

# Multistage Stochastic Decomposition: A Bridge between Stochastic Programming and Approximate Dynamic Programming

---

**First Version: May 27, 2009**

**Revision: February 3, 2012**

**Suvrajeet Sen**

Data Driven Decisions Laboratory  
ISE Department, Ohio State University  
Columbus, OH, 43210

**Zhihong Zhou**

MORE Institute, SIE Department  
University of Arizona, Tucson, AZ 85721

## Abstract

Multi-stage stochastic programs (MSP) pose some of the more challenging optimization problems. Because such models can become rather intractable in general, it is important to design algorithms that can provide approximations which, in the long run, yield solutions that are arbitrarily close to an optimum. In this paper, we propose a statistically motivated sequential sampling method that is applicable to multi-stage stochastic linear programs, and we refer to it as the multistage stochastic decomposition (MSD) algorithm. As with earlier SD methods for two-stage stochastic linear programs, this approach preserves one of the most attractive features of SD: asymptotic convergence of the solutions can be proven (with probability one) without any iteration requiring more than a small sample-size. This data-driven approach also allows us to sequentially update value function approximations, and the computations themselves can be organized in a manner that decomposes the scenario generation (stochastic) process from the optimization computations. As a by-product of this study, we also show that SD algorithms are essentially approximate dynamic programming algorithms for SP. Our asymptotic analysis also reveals conceptual connections between multiple SP algorithms.

**Keywords:** multi-stage stochastic programming, sequential sampling, stochastic decomposition, approximate dynamic programming

## 1 Introduction

Multi-stage Stochastic Linear Programming (MSLP) models have been applied in a variety of domains, including financial planning (Carino et al 1998), production systems (Boskma et al. 1977), power systems operations (Nowak and Römisich 2000), supply chain management (Tzur et al. 2006), and others. The key ingredient that makes MSLP attractive for these applications is the ability to plan in a manner that avoids myopic choices under continually evolving, and uncertain resource constraints.

Usually, the class of MSLP problems is computationally intractable even when the random variables in the MSLP have finite support. Although MSLP may be reformulated as linear programs, the number of constraints grows exponentially as the number of stages in MSLP increases. One might trace these difficulties to a result of Dyer and Stougie (2006) who show that the problem of optimizing a multi-dimensional expected value functional is inherently difficult because its decision counterpart is  $\#P$ -complete, in the worst case. In general, most applications of MSLP have been addressed via algorithms that use deterministic approximations of the value function (e.g. Birge 1985, Gassmann 1990, Rockafellar and Wets 1991, and Mulvey and Ruszczyński 1995). The underlying uncertainty and sequential evolution of data in these multistage sequential decision problems lead to a sequential optimization under uncertainty model. Unfortunately, as the stochastic process governing uncertainty becomes complicated (e.g. correlated exchange rates in financial models, Wu and Sen 2000), deterministic approximations for this class of problems can become unwieldy because of the need to represent uncertainty via a scenario tree. In order to avoid such an explosion, Dupačová et al (2003) have proposed effective ways to reduce the size of the scenario tree based on some predetermined limits on computational resources. Another approach which is applicable to certain classes of MSLP (e.g. problems with randomness only in the right-hand-side or the objective function) is provided in Casey and Sen (2005) where the approximations provide decisions which achieve a prescribed tolerance from optimality. For more general problems though, sample-based approximations provide the main route to scalable algorithms for stochastic programming.

Among sampling-based approaches for MSLP, the most popular ones are based on working with a sampled subtree in any given iteration. These approaches trace back to the work of Pereira and Pinto (1991) and more recently to Donahue and Birge (2006). The starting point of these methods is the deterministic nested Benders' decomposition approach, and sampling is used to create a sub-tree that is traversed to develop approximations. While these sample-based methods do provide a practical approach for approximate solutions of MSLP with large complicated

scenario trees, there are both computational and theoretical bottlenecks: a) even with discrete valued stochastic processes, it is unclear how one might interface with a simulation that might have the ability to generate a sample path, b) asymptotic convergence proofs require iterations which traverse the entire scenario tree at some steps of the method (Linowsky and Philpott 2005), and c) stagewise independence of the stochastic process appears to be critical (Shapiro 2010). In contrast, the multi-stage SD method proposed in this paper provides revised updates as more sample paths are observed sequentially. Indeed, as with its two-stage predecessor (Higle and Sen 1994, 1996), the multi-stage SD method learns to approximate value functions, as well as decisions in a sequential manner. Moreover, this sequential process is shown to provide asymptotic optimality without requiring us to traverse the entire tree in any one iteration, although each sample path with positive probability must be visited with probability one during execution of the algorithm.

The plan of this paper is as follows. Section 2 begins with a statement of the primal problem setting, and presents a review of some important concepts, such as scenario trees and duality in multi-stage stochastic programming. The algorithmic schema, as well as a discussion comparing MSD and other sampling methods is presented in section 3. In section 4, we prove its asymptotic convergence and our conclusions appear in section 5.

## 2 Multi-Stage Stochastic Programming Models

In our formulation, time will march ahead from now ( $t = 0$ ) to the end of horizon ( $t = T$ ), where  $T$  is a given positive integer. With this notation, there will be  $T + 1$  stages in the formulation, and thus a two-stage formulation will have stages indexed by  $t = 0$  and  $t = 1$ .

Let  $(\Omega, \mathcal{F}, \wp)$  denote a filtered probability space (i.e  $\mathcal{F}_t \subseteq \mathcal{F}$ , and for  $t = 1, \dots, T$   $\mathcal{F}_{t_1} \subseteq \mathcal{F}_{t_2}$  for  $t_1 < t_2$ ) which models the ever-increasing flow of data over time. Thus, the  $\sigma$ -algebras  $\mathcal{F}_t$  represent data available to the decision maker at time  $t$ . As usual,  $\Omega := \Omega_1 \times \dots \times \Omega_T$  ( $\Omega_t \subseteq \mathfrak{R}^{\nu_t}$  with  $\nu_t$  a positive integer) and, an outcome consisting of  $t$  periods of the process will be denoted  $\underline{\omega}_t := (\omega_1, \dots, \omega_t)$ , and the corresponding random variable will be denoted  $\underline{\tilde{\omega}}_t$ . We will assume that  $\underline{\tilde{\omega}}_t$  has finite support for all  $t$ , so that  $\mathcal{F}_t$  is finite, and so is the filtration  $\{\mathcal{F}_1, \dots, \mathcal{F}_T\}$ . Suppose that a finite partition  $\{\Theta_t^\ell\}$  of  $\Omega$  generates  $\mathcal{F}_t$ . Then the requirement of a filtration implies that for any set  $\bar{\Theta} \in \{\Theta_t^\ell\}$ ,  $\exists$  a collection of sets in  $\{\Theta_{t+1}^\ell\}$ , indexed by  $\mathbb{C}(\bar{\Theta})$ , say, such that  $\bar{\Theta} = \cup_{\ell \in \mathbb{C}(\bar{\Theta})} \Theta_{t+1}^\ell$ . The relationships between the sets in the partition  $\{\Theta_t^\ell\}$  and their children  $\mathbb{C}(\{\Theta_t^\ell\})$  can be encoded in the form of a tree which is termed as a scenario tree in SP. A node in period  $t$  represents a subset of paths (such as  $\bar{\Theta}$ ) which have the same events in the first  $t$  periods, and record some new event in period  $t + 1$ . Since algorithms will work by using the tree as a road-map, it will be convenient to index nodes of the scenario tree by  $n$ . We will denote any random child of node  $n$  by the notation  $\tilde{n}_+$ , so that the outcomes  $n_+ \in \mathbb{C}_n$ . By the

same token, arrival at node  $n$  of a scenario tree implies knowledge of the (unique) parent node which will be denoted by  $n -$ . Finally, letting  $p_n$  denote the probability of reaching node  $n$ , given that the process has arrived at node  $n -$ , we can simulate the generation of a sample path through the scenario tree. If  $\tilde{\omega}_t$  is a continuous stochastic process, then ensuring measurability of decisions requires greater care, and may resort to successive discretization (Casey and Sen 2005). We avoid such complications for this paper.

As with the two-stage SD algorithm, the multi-stage SD method will work with sample paths, although in this case, we will sample from the scenario tree with nodes  $n \in \mathcal{N}$ . For  $n \in \mathcal{N}$  we will associate a time index  $t(n)$ , and we will use the following notational convention.

- a) The root node is indexed by node 0.
- b) If node  $n$  belongs to the last stage, the expected value function of  $n +$  (the future) is 0.
- c) We define composite state variables  $\underline{s}_n = (x_n, \underline{\omega}_n)$ . Note that the composite state  $\underline{s}_n$  used here traces the entire history of the data state  $\underline{\omega}_n$ . In situations where we refer to an entire sample path, we will drop the subscript ( $n$ ) and simply represent the sampled data state by  $\underline{\omega}$ . We also note that while  $x_n$  is dependent on the history, we simplify the notation by showing dependence on node  $n$ , which is of course history dependent. Similarly, the history-dependence of constraints is also captured in the notation  $U_n(x_n)$ . We should clarify the distinction between the use of  $x_n$  as compared with  $\underline{s}_n$ : the former is an instantaneous algebraic variable, where as  $\underline{s}_n$  comprises of both the  $x_n$  as well as the history  $\underline{\omega}_n$ . Thus,  $\underline{s}_n$  denotes an outcome of a measurable random variable  $\tilde{\underline{s}}_n$ . Given the outcome  $\underline{s}_n$  the vector  $x_n$  as well as the history are assumed to be known.
- d) The initial state  $x_0$  is given

Given a scenario tree, a multi-stage decision model may be stated as follows.

$$(0) \quad \text{Min} \{d_0^\top u_0 + E[h_{0+}(\tilde{\underline{s}}_{0+})] : u_0 \in U_0, x_{0+} = a_{0+} + A_{0+}x_0 + B_{0+}u_0 \text{ a.s.}\}$$

where  $h_n$  are defined recursively for  $n \geq 1$  as

$$(1) \quad \text{Min} \left\{ \begin{array}{l} h_n(\underline{s}_n) = c_n^\top x_n + \\ d_n^\top u_n + E[h_{n+}(\tilde{\underline{s}}_{n+})] \\ u_n \in U_n(x_n) \\ U_n(x_n) = \left\{ u_n \mid \begin{array}{l} D_{t(n)} u_n \leq b_n - \\ C_n x_n \end{array} \right\} \text{ a.s.} \end{array} \right\}$$

where

$$(2) \quad x_{n+} = a_{n+} + A_{n+}x_n + B_{n+}u_n. \text{ a.s.}$$

Readers familiar with fixed recourse SP models might recognize that assuming  $D_n = D_{t(n)}$ , is the multi-stage version of the fixed-recourse assumption for two-stage problems. However, note that all other data elements are allowed to depend on the history of the stochastic process in node  $n$ .

In this formulation,  $u$  and  $x$  represent decision (or control) and state variables respectively and the constraints are required to hold almost surely. Constraints (2) represent system dynamics, while  $x_0$  is given as the initial state. In order to avoid certain algorithmic complications that arise from using non-negative state variables, we have stated the MSLP model without non-negativity restrictions on the state variables  $x_n$  which simplifies the presentation of the algorithm.

There are at least four advantages to the above formulation of the multi-stage SP model: a) the value function is stated in terms of the (state) variables  $s_n$  that couple successive stages in the model, and since these are usually in a lower dimensional space than the decision variables ( $u_n$ ), the approximations are more manageable, and scalable; b) in the course of the algorithmic development, it will become clear that SD provides a bridge between SP and ADP; c) the above notation is standard in dynamic systems theory and software (such as Matlab), and finally, d) our approach actually extends asymptotic convergence properties of both SP and ADP. Item a) above has also been observed in Powell (2007) who suggests that in resource allocation applications, most models have far fewer coupling (state) variables leading to approximations in lower dimensional spaces. Other applications, such as financial models, also satisfy such properties because the number of stocks that determine the set of decision variables is often much larger than the portfolio or the class of investments tracked over time. As for items b) and c) we note the role of SD as unifying algorithm between SP and ADP. In this sense, the notion of approximations has been a central focus of stochastic programming algorithms which allow both path dependent stochastic processes, as well as, constraints in the stochastic decision model. Finally recall that the implications of item d) have already been discussed in the introductory section.

### 3 The Multi-stage SD Algorithm

In the following we will use certain extensions of the regularized version of the two-stage SD algorithm (Higle and Sen 1994). In the multi-stage case, each non-terminal node  $n \in \mathcal{N}$  will be endowed with a mechanism to initialize and update incumbent decisions, denoted  $\hat{u}_n^{k-1}$  in iteration  $k$ . As with two-stage SD,  $\hat{u}_0^{k-1}$  will represent a solution that is estimated to be “the best” decision observed prior to iteration  $k$  (for stage 0). The incumbent decisions for the subsequent nodes are noted to be those future decisions that support the choice  $\hat{u}_0^{k-1}$ . Candidate decisions, denoted  $u_n^k$ , will refer to those solutions that are obtained by solving nodal decision simulations described below (see (3)). They are referred to as candidates because they may replace incumbent decisions. Because we will traverse only one path in any iteration, the algorithm will not visit other nodes that are not on the traversed path. Accordingly, candidate decisions will be generated for only a subset of nodes of the observed scenario tree, and these may become incumbent decisions if certain criteria are satisfied. Note that nodes that are not on the sample path for iteration  $k$  will not change their incumbent decisions. The states  $x_n$

generated using incumbent decisions will be referred to as incumbent states, and those associated with candidate decisions will be referred to as candidate states.

We begin with a summary of the algorithm which is discussed in greater depth subsequently.

---

1. Simulate decisions on a sample path ( $\omega^k$ ). Generate a path through the scenario tree, using the distribution provided by the model, but ensuring that the new path is generated independently of all previous paths. The tree revealed in the first  $k$  iterations will be denoted  $\mathcal{N}_k$ . We will refer to nodes along the sample path by the notation  $n \in \omega^k$ . The rest of step 1 executes either 1.1 or 1.2.

1.1. (No nodes on the path are new) If the current sample path, denoted  $\omega^k$ , has been revealed in some prior iteration, then assume that approximations  $f_n^{k-1}$  are available for all  $n \in \omega^k$ , as well as incumbent states and decisions for all  $n \in \omega^k$ . Starting with  $n = 0$ , we will optimize  $f_n^{k-1}$  for a given state  $x_0$ , denote the solution as  $u_n^k$  and then identify a candidate state  $x_{n+}^k = a_{n+} + A_{n+}x_n^k + B_{n+}u_n^k$ . Using the latter state, we obtain a candidate decision ( $u_{n+}^k$ ) by solving a nodal decision simulation (NDS, see (3)). This process is repeated for all non-terminal nodes on the sample path.

1.2. (Some nodes on the path are new). If some nodes  $n \in \omega^k$  have not been visited in previous iterations, then starting from node 0, move forward performing operations of step 1.1 until one arrives at a node that has never been visited by the algorithm. We then solve two scenario LPs associated with the remainder of the path. One LP is initialized with the incumbent state, and the other with the candidate state for the current iteration. The resulting solutions yield the incumbent and candidate sequences for nodes on the entire path  $\omega^k$ .

2. Solve the nodal dual approximation (see (5)). If control to this step is passed from step 1, then, update counts as well as frequencies reflecting the number of visits, the empirical conditional probability of visits, and use the terminal node for the sample path as node  $n$ ; otherwise use  $n$  as dictated by step 3 below. Solve  $NDA(n)$  using the incumbent state  $\hat{s}_n^{k-1}$ , as well as the candidate state  $\underline{s}_n^k$ , and proceed to step 3 using node  $n \leftarrow n -$ . (Note that this step is only performed for non-root nodes).

3. Update approximations. Using  $n$  provided by step 2, collect information regarding subgradients for  $h_{n+}^k$  (see(5)), and form two affine lower bounding approximations for  $h_n^k$  (One is the approximations obtained for the incumbent state  $\hat{s}_n^{k-1}$ , while the other is for the candidate state  $\underline{s}_n^k$ ). These updates provide functions  $f_n^k$  (see (10)). If  $n = 0$ , we proceed to step 4; otherwise, we return to step 2 with the current node  $n$ . (This step is only performed for non-root or non-terminal nodes).

4. Update incumbents. Let  $q \in (0,1)$ ,  $\bar{\sigma} > \underline{\sigma} > 0$  be given.

**if**  $f_0^k(u_0^k, x_0) - f_0^k(\hat{u}_0^{k-1}, x_0) \leq q[f_0^{k-1}(u_0^k, x_0) - f_0^{k-1}(\hat{u}_0^{k-1}, x_0)]$   $\hat{u}_0^k \leftarrow u_0^k$ ,  $\sigma_k \leftarrow \text{Max} \left\{ \frac{\sigma_k}{2}, \underline{\sigma} \right\}$   
**else**, we continue with  $\hat{u}_0^k \leftarrow \hat{u}_0^{k-1}$ ;  $\sigma_k \leftarrow \text{Min}\{2\sigma^k, \bar{\sigma}\}$ .

For nodes that do not belong to the  $k^{\text{th}}$  sample path, set  $h_n^k \leftarrow h_n^{k-1}$ . Increment the iteration counter  $k$ , and repeat from 1.

---

Full details of step  $r$  of the algorithmic steps are provided in subsection 3.  $r$

The following figure presents a caricature of one iteration of the algorithm. In the following, the solid lines joining the blank nodes represent the path generated in a forward pass of the algorithm, and the nodes marked with a cross represent previously visited children of nodes

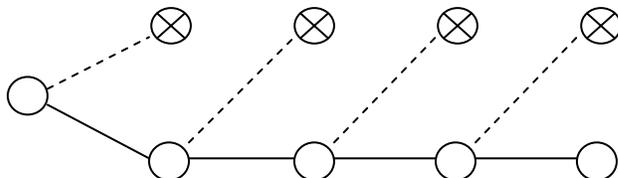


Figure 1: Caricature of visits to nodes within one iteration

associated with the current sample path. The calculations in step 1.1 generate the scenario shown by the blank nodes, and the computations necessary for obtaining a candidate sequence for that scenario. The calculations in step 1.2 are intended to generate initial incumbents for a new scenario. As for other calculations, step 2 requires the solution of one LP for each non-root node on the sample path. Indeed, step 2 starts with the terminal node (on the sample path), and upon solving the NDA for that node (referred to as  $n$ ), the algorithm proceeds to calculate the approximations for node  $n -$ , which will be designated as node  $n$  for step 3. Once the approximation is developed in step 3, it returns to step 2, with node  $n$  and one solves a new  $NDA(n)$ . In this manner, the algorithm moves backward in time until step 3 processes the root node (indexed by 0). At this point, a new approximation for the root node is at hand, and the algorithm proceeds to step 4.

It is important to recognize that in any iteration, approximations of the expected value functions are developed only for those nodes  $n$  that belong to the sample path generated in iteration  $k$  (i.e.  $n \in \underline{\omega}^k$ ). Thus, unlike other sampled approaches in stochastic programming (e.g. Donahue and Birge 2004, Linowsky and Philpott 2005), the multi-stage SD algorithm creates approximations of the expected value function of non-terminal nodes of the sample path  $\underline{\omega}^k$  generated in iteration  $k$ . Further details of each step are provided below.

Borrowing from the case of two-stage SD, we will make the following assumptions:

A1) the set of first-stage decisions ( $U_0$ ) is compact

A2) the complete recourse assumption is satisfied at every stage (i.e. (5) has a finite optimum for any setting of feasible state  $x_n$ )

A3) zero provides lower bound on all conditional expectations. (This assumption can be easily relaxed as in the two-stage case)

A4) Assume that for all  $t$ ,  $D_t$  has full row rank. In addition, we reiterate the fixed recourse assumption ( $D_n = D_{t(n)}$ ), as well as the requirement that the stochastic process has compact support.

A5) the scenario tree represents only nodes with conditional probability  $p_n > 0$ .

### 3.1. Simulate decisions on a sample path

We assume that the simulation will be consistent with the given stochastic process denoted  $\tilde{\omega}$ , and the generation of a sample path in any iteration must be accomplished via i.i.d sampling. Step 1 is called the forward pass because it is used to generate a sequence of states based on the first-stage incumbent ( $\hat{u}_0^{k-1}$ ) and candidate ( $u_0^k$ ) decisions. As in two-stage problems, we put  $u_0^k \in \operatorname{argmin} \{f_0^{k-1}(u_0, x_0) + \frac{\sigma_k}{2} \|u_0 - \hat{u}_0^{k-1}\|^2 : u_0 \in U_0\}$ . Given a decision  $u_n^k$ , one is able to obtain the next states  $\{x_{n+}^k\}$  using the dynamics by traversing the sampled path  $\underline{\omega}^k$  forward in time, starting with  $n = 0$  and then recursively calculating a candidate state trajectory by using the dynamics:  $x_{n+}^k = a_{n+} + A_{n+}x_n^k + B_{n+}u_n^k$ , followed by the decisions

$$(3) \quad u_{n+}^k \in \operatorname{argmin} \underbrace{\{f_{n+}^{k-1}(u_{n+}, x_{n+}^k) + \frac{\sigma_k}{2} \|u_{n+} - \hat{u}_{n+}^{k-1}\|^2 : u_{n+} \in U_{n+}(x_{n+}^k)\}}_{\text{Nodal Decision Simulation}},$$

where  $n+ \in \underline{\omega}^k$  and follows  $n$ . We refer to a decision problem in (3) as a nodal decision simulation because these forward passes generate decisions along the sample path during the forward simulation. Because sample paths may change from iteration to iteration, incumbent state trajectories  $\{\hat{x}_n^{k-1}\}$  along a sampled path must be updated to be consistent with the current incumbent  $\hat{u}_0^{k-1}$ . However, due to the relatively complete recourse assumption, the previously calculated incumbent decisions ( $\{\hat{u}_n^{k-1}\}$ ) at a sampled node need not be changed. Thus one solves only one nodal decision simulation for any non-terminal node on the sample path for iteration  $k$ .

The MSD algorithm will update the approximate functions  $\{f_n^k\}$  from iteration to iteration by using piecewise linear approximations developed during the backward pass (step 3). The parameter  $\sigma_k$  is also updated for computational efficiency; however it is important to maintain its value within a range that does not contain zero, and its upper limit should not be so large as to cause difficulties due to scaling (Higle and Sen 1994). Without loss of generality, we can therefore assume a lower bound to be  $\underline{\sigma} = 1$ .

Steps 1.1 and 1.2 represent two cases: the former applies when all nodes of the sampled path have been revealed in some previous iteration, while the latter case (1.2) plays a role in those iterations in which the sampling process discovers new nodes of the scenario tree. For such nodes, function approximations have not been created in any previous iteration, and as a result it is not possible to perform (3). What we recommend in this case is that a linear program be solved, starting from the first newly revealed node, with data corresponding to the rest of the sample path. The solution of this LP will provide the first primal incumbent for nodes on the path, as well as dual multiplier estimates to be used in the backward pass discussed next.

### 3.2. Solve nodal dual approximations

These calculations are carried out backward in time, along the path that was generated in the forward pass. Accordingly, the definitions that follow are best carried out in a recursive manner, starting from a terminal node. For nodes along the sample path  $\underline{\omega}^k$  the backward pass updates the functions  $f_n^k$  using two state trajectories  $\{\hat{x}_n^{k-1}\}$ , and  $\{x_n^k\}$ ,  $n \in \underline{\omega}^k$ , and the corresponding decisions  $\{\hat{u}_n^{k-1}\}$  and  $\{u_n^k\}$ ,  $n \in \underline{\omega}^k$  respectively. Whenever a particular calculation applies to both incumbent, and candidate solutions, we will use the notation  $x_n$  to denote either trajectory.

In developing the approximations, we will use the empirical distribution observed for transitions from node  $n$  to a child node  $n+$ . At iteration  $k$ , this estimate of the quantity  $(p_{n+})$  will be denoted by  $p_{n+}^k$ . Our estimates of probabilities will reflect the number of times the simulation has visited a given node, given that its parent node was visited.

One of the main ideas behind the multi-stage SD setup is that it highlights the use of state variables  $(\underline{s}_n := (x_n, \underline{\omega}_n))$  in the approximation process.

- For a terminal node  $n$ , we define  $E[h_{n+}] = 0$  and let  $h_n^k$  provide a lower bound on  $h_n$ .
- In order to define approximations for non-terminal nodes, we set forth a recursive definition by assuming that for all  $n+ \in \mathbb{C}_n$ , we have approximations  $h_{n+}^{k-1}$ , as well as empirical probabilities  $\{p_{n+}^k\}_{n+ \in \mathbb{C}_n}$ . As with the two-stage case, we initialize  $h_{n+}^0 := -\infty$ . Then, we will obtain  $h_n^k$  as an empirical *lower bounding approximation* to satisfy the following:

$$(4) \quad h_n^k(\underline{s}_n) \leq c_n^\top x_n + \text{Min} \{ d_n^\top u_n + \sum_{n+ \in \mathbb{C}_n} p_{n+}^k h_{n+}^k(\underline{s}_{n+}) : D_{t(n)} u_n \leq b_n - C_n x_n \}.$$

If the quantities  $h_{n+}^k$  on the right hand side of (4) are replaced by exact values  $h_{n+}$ , then the above statement becomes an equality, and in that case (4) is simply an application of the DP principle of optimality. Of course, if  $n$  is a terminal node, the above inequality will always hold as an equation because  $h_{n+}^k = h_{n+} = 0$ .

Suppose that for all  $n+ \in \mathbb{C}_n$ , we have already calculated a subgradient (with respect to  $x_{n+}$ )  $\beta_{n+}^k \in [\partial h_{n+}^k(\underline{s}_{n+}) - c_{n+}]$  and let  $\alpha_{n+}^k$  denote the constant term associated with the corresponding affine function (hyperplane). As with its two-stage counterpart, our MSD algorithm will derive two such approximations, one for candidate sequence, and another for the incumbent sequence. Since the calculations for both will be similar, we only derive these quantities for the candidate sequence, although we distinguish the two approximations by using a " $\wedge$ " above the approximation generated for the incumbent sequence (see (9)). Assuming relatively complete recourse, substituting the state variables  $x_{n+} = a_{n+} + A_{n+} x_n + B_{n+} u_n$ , and

replacing the primal value function by its dual representation, we obtain an affine lower bounding approximation as follows:

$$(5) \quad h_n^k(\underline{s}_n) \leq c_n^\top x_n + \sum_{n+ \in \mathbb{C}_n} p_{n+}^k \{ \alpha_{n+}^k + (c_{n+} + \beta_{n+}^k)^\top [a_{n+} + A_{n+} x_n] \} +$$

$$\underbrace{\text{Max } \pi_n^\top [b_n - C_n x_n] : \pi_n \leq 0, D_{t(n)}^\top \pi_n = d_n + \sum_{n+ \in \mathbb{C}_n} p_{n+}^k B_{n+}^\top (c_{n+} + \beta_{n+}^k)}_{\text{Nodal Dual Approximation for node } n}.$$

The LPs in (5) will be referred to as nodal dual approximations (NDA). In order to instantiate the NDA for node  $n$ , denoted  $NDA(n)$ , note that the right-hand side requires quantities  $p_{n+}^k, B_{n+}, c_{n+}, \beta_{n+}^k$  for  $n+ \in \mathbb{C}_n$ . Hence the quantities in NDA and the rest of the right-hand-side of (5) can be calculated recursively. Let  $\pi_n^k$  denote the basic feasible optimal solution with the least 2-norm obtained by setting  $x_n = x_n^k$  (a candidate) for the LPs in (5). Hence using  $\pi_n^k$ , and noting that the approximations  $h_n^k$  will be the maximum of the new affine function, as well as any previously generated affine functions, we have

$$(6) \quad h_n^k(\underline{s}_n) \geq (\pi_n^k)^\top b_n + (c_n - C_n^\top \pi_n^k)^\top x_n + \sum_{n+ \in \mathbb{C}_n} p_{n+}^k \{ \alpha_{n+}^k + (c_{n+} + \beta_{n+}^k)^\top [a_{n+} + A_{n+} x_n] \}.$$

Letting

$$(7) \quad \beta_n^k = -C_n^\top \pi_n^k + \sum_{n+ \in \mathbb{C}_n} p_{n+}^k A_{n+}^\top (c_{n+} + \beta_{n+}^k)$$

$$(8) \quad \text{and } \alpha_n^k = (\pi_n^k)^\top b_n + \sum_{n+ \in \mathbb{C}_n} p_{n+}^k \{ \alpha_{n+}^k + (c_{n+} + \beta_{n+}^k)^\top a_{n+} \},$$

we recursively obtain a subgradient  $\beta_n^k \in [\partial h_n^k(\underline{s}_n^k) - c_n]$  (with respect to  $x_n$ ) and the quantity  $\alpha_n^k$  represents the ‘‘constant’’ term of the affine function (hyperplane). These quantities will be used for approximations at the parent node  $n -$ .

### 3.3 Update approximations

Using the pairs (7), (8) (and their analogs for incumbent trajectories), we can now summarize the approximations developed at the candidate (and incumbent) state trajectories for the multi-stage SD algorithm with two new affine functions of the form  $\alpha_n^k + (c_n + \beta_n^k)^\top x_n$  for all  $n \in \underline{\omega}^k$ . This leads to the following updated approximation which satisfies (5) and (6) for nodes on the sample path in iteration  $k$ .

$$(9) \quad h_n^k(\underline{s}_n) := \begin{cases} \text{Max} \left\{ \alpha_n^k + (c_n + \beta_n^k)^\top x_n, \hat{\alpha}_n^k + (c_n + \hat{\beta}_n^k)^\top x_n, h_n^{k-1}(\underline{s}_n) \right\}, & \text{if } t(n) = T; \\ \text{Max} \left\{ \alpha_n^k + (c_n + \beta_n^k)^\top x_n, \hat{\alpha}_n^k + (c_n + \hat{\beta}_n^k)^\top x_n, \frac{\kappa_n^k - 1}{\kappa_n^k} h_n^{k-1}(\underline{s}_n) \right\}, & \text{if } 1 \leq t(n) < T; \end{cases}$$

where  $\kappa_n^k$  denotes the number of visits to node  $n$  when the  $k^{\{th\}}$  approximation is created. Note that if node  $n$  belongs to the sample path for iteration  $k$ , then  $\kappa_n^{k-1} = \kappa_n^k - 1$ . Hence if sample mean approximations were previously constructed using  $\kappa_n^{k-1}$  visits to node  $n$ , the multiplier  $\left(\frac{\kappa_n^k - 1}{\kappa_n^k}\right)$  reflects the increase in sample size by using the lower bound (assumed to be zero) as a new observation for all newly generated affine approximations for a non-terminal node  $n$ . Of course, in case of terminal nodes, there is no uncertainty in the future, and as a result the approximations require no further estimation (because sampling produces the same scenario).

In order to keep the notation manageable, we introduce an index set  $I_n^k$  that will index all affine functions defining  $h_n^k$  as follows:  $h_n^k(\underline{s}_n) := \text{Max} \left\{ \alpha_{in}^k + (c_n + \beta_{in}^k)^\top x_n : i \in I_n^k \right\}$ . Note that this is precisely the form of approximation used in L-shaped method/Benders' decomposition, and its variants. Finally, we put  $h_n^k \leftarrow h_n^{k-1}$  for all those nodes that do not belong to the path sampled in iteration  $k$ . Thus the approximations associated with all other nodes remain unaltered, although their impact on the parent node changes via the estimated probability update  $p_n^k$ . In any event, setting  $n \leftarrow n -$ , we obtain an updated approximation of the following form.

$$(10) \quad f_n^k(u_n, x_n) := c_n^\top x_n + d_n^\top u_n + \sum_{n+ \in \mathcal{C}_n} p_{n+}^k h_{n+}^k(\underline{s}_{n+}) : x_{n+} = a_{n+} + A_{n+} x_n + B_{n+} u_n \quad (\text{a.s.})$$

where the ‘‘almost sure’’ requirement is to be interpreted with respect to the empirical probability distribution observed at node  $n$ . Note that since  $\underline{s}_{n+} = (x_{n+}, \underline{u}_{n+})$ , the constraints in (10) provide terminal conditions whose ‘‘cost-to-go’’ is  $\sum_{n+ \in \mathcal{C}_n} p_{n+}^k h_{n+}^k(\underline{s}_{n+})$ , the expected recourse (value) function. Clearly, this approximation scheme which is based on the arguments of SD, has the same form as ADP, and hence forms a bridge between SP and ADP. Moreover, using (10) in (3) further clarifies the role of the principle of optimality in both SP and ADP.

It is interesting to recognize that the functions  $(f_n^k)$  which are optimized in the nodal decision simulation (i.e. (10)) can be accommodated in several ways: a) as in the original L-shaped method using aggregated cuts from all observations (Van Slyke and Wets 1969, Hight and Sen 1991), or b) via a multi-cut version as in Birge and Louveaux (1997), Ruszczyński (1986). Depending on the degree of uncertainty and nonlinearity of the value function, one can use either of these extremes (or some combination of the two) in solving the approximations. Furthermore we remind the reader that the forward simulation pass is performed by fixing  $x_n^k$ , and optimizing  $f_n^{k-1}(u_n, x_n^k)$ , whereas, the backward pass uses subgradients at the candidate and incumbent state trajectories (i.e.  $\beta_n^k \in \partial h_n^k(x_n^k) - c_n$  and  $\hat{\beta}_n^k \in \partial h_n^k(\hat{x}_n^k) - c_n$ ) to generate new

approximations. Note that unlike nested Benders' decomposition, and related stochastic dual dynamic programming, the backward pass in MSD uses specific future subgradients of  $NDA(n)$  (5), rather than optimizing over the entire collection of available subgradients. This has the advantage of maintaining a pure LP structure, rather than a piecewise LP structure as done in other previously mentioned multi-stage SP algorithms. An alternative, somewhat weaker version of these updates, appears in a related paper (Sen and Zhou 2011).

For situations in which the relatively complete recourse assumption is not justifiable, the algorithm can be modified to accommodate so-called feasibility cuts, which would have the form  $[\delta_{n+}^k]^\top [b_{n+} - C_{n+}x_{n+}] \leq 0$ , where  $\delta_{n+}^k$  denote some dual extreme direction along which the objective in (5) recedes to  $+\infty$ . For the purposes of this paper however, we continue with the relatively complete recourse assumption as stated earlier.

### 3.4 Update incumbent solutions

As with the two-stage regularized SD algorithm (Higle and Sen 1994) and other non-smooth optimization methods, the choice of an incumbent is based on predicted objective value reduction  $f_0(u_0, x_0)$  at the root node; that is,  $\hat{u}_0^k \leftarrow u_0^k$  if

$$(11) \quad f_0^k(u_0^k, x_0) - f_0^k(\hat{u}_0^{k-1}, x_0) \leq q[f_0^{k-1}(u_0^k, x_0) - f_0^{k-1}(\hat{u}_0^{k-1}, x_0)]$$

If the above inequality is satisfied, then all incumbent solutions on the sample path are updated to assume the values of the candidate decisions, although nodes that are not on the sample path retain previous values of incumbent decisions.

## 4 Asymptotic Convergence

While the algorithmic approach described above is entirely recursive, the analysis that follows will be more akin to the analysis of scenario-based SP algorithms, thus making SD an appropriate bridge between the SP and DP approaches. Because SP allows very general dependence structures, the multi-stage SD algorithm also inherits this generality. We emphasize that while the computational constructs for the algorithmic process have already been introduced in the previous section, some of the notations introduced below are simply intended for the purposes of analysis.

The collection of all scenarios generated will be denoted  $\mathcal{S}_0$ , and for nodes  $n+ \in \mathbb{C}_0$ , we define  $\mathcal{S}_{0+} \subseteq \mathcal{S}_0$  as the subset of scenarios that visit node "0 +". Recursively then, let  $\mathcal{S}_{n+}$  denote the subset of scenarios of  $\mathcal{S}_n$  passing through node  $n+$ . This process is the same as that used to form scenario trees from filtrations in section 2, and consequently, the frequency estimates lead to the conditional probability estimates, asymptotically. For any  $j \in (\cup_{n+ \in \mathbb{C}_n} \mathcal{S}_{n+})$  let  $x_{n+}^j$

and  $\mathcal{U}_{n+}^j$  denote sequences of state and decision vectors associated with scenario  $j$ , starting with node  $n +$ . Accordingly, assume that the vectors  $(x_n, \mathcal{X}_{n+}^j)$  and  $(u_n, \mathcal{U}_{n+}^j)$  satisfy feasibility, non-anticipativity, as well as dynamics as stated in (2), although it is the sampled subproblem starting at node  $n$ . Then, define a nodal sample mean approximation ( $NSMA(n)$ ) as

$$(12) \quad \mathcal{H}_n(\underline{s}_n | \mathcal{S}_n) := c_n^\top x_n + \text{Min} \left\{ d_n^\top u_n + \sum_{j \in (\cup_{n+ \in \mathbb{C}_n} \mathcal{S}_{n+})} \left( \frac{\kappa_j}{\kappa_n} \right) [\mathcal{C}_j(\mathcal{X}_{n+}^j) + \mathcal{D}_j(\mathcal{U}_{n+}^j)] \right\} = \\ c_n^\top x_n + \text{Min} \left\{ d_n^\top u_n + \sum_{n+ \in \mathbb{C}_n} \left( \frac{\sum_{j \in \mathcal{S}_{n+}} \kappa_j}{\kappa_n} \right) [\mathcal{H}_{n+}(\underline{s}_{n+} | \mathcal{S}_{n+})] \right\},$$

where  $\kappa_n = |\mathcal{S}_n|$ , and note that  $\kappa_n = \sum_{j \in \cup_{n+ \in \mathbb{C}_n} \mathcal{S}_{n+}} \kappa_j$ . To avoid further complicating the notation, we are not distinguishing between counters for scenarios ( $\kappa_j$ ) and counters for visits to node ( $\kappa_n$ ), and because the context (scenario/node) will be clear there should be no confusion. The functions  $\mathcal{C}_j(\mathcal{X}_{n+}^j)$  and  $\mathcal{D}_j(\mathcal{U}_{n+}^j)$  represent the cost for scenario  $j$ , starting with state  $\underline{s}_n$ . Since we are interested in asymptotic analysis, we assume that  $\kappa_n$ , the number of sampled paths is large enough for all  $n \in \mathcal{N}$ . Next assume that both non-anticipativity and dynamics are stated in polyhedral form (see e.g. Mulvey and Ruszczyński 1995, and Higle and Sen 2006), and the frequency estimates  $\left( \frac{\kappa_j}{\kappa_n} \right)$  appear only in the objective function, but not in the constraints. The notation  $\mathcal{H}_n(x_n | \mathcal{S}_n)$  is intended to convey the sense that the sample is given, and accordingly, its size  $\kappa_n$  as well as the counts  $\kappa_j$  are also given. In the following we will indicate the dependence of  $NSMA(n)$  on the iteration counter ( $k$ ) as well as the frequency estimates  $\left( \frac{\kappa_j}{\kappa_n} \right)$  by using the sample  $\mathcal{S}_n^k$  in  $\mathcal{H}_n(x_n | \mathcal{S}_n^k)$ .

In the design of SD presented in the previous section, we do not solve any of the nodal sample mean approximations; instead we sequentially develop approximations of these nodal sample mean approximations, and they will be shown to provide asymptotically accurate estimates of the objective function. To see how the approximations in SD compare with (12), let us relate the latter to the nodal calculations of SD. We first observe that  $p_{n+}^k = \left( \sum_{j \in \mathcal{S}_{n+}^k} \kappa_j^k \right) / \kappa_n^k$ , and for large enough  $k$  such that  $p_{n+}^k > 0$  for all  $n+ \in \mathbb{C}_n$ , let us examine how the approximations  $f_n^k$  in (10) compare with  $NSMA(n)$  in (12). Both (10) and (12) require three properties: feasibility of  $u_n$  (in the sense that  $u_n \in U(x_n)$ ), as well as feasibility, non-anticipativity and dynamics of all future states and decisions. However, there is an important difference: (12) requires optimality with respect to the exact sample mean objective for nodes  $n +$  and beyond, whereas (10) requires optimality with respect to future sample mean approximations  $h_{n+}^k$ . Through the nodal decision simulations in the forward pass, SD chooses  $u_n$ , and based on the choice of nodes dictated via sampling, the backward pass uses one child node in  $\mathbb{C}_n$  whose piecewise linear approximation is updated. Other approximations remain unchanged during any iteration. Let

$$(13) \quad u_n^{k+1}(x_n) \in \text{argmin} \left\{ d_n^\top u_n + \sum_{n+ \in \mathbb{C}_n} [p_{n+}^k h_{n+}^k(x_{n+})] + \frac{\sigma_k}{2} \|u_n - \hat{u}_n^k\|^2 \right\},$$

where  $u_n \in U_n(x_n)$ , and  $x_{n+} = a_{n+} + A_{n+}x_n + B_{n+}u_n$ .

Using  $x_{n+}^{k+1}(x_n) := a_{n+} + A_{n+}x_n + B_{n+}u_n^{k+1}(x_n)$ , define

$$(14) \quad F_n^k(\underline{s}_n) := c_n^\top x_n + d_n^\top u_n^{k+1}(x_n) + \sum_{n+ \in \mathcal{C}_n} \left[ p_{n+}^k h_{n+}^k \left( x_{n+}^{k+1}(x_n) \right) \right],$$

$$(15) \quad \phi_n^k(\underline{s}_n) := F_n^k(\underline{s}_n) + \frac{\sigma_k}{2} \|u_n^{k+1}(x_n) - \hat{u}_n^k\|^2.$$

Note that as with the sample mean approximation  $\mathcal{H}_n(x_n | \mathcal{S}_n^k)$ , the approximations defined in (14) and (15) are also dependent on the sample  $\mathcal{S}_n^k$ . Hence, strictly speaking it would have been appropriate to indicate the sample dependence for these functions too. However, recognizing that all three functions depend on the same sample  $\mathcal{S}_n^k$ , we have chosen to simplify the notation for (14) and (15) by simply using the superscript  $k$ .

In order to study for the asymptotic behavior of the MSD algorithm, we investigate how the functions  $\phi_n^k(\underline{s}_n)$ ,  $F_n^k(\underline{s}_n)$ ,  $\mathcal{H}_n(\underline{s}_n | \mathcal{S}_n^k)$  and  $h_n(\underline{s}_n)$  compare at limiting states (if such exist). The proof is based on a dynamic version of regularization (Ruszczynski 1986), subdifferential compatibility (Higle and Sen 1992, Rockafellar and Wets 1997), and epi-consistency (King and Wets 1991). These concepts were combined previously for asymptotic results two-stage SD in Higle and Sen (1994, 1996). The following concept will be useful in the analysis.

Definition (Higle and Sen 1992): A sequence of functions  $\{\phi^k\}$  and a function  $\phi$  are said to be subdifferentially compatible with respect to a sequence of points  $\{x^k\}$  if for any subsequence such that  $\{x^k\}_{k \in K} \rightarrow \bar{x}$  one has  $\limsup_{k \in K} \partial \phi^k(x^k) \subseteq \partial \phi(\bar{x})$ . A shorthand representation of this notion is  $\{\phi^k\} \xrightarrow{\partial} \phi$  (wrt  $\{x^k\}$ ).

Corollary 8 of Higle and Sen (1992) states that under subdifferential compatibility, the decisions generated via minimization of the approximation creates a sequence whose accumulation points belong to the set of optimal solutions to the original problem of optimizing  $\phi$ . The terrain that we explore in proving asymptotic optimality can be summarized by the following relations,

$$\phi_n^k \xrightarrow{\partial} F_n^k \xrightarrow{\partial} \mathcal{H}_n(\cdot | \mathcal{S}_n^k) \xrightarrow{\text{wp1}} h_n \quad (\text{wrt } \{\hat{\underline{s}}_n^k\}),$$

which requires us to show that for the sequence of states  $\{\hat{\underline{s}}_n^k\}$ , the sequence of function values and subgradients on the left hand side of an arrow satisfy subdifferential compatibility for the sequence on the right hand side of the arrow. Of the three arrows above, the right-most one is classical (see e.g. Rubin 1956, Rudin 1976): it focuses attention on the so-called sample mean (average) approximation (for multi-stage problems). This is what one relies on when an algorithm samples the original scenario tree to create a more manageable sub-tree, as in previously studied sampling algorithms for MSLP. The middle arrow is what distinguishes SD from sample mean approximation, whereby piecewise linear approximations learn the sample

mean approximation, asymptotically. Finally, the left-most arrow is used to ensure certain regularizing properties such as compactness and convergence of state and decision trajectories. We start with the regularizing properties (e.g. compactness).

#### 4.1 Lemma for Subgradient Compactness.

Suppose that the multi-stage SD algorithm runs for infinitely many iterations. For any node  $n$ , let  $I_n^k$  denote the index set of affine functions in the definition of  $h_n^k$  (see (9)). Under assumptions A1, A2 and A3, the collection of vectors  $\{V_n^k := (\alpha_{in}^k, \beta_{in}^k), i \in I_n^k\}$  belongs to a compact set for all  $k$  and  $n$ .

**Proof.** It is sufficient to show that the subgradients generated in each iteration, denoted  $\beta_{in}^k$ , belong to a compact set because all subsequent updates can be interpreted as convex combinations of  $\beta_{in}^k$  and 0. It is therefore convenient to drop the index  $i$  for the remainder of the proof. For any terminal node  $n$ , the complete recourse assumption ensures that an optimal solution of the nodal dual approximation is obtained at a basic feasible solution, and using  $p_{n+}^k = 0$ , we conclude that for all  $n$  with  $t(n) = T$ ,  $\beta_n^k = -C_n^\top \pi_n^k$  belongs to a compact space (because  $\pi_n$  are vertices of a fixed dual polyhedron). Hence the result is true for all terminal nodes. Next assume that for some fixed  $\tau \leq T$  this property is true for all nodes  $n$  such that  $\tau = t(n) \leq T$ . We show that this implies compactness of  $\{\beta_{n-}^k\}$  also holds for all parent nodes  $n-$ , where  $t(n-) = \tau - 1$ . From (7), it is clear that for nodes  $n-$  such that  $t(n-) = \tau - 1$ , the future reflected by  $\sum_{n \in \mathbb{C}_{n-}} p_n^k A_n^\top (c_n + \beta_n^k)$  is bounded due to the induction hypothesis. Hence  $\beta_{n-}^k$  is bounded if and only if  $\pi_{n-}^k$  is bounded. Since the complete recourse assumption allows us to restrict our attention to optimal nodal dual solutions that are basic feasible solutions of (5), the boundedness of  $\{d_{n-} + \sum_{n \in \mathbb{C}_{n-}} p_n^k A_n^\top (c_n + \beta_n^k)\}$  and the fixed matrix  $D_{\tau-1}$  imply that  $\pi_{n-}^k$  is bounded and the result follows. ■

#### 4.2 Lemma for Solution Stability and Compactness.

Suppose assumptions A1-A5 hold, and  $\underline{\sigma} \geq 1$ . Let let  $\{\hat{u}_0^k\} \subseteq U_0$  denote any infinite sequence of first stage incumbent decisions. Then for each  $n \in \mathcal{N}$  there exists a subsequence of iterations indexed by  $K_n$  such that the incumbent states  $\{\hat{x}_n^k\}_{K_n}$  as well as incumbent decisions  $\{\hat{u}_n^k\}_{K_n}$  have accumulation points.

**Proof.** First consider any node  $n$  with  $t(n) = 1$ . Since  $U_0(x_0)$  is compact set by assumption, there exists a subsequence indexed by  $K_0$  such that for  $\in K_0$ ,  $\{\hat{u}_0^k\}_{K_0} \rightarrow \bar{u}_0$ . Then the linearity of the state dynamics implies that for any  $n$  such that  $t(n) = 1$ ,  $\{\hat{x}_n^k\}_{K_n} \rightarrow \bar{x}_n := a_n + A_n x_0 + B_n \bar{u}_0$ , where  $K_n$  is a subsequence of  $K_0$ , starting with the earliest iteration in  $K_0$  after the first visit to node  $n$ . Next we consider the incumbent solutions  $\hat{u}_n^k$ , where once again  $k \in K_n$ . Since incumbent solutions are a subsequence of candidate solutions, we study the stability of candidate solutions. Note that the variables in the nodal decision simulation (NDS) can be written in the

form  $u_n^k = \hat{u}_n^{k-1} + \Delta u_n^k$ , where  $\Delta u_n^k \in \operatorname{argmin} \{f_n^{k-1}(\hat{u}_n^{k-1} + \Delta u_n, \hat{x}_n^k) + \frac{\sigma_k}{2} \|\Delta u_n\|^2 : \hat{u}_n^{k-1} + \Delta u_n \in U_n(\hat{x}_n^k)\}$ . As before, this optimization problem is denoted  $NDS(n)$ , and we denote its solution set by  $\operatorname{argmin}(NDS(n))$ . Since  $\sigma_k > \underline{\sigma} \geq 1$  (by construction), this is a positive definite piecewise linear quadratic program with linear constraints, which has unique primal and dual optimal solutions (see Chapter 4 in Higle and Sen 1996). Let  $\theta_n^k$  denote the vector of dual multipliers corresponding to the subgradient inequalities in  $NDS(n)$ . Dualizing all subgradient inequalities by using the optimal dual multipliers  $\theta_n^k$ , one obtains a quadratic programming subproblem whose solution set is identical to that of  $NDS(n)$ . Since this quadratic programming subproblem is stable to perturbations of the right-hand side and the linear part of the objective (see Guddat 1976, section 5), it follows that the solution set mapping (for  $NDS(n)$ )  $\psi_n: \hat{x}_n \rightarrow \operatorname{argmin}(NDS(n))$  is also continuous. Consequently it follows that  $\{\hat{x}_n^k\}_{k_n} \rightarrow \bar{x}_n$  implies that  $\{\hat{u}_n^k\}_{k_n} \rightarrow \bar{u}_n$  for all nodes  $n$  such that  $t(n) = 1$ . Using the above argument recursively, one concludes the validity of the result for all nodes  $n \in \mathcal{N}$ . ■

### 4.3 Corollary for Uniform Convergence.

Suppose assumptions A1-A5 hold.

- a) The sequence of functions  $\{h_n^k\}_k$  is uniformly equicontinuous, and uniformly convergent for all  $n$ .
- b) The sequence of functions  $\{F_n^k\}_k$  is uniformly equicontinuous, and uniformly convergent for all  $n$ .
- c) The sequence of functions  $\{\mathcal{H}_n(\cdot | \mathcal{S}_n^k)\}_k$  converges uniformly with probability one to the expectation  $h_n$ .

**Proof.** a) From Lemma 4.2 one is able to restrict attention to a compact subset of decisions for each  $n$ . As observed in concluding line of the proof of Lemma 4.1, the boundedness of  $\{d_{n-} + \sum_{n \in \mathbb{C}_{n-}} p_n^k A_n^\top (c_n + \beta_n^k)\}$  together with the fixed recourse matrix  $D_{t(n-)}$  implies that  $\{h_n^k\}$  must have a uniform Lipschitz constant, leading to the conclusion stated in the corollary.

b) The complete recourse assumption ensures that the functions  $F_n^k$  have finite value, and recall from quadratic programming optimality conditions that the solution mapping  $u_n^{k+1}(x)$  is piecewise linear. Hence the sequence of functions  $F_n^k(\underline{s}_n) := c_n^\top x_n + d_n^\top u_n^{k+1}(x_n) + \sum_{n+ \in \mathbb{C}_n} [p_{n+}^k h_{n+}^k(x_{n+}^{k+1}(x_n))]$  is also bounded over a compact set of decisions, and moreover, a uniform Lipschitz constant also exists, thus leading to the same conclusion as in part a).

c) This is the classical result of sample mean approximations converging uniformly to the mean with probability one, so long as the approximations are constructed by i.i.d sampling with an ever increasing sample size, the existence of a uniform Lipschitz constant (Lemma 4.1) and compactness of the decision/control/parameter space (Lemma 4.2) can be established (Rubin 1956, Rudin 1976). ■

#### 4.4 Theorem for Convergence of Decisions.

Suppose assumptions A1-A5 hold, and  $\underline{\sigma} \geq 1$ . Then there exists  $\bar{u}_0 \in U_0$ , such that  $\{\hat{u}_0\} \rightarrow \bar{u}_0$ , and more generally, there exist  $\bar{u}_n(\bar{x}_n) \in U_n(\bar{x}_n)$ , where the dynamics in (2) are satisfied, and  $\{\hat{u}_n^k(\hat{x}_n^k)\} \rightarrow \bar{u}_n(\bar{x}_n)$  for all  $n \in \mathcal{N}$ .

**Proof.** If the incumbent at the root node changes only finitely many times, then the result is obviously true. So, consider the case in which the incumbent changes infinitely many times. First consider node 0. Since  $x_0$  is given, it will be convenient to refer to the objective function at node 0, simply by  $f_0^k(u_0)$ , rather than the more cumbersome  $f_0^k(u_0, x_0)$ . The optimality conditions for the regularized approximation at the root node (see equation (2.6) on page 115 of Hige and Sen 1996) and our choice  $\underline{\sigma} \geq 1$  implies that

$$(16) \quad f_0^k(u_0^{k+1}) - f_0^k(\hat{u}_0^k) \leq -\|u_0^{k+1} - \hat{u}_0^k\|^2 \leq 0.$$

Let  $K_0$  denote iterations at which the incumbent changes, and for  $m$  successive indices in  $K_0$ , we have  $\frac{1}{m} \sum_{\ell} [f_0^{k_\ell}(u_0^{k_\ell+1}) - f_0^{k_\ell}(\hat{u}_0^{k_\ell})] \leq -\frac{1}{m} \sum_{\ell} \|\hat{u}_0^{k_\ell+1} - \hat{u}_0^{k_\ell}\|^2 \leq 0$ . But since  $k_\ell \in K_0$ , we have  $\hat{u}_0^{k_\ell+1} = u_0^{k_\ell+1}$ , and the left hand side of this inequality can be written as

$$(17) \quad \Delta_m := \frac{1}{m} [f_0^{k_m}(\hat{u}_0^{k_{m+1}}) - f_0^{k_0}(\hat{u}_0^{k_0})] + \frac{1}{m} \sum_{\ell=0}^{m-1} [f_0^{k_\ell}(\hat{u}_0^{k_{\ell+1}}) - f_0^{k_{\ell+1}}(\hat{u}_0^{k_{\ell+1}})].$$

From Corollary 4.3 the summation term in (17) converges to 0, whereas, the first term (in square brackets) has a finite upper bound (due to upper and lower bounds on the approximations  $f_0^k$ ).

Hence  $\lim_{m \rightarrow \infty} \Delta_m \rightarrow 0$ . It follows that  $\lim_{m \rightarrow \infty} \frac{1}{m} \sum_{\ell} \|\hat{u}_0^{k_\ell+1} - \hat{u}_0^{k_\ell}\|^2 \rightarrow 0$ . Hence the entire sequence of incumbents generated at node 0 must converge, and we denote such a point by  $\bar{u}_0 \in U_0$ .

Let us now proceed to other nodes on the tree. If a node is visited finitely many times, the result is clearly true. Next consider nodes that are visited infinitely many times. Since the incumbent decisions converge for  $t = 0$ , the future incumbent states  $\{\hat{x}_n^k\}$  also converge (as in Lemma 4.2). Hence there exists a sequence of incumbent states that satisfy the equivalent of (16); that is,  $F_n^k(\hat{x}_n^{k+1}) - F_n^k(\hat{x}_n^k) \leq -\|u_n^{k+1}(x_n^{k+1}) - \hat{u}_n^k(\hat{x}_n^k)\|^2 \leq 0$ , where  $k \in K_n$  (as in the proof of Lemma 4.2 and the previous paragraph). Once again, using the same arguments as in the previous paragraph, we conclude that  $\{\hat{u}_n^k(\hat{x}_n^k)\} \rightarrow \bar{u}_n(\bar{x}_n)$  for all  $n \in \mathcal{N}$ . ■

Finally, we present the main asymptotic optimality result of MSD.

#### 4.5 Theorem for Asymptotic Consistency and Optimality.

Suppose assumptions A1-A5 hold, and  $\underline{\sigma} \geq 1$ . For all  $n$  such that  $1 \leq t(n) \leq T$ , we have

$$\phi_n^k \xrightarrow{\partial} F_n^k \xrightarrow{\partial} \mathcal{H}_n(\cdot | \mathcal{S}_n^k) \text{ (wrt } \{\hat{\underline{s}}_n^k\}),$$

(which requires us to show that for the sequence of states  $\{\hat{\underline{s}}_n^k\}$ , the sequence of function values and subgradients on the left hand side of an arrow satisfy subdifferential compatibility for the sequence on the right hand side of the arrow). Moreover, subdifferential compatibility for node 0 implies that  $\bar{u}_0 \in U_0$  is optimal (wp1).

**Proof.** First note that  $\{\hat{u}_n^k(\hat{x}_n^k)\} \rightarrow \bar{u}_n(\bar{x}_n)$  implies that the sequence of state  $\{\hat{\underline{s}}_n^k\}$  converges for all  $n$ . Now, let us investigate the subdifferential compatibility as stated. The definition of subgradients and the updates in (9) together with Theorem 4.4. imply subdifferential compatibility of  $\phi_n^k$  and  $F_n^k$  for all  $n$ . In order to prove the validity of the next arrow, consider  $n$  such that  $t(n) = T$  (terminal nodes). Since  $\{\hat{\underline{s}}_n^k\} \rightarrow \bar{\underline{s}}_n$  and the functions  $h_n^k$  and  $\mathcal{H}_n(\cdot | \mathcal{S}_n^k)$  are equal for such (terminal)  $n$  and all states  $\hat{\underline{s}}_n^k$ , they are also subdifferentially compatible. Next we consider any parent node  $n$ , such that  $t(n) = T - 1$ . Let  $F_n := \lim_{k \rightarrow \infty} F_n^k$ . Because the incumbent sequence is created from the candidate sequence (i.e.  $\hat{u}_n^k = u_n^k$  for the chosen indices  $k$ ) and  $\{\hat{\underline{s}}_{n+}^k\} \rightarrow \bar{\underline{s}}_{n+}$ , (10), (13), and (14) imply

$$(18) \quad \lim_{k \rightarrow \infty} h_n^k(\hat{\underline{s}}_n^k) = F_n(\bar{\underline{s}}_n).$$

Moreover, the induction hypothesis implies that  $h_{n+}^k(\cdot)$  and  $\mathcal{H}_{n+}(\cdot | \mathcal{S}_{n+}^k)$  are sub-differentially compatible wrt  $\{\hat{\underline{s}}_{n+}^k\} \rightarrow \bar{\underline{s}}_{n+}$  for  $n+ \in \mathbb{C}_n$ . From the development immediately preceding (13), we recall that  $p_{n+}^k = (\sum_{j \in \mathcal{S}_{n+}^k} \kappa_j^k) / \kappa_n^k$ . Consequently,  $\{\hat{\underline{s}}_n^k\} \rightarrow \bar{\underline{s}}_n$ , and  $\{\hat{\underline{s}}_{n+}^k\} \rightarrow \bar{\underline{s}}_{n+}$  for all  $n+ \in \mathbb{C}_n$ , together with (12), (13) and (14) imply that  $F_n(\bar{\underline{s}}_n) = \lim_{k \rightarrow \infty} \mathcal{H}_n(\underline{s}_n^k | \mathcal{S}_n^k)$ . Combining this asymptotic behavior with that in (18), and the lower bounding behavior (5) obeyed by (9), we conclude that  $F_n^k$  and  $h_n^k$  are both subdifferentially compatible with  $\mathcal{H}_n(\underline{s}_n^k | \mathcal{S}_n^k)$  with respect to  $\{\hat{\underline{s}}_n^k\} \rightarrow \bar{\underline{s}}_n$ , where  $n$  satisfies  $t(n) = T - 1$ . Using this reasoning backwards recursively leads us to the conclusion that  $F_{0+}^k$  and  $\mathcal{H}_{0+}(\hat{\underline{s}}_{0+}^k | \mathcal{S}_{0+}^k)$  are also subdifferentially compatible. Finally, applying Corollary 4.3(c) and Corollary 8 of Higle and Sen (1992) to approximations  $\{f_0^k\}_k$  at node 0 we conclude that  $\bar{u}_0 \in U_0$  is optimal (wp1) ■

#### 4.6 Remarks on the asymptotic analysis

A review of the convergence analysis reveals that our approach can be easily applied to models for which the nodal problems are convex programs whose objective and constraints are separable by states and decisions, and the stochastic dynamics are linear. For the case of stochastic non-linear dynamics however, one may not achieve global optimality.

With a few exceptions (e.g. Powell 2007), approximations in ADP do not rely on convexity. The asymptotic analysis presented above highlights some of the main differences between SP and

ADP: the former uses convex approximations which satisfy certain properties (e.g. subdifferential compatibility) to ensure asymptotic convergence. This is achieved through dual problems, as in the nodal dual approximations used in MSD. ■

## 5 Conclusions

The main goal for this paper was to present a unified framework for both two-stage as well as multi-stage stochastic decomposition algorithms. In a sense, SD is similar to Stochastic Dual Dynamic Programming (SDDP, Pereira and Pinto 1991, Infanger and Morton 1996, Donahue and Birge 2006), although the differences are significant: a) SD uses sample means for recursive estimates of subgradients, where as SDDP uses the probability given via the scenario tree, b) the forward simulation solves regularized (quadratic) approximations leading to unique nodal decisions, c) generating a cut requires one LP per non-terminal node of a sample path generated during forward simulation, d) SD allows subgradient generation to use special structure (e.g. network structure), e) convergence (with probability one) does not require stagewise independence of the stochastic process, and f) unlike most SP algorithms which require that the probability distribution be specified a priori, the SD approach allows scenarios to be generated via simulations, and so long as the outputs of the simulated process are discrete (e.g. finite state stochastic models), one can directly use the sample paths within the SD framework. In our opinion, these differences, especially (c-f) are particularly important for large scale applications. Items e) and f) are quite far reaching because they allow for the possibility of including more general simulators than ordinary Monte Carlo simulation as in this paper. In contrast to “Simulation Optimization” which injects optimization into a simulation model, our new paradigm allows us to inject simulators into an optimization model, leading to a new approach for stochastic programming (“Optimization Simulation” in Sen and Zhou 2011).

We have also explored common ground between SP and DP, illustrating how SP treats its state variables in a manner that tends to reduce the curse of dimensionality by distinguishing endogenous state variables, with exogenous state variables  $\underline{\omega}$ . As a result of this framework, we have opened the door of SP methods to dynamic systems optimization, including Approximate DP, Differential DP, and Model Predictive Control. While these variants rely on differentiability, SD and more generally SP, handles non-differentiable objective functions.

**Acknowledgements.** The unifying nature of this framework should be apparent, and we offer it as our tribute to the originators of Stochastic Programming (George B. Dantzig) and Dynamic Programming (Richard E. Bellman). We are grateful to Darinka Dentcheva and two anonymous referees who requested that we clarify the connections to DP and ADP. We also thank Warren Powell for several discussions regarding connections between SD and ADP. This research was supported in part by AFOSR Grant: FA9950-08-1-0154, and CMMI 0900070 from the National Science Foundation. We are grateful for continued support of this line of research.

## 6 References

- [1] Benders, J.F. 1962, "Partitioning procedures for solving mixed integer variables programming problems," *Numerische Methemattick*, vol. 4, pp. 238-252.
- [2] Birge, J. R. 1985, "Decomposition and partitioning methods for multistage stochastic linear programs," *Operations Research*, 33:989–1007.
- [3] Birge, J.R. and F. Louveaux 1997, *Introduction to Stochastic Programming*, Springer-Verlag.
- [4] Casey, M. and S. Sen 2005, "The Scenario Generation Algorithm for Multi-stage Stochastic Linear Programming," *Mathematics of Operations Research*, 30:615-631.
- [5] Carino, D. R., D. H. Myers and W. T. Ziemba 1998, "Concepts, technical issues and uses of the Russell-Yasuda-Kasai financial planning model," *Operations Research*, 46(4):450–462.
- [6] Donohue, C.J. and J. R. Birge 2006 "The abridged nested decomposition method for multistage stochastic linear programs with relatively complete recourse," *Algorithmic Operations Research*, 1:20-30.
- [7] Dupačová, J. N. Gröwe-Kuska and W. Römisch 2003, "Scenario reduction in stochastic programming: An approach using probability metrics," *Mathematical Programming*, 95:493–511, 2003.
- [8] Dyer, M, and L. Stougie 2006, "Computational Complexity of Stochastic Programming Problems," *Mathematical Programming*, 106:423-432.
- [9] Gassmann, H. 1990, "Mslip: a computer code for the multistage stochastic linear programming problem," *Mathematical Programming*, 47:407–423.
- [10] Guddat, J. 1976, "Stability in convex quadratic parametric programming." *Statistics*, v. 7 issue 2, pp. 223-224.
- [11] Higle, J.L. and S. Sen 1991, "Stochastic Decomposition: An algorithm for two stage linear programs with recourse," *Mathematics of Operations Research*, vol. 16:650-669.
- [12] Higle, J.L. and S. Sen 1992, "On the convergence of algorithms with implications for stochastic and non-differentiable optimization," *Mathematics of Operations Research*, 17:112-131.
- [13] Higle, J.L. and S. Sen 1994, "Finite master programs in stochastic decomposition," *Mathematical Programming*, 67:143-168.
- [14] Higle, J. L. and S. Sen 1996, *Stochastic Decomposition: A Statistical Method for Large Scale Stochastic Linear Programming*, volume 8 of *Nonconvex Optimization and Its Applications*. Kluwer Academic Publisher.
- [15] Higle, J.L. and S. Sen 2006, "Duality for Multistage Convex Stochastic Programs," *Annals of Operations Research*, 142, pp. 129-146.
- [16] Infanger, G. and D.P. Morton 1996, "Cut sharing for multistage stochastic linear programs with interstage dependency," *Mathematical Programming* 75: 241-256.
- [17] King, A.J. and R. J-B. Wets 1991, "Epi-consistency of convex stochastic programs," *Stochastics and Stochastics Reports*, 34:83-92.

- [18] Linowsky, K. and A. B. Philpott 2005, “On the convergence of sampling-based decomposition algorithms for multistage stochastic programs,” *Journal of Optimization Theory and Applications*, 125: 349-366.
- [19] Mulvey, J. M. and A. Ruszczyński 1995, “A new scenario decomposition method for large scale stochastic optimization,” *Operations Research*, 43:477–490.
- [20] Nowak M. P. and W. Römisch 2000, “Stochastic lagrangian relaxation applied to power scheduling in a hydrothermal system under uncertainty,” *Annals of Operations Research*, 100:251–272.
- [21] Pereira, M.V. and L. M. Pinto 1991, “Multi-stage stochastic optimization applied to energy planning,” *Mathematical Programming*, 52:359-375.
- [22] Peters, R. J. , K. Boskma and H. A. E. Kuper 1977, “Stochastic programming in production planning: a case with nonsimple recourse,” *Statistica Neerlandica*, 31:113–126.
- [23] Philpott, A.B. and Z. Guan 2008, “On the convergence of stochastic dual dynamic programming and related methods,” *Operations Research Letters*, 36: 450-455.
- [24] Powell, W. B. 2007, *Approximate Dynamic Programming: Solving the curses of dimensionality*, Wiley Series in Probability and Statistics.
- [25] Ruszczyński, A. 1986, “A regularized decomposition method for minimizing a sum of polyhedral functions,” *Mathematical Programming*, 35:309- 333.
- [26] Rockafellar, R.T. and R. J-B. Wets 1991, “Scenarios and policy aggregation in optimization under uncertainty,” *Mathematics of Operations Research*, 16(1):119–147.
- [27] Rockafellar, R.T. and R.J-B. Wets 1997, *Variational Analysis*, Springer-Verlag, Berlin, Germany.
- [28] Rubin, H. 1956, “Uniform convergence of random functions with applications to statistics,” *The Annals of Mathematical Statistics*, 27(1): 200-203.
- [29] Rudin, W. 1976, *Principles of Mathematical Analysis*, McGraw-Hill, NY.
- [30] Sen, S. and Z. Zhou 2011, “Optimization Simulation: The case of multi-stage decision models,” Proceedings of the Winter Simulation Conference, S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu ( eds.)
- [31] Shapiro, A. 2010, “Analysis of Stochastic Dual Dynamic Programming Method,” *Optimization Online*.
- [32] Tzur M., Y.T. Herer, and E. Yucesan 2006, “The multilocation transshipment problem,” *IIE Transactions*, 38(3):185–200.
- [33] Van Slyke,R. and R. J-B. Wets 1969, “L-shaped linear programs with applications to optimal control and stochastic programming,” *SIAM Journal on Applied Mathematics*, 17:638-663.
- [34] Wu, J. and S. Sen 2000, “A stochastic programming model for currency option hedging,” *Annals of Operations Research*, 100:227–250.