in (22) over a sequence of converging first-stage decisions. A converging sequence results in perturbed stage problems with linear constraints in subsequent stages. A central argument in the above proof relies upon the local Lipschitz continuity of the argmin mapping. Such mappings have previously been studied in [41]. We refer the reader to this reference for a more thorough treatment of inf-projections and the argmin mapping for non-linear optimization problems with linear constraints.

To facilitate the presentation in the remainder of this section, let $\mathcal{P}^k_{(t+)} \in \Omega^k_{t+} \times \ldots \times \Omega^k_T$ denote the set of all possible scenarios from stage-$(t+1)$ to the end of horizon which traverse through observations encountered by the algorithm in the first $k$ iterations. Note that $\mathcal{P}^k_{(t+)}$ represents the set of possible paths in the future and should not be confused with $\mathcal{P}^k_t$ which represents the set of traversed paths. Stagewise independence allows us to compute the probability estimate of a sample-path $\omega^j_{(t+)} \in \mathcal{P}^k_{(t+)}$ as product of frequencies associated with observations along that sample-path, i.e. $p^k(\omega^j_{(t+)}) = p^k(\omega^j_{t+1}) \times \ldots \times p^k(\omega^j_T)$. Let $x^j_{(t+)}$ and $u^j_{(t+)}$ denote endogenous state and decision vector, respectively, associated with sample-path $\omega^j_{(t+)}$. While Theorem

4.1 captured the behavior of solutions generated using the incumbent mapping in (21) during prediction pass, the next result captures the behavior of solutions generated in optimization pass of the algorithm.

*Theorem* 4.2. *Suppose assumptions (A1) - (A5) hold, and* $\sigma \geq 1$. *Then there exists* $\bar{u}_0 \in \mathcal{U}_0(s_0)$ *such that the sequence of root-node incumbent decisions generated by the algorithm satisfy* $\{\hat{u}_0^k\} \to \bar{u}_0$. *Moreover in every subsequent stage, there exists* $\bar{u}_t(s_{[t]}) \in \mathcal{U}_t(\bar{s}_t(s_{[t]}))$ *which satisfy dynamics in* (1) *and the sequence of solutions generated by the algorithm* $\{u_t^k(s_{[t]})\}_{\mathcal{K}_t(s_{[t]})} \to \bar{u}_t(s_{[t]})$ *for all paths* $\omega_{[t]} \in \mathcal{P}_t$.

*Proof.* The proof for the root-stage follows that of regularized master in 2-SD (Theorem 5, [19]) and the root-node of MSD algorithm ([36]). Here we present the main parts of the proof and refer the reader to earlier works for detailed exposition. If the incumbent solution $\hat{u}_0^k$ changes infinitely many times, then the optimality condition for regularized approximation (see equation (5) in [19]) and our choice of $\sigma \geq 1$ suggests that for any candidate solution $u_0^k$ the following holds:

$$f_0^{k-1}(s_0, u_0^k) - f_0^{k-1}(s_0, \hat{u}_0^{k-1}) \leq -\|u_0^k - \hat{u}_0^{k-1}\|^2 \leq 0 \qquad \forall k \geq 1. \tag{24}$$

In particular, the above condition holds at the iterations when the incumbent was updated by assigning the candidate solution as the new incumbent solution, i.e. $\hat{u}_0^k = u_0^k$. Let $\{k_1, k_2, \ldots, k_m\} \in \mathcal{K}_0$ denote the set of $m$ successive iterations when the incumbent solution was updated starting with an incumbent $\hat{u}_0^{k_0}$. Note that, for any $k_n \in \mathcal{K}_0$, $\hat{u}_0^{k_n-1} = \hat{u}_0^{k_{n-1}}$. Denote by $\Delta^{k_n} := f_0^{k_n-1}(s_0, \hat{u}_0^{k_n}) - f_0^{k_n-1}(s_0, \hat{u}_0^{k_{n-1}})$. Using (24) over these $m$ updates, we have

$$\begin{aligned}
\frac{1}{m}\sum_{l=1}^{m} \Delta^{k_n} &= \frac{1}{m}\sum_{l=1}^{m} [f_0^{k_n-1}(s_0, \hat{u}_0^{k_n}) - f_0^{k_n-1}(s_0, \hat{u}_0^{k_{n-1}})] \\
&= \frac{1}{m}\Bigg[ \underbrace{[f_0^{k_m-1}(s_0, \hat{u}_0^{k_m}) - f_0^{k_1-1}(s_0, \hat{u}_0^{k_0})]}_{(a)} + \sum_{n=1}^{m} \underbrace{[f_0^{k_n-1}(s_0, \hat{u}_0^{k_n}) - f_0^{k_{n+1}-1}(s_0, \hat{u}_0^{k_n})]}_{(b)} \Bigg].
\end{aligned}$$

The boundedness of functions $\{f_0^k\}$ implies that (a) above approaches zero, as $m \to \infty$, and their uniform convergence (Theorem 3.3) implies that (b) converges to zero. Hence,

$$\lim_{m\to\infty} \frac{1}{m}\sum_{l=1}^{m} \Delta^{k_n} = 0, \tag{25}$$

with probability one. Further, the above result, along with (24) implies that $\lim_{m\to\infty} \frac{1}{m}\sum_{n=1}^{m} \|\hat{u}_0^{k_n} - \hat{u}_0^{k_{n-1}}\|^2 = 0$. Therefore, we conclude that the sequence of root-node incumbent solutions converges to $\bar{u}_0 \in \mathcal{U}_0$.

At non-root stages, the incumbent solutions are selected using the BFP described in §3.3.2. The BFP is built using the bases of (11) discovered during the course of the algorithm that are identified by the collection of index sets $\mathcal{B}_t$. Since there is a finite collection $\mathcal{B}_t$ of index sets, there exists iteration count $K_t$ large enough such that $\mathcal{B}_t^{k'} = \mathcal{B}_t$ for all $k' \geq K_t$. Let us consider $k > \max_t K_t$ when all the index sets for all non-root, non-terminal stages have been discovered. In these iterations, the procedure in §3.3.2 results in an incumbent solution such that:

$$\hat{u}_t^k(\hat{s}_t^k) = \mathcal{M}_t^k(\hat{s}_t^k) \in \operatorname{argmin}_{u_t \in \mathcal{U}_t(\hat{s}_t^k)} \langle d_t, u_t \rangle + \sum_{\omega_{t+} \in \Omega_{t+}^{k-1}} p^{k-1}(\omega_{t+}) h_{t+}^{k-1}(\mathcal{D}_t(\hat{x}_t^k, \omega_{t+}, u_t), \omega_{t+}).$$

Consequently, the value associated with $\hat{u}_t^k(\hat{s}_t^k)$ is $H_t^{k-1}(\hat{s}_t^k)$ (see (9)). The forward pass optimal value associated with the candidate solution $u_t^k(\hat{s}_t^k)$ differs from $H_t^{k-1}(\hat{s}_t^k)$ in only the quadratic term. Therefore, we have $H_t^{k-1}(\hat{s}_t^k) \leq F_t^{k-1}(\hat{s}_t^k)$ that can be restated using (15) as:

$$f_t^{k-1}(\hat{s}_t^k, \hat{u}_t^k(\hat{s}_t^k)) - f_t^{k-1}(\hat{s}_t^k, u_t(\hat{s}_t^k)) \leq 0,$$

where $u_t(\hat{s}_t^k)$ is the solution obtained by optimizing the regularized problem used during forward recursion. The quadratic programming optimality conditions of this regularized problem allow us to write the following inequality:

$$f_t^{k-1}(\hat{s}_t^k, u_t(\hat{s}_t^k)) - f_t^{k-1}(\hat{s}_t^k, \hat{u}_t^k(\hat{s}_t^k)) \leq 0.$$

The two preceding inequalities together with ((24) for stage $t$) implies that $\|u_t(\hat{s}_t^k) - \hat{u}_t^k(\hat{s}_t^k)\|^2 = 0$. For a sample-path $\omega_{[t]} \in \mathcal{P}_t$, let $\mathcal{K}_t(s_{[t]})$ denote the subsequence constructed in the proof of Theorem 4.1. Over this subsequence, the result of Theorem 4.1 shows the existence of an accumulation point of $\{\hat{u}_t^k(s_t^k)\}_{k\in\mathcal{K}_t(s_{[t]})}$, and subsequently, an accumulation point $\bar{u}_t(s_{[t]})$ of $\{u_t^k(s_t^k)\}_{k\in\mathcal{K}_t(s_{[t]})}$. Applying the argument to all sample-paths in $\omega_{[t]} \in \mathcal{P}_t$ completes the proof. □

The limit in (25) plays a critical role in showing the existence of an optimal accumulation point of incumbent solutions at the root-stage. Notice that the limit holds when the incumbent is updated infinitely often, i.e., $m \to \infty$. On the other hand, if the incumbent solution is updated only a finite number of times, then there exists a $K < \infty$ such that $\hat{u}_0^k = \bar{u} \in \mathcal{U}_0$, for all $k > K$. In this case, the optimality of $\bar{u}_0$ is attained only if $\Delta^k \to 0$. Before we present the optimality of solution sequence, we present the convergence of the value function estimates.

### Convergence of Value Function Estimates

Since our algorithm uses sequential sampling, path-wise forward and backward recursion updates, estimates of probability and sampled minorants we use benchmark functions to verify optimality of the value functions and solutions obtained from them. We next present the construction of these benchmark functions. Note that these function are not computed during the course of the algorithm and are intended only for the purpose of analysis.

For a given input $s_t$, the following is an extensive formulation of the cost-to-function:

$$\begin{aligned} \mathcal{H}_t^k(s_t) = \ & \langle c_t, x_t\rangle + \min \ \langle d_t, u_t\rangle + \\ & \sum_{j\in\mathcal{P}_{(t+)}^k} p^k(\omega_{(t+)}^j) \times [\langle c_{(t+)}, x_{(t+)}^j\rangle + \langle d_{(t+)}, u_{(t+)}^j\rangle] \\ & s.t.\ u_t \in \mathcal{U}_t(u_0, s_t),\ \{u_{t'}^j \in \mathcal{U}_{t'}(u_0, s_{t'}^j)\}_{t'>t} \text{ and non-anticipative}, \\ & \{x_{t'+}^j = \mathcal{D}_{t'+}(x_{t'}^j, \omega_{t'+}^j, u_{t'}^j)\}_{t'\geq t}. \end{aligned} \tag{26}$$

In the above formulation, dynamics and non-anticipativity are satisfied starting at stage $t$, and are relative to input $s_t$. This sample average function $\mathcal{H}_t^k$ represents the value associated with input $s_t$ for the remainder of horizon with respect to current observations $\{\Omega_i^k\}_{i=t+}^k$. In order to

simplify notation, the dependence of the sample average function on the set $\mathcal{P}^k_{(t+)}$ is conveyed through the index $k$ in $\mathcal{H}^k_t(s_t)$, as opposed to the more complete $\mathcal{H}^k_t(s_t|\mathcal{P}^k_{(t+)})$.

During forward recursion decisions, $\{u_t\}$ are simulated using approximation $f^{k-1}_t$ in (4) along the observations dictated by sampling, and during the backward recursion the approximations using subgradients observed along the same sample-path. Next we relate the objective function values encountered during forward and backward recursions. In order to do this, we define $u_t(s_t)$ to be the optimal solution obtained using (4) during forward recursion with input $s_t$. The forward recursion objective function value $F^{k-1}_t$ associated with this decision is therefore given by:

$$F^{k-1}_t(s_t) := \langle c_t, x_t\rangle + \langle d_t, u_t(s_t)\rangle + \sum_{\omega_{t+}\in\Omega^{k-1}_{t+}} p^{k-1}(\omega_{t+})\; h^{k-1}_{t+}(s^k_{t+}(\omega_{t+})). \tag{27}$$

Here $s^k_{t+}(\omega_{t+}) = \mathcal{D}_{t+}(x_t, \omega_{t+}, u_t(s_t))$. In order to study the asymptotic behavior of our algorithm, we investigate how the functions $\mathcal{H}^k_t$, $F^k_t$ and $h^k_t$ relate in value at limiting states. It is worthwhile to note that the SAA in (10), the extensive formulation in (26) and the forward recursion objective value in (27) are defined only for non-terminal stages as $H^k_T(s_T) = \mathcal{H}^k_T(s_T) = F^k_T(s_T) = h_T(s_T)$ for terminal stage $\forall k$.

*Lemma* 4.3. *Suppose Assumptions (A1)-(A5) hold.*

*(i) The sequence of functions $\{F^k_t\}_k$ is equicontinuous and uniformly convergent for all $t$.*

*(ii) The sequence of SAA functions $\{\mathcal{H}^k_t\}_k$ converges uniformly to the value function $h_t(\cdot)$ in (2) for all $t > 0$, with probability one.*

*Proof.* Under Assumption 2, we have $F^k_t < \infty$ for all $t \in \mathcal{T}\setminus\{T\}$ and $k \geq 1$. (i) Since a regularized problem (4) with quadratic proximal parameter is used to identify the sequence of solutions in the forward recursion of the algorithm, the optimality conditions of affinely constrained quadratic programs indicate that the solutions $u_t(s_t)$ are piecewise linear. Therefore, the sequence $\{F^k_t\}$ is bounded over a compact space and must have a uniform Lipschitz constant. This leads to the conclusion stated in part (i) of the lemma. Part (ii) follows from Theorem 7.53 in [39]. □

Following the above result, we use $\mathcal{H}^k_t$ as a benchmark for assessing optimality of the SDLP algorithm. We first show the convergence of approximations generated during the course of the algorithm to the true value function in the following theorem. In the two-stage setting, the equivalent result appears as Theorem 3 and Corollary 5 in [18].

*Theorem* 4.4. *Suppose Assumptions (A1)-(A3) hold. At any non-terminal stage $t$, if subsequence $\mathcal{K}_t$ is such that $\{\hat{s}^k_t\}_{k\in\mathcal{K}_t} \to \bar{s}_t$, then*

$$\lim_{k\in\mathcal{K}_t} f^k_t(\hat{s}^k_t, \hat{u}^k_t(\hat{s}^k_t)) = \lim_{k\in\mathcal{K}_t} f^{k+1}_t(\hat{s}^k_t, \hat{u}^k_t(\hat{s}^k_t)) = f_t(\bar{s}_t, \bar{u}_t), \tag{28}$$

*with probability one.*

*Proof.* For terminal stage ($t = T$), continuity of linear programming value function implies that $\lim_{k\in\mathcal{K}_T} h_T(\hat{s}_T^k) = h_T(\bar{s}_T(s_{[t]}))$. Since $F_T^k$, $H_T^k$ and $\mathcal{H}_T^k$ are all equivalent to $h_T$, the above relation holds from linear programming. Consequently we have, $\lim_{k\to\mathcal{K}_T} \hat{\ell}_T^k(\hat{s}_T^k) = h_T(\bar{s}_T)$ and $\lim_{k\to\mathcal{K}_T} \hat{\beta}_T^k \in \partial h_T(\bar{s}_T)$, where $\hat{\beta}_T^k \in \partial\hat{\ell}_T^k(\hat{s}_T^k)$. Recall that $\hat{\ell}_T^k(\cdot)$ is an affine function with coefficients given in (13).

For a non-terminal stage, let $k-\tau$ and $k$ be two successive iterations of subsequence $\mathcal{K}_t$. From the definitions of $H_t^{k-1}$ and $F_t^{k-1}$ in (9) and (27), respectively, we have $H_t^{k-1}(s_t) \le F_t^{k-1}(s_t)$ for all $s_t \in \mathcal{S}_t$. In the following, we focus on functions evaluated at $\hat{s}_t^k$, and use $m = \omega_{t+}^k$ and $n$ as an index for set $\Omega_{t+}^k$. The forward recursion objective function value at the current input state can be written as:

$$F_t^{k-1}(\hat{s}_t^k) = \langle c_t, \hat{x}_t^k\rangle + \langle d_t, u_t(\hat{s}_t^k)\rangle + \sum_{n\in\Omega_{t+}^{k-1}} p^{k-1}(n) h_{t+}^{k-1}(\hat{s}_{t+}^k(n)).$$

The optimality of $u_t(\hat{s}_t^k)$ ensures that the objective function value is associated with $u_t(\hat{s}_t^k)$ is lower than any other feasible solution. If we specifically consider the optimal solution of the dual in (11), denoted $\tilde{u}_t(\hat{s}_t^k)$, we have

$$F_t^{k-1}(\hat{s}_t^k) \le \langle c_t, \hat{x}_t^k\rangle + \langle d_t, \tilde{u}_t(\hat{s}_t^k)\rangle + \sum_{n\in\Omega_{t+}^{k-1}} p^{k-1}(n) h_{t+}^{k-1}(\tilde{s}_{t+}^k(n)).$$

By adding and subtracting the current approximation of future cost, i.e., $\sum_{n\in\Omega_{t+}^k} p^k(n) h_{t+}^k(\hat{s}_{t+}(n)) = \sum_{n\in\Omega_{t+}^{k-1}\setminus\{m\}} p^k(n)\ h_{t+}^k(\hat{s}_{t+}(n)) + p^k(m) h_{t+}^k(\hat{s}_{t+}(m))$ we obtain

$$\begin{aligned} F_t^{k-1}(\hat{s}_t^k) \le & \underbrace{\langle c_t, \hat{x}_t^k\rangle + \langle d_t, \tilde{u}_t(\hat{s}_t^k)\rangle + \sum_{n\in\Omega_{t+}^k} p^k(n) h_{t+}^k(\hat{s}_{t+}^k(n))}_{=\, H_t^k(\hat{s}_t^k) \text{ from (11)}} \\ & + \sum_{n\in\Omega_{t+}^{k-1}} p^{k-1}(n) h_{t+}^{k-1}(\tilde{s}_{t+}^k(n)) \\ & - \Bigg[\sum_{n\in\Omega_{t+}^{k-1}} \left(\frac{k-1}{k}\right) p^{k-1}(n)\ h_{t+}^k(\hat{s}_{t+}^k(n)) + \frac{1}{k} h_{t+}^k(\hat{s}_{t+}^k(m))\Bigg]. \end{aligned}$$

Note that we have substituted $p^k(n) = \left(\frac{k-1}{k}\right) p^{k-1}(n)$ in the bracketed term on the third line of the left-hand side of above inequality. Using the fact that $h_t(s_t) \ge 0$, we have

$$F_t^{k-1}(\hat{s}_t^k) \le H_t^k(\hat{s}_t^k) + \sum_{n\in\Omega_{t+}^{k-1}} p^{k-1}(n)\left[h_{t+}^{k-1}(\tilde{s}_{t+}^k(n)) - \left(\frac{k-1}{k}\right) h_{t+}^k(\hat{s}_{t+}^k(n))\right].$$

From Theorem 3.3, we have $h_{t+}^k \ge \left(\frac{k-1}{k}\right)^{T-t-1} h_{t+}^{k-1}$. This yields

$$F_t^{k-1}(\hat{s}_t^k) \le H_t^k(\hat{s}_t^k) + \sum_{n\in\Omega_{t+}^{k-1}} p^{k-1}(n)\left[h_{t+}^{k-1}(\tilde{s}_{t+}^k(n)) - \left(\frac{k-1}{k}\right)^{T-t} h_{t+}^{k-1}(\hat{s}_{t+}^k(n))\right].$$

Let us focus on the terms within the summation on the right hand side of above inequality, i.e., $\Delta_n^k = h_{t+}^{k-1}(\tilde{s}_{t+}^k(n)) - \left(\frac{k-1}{k}\right)^{T-t} h_{t+}^k(\hat{s}_{t+}^k(n))$. Then

$$\begin{aligned}\Delta_n^k = \; & h_{t+}^{k-1}(\tilde{s}_{t+}^k(n)) - h_{t+}^{k-1}(\hat{s}_{t+}^k(n)) + \left(1 - \left(\frac{k-1}{k}\right)^{T-t}\right) h_{t+}^{k-1}(\hat{s}_{t+}^k(n)) \\ \leq \; & |h_{t+}^{k-1}(\tilde{s}_{t+}^k(n)) - h_{t+}^{k-1}(\hat{s}_{t+}^k(n))| + \left|\left(1 - \left(\frac{k-1}{k}\right)^{T-t}\right) h_{t+}^{k-1}(\hat{s}_{t+}^k(n))\right|.\end{aligned}$$

The second absolute value term in the above expression vanishes as $k \to \infty$. Further, since $\{\hat{s}_t^k\}_{k\in\mathcal{K}_t} \to \bar{s}_t(s_{[t]})$, for every $\delta > 0$ there exists a $K(\delta) \in \mathcal{K}_t$ such that $\|\hat{s}_t^k - \tilde{s}_t^k\| < \delta$ for all $k > K(\delta)$. Using the uniform equicontinuity of $\{h_t^k\}$ (Theorem 3.3), we have $|h_{t+}^{k-1}(\tilde{s}_{t+}^k(n)) - h_{t+}^{k-1}(\hat{s}_{t+}^k(n))| < \epsilon$. Therefore, we can conclude that $\lim_{k\in\mathcal{K}} F_t^{k-1}(\hat{s}_t^k) - H_t^k(\hat{s}_t^k) \leq \epsilon$, for any $\epsilon > 0$.

To show the inequality in the other direction, we use the fact that $H_t^k(\hat{s}_t^k) \leq F_t^k(\hat{s}_t^k)$ and the uniform convergence of the sequence $\{F_t^k\}$. This gives us $\lim_{k\in\mathcal{K}_t} H_t^k(\hat{s}_t^k) - F_t^{k-1}(\hat{s}_t^k) \leq \epsilon$. Since inequalities hold in both directions for an arbitrary $\epsilon > 0$, we have

$$\lim_{k\in\mathcal{K}_t} |F_t^{k-1}(\hat{s}_t^k) - H_t^k(\hat{s}_t^k)| = 0 \quad (w.p.1). \tag{29}$$

Now consider the benchmark function $\mathcal{H}_t^k(\hat{s}_t^k)$ that is optimal across all possible sample-paths. Optimality of $\mathcal{H}_t^k(\hat{s}_t^k)$, along with the fact that $h_t^k \leq H_t^k$ (Theorem 3.2), we have

$$\lim_{k\in\mathcal{K}_t} \mathcal{H}_t^k(\hat{s}_t^k) \leq \lim_{k\in\mathcal{K}_t} h_t^k(\hat{s}_t^k) \leq \lim_{k\in\mathcal{K}_t} H_t^k(\hat{s}_t^k) \quad (w.p.1),$$

Moreover, the forward recursion objective function value satisfies $\lim_{k\in\mathcal{K}_t} F_t^{k-1}(\hat{s}_t^k) \leq \lim_{k\in\mathcal{K}_t} \mathcal{H}_t^k(\hat{s}_t^k)$ $(w.p.1)$. Therefore we have

$$\lim_{k\in\mathcal{K}_t} F_t^{k-1}(\hat{s}_t^k) \leq \lim_{k\in\mathcal{K}_t} \mathcal{H}_t^k(\hat{s}_t^k) \leq \lim_{k\in\mathcal{K}_t} h_t^k(\hat{s}_t^k) \leq \lim_{k\in\mathcal{K}_t} H_t^k(\hat{s}_t^k) \quad (w.p.1). \tag{30}$$

Using (29) in the above relation and the results in Lemma 4.3, we conclude that the expresstion (30) holds with equality, with probability one.

Since $\{\hat{s}_t^k\}_{k\in\mathcal{K}_t} \to \bar{s}$, the result in Theorem 4.2 shows the existence a subsequence $\overline{\mathcal{K}}_t$ such that $\{(\hat{s}_t^k, \hat{u}_t^k)\}_{k\in\overline{\mathcal{K}}_t} \to (\bar{s}_t, \bar{u}_t)$. Using, the uniform convergence of the sequence of minorants $\{h_t^k\}$ and benchmark function $\{\mathcal{H}_t^k\}$ (Theorem 3.3 and Lemma 4.3, respectively), we conclude that the function values $\{f_t^k(\hat{s}_t^k, \hat{u}_t^k(\hat{s}_t^k))\}$ converge to the optimal value at the accumulating state $\bar{s}_t$, with probability one. □

The above result holds at all non-terminal stages over any converging subsequence of states $\{\hat{s}_t^k\}_{k\in\mathcal{K}_t}$. Under the assumption of finite support (A5), the algorithm will generate such subsequences as asserted in Theorem 4.1 and Theorem 4.2.

### Optimality of the Incumbent Solution Sequence

Before establishing the optimality of the root-stage incumbent solution sequence, we establish the limiting relationship between the value function estimate at the candidate and incumbent solutions, $f_0^{k-1}(s_0, u_0^k)$ and $f_0^{k-1}(s_0, \hat{u}_0^{k-1})$, respectively. As a consequence of Theorem 4.4, the

root-stage value function is equivalent to the value function of a 2-SLP. This equivalent 2-SLP has the first-stage cost equal to $\langle c_0, x_0\rangle + \langle d_0, u_0\rangle$ and the expected recourse value given by $\sum_{\omega_1\in\Omega_1} p(\omega_1)h_1(s_1(\omega))$. The function $h_1(\cdot)$ is the optimal cost-to-go value starting from stage 1 which is attained for the limiting states $\bar{s}_1(\omega_1)$ for all $\omega_1 \in \Omega_1$. With this perspective, the following lemma parallels a result from [19] (Theorem 3). We present the proof for the case when the incumbent changes infinitely often and refer the reader to [19] for the case when the incumbent changes finitely often.

*Lemma* 4.5. *Let* $\{u_0^k\}_{k=1}^\infty$ *and* $\{\hat{u}_0^k\}_{k=1}^\infty$ *denote the sequence of candidate and incumbent solutions identified by SDLP, respectively. With probability one,*

$$\limsup_{k\to\infty} f_0^{k-1}(s_0, u_0^k) - f_0^{k-1}(s_0, \hat{u}_0^{k-1}) = 0. \tag{31}$$

*Proof.* Let $\{k_n\}_{n\in\mathcal{K}_0}$ represent the sequence of iterations at which the incumbent is changed. If $\mathcal{K}_0$ is an infinite set, then as a consequence of the incumbent update rule (23) and Theorem 4.4, we have

$$\lim_{m\to\infty} \frac{1}{m}\sum_{n=1}^{m} \Delta^{k_n} \le \limsup_{n\to\infty} \Delta^{k_n} \le 0.$$

From (25), there exists a subsequence $\mathcal{K}_0^* \subset \mathcal{K}_0$ such that

$$\lim_{k\in\mathcal{K}_0^*} \Delta^k = 0.$$

Since, $\Delta^k = f_0^{k-1}(s_0, \hat{u}_0^k) - f_0^{k-1}(s_0, \hat{u}_0^{k-1})$ and $\hat{u}_0^k = u_0^k$, for all $k \in \mathcal{K}_0^*$, establishes (31). □

The following result captures the asymptotic behavior of the directional derivatives of the sequence of first-stage objective function approximations. Specifically, it relates the directional derivatives of the approximate value function to that of the true value function.

*Lemma* 4.6. *Let* $u_t \in \mathcal{U}_t(s_t)$. *Define* $\delta_t^k(u_t) = \frac{u_t - u_t^{k-1}}{\|u_t - u_t^{k-1}\|}$ *and* $\bar{\delta}_t(u_t) = \frac{u_t - \bar{u}_t}{\|u_t - \bar{u}_t\|}$. *For any sequence* $\mathcal{K}$ *such that* $\{u_t^k\}_{k\in\mathcal{K}} \to \bar{u}_t$, $\bar{u}_t \in \mathcal{U}_t(s_t)$, *then*

$$\lim_{k\in\mathcal{K}} (f_t^k)'(s_t, u_t^{k-1}; \delta_t^k(u_t)) \le f_t'(s_t, \bar{u}_t; \bar{\delta}_t(u_t)), \tag{32}$$

*with probability one.*

*Proof.* Since $\{u_t^k\}_{k\in\mathcal{K}} \to \bar{u}_t$, we have $\delta_t^k(u_t) \to \bar{\delta}_t(u_t)$, for all $u_t \in \mathcal{U}_t(s_t)$. Note that,

$$\beta_{t+}^k(\omega_{t+}) = (h_{t+}^k)'(x_{t+}(\omega_{t+}), \omega_{t+}).$$

Following Lemma 3.1 (b) and Theorem 4.4, we have $\limsup_{k\in\mathcal{K}} (f_t^k)'(s_t, u_t^{k+1}; \delta_t^k(u_t)$ is finite and further, there is exists a subsequence $\overline{\mathcal{K}} \subset \mathcal{K}$ such that

$$\{\beta_{t+}^k(\omega_{t+})\}_{k\in\overline{\mathcal{K}}} \to \bar{\beta}_{t+}(\omega_{t+}) \in \partial h_{t+}(x_{t+}(\omega_{t+}), \omega_{t+}). \tag{33}$$

Using the definition of $f_t^k$ in (15), we have

$$(f_t^k)'(s_t, u_t^{k+1}; \delta_t^k(u_t)) = \langle d_t + \sum_{\omega_{t+}\in\Omega_{t+}^k} p^k(\omega_{t+})(h_{t+}^k)'(x_{t+}(\omega_{t+}), \omega_{t+}), \delta_t^k(u_t\rangle.$$

This implies that

$$\begin{aligned}\lim_{k\in\overline{\mathcal{K}}}(f_t^k)'(s_t,u_t^{k+1};\delta_t^k(u_t)) &= \limsup_{k\in\mathcal{K}}(f_t^k)'(s_t,u_t^{k+1};\delta_t^k(u_t))\\ &= \langle d_t+\sum_{\omega_{t+}\in\Omega_{t+}^k}p^k(\omega_{t+})(h_{t+}^k)'(x_{t+}(\omega_{t+}),\omega_{t+}),\delta_t^k(u_t)\rangle.\end{aligned}$$

Let $\bar{d}_t = d_t + \mathbb{E}[\bar{\beta}(\tilde{\omega})]$. Using (33) and the fact that $p^k(\omega)\to p(\omega)$, almost surely, we have

$$\begin{aligned}\limsup_{k\in\mathcal{K}}(f_t^k)'(s_t,u_t^{k+1};\delta_t^k(u_t)) &= \langle\bar{d}_t,\bar{\delta}_t(u_t)\rangle\\ &\leq \max\{\langle v,\bar{\delta}_t(u_t)\rangle \mid v\in\partial f_t(s_t,\bar{u}_t;\bar{\delta}(\bar{u}_t)\} = f'(s_t,\bar{u}_t;\bar{\delta}(\bar{u}_t).\end{aligned}$$

□

The above lemma mirrors a similar result for regularized 2-SD that appeared in [19] (as Lemma 4). We are now in a position to establish the optimality of the accumulation point of the sequence of root-stage incumbent solutions.

*Theorem* 4.7. *Suppose Assumptions (A1)-(A5) hold and $\underline{\sigma}\geq 1$, then the SDLP algorithm produces a sequence incumbent solutions at root-stage $\{\hat{u}_0^k\}\to u_0^*$ and $u_0^*$ is optimum, with probability one.*

*Proof.* Using the optimality condition of regularized root-stage problem (24), the result in Lemma 4.5 implies that there exists a subsequence $\mathcal{K}_0^*$ such that

$$\lim_{k\in\mathcal{K}_0^*} f_0^{k-1}(s_0,u_0^k) - f_0^{k-1}(s_0,\hat{u}_0^{k-1}) + \|u_0^k-\hat{u}_0^{k-1}\| = 0,$$

with probability one. Let $\overline{\mathcal{K}}_0^*\subset\mathcal{K}_0^*$ be such that $\{\hat{u}_0^k\}_{k\in\overline{\mathcal{K}}_0^*}\to\bar{u}_0$. Let $u_0\in\mathcal{U}_0$ be such that $u_0\neq\bar{u}_0$. We define

$$\delta_0^k(u_0) = \frac{u_0-u_0^k}{\|u_0-u_0^k\|},\text{ and }\bar{\delta}_0(u_0) = \frac{u_0-\bar{u}_0}{\|u_0-\bar{u}_0\|}.$$

Selecting $\delta_0^k(u_0)>0$ and for $0<\vartheta\leq\|u_0-u_0^k\|$, optimality of $u_0^k$ implies that

$$\begin{aligned}&f_0^k(s_0,u_0^k)+\frac{\sigma}{2}\|u_0^k-\hat{u}_0^{k-1}\|^2 \leq\\ &\qquad f_0^k(s_0,u_0^k+\vartheta\delta_0^k(u_0))+\frac{\sigma}{2}\|(u_0^k+\vartheta\delta_0^k(u_0))-\hat{u}_0^{k-1}\|^2.\\ \Rightarrow&[f_0^k(s_0,u_0^k+\vartheta\delta_0^k(u_0))-f_0^k(s_0,u_0^k)]+\\ &\qquad\frac{\sigma}{2}[\|(u_0^k+\vartheta\delta_0^k(u_0))-\hat{u}_0^{k-1}\|^2-\|u_0^k-\hat{u}_0^{k-1}\|^2]\geq 0.\\ \Rightarrow&(f_0^k)'(s_0,u_0^k;\delta_0^k(u_0))+\sigma\langle\delta_0^k(u_0),u_0^k-\hat{u}_0^{k-1}\rangle\geq 0.\end{aligned}$$

Note that the term $\langle\delta_0^k(u_0),u_0^k-\hat{u}_0^{k-1}\rangle$ is the directional derivative of the proximal term evaluated at $u_0^k$. Since $\lim_{k\in\overline{\mathcal{K}}_0^*}u_0^k = \lim_{k\in\overline{\mathcal{K}}_0^*}\hat{u}_0^{k-1}=\bar{u}_0$, the directional derivative of the proximal term vanishes in the limit. Therefore, we have

$$0\leq\liminf_{k\in\mathcal{K}_0^*}(f_0^k)'(s_0,u_0^k;\delta_0^k(u_0))\leq\limsup_{k\in\mathcal{K}_0^*}(f_0^k)'(s_0,u_0^k;\delta_0^k(u_0))\leq f'(s_0,\bar{u}_0;\bar{\delta}_0(u_0)).$$

The last inequality follows from Lemma 4.6. Since $f_0(\cdot)$ is convex function and the above statement implies that the directional derivatives $\bar{\delta}_0(u_0)$ are non-negative for an arbitrary $u_0\in\mathcal{U}_0$, it follows that $\bar{u}_0$ is an optimal solution. □

# 5 Conclusions

The SDLP algorithm extends the regularized 2-SD algorithm [19] to the MSLP setting where the underlying stochastic process exhibits stagewise independence. The algorithm addresses the state variable formulation of MSLP problems by employing sequential sampling. In this sense, it is a counterpart to the MSD algorithm of [36] which was designed for a case where the underlying uncertainty has a scenario tree structure. The algorithm presented in this paper incorporates several additional advantages granted by the stagewise independence property. We conclude here by noting the salient features of the SDLP algorithm:

1. The algorithm uses a single sample-path both for simulating decisions during the forward recursion and updating approximations during backward recursion. In any iteration, compared to SDDP which requires solving subproblems corresponding to all outcomes at all stages and for all sample-paths simulated during the forward pass, SDLP uses two subproblem solves at each stage. This significantly reduces the computational burden of solving MSLP problems.

2. The method is a stagewise proximal method with quadratic regularization terms at all non-terminal stages which alleviates the need to retain all the minorants generated. This allows us to retain a finite-sized approximation in all stages, further improving its computational advantage.

3. The BFP described in §3.3.2 is the first to provide a data-driven policy for MSLP. This mapping overcomes the need to store incumbent solutions that, either explicitly or implicitly, depending on the entire history of state evolution, and can be used with other regularized MSLP algorithms. Our convergence results show that the optimality of the accumulation points of a subsequence of incumbent solutions is preserved even when such a mapping is employed.

4. SDLP incorporates sampling within the optimization step, and thereby, optimizes an SAA with increasing sample size. This feature enables SDLP to solve the MSLP problems to greater accuracy by incorporating additional observations at any stage without having to re-discover the structural information of an instance to build/update the approximations. The adaptive nature allows the algorithm to be terminated upon attaining a desired level of accuracy. This opens the avenue to design statistical optimality rules for multistage settings akin to those developed for 2-SLP [21, 37].

The computational advantages of SDLP were revealed in our companion paper [12]. In that paper, we applied the SDLP algorithm to an MSLP model for distributed storage control in the presence of renewable generation uncertainty. The computational results compare our algorithm with SDDP applied to an SAA of the original model. The sample-paths used to set up the SAA and those used within the SDLP algorithm were simulated using an autoregressive moving-average time series model. The computational results of that paper indicate that SDLP provides solutions that are not only reliable but are also statistically indistinguishable from SDDP, while significantly improving the computational times. The computational advantage

of SDLP over SDDP can be attributed to the algorithm design. Namely, (i) the forward and backward recursion calculations are carried along only one sample-path in each iteration, and (ii) the use of regularization helps us maintain a finite-sized optimization problem at every non-terminal stage. Note that we are only referring to calculations within any particular iteration. In this sense our comparison is incomplete. However, carrying out a full theoretical comparison of SDLP and SDDP is beyond the scope of this paper. Nevertheless, we point the reader to recent results related to iteration complexity of the SDDP algorithm [25] and the sublinear rate of convergence for 2-SD in [27]. We plan to undertake the convergence rate analysis, (sample and iteration complexity) of SDLP in our future research endeavors. In any case, the results in [12] provide the first evidence of computational benefits provided by a sequential sampling approach in a multistage setting.

# A A Variant of Hoffman's Lemma

In this appendix we present a variant of the Hoffman's Lemma that is integral to the proof of Theorem 4.1.

*Lemma* A.1. *Consider a linear program* $\min_u \{\langle \rho, u\rangle$ *subject to* $Du \leq b - Cx\}$ *where* $x$ *and cost coefficient* $\rho$ *belong to compact sets* $\mathcal{X}$ *and* $\mathcal{R}$*, respectively. If the linear program is feasible for all* $x \in \mathcal{X}$ *and* $\mathfrak{U}(x,\rho)$ *be the set of optimal solutions, there exists exist positive constants* $\chi_1$ *and* $\chi_2$*, depending only on* $C$ *and* $D$*, such that for any* $(x,\rho),(x',\rho') \in$ *dom* $\mathfrak{U}$ *and any* $u \in \mathfrak{U}(x,\rho)$*,*

$$dist(u, \mathfrak{U}(x',\rho')) \leq \chi_1\|x - x'\| + \chi_2\|\rho - \rho'\|. \tag{34}$$

*Proof.* The linear program in (20) can be written in an equivalent form:

$$\min_{\eta\in\mathbb{R}} \ \eta \text{ subject to } Du \leq b - Cx, \ \langle \rho, u\rangle - \eta \leq 0. \tag{35}$$

Denote by $\mathcal{E}(x,\rho) := \{(u,\eta) \mid Du \leq b - Cx, \ \langle \rho, u\rangle - \eta \leq 0\}$ the set of feasible points of (35). Let $(x,\rho),(x',\rho') \in \text{dom } \mathfrak{U}$ and consider a point $(u,\eta) \in \mathcal{E}(x,\rho)$.

$$\begin{aligned}\text{dist}((u,\eta),\mathcal{E}(x',\rho')) &= \inf_{(u',\eta')\in\mathcal{E}(x',\rho')} \|(u,\eta) - (u',\eta')\| \\ &= \sup_{\substack{\lambda\geq 0,\ 0\leq\mu\leq 1 \\ \|\langle D,\lambda\rangle + \mu\rho'\|_*\leq 1}} \langle \lambda, [Du - (b - Cx')]\rangle + \mu[\langle \rho', u\rangle - \eta], \end{aligned} \tag{36}$$

where $\lambda$ and $\mu$ are the dual multipliers of constraints in (35), and $\|\cdot\|_*$ is the dual of the norm $\|\cdot\|$. The second equality is obtained using the same arguments as in the proof of Theorem 7.13 in [38].

Since the cost coefficient belongs to a compact set, there exists a constant $M_1 > 0$ such that $\|\rho\|_* < M_1$. Using this fact we can obtain a relaxation of the dual problem (36) by replacing the constraint $\|\langle D,\lambda\rangle + \mu\rho'\|_* \leq 1$ with the constraint $\|\langle D,\lambda\rangle\|_* \leq 1 + M_1$. Let $(\hat{\lambda},\hat{\mu})$ be an optimal solution of the dual problem. We can assume without loss of generality that $\|\cdot\|$ is the $\ell_1$ norm, and hence its dual is the $\ell_\infty$ norm. For such a choice of a polyhedral norm, we have that the feasible set of the dual problem (36) is a polytope. Therefore, $(\hat{\lambda},\hat{\mu})$ is an extreme point of the set $\{\lambda \mid \|\langle D,\lambda\rangle\|_* \leq 1 + M_1, \lambda \geq 0, 0 \leq \mu \leq 1\}$. This implies that $\|\lambda\|_*$ can be bounded by a

constant $\theta_1$ which depends only on the recourse matrix $D$. Let us examine the two terms in the objective of (36). Firstly,

$$\begin{aligned} \langle \hat{\lambda}, [Du - (b - Cx')] \rangle &= \langle \hat{\lambda}, [Du - (b - Cx)] \rangle + \langle \hat{\lambda}, C(x' - x) \rangle \\ &\leq \langle \hat{\lambda}, C(x - x') \rangle \leq \|\hat{\lambda}\|_* \|C\| \|x - x'\| \leq \theta_1 \theta_2 \|x - x'\|. \end{aligned} \tag{37}$$

The first inequality follows from the fact that $(u, \eta) \in \mathcal{E}(x, \rho)$ which implies that we have $Du - (b - Cx) \leq 0$. The third inequality is due to $\|\hat{\lambda}\|_* \leq \theta_1$ and $\|C\| \leq \theta_2$. The fact that $(u, \eta) \in \mathcal{E}(x, \rho)$ also implies $\langle \rho, u \rangle - \eta \leq 0$. Using this in the second term of (36), we have

$$\hat{\mu}[\langle \rho', u \rangle - \eta] = \hat{\mu}[\langle \rho, u \rangle - \eta] + \hat{\mu}\langle \rho' - \rho, u \rangle \leq \hat{\mu}\langle \rho' - \rho, u \rangle.$$

Recognizing that decision $u$ belongs to a compact set that depends on $x$ and $0 \leq \hat{\mu} \leq 1$, we have

$$\hat{\mu}[\langle \rho', u \rangle - \eta] \leq \hat{\mu}\|u\| \|\rho' - \rho\| \leq \theta_3(x) \|\rho' - \rho\|. \tag{38}$$

Using (37) and (38) in (36)

$$\begin{aligned} \text{dist}((u, \eta), \mathcal{E}(x', \rho')) &\leq \langle \hat{\lambda}, [Du - (b - Cx')] \rangle + \hat{\mu}[\langle \rho', u \rangle - \eta] \\ &\leq \theta_1 \theta_2 \|x - x'\| + \theta_3(x) \|\rho' - \rho\|. \end{aligned}$$

Setting $\chi_1 = \theta_1 \theta_2$ and $\chi_2 = \max_{x \in \mathcal{X}} \theta_3(x)$ completes the proof. □

## Acknowledgments

The authors are grateful to the Associate Editor Rene Henrion, and a referee for seeing the potential of the work presented in this paper. The second author is also grateful to several funding agencies (AFOSR, NSF, ONR) who have supported various aspects of his stochastic programming research. In particular, NSF funds have sustained his SP research continuously since 1991.

## References

[1] T. Asamov and W. Powell. Regularized decomposition of high-dimensional multistage stochastic programs with markov uncertainty. *SIAM Journal on Optimization*, 28(1):575–595, 2018.

[2] J.F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

[3] J. R. Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research*, 33(5):989–1007, 1985.

[4] M. S. Casey and S. Sen. The scenario generation algorithm for multistage stochastic linear programming. *Mathematics of Operations Research*, 30(3):615–631, 2005.

[5] Z.L. Chen and W.B. Powell. Convergent cutting-plane and partial-sampling algorithm for multistage stochastic linear programs with recourse. *Journal of Optimization Theory and Applications*, 102(3):497–524, 1999.

[6] Vitor L De Matos, Andy B Philpott, and Erlon C Finardi. Improving the performance of stochastic dual dynamic programming. *Journal of Computational and Applied Mathematics*, 290:196–208, 2015.

[7] C. Donohue and J.R. Birge. The abridged nested decomposition method for multistage stochastic linear programs with relatively complete recourse. *Algorithmic Operations Research*, 1(1), 2006.

[8] Jitka Dupačová, Giorgio Consigli, and Stein W Wallace. Scenarios for multistage stochastic programs. *Annals of operations research*, 100(1-4):25–53, 2000.

[9] J. Dupačová, N. Gröwe-Kuska, and W. Römisch. Scenario reduction in stochastic programming. *Mathematical Programming*, 95(3):493–511, 2003.

[10] M. Dyer and L. Stougie. Computational complexity of stochastic programming problems. *Mathematical Programming*, 106(3):423–432, 2006.

[11] Y. M. Ermol'ev. Stochastic quasigradient methods and their application to system optimization. *Stochastics*, 9, 1983.

[12] H. Gangammanavar and S. Sen. Two-scale stochastic optimization for controlling distributed storage devices. *IEEE Transactions on Smart Grid*, 9(4):2691–2702, July 2018.

[13] H. Gangammanavar, S. Sen, and V. M. Zavala. Stochastic optimization of sub-hourly economic dispatch with wind energy. *IEEE Transactions on Power Systems*, 31(2):949–959, March 2016.

[14] Harsha Gangammanavar, Yifan Liu, and Suvrajeet Sen. Stochastic decomposition for two-stage stochastic linear programs with random cost coefficients. *INFORMS Journal on Computing*, 33(1):51–71, January 2021.

[15] Vincent Guigues, Migual A. Lejeune, and Wajdi Tekaya. Regularized stochastic dual dynamic programming for convex nonlinear optimization problems. *Optimization and Engineering*, 21(3):1133–1165, jun 2020.

[16] Vincent Guigues and Werner Römisch. Sampling-based decomposition methods for multistage stochastic programs based on extended polyhedral risk measures. *SIAM Journal on Optimization*, 22(2):286–312, 2012.

[17] Grani A Hanasusanto, Daniel Kuhn, and Wolfram Wiesemann. A comment on "computational complexity of stochastic programming problems". *Mathematical Programming*, 159(1-2):557–569, 2016.

[18] J. L. Higle and S Sen. Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics of Operations Research*, 16(3):650–669, 1991.

[19] J. L. Higle and S Sen. Finite master programs in regularized stochastic decomposition. *Mathematical Programming*, 67(1-3):143–168, 1994.

[20] J. L. Higle and S Sen. *Stochastic Decomposition: A Statistical Method for Large Scale Stochastic Linear Programming*. Kluwer Academic Publishers, Boston, MA., 1996.

[21] J. L. Higle and S Sen. Statistical approximations for stochastic linear programming problems. *Annals of Operations Research*, 85(0):173–193, 1999.

[22] J.L. Higle and S. Sen. On the convergence of algorithms with implications for stochastic and nondifferentiable optimization. *Mathematics of Operations Research*, 17(1):112–131, 1992.

[23] G. Infanger and D. P. Morton. Cut sharing for multistage stochastic linear programs with interstage dependency. *Mathematical Programming*, 75(2):241–256, 1996.

[24] D. H. Jacobson and D. Q. Mayne. *Differential Dynamic Programming*. Elsevier, 1970.

[25] Guanghui Lan. Complexity of stochastic dual dynamic programming. *Mathematical Programming*, pages 1–38, 2020.

[26] K. Linowsky and A.B. Philpott. On the convergence of sampling-based decomposition algorithms for multistage stochastic programs. *Journal of Optimization Theory and Applications*, 125(2):349–366, 2005.

[27] Junyi Liu and Suvrajeet Sen. Asymptotic results of stochastic decomposition for two-stage stochastic quadratic programming. *SIAM Journal on Optimization*, 30(1):823–852, 2020.

[28] J. M. Mulvey and A. Ruszczyński. A new scenario decomposition method for large-scale stochastic optimization. *Operations Research*, 43(3):477–490, 1995.

[29] M.V.F. Pereira and L.M.V.G. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52(1-3):359–375, 1991.

[30] A. B. Philpott and V. L. de Matos. Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion. *European Journal of Operational Research*, 218(2):470 – 483, 2012.

[31] A. B. Philpott and Z Guan. On the convergence of stochastic dual dynamic programming and related methods. *Operations Research Letters*, 36(4):450 – 455, 2008.

[32] H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951.

[33] R. T. Rockafellar and R. J. B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Math. Oper. Res.*, 16(1):119–147, February 1991.

[34] W. Rudin. *Principles of mathematical analysis*. McGraw-Hill Book Co., New York, third edition, 1976. International Series in Pure and Applied Mathematics.

[35] A. Ruszczyński. A regularized decomposition method for minimizing a sum of polyhedral functions. *Mathematical Programming*, 35(3):309–333, 1986.

[36] S. Sen and Z. Zhou. Multistage Stochastic Decomposition: A bridge between Stochastic Programming and Approximate Dynamic Programming. *SIAM Journal on Optimization*, 24(1):127–153, 2014.

[37] Suvrajeet Sen and Yifan Liu. Mitigating uncertainty via compromise decisions in two-stage stochastic linear programming: Variance reduction. *Operations Research*, 64(6):1422–1437, 2016.

[38] A. Shapiro. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209(1):63 – 72, 2011.

[39] A. Shapiro, D. Dentcheva, and A. Ruszczynski. *Lectures on Stochastic Programming: Modeling and Theory, Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2014.

[40] R. M. Van Slyke and R. J. B. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663, 1969.

[41] Roger J-B Wets. Lipschitz continuity of inf-projections. *Computational Optimization and Applications*, 25(1-3):269–282, 2003.