Mitigating Uncertainty via Compromise Decisions in Two-stage Stochastic Linear Programming

Suvrajeet Sen, Yifan Liu
Daniel J. Epstein Department of Industrial and Systems Engineering
University of Southern California, Los Angeles, 90089
s.sen@usc.edu, yifanl@usc.edu

Stochastic Programming (SP) has long been considered as a well-justified yet computationally challenging paradigm for practical applications. Computational studies in the literature often involve approximating a large number of scenarios by using a small number of scenarios to be processed via deterministic solvers, or running Sample Average Approximation on some genre of high performance machines so that statistically acceptable bounds can be obtained. In this paper we show that for a class of stochastic linear programming problems, an alternative approach known as Stochastic Decomposition (SD) can provide solutions of similar quality, in far less computational time using ordinary desktop or laptop machines of today. In addition to these compelling computational results, we also provide a stronger convergence result for SD, and introduce a new solution concept which we refer to as the compromise decision. This new concept is attractive for algorithms which call for multiple replications in sampling-based convex optimization algorithms. For such replicated optimization, we show that the difference between an average solution and a compromise decision provides a natural stopping rule. We discuss three stopping criteria which enhance the reliability of the compromise decision, reducing both bias and variance associated with the result. Finally our computational results cover a variety of instances from the literature, including a detailed study of SSN, a network planning instance which is known to be more challenging than other test instances in the literature.

Key words: Stochastic Linear Programming, Stochastic Decomposition, Computational Experiments

1. Introduction

For certain stochastic linear programming (SLP) models, the associated probability space can be so large that identifying a deterministically verifiable optimum is impossible (in reasonable time) using any foreseeable computer. Nevertheless, such models arise in a variety of applications, and new notions of approximate (or near)- optimality, supported by statistically verifiable bounds, are important for decision support. While statistical optimality bounds have been studied in the literature for a while (e.g., Higle and Sen (1991b), Higle and Sen (1996a), Mak et al. (1999), Kleywegt et al. (2002), Bayraksan and Morton (2011), Glynn and Infanger (2013)), their use in identifying near-optimal *decisions* for realistic instances has been limited. It is important to note the emphasis on "decisions" as opposed to identifying bounds within which the optimal value might belong. Prior attempts to use statistical optimality bounds have either required some genre of high performance computing or they have been limited to relatively small instances (few decision variables/constraints and a small number of random variables or scenarios). The goal of this paper is to demonstrate that for certain classes of two-stage SLP models, we are able to provide statistically verifiable, near-optimal decisions even if uncertainty is modeled using continuous random variables. Building on previous work connected to Stochastic Decomposition (SD), we introduce the notion of a "compromise" decision, which allows us to not only confirm statistical bounds, but also recommend decisions with significant confidence. We report computational results for a suite of SLP test problems from the literature, and show that statistically verifiable decisions can be obtained within a few minutes of computing on desktops, laptops, and similar "run-of-the-mill" computing devices.

The class of SLP models discussed below fall under a category known as fixed-and-relatively-complete recourse models, and may be stated as follows.

$$\text{Min} \quad f(x) = c^\top x \ + \ E[h(x, \tilde{\omega})] \tag{1a}$$

$$\text{s.t. } x \in X = \{Ax \ \leq \ b\} \subseteq \Re^{n_1} \tag{1b}$$

where, $\tilde{\omega}$ denotes an appropriately defined (vector) random variable, and the function h is defined as follows.

$$h(x, \omega) = \text{Min} \quad d^\top y \tag{2a}$$

$$\text{s.t.} \quad Dy = \xi(\omega) - C(\omega)x \tag{2b}$$

$$y \geq 0, y \in \Re^{n_2}. \tag{2c}$$

The notation ω denotes any outcome of the random variable $\tilde{\omega}$. The fixed-and-relatively-complete recourse assumption of the above model implies that the matrix D is deterministic, and the function h has finite value for all solutions x satisfying $Ax \leq b$. Previous work on SD has restricted the second stage cost vector (d) to be fixed, and we continue with this setup. However current ongoing research has established that random elements in the cost vector (d) can also be accommodated within SD and will be reported in a subsequent paper.

As is common in decision-making under uncertainty, it is necessary to make decisions (x) in the first stage before an outcome ω is observed, and subsequently, the second stage decisions (y) are made. The quantity $\xi(\omega) - C(\omega)x$ often denotes the "position of resources" after the decision x has been made, and the outcome ω has been observed. In the terminology of Approximate Dynamic Programming (ADP, Powell (2007)), this vector may be looked upon as the "post-decision state" of a two-stage model.

In order to motivate this paper, we will begin by presenting some computational results for a well known instance called SSN (Sonet Switched Network, Sen et al. (1994)). This model has been used for a variety of purposes in the past; it was one of the early SLP models validated using discrete-event simulation; others have used it to illustrate whether certain algorithms scale well (Linderoth et al. (2006) and Nemirovski et al. (2009)), and still others have used it to illustrate what makes certain SLP models difficult for sampling-based algorithms (Shapiro et al. (2002)). More recently the Defense Science Board Report (Grasso and LaPlante (2011)) recommends SSN and models of this genre for DoD trade-off studies in which a very large number of contingency scenarios are necessary as part of the analysis to accompany recommendations for investment in new technologies. For these reasons, and because of its roots as an industrial-sized instance (SSN grew out of a 1990's Metropolitan network in Atlanta, GA), we use the performance on SSN to illustrate that decision-makers need not shy away from some classes of SLP models; certain current algorithms are up to the task of providing statistical optimality bounds and decisions within reasonable time using ordinary computing machinery. This evidence is provided in section 2.

Following a discussion of SSN, section 3 presents the methodology, which is based on Stochastic Decomposition (SD, Higle and Sen (1991a), Higle and Sen (1994), Higle and Sen (1996b)). While SD has close ties to some Simulation Optimization approaches (e.g., Shapiro and Homem de Mello (1998), and approaches described in Kim et al. (2015)), these methods are more general in scope because they allow "blackbox" functions, whereas, SD requires two-stage SLP. This focus of SD facilitates computations with large scale problems arising in practical applications. In addition to the ability to scale up using linear programming, focusing on SLP models also allows us to show that the algorithm produces a sequence which converges to a unique limit (with probability one). Moreover, as outputs of the algorithm, our procedure provides decision-makers two alternative choices which either reinforce each other, or provide indicators of "indecision". One of the alternative decisions will be the "compromise" decision, and the other will be an "average" solution. We will show that when these decisions are similar (i.e., they reinforce), they are both very close to optimum. On the other hand, if these decisions are not similar, then we suggest greater precision in calculating solutions for each replication, and these should be undertaken by using the "warm starting" capability that is naturally available via SD. When there is indecision between the average and the compromise, we will refer to both as average and compromise solutions. When they agree with each other (up to some given tolerance), then we refer to them as average and compromise decisions. In this sense it is different from the Polyak and Juditsky (1992) averaging process commonly used in stochastic approximation.

Following our presentation of the methodology, we present further computational results in section 4. The test instances, which are available in the literature cover a variety of applications ranging from inventory control and supply chain planning to power generation planning, freight transportation, and others. All of the test instances reported here require an algorithmic treatment of multi-dimensional random vectors, and hence instances with the simple recourse property (e.g., news-vendor models) are not included in this study. This paper addresses several questions of relevance to the SP community.

1. Given that SP problems can be demanding, greater accuracy may call for re-runs that should re-use previously discovered structures of an instance. How can such structures be re-used for the purposes of warm-starting?
2. Given that SLP models have a very special (convex) structure, how should sampling-based methods be designed to take advantage of such structure?
3. Sampling-based SP methods borrow variance reduction techniques from the simulation literature. Are there other variance reduction techniques that are appropriate for SP, but are not considered in the simulation literature?

4. Parallel architectures in SP have traditionally been used to process bunches of scenarios. Are there other ways to use parallel architectures which permit the solution of industrial-strength models?
5. Should SP algorithms report lower bound estimates for the "true" problem so that the quality of a recommended decision can be ascertained?

We will present the conclusions of this paper by placing our contributions in the context of these questions. For now, we begin by applying sampling-based algorithms to SSN.

2. Motivation: Computations with a Practical SLP

Formally speaking, SSN is a two-stage stochastic linear programming model. The basic "operations"-issue in the SSN model is to recommend link sizes of a given network so that the network will experience the least number of "lost calls" (in expectation), while operating under a given budget constraint. We refer the reader to Appendix B for its mathematical formulation. In the SP literature, such models are often classified as "here-and-now" models because the link capacities must be decided before actual demands are known. Models of this type, which are based on introducing randomness to linear programming models, must contend with multi-dimensional random vectors, which, in the SSN model represent point-to-point demand uncertainty. In this particular example, there are 86 point-to-point pairs, which, by standards of LP models, is modest. As is common today, these demands are available through forecasting systems, and *errors* in forecasts may be treated as independent random variables. For the model presented in Sen et al. (1994), each marginal error random variable was deemed to be sufficiently approximated by a discretization using about 5-9 outcomes per demand pair. It is not difficult to see that the total number of scenarios involves an astronomical number of parametric LPs (approximately of magnitude 10^{71}). Even if one had access to an exascale (10^{18} flops) computing platform, it would be pointless trying to seek a solution whose optimality could be verified in a deterministic sense. It is therefore pragmatic to seek approximate solutions which are near-optimum in a statistical sense. Other approximations to SLP have been suggested via linear decision rules (Kuhn et al. (2011), Chen et al. (2008)). However, these approaches are motivated by scalability, rather than statistical bounds of optimality.

The remainder of this section uses the SSN model to illustrate the level of computing resources that may be necessary to provide decision support using sampling approaches for two-stage SLP models. The two approaches presented below are Sample Average Approximation (SAA) and the Stochastic Decomposition (SD) algorithm.

2.1. Sample Average Approximation

Conceptually, the SAA approach consists of replacing the "Expectation" in the objective function (1a), by a collection of instances which optimize a sample average approximation defined by a relatively small number of outcomes, say N. Then the function $F_N(x)$ below is known as the sample average approximation, and the following optimization problem is an SAA *instance.*

$$\underset{x \in X}{\mathrm{Min}}\ F_N(x) \equiv c^\top x + \frac{1}{N} \sum_{i=1}^{N} h(x, \omega^i) \tag{3}$$

For the SSN model, one has $c = 0$, and in this sense the statement of (3) is agnostic to this fact.

In recent years, SAA has become the de facto standard for sampling-based SP methodology. This methodology draws inspiration from statistical inference (see Shapiro et al. (2009)), and is supported by stability results (Römisch (2003)) which justify approximations of SLP, and calculations of general confidence bounds (Vogel (2008)). The Statistical Validation procedure for SAA is usually based on Mak et al. (1999). As an overview, the SAA process may be summarized as follows.

1. Choose a sample size N, and choose a number M denoting the number of replications (batches).
2. (Optimization Step). For $m = 1, \dots, M$ create an approximation $F_N^m(x)$, and solve an SAA instance (3). Let $\hat{F}_N^m$ denote the optimal value of replication m.
3. (Statistical Validation Step). Using $\{\hat{F}_N^m\}_{m=1}^M$ estimate a Lower Bound confidence interval. Using M solutions (i.e., potential decisions) from the Optimization Step, estimate the best Upper Bound confidence interval to a specified level of accuracy.
4. If the lower end of the estimated Lower Bound confidence interval is acceptably close to the upper end estimated Upper Bound confidence interval, then stop. Else, increase the sample size N and repeat from step 2.

Because the sampling step is independent of the optimization step, SAA is sometimes referred to as an "external" sampling approach. This separation between the optimization algorithm and sampling leads to a fairly general setup. Some presentations in the literature refer to the "Optimization Step" as the "Training Step", and the "Statistical Validation Step" as simply the "Validation Step" (see Hastie et al. (2009)). While the notion of replications is often not emphasized in some segments of the literature, decision-makers who have experience with sample-based algorithms (e.g., simulation) seek variance estimates of any metric reported by an algorithm, which in this case consists of lower bound and upper bound estimates.

The calculation in step 4 reflects the worst case optimality gap, which we refer to as the Pessimistic-Gap. We should also note that variance reduction techniques, such as Latin Hypercube Sampling (LHS), have been found to reduce variance of SAA estimates (Linderoth et al. (2006)). When M is in the neighborhood of 30, one typically invokes the central limit theorem for estimating a Lower Bound confidence interval; however, when M is much smaller (say 10), then, confidence intervals should be derived using the t-distribution.

In order to complete the numerical illustration of SAA for SSN, we now draw upon results from Linderoth et al. (2006). The Optimization Step for their study was performed using the Asynchronous Trust Region algorithm of Linderoth and Wright (2003).

From Mak et al. (1999) it is well known that the optimality gap estimates reduce with increases in sample size as shown in Table 1, and with a significant increase in sample size (from 1000 to 5000), we see a significant improvement in the "pessimistic gap" in Table 1. Because the bounds in Table 1 provide 95% confidence intervals, the chance that the actual gap exceeds the pessimistic gap is less than 10%. The last row corresponding to a sample size of 5000 yields a pessimistic gap of about 2% of the upper bound, with very high probability.

Table 1 SAA with Latin Hypercube Sampling

Sample Size (N)	Lower Bound	Upper Bound	Pessimistic-Gap
50	10.10(±0.81)	11.380(±0.023)	2.113
100	8.90(±0.36)	10.542(±0.021)	2.023
500	9.87(±0.22)	10.069(±0.026)	0.445
1000	9.83(±0.29)	9.996(±0.025)	0.445
5000	9.84(±0.10)	9.913(±0.022)	0.195

The results of Linderoth et al. (2006) were obtained using a computational grid with hundreds of desktop PCs, although no more than one hundred machines were in operation at any one time. Even so, each SAA instance of the final row (with $N = 5000$) required about 30-45 minutes of wall clock time for solving one SAA instance of SSN. As it turned out, the solutions provided by the replications (about 6) were quite disparate even though these experiments were done using Latin-Hypercube Sampling. The use of a computational grid to establish the above results was a remarkable feat, solving millions of LPs on geographically dispersed and architecturally diverse machines. However, it also underscores the challenge of using SAA for real-world applications; if one resorts to sampling/simulation without exploiting the structure of the optimization problem, then the computing resources required to solve these instances can easily out-strip the available resources, thus restricting the potential of the SLP modeling paradigm.

2.2. Stochastic Decomposition (SD)

In keeping with the "high-level" description of SAA above, we provide a "high-level" description of SD. As with SAA, one may choose the number of replications (M), but instead of choosing a sample size, we allow the SD algorithm to determine what is a sufficiently large sample size during the Optimization Step. Unlike SAA, SD does not optimize one sample average function F_N^m; instead it creates successive approximations f_k^m at iteration k. These approximations satisfy

$$f_k^m(x) \leq F_k^m(x), \tag{4}$$

where F_k^m denotes a sample average function with a sample size of k (as in (3)). These minorants are referred to as Value Functions (VF) and each VF is the pointwise maximum of finitely many affine functions. Of these affine functions, two are lower bounds on subgradients of F_k^m denoted $h_{0,k}$ and $h_{\nu,k}$ while the rest are updated versions of previously generated lower bounding approximations of $F_t^m, t < k$, and are denoted $h_{t,k}$. As k grows, the approximations $h_{0,k}$ and $h_{\nu,k}$ become more accurate approximations of subgradients of F_k^m, and at such iterations they can be regarded as sample average subgradient approximations (SASA). In summary, each f_k^m is the maximum of $n_1 + 3$ affine pieces, with two SASA functions from the current iteration k and at most $n_1 + 1$ from previous iterations. We denote the index set for all of these affine functions as J_k with $|J_k| \leq n_1 + 3$.

Because of sampling, there are errors, and if these errors cause the sampled approximation to "cut off" portions of the epigraph of f, then they can thwart asymptotic convergence of the algorithm. Instead of letting the errors persist in future iterations (i), we actively modify these functions so that they satisfy $h_{t,i}^m(x) \leq F_i^m$, where $i \geq k$. For this reason, we prefer not to refer to each piece $h_{t,k}$ as a cut (which is the traditional terminology), and refer to them as minorants.

For iteration $k+1$, the next sample (of size $k+1$) will use all k previously generated outcomes and add one more (generated independently of previous outcomes) to arrive at a sample size of $k+1$. The earliest version of SD (Higle and Sen (1991a)) optimized the VF approximation of iteration k, that is, $\text{Min}\{f_{k-1}(x) \mid x \in X\}$ to obtain the next candidate solution $x^{m,k}$. More recent versions, including this paper, are based on Higle and Sen (1994) where optimization of a regularized version produces the next candidate solution. This form of the objective function is dual to the objective function commonly used in bundle methods.

A subset of the sequence of candidate solutions, denoted $\hat{x}^{m,k}$, will be referred to as "incumbents" (or incumbent solutions). In these earlier papers, it has been shown that if $k \to \infty$ then, we have asymptotic consistency of the values i.e., if $\hat{x}^{m,k} \to \mathbf{x}^m$, then $\lim_{k\to\infty} f_k^m(\hat{x}^{m,k}) = \lim_{k\to\infty} F_k^m(\hat{x}^{m,k}) = c^\top \mathbf{x}^m + E[h(\mathbf{x}^m, \tilde{\omega})]$ (wp1). A few more details regarding the algorithm are provided in the following section (see also Birge and Louveaux (1997)).

While results like consistency are based on long-run behavior of an algorithm, one stops each replication after a finite number of steps, which will be based on detecting that the approximations obtained for the current replication have stabilized sufficiently, based on a given tolerance level. This test is known as an In-Sample stopping rule (Higle and Sen (1999)), and signals whether a particular replication has enough information to propose a solution which we denote by $\mathbf{x}^m$. If $m < M$ (the desired number of replications), then, we proceed to the next replication; otherwise SD recommends a *"compromise decision"* $\mathbf{x}^c$ which presents a compromise between all replicated decisions $\mathbf{x}^m$. Using $\mathbf{x}^c$ as the proposed decision, we calculate a 95% confidence interval for the upper bound $f(\mathbf{x}^c)$. In addition, a 95% confidence interval for a lower bound estimate on the optimal value is also reported. As with SAA computations in the previous subsection, we will report the pessimistic gap. A high-level description of SD is provided next.

1. (Initialize). Let m denote a counter of completed replications, and set $m = 0$.
2. (Out-of-Sample loop). Increment m, and initialize (or continue) the next replication.
3. (In-Sample loop). Update the available sample by adding one sampled outcome (independent of previously generated outcomes), and update empirical frequencies.
4. (Update Value Function (VF) Approximation). Using previously generated approximations and the new outcome, update the VF approximation $f_k^m(x)$.

5. (Optimization Step). Optimize a regularization of the VF approximation, and update an incumbent decision for the first stage.

6. (In-Sample Stopping Rule). If an In-Sample stopping rule is satisfied, then output a lower bound estimate $\hat{\ell}^m := f^m(\mathbf{x}^m)$, an incumbent $\mathbf{x}^m$, and continue to step 7. Else repeat from step 3.

7. (Out-of-Sample Stopping Rule). If the number of replications is less than M, then go to step 2. Else, using the replicated solutions $\{\mathbf{x}^m\}$ calculate a compromise decision (denoted $\mathbf{x}^c$), and with this solution estimate a 95% Upper Bound confidence interval of specified accuracy. Using $\{\hat{\ell}^m\}_{m=1}^M$, the lower bound estimates, calculate a 95% Lower Bound confidence interval. If the Pessimistic-Gap is acceptably small then stop. Else, decrease the tolerance of the In-Sample Stopping Rule, reset $m = 0$, and resume all M replications from step 2.

It is important to observe that this sequential sampling procedure automatically includes variance reduction from Common Random Numbers (CRN). To see this, we mention that in step 5 above, the candidate replaces the incumbent as the new incumbent if the estimated value of the candidate solution is lower than that of the current incumbent using the same collection of outcomes $\{\omega^t\}_{t=1}^k$. This comparison is essentially based on using CRN (see Fu (2002)). In addition, the SASA functions generated in any iteration (one at the incumbent and another at the candidate) use the same stream of outcomes. Hence these two affine functions view all $x \in X$ through the same lens (of outcomes).

This procedure was executed for the SSN instance using three increasingly tighter relative tolerances: loose (0.01), nominal (0.001), and tight (0.0001), and the results for each run appear in Table 2.

Table 2 Stochastic Decomposition with Common Random Numbers (on a desktop PC with CPLEX12.4)

Stopping Tolerance	Avg. Sample Size (Std Dev)	Lower Bound	Upper Bound	Pessimistic-Gap	CPU Time (s) (Std Dev)
Loose	1030.83(182.31)	9.345(±0.240)	9.951(±0.05)	0.896	30.11(6.63)
Nominal	2286.90(341.71)	9.736(±0.118)	9.927(±0.05)	0.359	90.50(20.56)
Tight	3305.47(617.17)	9.852(±0.107)	9.923(±0.05)	0.228	162.01(55.43)

Upon examining Table 2, we first observe that the average sample size (per replication) increases with increased precision, as expected. The solution quality (as seen in the upper bound) does not improve dramatically from the first row to the third. However, the improvements in lower bounds are significant, ultimately, mitigating the uncertainty about the quality of the solution. In this sense, it reinforces a common observation in difficult optimization models: proving optimality is what requires extensive computations for difficult instances. Nevertheless, the average CPU secs. for even the row with "tight" tolerance is under 3 minutes on a desktop PC with the following specifications: Intel Core i7-3770S CPU@3.10GHz (Quad-Core), and 8 GB Memory @1600MHz. The processor used for the calculation reported in Table 2 is faster than the average processor of 2004/2005 vintage: Pentium IV processors of that era were approximately 2 GHz.

Since these processors are faster than the ones used in the computational grid study (Linderoth et al. (2006)), we also examine the performance on a slower machine. We execute the same SD code on a Mac Book Air running Intel Core i5 CPU@1.8GHz (Dual-Core) with 4 GB Memory @1600 MHz. Such (laptop) processors of today could be considered on par with (or slightly slower than) the standard Pentium IV processors of 2004/2005 vintage. The average solution times and solution quality for such a processor is reported in Table 3. Notice that the solution quality is very similar to that reported in Tables 1 and 2. Indeed, the average lower bound as well as the pessimistic lower bound (in Table 3) are the best reported to date for SSN. Moreover, the increase in CPU time (secs.) is marginal with "tight" tolerance, requiring just over 3 mins. of CPU time on average. In contrast, the grid study (Linderoth et al. (2006)) reported wall-clock time of the order of 30-45 minutes per replication using about 100 processors at any given time (for instances with $N = 5000$).

We recognize that it is impossible to provide a precise characterization of the level of speed-up for several reasons: a) in addition to processor speeds, LP software has also made significant progress since 2004/2005, and b) there is communications overhead involved with using the grid. Nevertheless, it is worth noting, as in a recent PCAST report (Holdren et al. (2010)), that software advances are just as meaningful for challenging numerical problems as improvements in processing power. As a rough estimate of such advances, let η represent the fraction of time that processors are either idle or communicating during the shortest run (30 mins. of wall-clock time) on the grid. Then the speed-up factor for software (LP-solvers and SD) using a laptop could be approximated as $\frac{30}{3} \times 100(1-\eta)$. To see how this might compare with the rate of Moore's Law, first note that the speed of processors used for Table 3 is about the same or slower than standard processors of 2004/2005 vintage. Hence the speed-up can be attributed entirely to software progress, which implies that as long as $\eta \leq 0.872$ (i.e., idling/communications are less than 87.2%) speed-ups in LP/SLP software outpace Moore's Law which calls for a factor of at least 128 in 10.5 years.

Table 3 **Stochastic Decomposition with Common Random Numbers (on a MacBook Air with CPLEX12.4)**

Stopping Tolerance	Avg. Sample Size (Std Dev)	Lower Bound	Upper Bound	Pessimistic-Gap	CPU Time (s) (Std Dev)
Loose	1023.33(167.62)	9.366(±0.244)	9.953(±0.050)	0.881	32.73(6.97)
Nominal	2353.43(343.33)	9.764(±0.120)	9.928(±0.050)	0.334	109.96(26.31)
Tight	3137.50(605.17)	9.876(±0.107)	9.925(±0.050)	0.206	189.79(74.57)

Let us now return to a closer examination of Table 2. The stepped curve in Figure 1 shows the spread of objective function estimates obtained for each tolerance level reported in Table 2. Despite the fact that SD is an asymptotically convergent algorithm, the objective function estimates (for each terminal incumbent) show variability due to the fact that each run is terminated upon achieving some level of accuracy in finitely many iterations. For each tolerance setting (Loose, Nominal and Tight), the objective function estimates are in the range [10.100, 10.460], [9.994, 10.330] and [9.950, 10.310] respectively, and note that both upper and lower limits of these ranges move in the appropriate direction (lower). Moreover, from Table 2 we notice that lower bounds increase steadily, starting with 9.345 (for loose tolerance) rising to 9.736 (for nominal tolerance), and finally 9.852 (for tight tolerance). Thus increasing precision in SD leads to less biased estimates of lower bounds. Moreover, comparing these lower bounds to the last two rows ($N = 1000, 5000$) in Table 1, we observe that lower end of confidence intervals for lower bounds in Table 1 are similar to those reported in Tables 2 and 3.

Figure 1 also shows three lines on the far left of the figure. These correspond to the upper bound (average value) reported in Table 2. They correspond to objective estimates for compromise decisions at each tolerance setting. As shown in Figure 1 the compromise decisions for each setting yields a lower objective function estimate than the incumbent solutions for the corresponding run. The compromise decisions are not only superior, but they also possess another important property: when the compromise decision and the average decision are reasonably close, we can also conclude that both decisions are reasonably

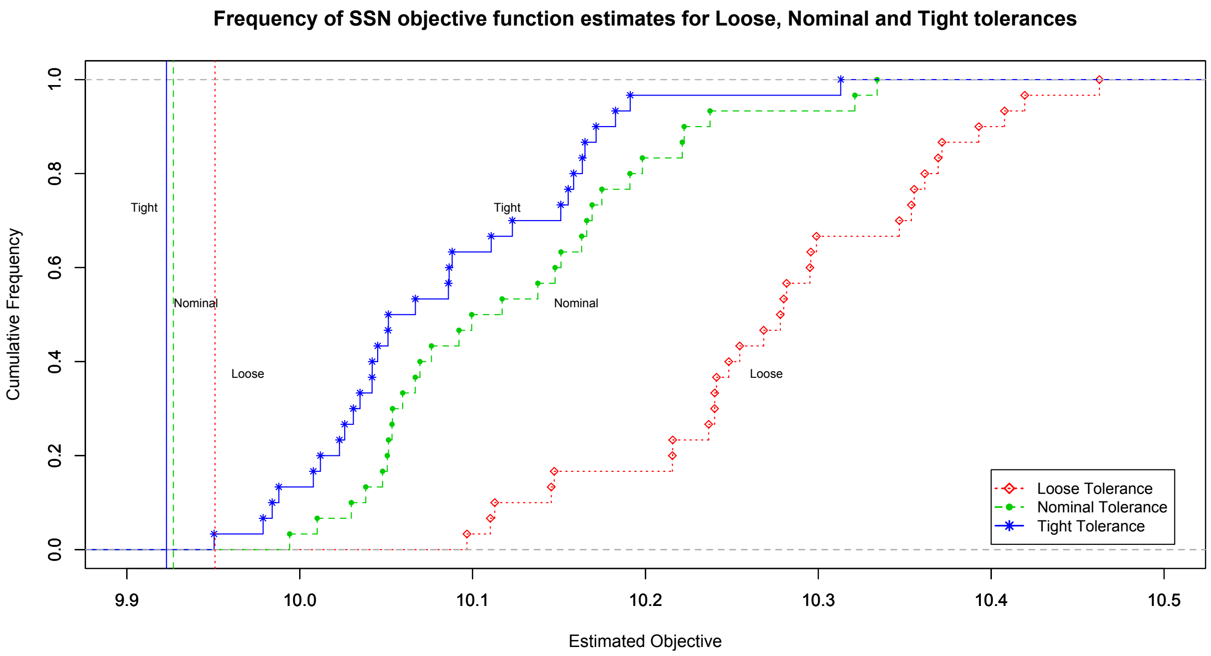


Figure 1 **Cumulative frequency of SSN objective function estimates for Loose, Nominal and Tight tolerances**

good. We will establish this result in the following section (Theorem 2). For now, examining the specific instance at hand (i.e., SSN), we present the maximum relative difference between the compromise and average decisions, that is $\{2|x_j^c - \bar{x}_j|/|x_j^c + \bar{x}_j|\}, \forall j$ in Figure 2. For variables that are almost zero we report the absolute difference $\{|x_j^c - \bar{x}_j|\}, \forall j$ instead of the relative difference. In these figures, the horizontal axis displays the index of first stage decision variables, and the vertical axis represents the difference between the compromise and average decisions for each first stage variable. Observe that for each tolerance setting, the maximum relative differences are on the order of 10^{-4} to 10^{-5}. Since the relative difference shown for all tolerance levels is pretty minimal, we can infer high quality decisions from compromise solutions for each tolerance level, even though the Lower Bound confidence intervals are weaker for the loose and nominal tolerances. Thus even for SSN, an instance considered to be ill-conditioned, the nominal setting provides reasonable accuracy. Such decision support is intended to mitigate the effects of uncertainty, without requiring extensive computational resources.

3. Algorithmic Concepts in Stochastic Decomposition

This section provides the algorithmic background for the computational results presented for SSN, as well as the more extensive computations presented in the next section. The

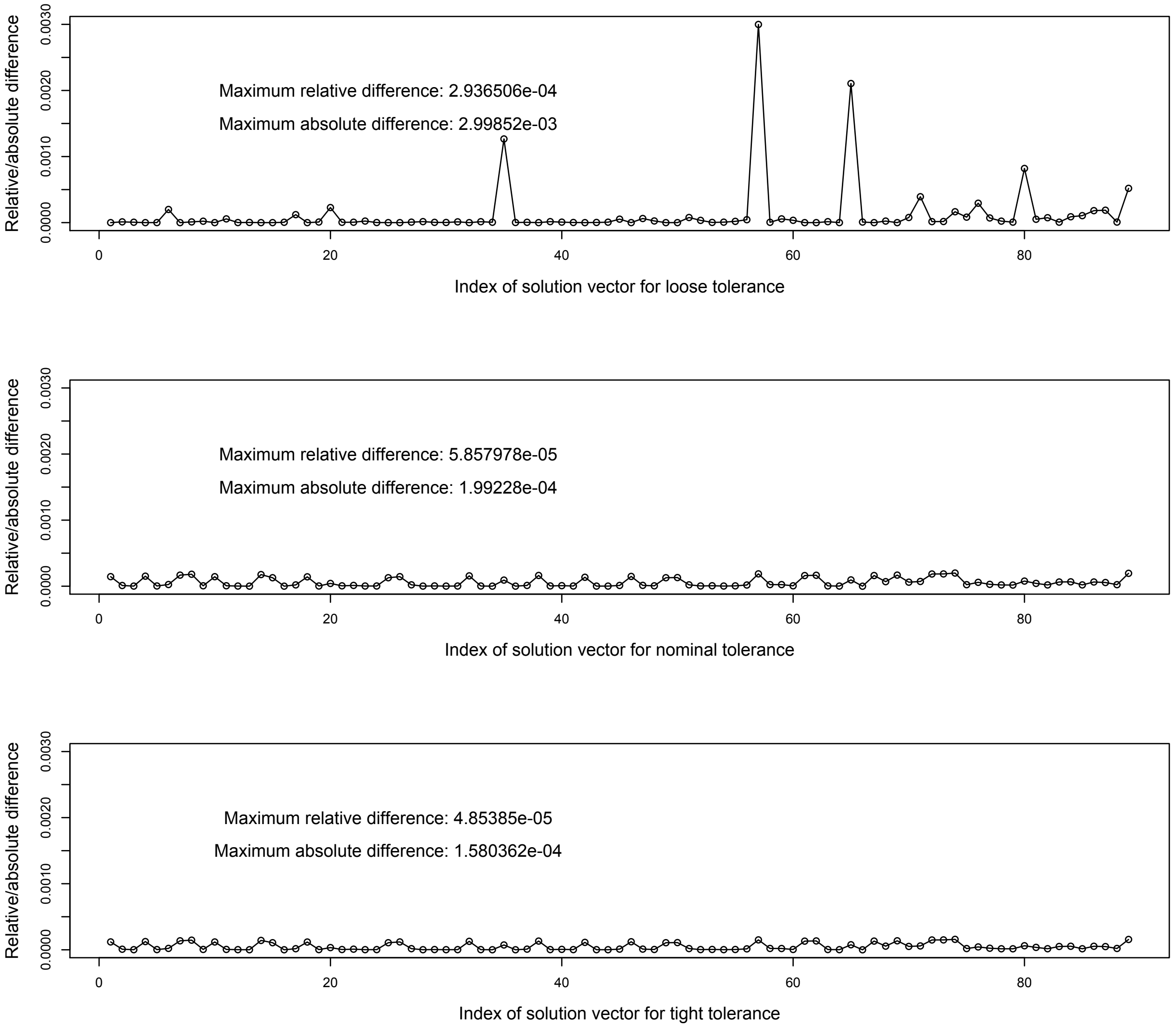


Figure 2 **Relative/absolute difference between compromise and average decisions calculated for Loose, Nominal and Tight tolerances**

algorithmic content will be presented in two subsections: one dealing with algorithm design and convergence, and another on stopping rules. The latter will be divided into two further subsections dealing with In-Sample and Out-of-Sample aspects of stopping within the SD framework. Before getting into the details, we mention some of the critical *assumptions for SD.*

Assumptions. The current SD algorithm operates under the following assumptions:

1. The set of first stage solutions, X, and the set of outcomes Ω are compact.
2. The relatively complete recourse assumption holds.

3. The matrix D is deterministic (i.e., fixed recourse) and the vector d is also deterministic.

4. The recourse function h is non-negative. This assumption can be relaxed in several ways (Higle and Sen (1996b)).

3.1. Algorithmic Details of SD and Convergence

The earliest paper on SD (Higle and Sen (1991a)) proposed using linear master programs as is common in Benders' decomposition (Birge and Louveaux (1997)). Subsequently the method was extended to a bundle-type method via a regularized algorithm (Higle and Sen (1994)). In order to keep this presentation self-contained, we provide a brief review together with some results.

For this, and the following section, we will suppress the index m because our focus will be on calculations during any replication. At iteration k, a VF approximation will be given by the pointwise maximum of some linear (formally affine) functions, that is, $f_{k-1}(x) = c^\top x + \max\{h_{j,k-1}(x), j \in J_{k-1}\}$. With each index j, we will record $t(j)$ as the iteration at which the linear function was created. This quantity $t(j)$ will also remind us of the sample size used to create the j^{th} function. It was shown in Higle and Sen (1994) that one only needs $n_1 + 3$ indices (at most) in J_{k-1}, where n_1 denotes the number of decision variables in the first stage, and functions that are deleted will be "forgotten" for all future iterations.

Any iteration of the SD algorithm works like an election. At iteration k, we start with a previously chosen incumbent decision and a VF approximation $f_{k-1}(x)$. The algorithmic strategy is to present a challenge to the incumbent by finding a solution to the following optimization problem.

$$x^k \in \text{argmin}\{f_{k-1}(x) + \frac{\rho}{2}\|x - \hat{x}^{k-1}\|^2 \mid x \in X\} \tag{5}$$

The decision x^k is referred to as the candidate. The quadratic term in the above problem is variously referred to as a proximal term or Tikhonov regularization. This form of decomposition (SD, Higle and Sen (1994)) predates recent bundle methods (e.g., Oliveira et al. (2011)) and is different in several ways: a) convergence of SD is proven for both continuous and discrete random variables, b) we work from statistical approximations of the objective function, which in turn requires a completely different convergence concept,

c) SD requires only approximate second stage solutions based on previously discovered dual extreme points, d) the master program (5) is dual to the one used in the bundle method, and e) SD is able to include statistical stopping rules as discussed subsequently in section 3.2. The quantity $\rho \geq 1$ and is chosen adaptively depending upon the progress observed during the algorithm. Formally, it should also be indexed by k, but since we will not discuss the procedure to update ρ, we prefer to maintain an un-indexed parameter.

We will pit the two competing decisions $\hat{x}^{k-1}$ (incumbent) and x^k (candidate) against each other using an *updated* value function $f_k(x)$. If the candidate happens to be significantly better (lower value) than the incumbent, then we accept it as the new incumbent. Otherwise, there is no incumbent update. The first question at this point is: how does one calculate and update the VF approximations $f_k(x)$? We accomplish the update in the following steps:

1. Generate a new outcome ω^k, independent of all previous outcomes.

2. Let π_k denote the optimal conditional shadow price for the second stage LP, given inputs (x^k, ω^k). Assuming V_{k-1} (possibly empty) is available from previous iterations, $V_k \leftarrow V_{k-1} \cup \{\pi_k\}$.

3. Derive two SASA functions denoted $h_{\nu,k}, h_{0,k}$, with the former representing the candidate, and the latter representing the incumbent. Let $J_k = \{0, 1, \ldots, \nu\}$ be the index set of SASA functions at iteration k. We simply present calculations for $h_{\nu,k}$, and recognize that $h_{0,k}$ is calculated similarly (by replacing x^k with $\hat{x}^{k-1}$ in (7)).

$$h_{\nu,k}(x) = \frac{1}{k}\sum_{i=1}^{k} \pi_{i,k}^{\top}[\xi(\omega^i) - C(\omega^i)x] \tag{6}$$

where, $i = 1, \ldots, k$ and $\pi_{i,k}$ is a conditional shadow price for outcome i, and is calculated as follows.

$$\pi_{i,k} \in \text{ argmax } \{\pi^{\top}[\xi(\omega^i) - C(\omega^i)x^k] \mid \pi \in V_k\} \tag{7}$$

4. Delete one of those linear functions $h_{j,k-1}$ for which the dual multiplier obtained by solving (5) is zero. Create an updated index set J_k. (One can ensure that at most $n_1 + 3$ indices are present in J_k.)

5. Let $t(j)$ denote the iteration at which inequality $j \in J_k$ was created. Under the assumption that $h \geq 0$ for all (x, ω), the functions $j \notin \{0, \nu\}$ are updated as follows.

$$h_{j,k}(x) \leftarrow \left(\frac{t(j)}{k}\right) h_{j,t(j)}(x), \quad j \in J_k \backslash \{0, \nu\}. \tag{8}$$

Remark 1. The approximations used in SD depends critically on the SASA functions (see (6)) derived for both the candidate and the incumbent solutions. These functions are similar in spirit to Benders' cuts; however, they are different in several important ways: a) they use the empirical distribution induced by the sample, b) as seen in (7) they can be derived by using approximately optimum shadow prices (in V_k), and c) as iterations proceed, the SASA functions that were generated in past iterations are given less weight because they were created using a smaller sample size (than k). Thus unlike cuts in Benders' decomposition, each SASA function ultimately fades away, thus avoiding the persistence of poor (sampled) approximations. Because they do not cut away any part of the epigraph permanently, we do not refer to these approximations as "cuts" (as in traditional Benders' decomposition).

Let $\Delta_k = f_{k-1}(x^k) - f_{k-1}(\hat{x}^{k-1})$ denote the predicted change under the VF approximation f_{k-1}. We allow $\rho \geq 1$ to be chosen in such a way that $\Delta_k \leq 0$. Then the winner of the election using the approximation f_k will be the next incumbent $\hat{x}^k$ as obtained below, with a fixed parameter $r \in (0,1)$ (set at 0.2 in our experiments).

$$\hat{x}^k = \begin{cases} x^k & \text{if} \quad f_k(x^k) < f_k(\hat{x}^{k-1}) + r\Delta_k \\ \hat{x}^{k-1} & \text{otherwise} \end{cases} \tag{9}$$

In the event that the incumbent changes, the positioning of the incumbent and candidate inequalities in J_k must also be swapped.

Before presenting the results of this paper, we summarize some key results from two previous papers in the following proposition. Parts (a) and (b) below pertain to both linear master problems Higle and Sen (1991a) as well as the regularized master problems Higle and Sen (1994).

PROPOSITION 1. Let $\{f_k\}$, $\{x^k\}, \{\hat{x}^k\}$ be the sequences of objective function approximations, candidate solutions and incumbent solutions, respectively, generated by the algorithm. Let K denote the sequence of iterations where the incumbent changes.

(a) If $\{\hat{x}^k\}_{k\in K} \to \hat{x}$, then $\{f_k(\hat{x}^k)\}_{k\in K} \to f(\hat{x})$ (with probability one). Furthermore, every limit of $\{(\alpha_{0,k}, c + \beta_{0,k})\}_{k\in K}$ defines a support of f at $\hat{x}$ (with probability one).

(b) Let $\{\hat{x}^k\}_{k=0}^{\infty}$ and $\{x^k\}_{k=1}^{\infty}$ be the sequence of incumbent and candidate solutions, respectively, generated by the regularized SD algorithm. With probability one there exists a subsequence of iterations, indexed by a set K_*, such that $\lim_{k\in K_*} f_{k-1}(x^k) + \frac{1}{2}\|x^k - \hat{x}^{k-1}\| - f_{k-1}(\hat{x}^{k-1}) = 0$, and every accumulation point of $\{\hat{x}^k\}_{k\in K_*}$ is an optimal solution of (1).

PROOF. (a) See Theorem 2 in Higle and Sen (1991a).

(b) See Theorem 5 in Higle and Sen (1994).

Part (a) of the above proposition shows that if the SD algorithm obtains asymptotically accurate objective function values (with probability one) whereas part (b) shows the existence of a subsequence whose accumulation points are optimal. Note that this does not rule out the possibility of other subsequences. In Theorem 1 we show that the entire sequence of incumbents does converge to an optimal solution with probability one. The existence of the limit is particularly important in a stochastic algorithm because the approximations become more dependable in this case. However, observe that the uniqueness of the limit point does not require the instance to have a unique solution.

Theorem 1. Assume X is a compact set, and the fixed-and-relatively-complete recourse assumption holds. In addition, assume that $\rho \geq 1$, and the recourse function $h(x,\omega)$ is non-negative for all $x \in X$ almost surely. Then the sequence $\{\hat{x}^k\}$ converges to an optimal solution with probability one.

Proof. See Appendix A.

A few comments on analytical predictions (as opposed to computational experiments) are also in order. In this regard, recall that any SAA implementation separates the computational work along two dimensions: numerical optimization and statistical validation. Royset and Szechtman (2013) explores a variety of combinations based on asymptotic rates for using numerical optimization within SAA. A related method, known as Retrospective Approximations (RA), has been studied in Pasupathy (2010) where the sample average function is allowed to use larger sample sizes as iterations proceed. It is easy to see that RA has similarities with both SAA as well as SD. Like SAA, it seeks a near-optimal solution to an approximate problem of the same form as SAA (3); although like SD, it uses a growing set of outcomes. Indeed, RA is perhaps closest in spirit to a precursor of SD known as a Stochastic Cutting Plane method (SCP, Higle and Sen (1996b)), and for smooth problems, one might expect SCP to have similar convergence rates as RA, asymptotically.

3.2. Stopping Rules

As with any decision-making algorithm, SD must be terminated in finitely many iterations. Because the expectation operator requires multi-dimensional integration, providing deterministic certificates of optimality for practical instances (with several random variables) is intractable in general. As a result, a tandem of stopping rules, one based on In-Sample estimates, and the other on Out-of-Sample tests have been studied previously in a series of papers (Higle and Sen (1991b, 1996a, 1999)). We will comment on the performance of these tests in the appropriate subsections below. At this point, it suffices to say that previously known hurdles (e.g., the inability to reconcile multiple solutions from replications, relatively large gap estimates, and in some cases time-consuming LP-based bootstrap processes) have been overcome through a fresh view of the Out-of-Sample tests. These stopping rules are presented next.

3.2.1. In-Sample Stopping Rule. As proposed in Higle and Sen (1999), the In-Sample rule is intended to address two issues:

1. (Shadow Price Stability.) To assess whether the approximation due to (7) exhibits any sensitivity to additional information in the form of new shadow prices.

2. (Primal-Dual Gap Stability.) To recognize whether the estimated primal and dual solutions associated with the (5) are sensitive to variability due to sampling.

Shadow Price Stability. At iteration k, we assess the impact of new information (new outcomes, new first-stage candidate solutions and most importantly new shadow prices) on VF approximations. Note however that the first term of any VF approximation is linear ($c^\top x$) and c is known. Hence the predictive capacity of a VF approximation depends on how well the shadow prices in V_k predict the recourse function realizations $h(x,\omega)$ for any pair (x,ω) encountered by the algorithm. For any set of runs, suppose that we fix a tolerance level denoted ϵ. For such a run, let $q \in [2, k-1]$ define a window of iterations which will be used to ascertain whether further iterations are meaningful for the purpose of improving the approximation of the recourse function. Towards this end we observe the difference in approximation-quality when the recourse function is estimated using V_q versus the larger set V_k in (7). The value q represents a window of iterations. We look at what might have happened if we stop collecting dual vertices at iteration q. And the impact of extra $(k-q)$ iterations on the VF approximations is estimated using the following ratio:

$$R_q^k(x) = S_q^k(x)/S_k^k(x), \text{ where} \tag{10a}$$

$$S_q^k(x) = \sum_{i=1}^{k} \max\{\pi^\top[\xi(\omega^i) - C(\omega^i)x] | \pi \in V_q\} \tag{10b}$$

$$S_k^k(x) = \sum_{i=1}^{k} \max\{\pi^\top[\xi(\omega^i) - C(\omega^i)x] | \pi \in V_k\}. \tag{10c}$$

These ratios are calculated whenever we calculate a subgradient at either a candidate (x^k) or an incumbent $(\hat{x}^{k-1})$. By assumption, $S_k^k(x) > 0$, and since we have $S_q^k(x) \leq S_k^k(x)$, we have $0 \leq R_q^k(x) \leq 1$. Then, we assess the stability of $R_q^k(x)$ by examining its sample mean and variance over the most recent $w(\epsilon)$ iterations. When these measures are sufficiently close to target thresholds (0.95 for the sample mean, and 0.00001 for the sample variance), then we declare the set of shadow prices to be stable. Appendix C provides figures showing the evolution of $R_q^k(x)$ associated with candidate and incumbent solutions for the industrial-strength instances in our study. In our implementation, we use $w(\epsilon) \in \{64, 256, 512\}$ for $\epsilon \in \{\text{Loose }(0.01), \text{Nominal}(0.001), \text{Tight}(0.0001)\}$ respectively. Finally, to ensure an unbiased estimate of the objective value, we verify that

$$h_{0,k}(\hat{x}^{k-1}, \omega^t) = h(\hat{x}^{k-1}, \omega^t), t = 1, \ldots, k. \tag{11}$$

Primal-Dual Gap Stability. Formally we wish to estimate the probability $P(f(\hat{\mathbf{x}}) - f^* \leq \epsilon)$ where f^* denotes the optimal value of the "true" problem. Recall that (5) is defined using several SASA functions and hence the primal as well as its dual are random objects. We will consider a primal-dual pair of solutions $(\hat{x}^{k-1}, \hat{\theta}^k, \hat{\lambda}^k)$ which are primal and dual optimum for (5) in iteration k. We wish to ascertain whether the variability of the gap estimate is small enough that this pair of primal and dual solutions may be considered to be close enough to optimality i.e., within a specified tolerance. To do so, we use the general concept of bootstrapping as set forth in Efron (1979) (see also Singh (1981)). In the context of SD, bootstrapping involves re-sampling each SASA function in J_k to create a re-sampled instance of (5). This application of bootstrapping was first used in Higle and Sen (1991b) where primal and dual pairs were both linear programs. Because re-sampled versions of linear programming approximations may render the dual multipliers $(\hat{\theta}^k, \hat{\lambda}^k)$ infeasible for the re-sampled approximation, the above implementation of the bootstrapping procedure required solving each re-sampled dual problem, thus making it somewhat computationally burdensome. In a subsequent paper, Higle and Sen (1999) proposed the primal-dual pair of QPs below which overcome the computational demands posed by the earlier LP counterpart.

Let β^j denote an affine piece and B_k denote the corresponding matrix of all affine functions in iterations k. Accordingly, let H_k denote the vector of scalars $h_{j,k}(\hat{x}^{k-1})$. As shown in Higle and Sen (1994), the set J_k has cardinality of at most $n_1 + 3$, thus maintaining a finite bound on the size of the primal master problem. Let A and b define the polyhedron $X = \{Ax \leq b\}$ and let b_k denote the quantity $b - A\hat{x}^{k-1}$. Then using primal decisions as $\upsilon = x - \hat{x}^{k-1}$ the primal and dual problems may be stated as

$$u = \text{Min} \quad h + c^\top \upsilon + \frac{\rho}{2}\|\upsilon\|^2 \tag{12a}$$

$$\text{s.t. } h - (\beta^j)^\top \upsilon \geq h_{j,k}(\hat{x}^{k-1}) \quad \forall j \in J_k \tag{12b}$$

$$\hat{x}^{k-1} + \upsilon \in X \tag{12c}$$

$$\ell = \text{Max} \quad H_k^\top \theta + b_k^\top \lambda - \frac{1}{2\rho}\|c + B_k^\top \theta + A^\top \lambda\|^2 \tag{13a}$$

$$\text{s.t.} \quad \mathbf{1}^\top \theta = 1, \theta \geq 0, \lambda \geq 0 \tag{13b}$$

With the above formulations, the point $\hat{x}^{k-1}$ gets mapped to $\upsilon = 0$ in the primal, and the multipliers $(\hat{\theta}^k, \hat{\lambda}^k)$ are to be used for the dual. The gap estimate for this pair of solutions will be estimated by constructing a primal-dual pair in which each affine piece is represented by a re-sampled estimate. Very briefly, the idea is as follows. Let $\{\omega^1, \ldots, \omega^k\}$ be a random i.i.d. sample of size k with distribution $\mathcal{F}$ and let $\mathcal{F}_k$ denote the empirical distribution of $\{\omega^1, \ldots, \omega^k\}$. Define a random object $T(\omega^1, \ldots, \omega^k; \mathcal{F})$, which depends on the distribution $\mathcal{F}$. The bootstrap method is used to approximate the distribution of $T(\omega^1, \ldots, \omega^k; \mathcal{F})$ by that of $T(\psi^1, \ldots, \psi^k; \mathcal{F}_k)$ under $\mathcal{F}_k$, where $\{\psi^1, \ldots, \psi^k\}$ denotes a random sample of size k under distribution $\mathcal{F}_k$. Next, we summarize an important theorem by Singh (1981).

Proposition 2. Let $\mu = E_{\mathcal{F}}[\omega]$, $\bar{\omega}^k = 1/k \sum_{i=1}^k \omega^i$, $\bar{\psi}^k = 1/k \sum_{i=1}^k \psi^i$ and assume $E_{\mathcal{F}}[\omega^2] < \infty$. Let P and P_k denote the probabilities under $\mathcal{F}$ and $\mathcal{F}_k$ respectively. Then

$$\lim_{k\to\infty} |P(k^{1/2}(\bar{\omega}^k - \mu) \leq s) - P_k(k^{1/2}(\bar{\psi}^k - \bar{\omega}^k) \leq s)| = 0 \quad a.s. \tag{14}$$

Proof. See Singh (1981). ■

Basically, Proposition 2 studies the convergence (to zero) of the discrepancy between distribution of $k^{1/2}(\bar{\omega}^k - \mu)$ and the bootstrap approximation. Essentially, pivotal statistics like the sample average are appropriate for bootstrapping because they are based on linear operators. In order to apply this idea to our setting, we re-sample every SASA function defining the primal (12) to obtain a re-sampled primal. The dual corresponds to this re-sampled primal, and therefore has the form (13) using the re-sampled data. Thus in our re-sampling process we set up primal and dual problems from which we compare the gap associated with the given primal-dual pair. Note that this procedure re-samples pivotal statistics (i.e., sample averages) and not the duality gap directly because the latter is not necessarily pivotal. Moreover, since the primal solution $\hat{x}^{k-1}$ and the dual solution $(\hat{\theta}^k, \hat{\lambda}^k)$ are feasible to the respective re-sampled problems, we are able to calculate the re-sampled gap estimates by simply computing the primal and dual objective functions for each re-sampled instance. This eliminates the need to solve LPs as in the original bootstrapping method of Higle and Sen (1991b). Let $u_i - \ell_i$ denote the gap obtained for the i^{th} re-sampled instance, and from these we compute the empirical frequency distribution $\mathcal{F}_k(\epsilon)$ to estimate $P(u - \ell \leq \epsilon)$. Thus when $\mathcal{F}_k(\epsilon) \geq 1 - \delta$, then the replication is terminated, and the next SD replication can begin. In fact, the actual implementation uses a relative tolerance, so that a replication terminates when duality gap is small relative to the current incumbent value. For our computational experiments, we use $\delta = 0.05$, so that the In-Sample test requires 95% of the bootstrapped gap estimates to pass the test.

3.2.2. Out-of-Sample Stopping Rule. The idea of replication in stochastic programming has been adopted in many papers dealing with sample-based algorithms (Mak et al. (1999)). Because sampling introduces randomness into the algorithm, it is important to characterize errors in a manner that provides statistical performance guarantees. As the reader may have observed from Figure 1, the variability of solutions as well as objective values can be significant. Indeed, Linderoth et al. (2006) also report wide disparity of solutions of sampled instances of SSN with a sample size of 5000. A common suggestion is to obtain a preliminary objective estimate for solutions obtained from each replication, and then to successively prune (solutions) and refine objective estimates, until the subset of solutions is small enough to recommend a decision (Linderoth et al. (2006)). In order to overcome issues related to the complexity of multiple replications, Bayraksan and Morton (2011) proposed a sequential sampling scheme, where the increase in sample size was controlled. To the best of our knowledge, this idea has been tested on some of the smaller instances (e.g., GBD, PGP2, 4TERM) and the computational results suggest that for instances with higher variability (e.g., PGP2), multiple replications provide more reliable estimates. In contrast, Nesterov and Vial (2008) suggest multiple replications with small increments to the sample size, and then to use an average solution, instead of trying to find the best among the replications. This idea has some similarity with the concept of compromise decisions which we report below. Our Out-of-Sample test will leverage not only the primal solutions $\{\mathbf{x}^m\}_{m=1}^M$, but also the value function approximations $\{f^m(x)\}_{m=1}^M$.

To motivate the out-of-sample stopping rule, we use certain concepts from prox-mapping theory, which we summarize below. In this context, the master program (5) may be looked upon as creating a sequence $\{x^k\}$ such that

$$x^k = \mathbf{prox}(\rho^m; f^m_{k-1}; \hat{x}^{k-1}),$$

where we write ρ^m to denote the proximal parameter for the m^{th} replication. In contrast, deterministic algorithms use a deterministic function f^m, instead of a stochastic approximation f^m_{k-1}. As a result, optimality for deterministic algorithms can be recognized when one arrives at a fixed point x^* of the prox-mapping, that is, $x^* = \mathbf{prox}(\rho^m; f^m; x^*)$, or simply $x^* = \mathbf{prox}(\rho^m; f^m)$. Despite in-sample optimality tests, stochastic algorithms like SD use sampled approximations f^m_{k-1}, and the variability does not vanish for finite k. Nevertheless, we will use ideas from deterministic optimization to seek reduced variance solutions to the "true" problem. When the functions f^m are deterministic, one enjoys the following identity

$$\frac{1}{M}\sum_{m=1}^{M}\mathbf{prox}(\rho^m; f^m) = \mathbf{prox}(\frac{1}{M}\sum_{m=1}^{M}\rho^m; \frac{1}{M}\sum_{m=1}^{M}f^m). \tag{15}$$

In a stochastic setting, there is variance associated with the expressions on both sides of (15). Nevertheless, the variance on the right hand side can be expected to be smaller, and we formalize this idea in Theorem 2 of this section. With this motivation, we propose to solve the following.

$$\underset{x\in X}{\text{Min}} \quad \frac{1}{M}\sum_{m=1}^{M}[f^m(x) + \frac{\bar{\rho}}{2}\|x - \mathbf{x}^m\|^2], \tag{16}$$

where $\bar{\rho}$ is the sample average of $\{\rho^m\}$. We refer to (16) as the Compromise Problem and its solution as the Compromise Decision. The Compromise Problem represents two objectives: the average value function approximation and the sum of squared errors. Let $\mathbf{x}^c$ denote the compromise decision, and $\bar{\mathbf{x}} = \frac{1}{M}\sum_m \mathbf{x}^m$. Intuitively, $\bar{\mathbf{x}}$ is an optimal solution to the squared errors part of the objective and if $\mathbf{x}^c$ and $\bar{\mathbf{x}}$ agree, then clearly, both are optimal to (16).

On a related note, we observe that while it might seem that (16) is even more onerous than the individual replications, that is not the case. Because the In-Sample rule already provides $\mathbf{x}^m$, the sample average solution $\bar{\mathbf{x}}$ is already available, and so are the VF approximations $\{f^m\}$. Hence all that is required is to warm-start the quadratic program (16) using $\bar{\mathbf{x}}$, and if it declares optimality at the start, one can directly use $\bar{\mathbf{x}}$ as the compromise decision. However, since the marginal amount of computations to solve the compromise problem (independently of the sample average decision) is not very resource intensive, we calculate both $\mathbf{x}^c$ and $\bar{\mathbf{x}}$ independently so that the reader can verify similarities between these solutions.

Let $K_m(\epsilon)$ denote the sample size used to construct the SASA function at $\mathbf{x}^m$ at termination for a given trial m and $\epsilon > 0$. Define $N = \min\{K_m(\epsilon), m = 1, \ldots, M\}$, the smallest sample size of the SASA functions used by any of the M approximations at their terminal solutions $\mathbf{x}^m$. Of course, N depends on ϵ, but we suppress this dependence for notational convenience.

Remark 2. The terminal solution $\mathbf{x}^m$ was used to generate a SASA function using $K_m(\epsilon)$ observations. Recall that these observations are i.i.d, and the in-sample stopping rule ensures (11). Then the Central Limit Theorem (CLT) implies that $[K_m(\epsilon)]^{\frac{1}{2}}[f(\mathbf{x}^m) - f^m(\mathbf{x}^m)]$ is asymptotically normal $\mathbf{N}(0, \sigma_m^2)$, where $\sigma_m^2 < \infty$ denotes the variance of $f^m(\mathbf{x}^m)$. Since $0 < N \leq K_m(\epsilon)$, it follows that the error $[f(\mathbf{x}^m) - f^m(\mathbf{x}^m)]$ is no greater than $O_p(N^{-\frac{1}{2}})$. This property is useful in the following lemma.

Theorem 2. Suppose that assumptions 1-4 stated at the beginning of this section hold. If $\mathbf{x}^c = \bar{\mathbf{x}}$,

a) then $\mathbf{x}^c$ solves

$$\min_{x \in X} \quad \bar{F}_M(x) := \frac{1}{M} \sum_{m=1}^{M} f^m(x), \tag{17}$$

b) and

$$f(\mathbf{x}^c) \leq \bar{F}_M(\mathbf{x}^c) + O_p((NM)^{-\frac{1}{2}}). \tag{18}$$

Proof: See Appendix A.

The first claim in the theorem shows that under replication, a sufficient condition for optimality is $\mathbf{x}^c = \bar{\mathbf{x}}$, and as with any optimality test, it is customary to verify this condition by checking whether $\|\mathbf{x}^c - \bar{\mathbf{x}}\| \leq \epsilon$ for a pre-specified ϵ. What is more important in Theorem 2 is that the compromise solution provides a decision whose total (objective value) error, via bias and variance, is of the order of $(NM)^{-\frac{1}{2}}$. In contrast, each solution $\mathbf{x}^m$ has a total (objective value) error of the order of $(N)^{-\frac{1}{2}}$, which is higher for $M \geq 2$. This variance reduction is one of the key properties of the compromise decision. As shown below, it also provides asymptotic consistency. In this regard, note that when $\epsilon \to 0$, we have $K_m(\epsilon) \to \infty$, and $N \to \infty$. As the reader might recall, N depends on ϵ, and in fact, so does $\bar{F}_M$. If we make this dependence explicit by using $\bar{F}_{\epsilon,M}$.

Theorem 3. Let the assumptions of Theorem 1 hold and, let f^* denote the optimal value of the problem. Let $\mathbf{x}^c(\epsilon)$ and $\mathbf{x}^m(\epsilon)$ denote the quantities analogous to those in Theorem 2 for a given ϵ. Suppose that $\lim_{\epsilon\to 0}\mathbf{x}^c(\epsilon) = \mathbf{x}^c$ and $\lim_{\epsilon\to 0}\mathbf{x}^m(\epsilon) = \mathbf{x}^m$ and $\bar{\mathbf{x}} = \frac{1}{M}\sum_{m=1}^{M}\mathbf{x}^m$. If $\mathbf{x}^c = \bar{\mathbf{x}}$, then, $\lim_{\epsilon\to 0} P(|\bar{F}_{\epsilon,M}(\mathbf{x}^c(\epsilon)) - f^*| < t) = 1 \ \forall t > 0$.

Proof. See Appendix A.

The increased reliability offered by compromise decisions reported for SSN (see Figure 2) might have come as a surprise for a problem that has been characterized as "ill-conditioned". The reader might recall that for the loose tolerance run, Figure 2 already showed good agreement between the average and compromise decisions, and a decision-maker might have been satisfied with an objective function upper bound of 9.951 (see Table 2). However, if the decision-maker chooses to obtain greater accuracy, then, it is easy to resume the SD process for the nominal tolerance using all the information obtained in the course of the previous run (with loose tolerance). Thus, while greater reliability may require more sampling (for instances like SSN), the marginal effort is simply the additional iterations and not an entire run from scratch. Such warm-starting is critical for those decision-makers who seek greater accuracy in decision support.

To give the reader a sense of the computational time involved for each of tolerance level of SSN, we note that the CPU seconds for the compromise problems were 5.78 (Loose Tol.), 5.86 (Nominal Tol.) and 6.21 (Tight Tol.) respectively. When compared with the effort required to estimate the objective function at one or more of the points $\{\mathbf{x}^m\}$, the solution time of the Compromise Problem is minimal. To close this section, we note that there are several instances in the literature (e.g., CEP1 and STORM) for which longer runs (and even replications) are an overkill. By monitoring the coefficient of variation of each decision variable across successive replications, several instances can be accurately solved without excessively long runs. We shall report these computational results in the following section.

4. Further Computational Results

Table 4 summarizes all test instances solved by SD under nominal tolerance. BAA99 is a single period multi-product inventory model (see Bassok et al. (1999)) and BAA99-20 is a larger version (20 products) of BAA99. CEP is a capacity expansion model which appears in Higle and Sen (1996b). LandS, LandS2 and LandS3 are three versions of a power generation planning problem (Louveaux and Smeers (1988)). The original LandS instance has one random variable with 3 outcomes. LandS2 has 3 random variables with a total of 64 scenarios. LandS3 has 3 random variables each discretized to a finer normal distribution with a total of 10^6 scenarios. PGP2 is the same genre of instances, but has greater objective function variability in value as well as solution variability (Higle and Sen (1996b)). The multi-location transshipment model called RETAIL can be found in Herer et al. (2006). SSN is the network expansion model presented at the beginning of this paper. STORM is an air freight scheduling model (Mulvey and Ruszczyński (1995)). 4NODE and 20TERM are both freight fleet scheduling problems, with the former data set appearing on a web site due to A. Felt (http://www4.uwsp.edu/math/afelt/slptestset/download.html), and the latter came to us from Infanger (1999). Along with SSN, STORM and 20TERM are based on industrial applications, and are indicated by an asterisk in Table 4. One of the larger data sets in this set is STORM. While it is large both in size and number of scenarios, the variability of its objective function is relatively small, as confirmed by the ratios $R_k(x)$ reported for STORM in Appendix C.

Because the early computational tests in the Stochastic Programming literature were performed using deterministic algorithms, it became customary to report performance by creating alternative instances using different sample sizes. For example, Zverovich et al. (2012) reported 16 different sample sizes for 4NODE, corresponding to scenarios ranging from $2^0, 2^1, \ldots, 2^{15}$. In our study, we do not consider these as 16 different instances, but one instance with 2^{15} scenarios, and our goal is to provide statistical guarantees for objective value upper and lower bounds, together with recommendations of the compromise and average solutions with the former being generated without warm-starting (using the average solution). We made this choice to avoid prompting the compromise problem in any way.

In Table 5, we report the quality of average solutions as well as compromise solutions for the instances shown in Table 4. Since the bounds on the problems are statistically

Table 4 Test instances with SD under Nominal Tolerance

Instance Name	1^{st} stage variables/constraints	2^{nd} stage variables/constraints	# of random variables	Magnitude of scenarios
BAA99	2/0	7/4	2	615
BAA99-20	20/0	250/40	20	10^{34}
CEP	8/5	15/7	3	216
LandS	4/2	12/7	1	3
LandS2	4/2	12/7	3	64
LandS3	4/2	12/7	3	10^{6}
PGP2	4/2	16/7	3	576
RETAIL	7/0	70/22	7	10^{11}
SSN*	89/1	706/175	86	10^{70}
STORM*	121/185	1259/528	117	10^{81}
4NODE	52/14	186/74	12	32768
20TERM*	63/3	764/124	40	10^{12}

estimated, we report 95% confidence intervals for all upper and lower bounds in the table. The upper bound confidence intervals are calculated for both average decisions as well as compromise decisions. These confidence intervals are derived by running Monte Carlo experiments to estimate $f(\bar{\mathbf{x}})$ and $f(\mathbf{x}^c)$. We chose the sample size for these estimates to ensure that the half width is within 1% of the value.

The lower bounds in Table 5 are calculated by using SD values $f^m(\mathbf{x}^m), m = 1, \ldots, M$. To do so, let $L_M = \frac{1}{M}\sum_{m=1}^{M} f^m(\mathbf{x}^m)$, and calculate $s_M^2 = \frac{1}{M-1}\sum_{m=1}^{M}(f^m(\mathbf{x}^m) - L_M)^2$. Since $M = 30$, we chose[1] $z_{\alpha/2}$ such that $P\{\mathbf{N}(0,1) \le z_\alpha\} = 1 - \alpha$ and report

$$\left[L_M - \frac{z_{\alpha/2}s_M}{\sqrt{M}}, L_M + \frac{z_{\alpha/2}s_M}{\sqrt{M}}\right], \text{ where } \alpha = 0.05. \qquad (19)$$

We see that for all instances, the confidence levels obtained using the nominal tolerance is acceptably close to the lower bound confidence interval. Moreover, in comparing the estimated value of the compromise decision with that of the average decision, we find that the former has either equal or better objective value estimates, except for BAA99 and

[1] As is standard in the statistical literature, α denotes the confidence level, whereas elsewhere in the paper, α denotes constants of the minorants in SD.

LandS2, where the discrepancy is only in the third decimal place. Such minor deviations are inevitable in sample-based algorithms. These solutions are obtained with nominal tolerance, although the user has the option to be more demanding, and seek solutions with tighter tolerance without having to re-start the process "from scratch". This is because SD records all shadow prices, and SASA functions for each run, and they can be used immediately to continue further iterations if the user desires. For large scale models like SSN, this is a powerful setup. Another aspect of our study that is unique to SD is its ability to report the sample sizes that were necessary to obtain the quality of solution. This is particularly important in large scale applications where optimization and simulation are tightly coupled (see the wind energy integration model for the state of Illinois in Gangammanavar et al. (2015)).

In Table 6 we report the sample sizes used for each instance (together with its variability over 30 replications). From this table one can observe that although STORM is very large in terms of the variables, constraints and scenarios, it requires a sample size of only 300 to get reasonable solutions. This is about the same number of samples required for much smaller instances such as PGP2 and BAA99. From our study, we also observe that 20TERM and RETAIL take more iterations, and in fact, the latter takes longer for objective function estimation even though the instance (RETAIL) is smaller than 20TERM in terms of decision variables and constraints. As expected it is the variability that makes instances take longer. Finally, it is clear that SSN is by far the most demanding of these instances, but each replication can be completed in about 90 secs for nominal tolerance. It was shown in Figure 2 that this solution has very low error from the average solution. From this point of view, the user may decide to accept the decision based on the fact that the difference between the compromise and average solutions is small enough. However, as shown in section 2 tighter statistical bounds can also be obtained by demanding greater precision on the bounds. Such solution reports are likely to provide decision makers a great deal of confidence in adopting decisions recommended by the SLP solver.

Table 5 Objective upper and lower bounds and corresponding 95% confidence intervals of all instances solved by SD

Instance	Lower Bound	Average Decision	Compromise Decision
		Upper Bound	Upper Bound
Name	(95% CI's)	(95% CI's)	(95% CI's)
BAA99	-240.864 (±5.988)	-236.204 (±5.451)	-236.203 (±5.451)
BAA99-20	-322870.496 (±1337.854)	-318393.648 (±3994.053)	-318451.800 (±3994.502)
CEP	353080.809 (±10530.916)	354175.320 (±1687.537)	354175.320 (±1687.537)
LandS	381.120 (±1.174)	381.761 (±1.309)	381.761 (±1.309)
LandS2	227.789 (±1.628)	227.393 (±0.668)	227.395 (±0.668)
LandS3	225.712 (±1.319)	225.541 (±0.640)	225.541 (±0.640)
PGP2	447.339 (±2.157)	447.955 (±1.406)	447.928 (±1.405)
RETAIL	153.995 (±2.573)	154.411 (±0.772)	154.406 (±0.772)
SSN	9.736 (±0.118)	9.927 (±0.050)	9.927 (±0.050)
STORM	15493958.503 (±8826.814)	15481852.286 (±48193.646)	15481760.131 (±48208.190)
4NODE	446.979 (±0.068)	447.058 (±0.392)	446.951 (±0.394)
20TERM	253649.385 (±168.573)	254515.479 (±1005.008)	254514.672 (±1004.934)

Table 6 Sample size (std dev), solution time for SD(std dev), solution time for Compromise Problem and evaluation time (of compromise decision and average decision) for all instances solved by SD

Instance Name	Average Sample Size (Std Dev) for SD	Average CPU secs (Std Dev) for SD	CPU secs to Solve Compromise Problem	CPU secs to Estimate Obj of Compromise Decision	CPU secs to Estimate Obj of Average Decision
BAA99	298.03(58.82)	0.89(0.23)	0.01	0.25	0.21
BAA99-20	330.50(13.68)	1.49(0.13)	0.04	0.13	0.14
CEP	263.03(6.73)	0.79(0.08)	0.01	5.61	5.43
LandS	260.27(2.77)	0.76(0.07)	0.01	0.21	0.22
LandS2	264.27(4.86)	0.83(0.08)	0.01	1.29	1.26
LandS3	263.57(5.99)	0.78(0.07)	0.00	0.78	0.74
PGP2	284.63(27.60)	0.74(0.10)	0.01	0.31	0.31
RETAIL	460.47(150.55)	1.53(0.67)	0.00	4.01	3.98
SSN	2286.90(341.71)	90.50(20.56)	5.86	112.31	111.52
STORM	300.50(5.33)	19.56(1.46)	8.83	0.06	0.05
4NODE	406.07(49.44)	5.66(0.85)	3.04	0.01	0.01
20TERM	453.07(5.98)	7.28(0.39)	1.13	0.13	0.12

5. Conclusions

Our contributions to the SP literature may be viewed through several lenses. These points of view are captured by the questions that were posed in section 1, and in this context, we offer the following comments.

1. Given that SP problems can be demanding, greater accuracy may call for re-runs that should re-use previously discovered structures of an instance. How can such structures be re-used for the purposes of warm-starting?

- Because SD uses "sampling-on-the-fly" (i.e., adaptive sampling), it creates a sequence of SASA functions based on re-using shadow prices. This approach records information gathered during the course of the algorithm (e.g., shadow prices, and SASA functions), so that future iterations can be resumed without having to re-discover them from scratch. This capability is also important in the context of Stochastic MIPs, just as fast LP solvers, and warm-starting are important for deterministic MIPs.

2. Given that SLP models have a very special (convex) structure, how should sampling-based methods be designed to take advantage of such structure?

- Convex non-smooth optimization forms the backbone of SD, which derives its stability from using a proximal term. The latter also suggests a quadratic programming dual problem which is used for the In-Sample tests (using bootstrapping of primal and dual estimates) as well as the Out-of-Sample tests (using low variance value functions in the Compromise Problem). Such integration of numerical optimization and statistical computing promotes algorithmic synergy for more efficient solution.

3. Sampling-based SP methods borrow variance reduction techniques from the simulation literature. Are there other variance reduction techniques that are appropriate for SP, but are not considered in the simulation literature?

- Variance reduction in simulation is primarily aimed at performance (objective function) estimation. This is available in SD through common random numbers when comparing incumbent and candidate solutions. While this is important for SP models, it needs to go beyond performance estimation because of the emphasis on decision-making. In order to appreciate this point, note that past attempts at solving SSN have not only required high performance computing, but disagreement between alternative runs can leave users in a quandary. Instead, concurrence between compromise and sample average decisions, provides a principled and computationally effective approach. In addition, Theorem 2 provides variance reduction of replications from $O_p((N)^{-\frac{1}{2}})$ to $O_p((NM)^{-\frac{1}{2}})$. Moreover, the bias is reduced as well. Figure 1 highlights the empirical success of this approach.

4. Parallel architectures in SP have traditionally been used to process bunches of scenarios. Are there other ways to use parallel architectures which permit the solution of industrial-strength models?

• The speed of replications can be improved significantly by using parallelization. This approach to SP (and of course SD) needs greater attention. In the event that computational deadlines preclude a meaningful number of replications, one should verify whether the objective function estimate reported by SD falls in the 95% confidence interval for the upper bound resulting from one run. If it does not belong to the confidence interval, the analyst ought to be careful to consider faster machines, parallel processing and perhaps tighter tolerance.

5. Should SP algorithms report lower bound estimates for the "true" problem so that the quality of a recommended decision can be ascertained?

• It is difficult to establish the quality of a solution without both upper bound and lower bound estimates. The former is usually a statistical estimation problem, if a decision is available. The latter (lower bounds) is an optimization issue since it is the optimization step that provides the lower bounding estimate. Clearly this is an essential ingredient of any optimization method.

In summary we have demonstrated that two-stage SLP models are now solvable to very reasonable levels of accuracy, even for industrial-strength models, in reasonable time, using "run-of-the-mill" computing devices. This bodes well for users of the SLP modeling paradigm.

Acknowledgements: We are grateful to the National Science Foundation (Grant CMMI: 0900070) and AFOSR (Grant FA9550-13-1-0015) for their continued support of our research. We are also grateful to Shane Henderson and Raghu Pasupathy for their comments on an earlier version of the paper. We also thank the Area and Associate Editors, and the anonymous referees for their prompt, thorough and insightful comments while handling this paper.

Appendix A: Proofs

Proof of Theorem 1

In the course of the proof of part (b) of Theorem 1, Higle and Sen (1994) shows that if there are only finitely many distinct incumbent solutions generated by the algorithm, then the final incumbent solution is an optimum with probability one. In this case, there is clearly a unique solution generated by the algorithm. We prove this result for the case of infinitely many incumbents in Theorem 1. Let K denote the sequence of iterations where the incumbent changes, and for $\tau > 1$, let $k_0, k_1, k_2, \dots, k_\tau \in K$. Consider the quantity

$$\gamma_\tau = \frac{1}{\tau}\sum_{\ell=1}^{\tau}\big(f_{k_\ell-1}(\hat{x}^{k_\ell}) - f_{k_\ell-1}(\hat{x}^{k_{\ell-1}})\big) \tag{20a}$$

$$= \frac{1}{\tau}[f_{k_\tau-1}(\hat{x}^{k_\tau}) - f_{k_1-1}(\hat{x}^{k_0})] + \frac{1}{\tau}\sum_{\ell=1}^{\tau-1}\big(f_{k_\ell-1}(\hat{x}^{k_\ell}) - f_{k_{\ell+1}-1}(\hat{x}^{k_\ell})\big) \tag{20b}$$

Because of Proposition 1 part (a), the summation term in (20b) must approach zero (wp1) as $\tau \to \infty$. Hence

$$\lim_{\tau\to\infty}\gamma_\tau = \lim_{\tau\to\infty}\frac{1}{\tau}[f_{k_\tau-1}(\hat{x}^{k_\tau}) - f_{k_1-1}(\hat{x}^{k_0})] \text{ (wp1)}. \tag{21}$$

Because of the fixed and complete recourse assumption, and the compactness of X. The term $f_{k_\tau-1}(\hat{x}^{k_\tau}) - f_{k_1-1}(\hat{x}^{k_0})$ is bounded. Hence as $\tau \to \infty$. Then (21) implies

$$\lim_{\tau\to\infty}\gamma_\tau = 0 \text{ (wp1)}. \tag{22}$$

Now with $\rho \geq 1$, the optimality conditions for (5) at a candidate solution x^k imply that (see equation (2.6) on page 115 of Higle and Sen (1996b)).

$$f_{k-1}(x^k) - f_{k-1}(\hat{x}^{k-1}) \leq -\|x^k - \hat{x}^{k-1}\|^2 \leq 0. \tag{23}$$

Focusing on those iterations in which incumbents change (as in the index set K above), we note that $\hat{x}^{k_{\ell-1}} = \hat{x}^{k_\ell-1}$. Combining (20a), (22) and (23), we conclude that (with probability 1)

$$0 = \lim_{\tau\to\infty}\gamma_\tau = \lim_{\tau\to\infty}\frac{1}{\tau}\sum_{\ell=1}^{\tau}\big(f_{k_\ell-1}(\hat{x}^{k_\ell}) - f_{k_\ell-1}(\hat{x}^{k_{\ell-1}})\big) \tag{24a}$$

$$\leq -\frac{1}{\tau}\sum_{\ell=1}^{\tau}\|\hat{x}^{k_\ell} - \hat{x}^{k_{\ell-1}}\|^2 \leq 0, \quad k_\ell \in K. \tag{24b}$$

Therefore we conclude that the sample average change between all incumbent solutions vanishes (wp1).

Due to Proposition 1(b), there exists a subsequence indexed by K_* such that $\{\hat{x}^{k_\ell}\}_{k_\ell \in K_*} \to \hat{x}$, an optimal solution with probability one. Hence, there exists a positive integer $L_1 < \infty$ such that $\|\hat{x}^{k_\ell} - \hat{x}\| < \frac{\epsilon}{2\mu}$ for $k_\ell \in K_*$ and $\ell > L_1$, and μ is the diameter of the ball containing the compact set X. Define $\bar{K}_* = K \setminus K_*$. If the claim of the theorem is false, then there is a positive probability that $\bar{K}_*$ is an infinite set of iterations for which $0 < \sqrt{2\epsilon} < \|\hat{x}^{k_\ell} - \hat{x}\| < \mu$ for $k_\ell \in \bar{K}_*$ where $\ell > L_2$, a positive integer. Denote $L = \max\{L_1, L_2\}$. Now note that the summation in (24b) can be written as

$$\begin{aligned}
&\frac{1}{\tau}\sum_{\ell=1}^{\tau}\|\hat{x}^{k_\ell} - \hat{x}^{k_{\ell-1}}\|^2 \\
&= \frac{1}{\tau}\Bigg[\sum_{\substack{\ell=1 \\ (k_\ell,\ k_{\ell-1})\in K_*}}^{\tau} \|\hat{x}^{k_\ell} - \hat{x}^{k_{\ell-1}}\|^2 + \sum_{\substack{\ell=1 \\ (k_\ell,\ k_{\ell-1})\in \bar{K}_*}}^{\tau} \|\hat{x}^{k_\ell} - \hat{x}^{k_{\ell-1}}\|^2 \\
&+ \sum_{\substack{\ell=1 \\ k_\ell\in K_*,\ k_{\ell-1}\in \bar{K}_*}}^{\tau} \|\hat{x}^{k_\ell} - \hat{x}^{k_{\ell-1}}\|^2 + \sum_{\substack{\ell=1 \\ k_\ell\in \bar{K}_*,\ k_{\ell-1}\in K_*}}^{\tau} \|\hat{x}^{k_\ell} - \hat{x}^{k_{\ell-1}}\|^2\Bigg] \qquad (25a) \\
&\geq \frac{1}{\tau}\Bigg[\sum_{\substack{\ell>L \\ k_\ell\in K_*,\ k_{\ell-1}\in \bar{K}_*}}^{\tau} \|(\hat{x}^{k_\ell} - \hat{x}) - (\hat{x}^{k_{\ell-1}} - \hat{x})\|^2 + \sum_{\substack{\ell>L \\ k_\ell\in \bar{K}_*,\ k_{\ell-1}\in K_*}}^{\tau} \|(\hat{x}^{k_\ell} - \hat{x}) - (\hat{x}^{k_{\ell-1}} - \hat{x})\|^2\Bigg] \qquad (25b) \\
&\geq \frac{1}{\tau}\Bigg[\sum_{\substack{\ell>L \\ k_\ell\in K_*, k_{\ell-1}\in \bar{K}_*}}^{\tau} \left(\|\hat{x}^{k_\ell} - \hat{x}\|^2 + \|\hat{x}^{k_{\ell-1}} - \hat{x}\|^2 - 2\|\hat{x}^{k_\ell} - \hat{x}\|\|\hat{x}^{k_{\ell-1}} - \hat{x}\|\right) \\
&+ \sum_{\substack{\ell>L \\ k_\ell\in \bar{K}_*, k_{\ell-1}\in K_*}}^{\tau} \left(\|\hat{x}^{k_\ell} - \hat{x}\|^2 + \|\hat{x}^{k_{\ell-1}} - \hat{x}\|^2 - 2\|\hat{x}^{k_\ell} - \hat{x}\|\|\hat{x}^{k_{\ell-1}} - \hat{x}\|\right)\Bigg] \qquad (25c) \\
&\geq \frac{1}{\tau}\Bigg[\sum_{\substack{\ell>L \\ k_\ell\in K_*,\ k_{\ell-1}\in \bar{K}_*}}^{\tau} \epsilon + \sum_{\substack{\ell>L \\ k_\ell\in \bar{K}_*,\ k_{\ell-1}\in K_*}}^{\tau} \epsilon\Bigg]. \qquad (25d)
\end{aligned}$$

To make the above arguments completely transparent, observe that in going from (25a) to (25b), we dropped some non-negative terms, and added/subtracted $\hat{x}$; from (25b) to

(25c), we applied Cauchy-Schwartz inequality; from (25c) to (25d), we used the bounds on the appropriate distance between points of K_* and $\bar{K}_*$.

From (25), we have

$$\frac{1}{\tau}\sum_{\ell=1}^{\tau}\|\hat{x}^{k_\ell}-\hat{x}^{k_{\ell-1}}\|^2 \geq \frac{C_\tau}{\tau}\epsilon \text{ where } C_\tau \in (0,\tau-L], \tag{26}$$

where C_τ reflects the number of crossovers between K_* and $\bar{K}_*$. If the probability of crossovers is nonzero, then $\lim_{\tau\to\infty}\frac{C_\tau}{\tau}$ can not be zero wp1. Hence, the left hand side of (26) is strictly greater than zero wp1, contradicting (24). Therefore, the supposition that the set $\bar{K}_*$ has positive probability is untenable, which concludes the proof. ■

Proof of Theorem 2

a) To show that $\mathbf{x}^c$ solves (17), note that the problem in (16) is a convex program and its optimality conditions imply that

$$0 \in \frac{1}{M}\sum_{m=1}^{M}\partial f^m(\mathbf{x}^c) + \mathcal{N}_X(\mathbf{x}^c) + \bar{\rho}(\mathbf{x}^c - \frac{1}{M}\sum_{m=1}^{M}\mathbf{x}^m) \tag{27}$$

where $\mathcal{N}_X(\mathbf{x}^c)$ denotes the normal cone at $\mathbf{x}^c$. If $\mathbf{x}^c = \bar{\mathbf{x}}$, then the above equation reduces to the optimality conditions of (17).

b) In order to show (18), note that

$$f(\mathbf{x}^c) = f(\bar{\mathbf{x}}) \leq \frac{1}{M}\sum_{m=1}^{M} f(\mathbf{x}^m) \tag{28a}$$

$$= \frac{1}{M}\sum_{m=1}^{M} f^m(\mathbf{x}^m) + \frac{1}{M}\sum_{m=1}^{M}(f(\mathbf{x}^m) - f^m(\mathbf{x}^m)) \tag{28b}$$

$$\leq \frac{1}{M}\sum_{m=1}^{M} f^m(\mathbf{x}^c) + \frac{1}{M}\sum_{m=1}^{M}(f(\mathbf{x}^m) - f^m(\mathbf{x}^m)) \tag{28c}$$

Here (28a) is due to the convexity of f, and (28c) is due to $f^m(\mathbf{x}^m) \leq f^m(\mathbf{x}^c)$. As indicated in Remark 2, the SD algorithm and CLT ensure that the error terms $[f(\mathbf{x}^m) - f^m(\mathbf{x}^m)]$ are bounded by an asymptotically distributed normal random variable: $\mathbf{N}(0, \sigma_m^2/N)$. Due to assumptions 1 and 2, there exists $\hat{\sigma} < \infty$ such that $\sigma_m \leq \hat{\sigma}$. Moreover, the replications are also independent. Hence the average of M errors terms $\frac{1}{M}\sum_m(f(\mathbf{x}^m) - f^m(\mathbf{x}^m))$ is bounded by $O_p((NM)^{-\frac{1}{2}})$. Hence, in combination with the definition of $\bar{F}_M$ in (17), equation (28c) and the above arguments imply (18).■

Proof of Theorem 3

Our path to this result starts by first adopting a stochastic pointwise convergence argument (see a) below), and then extending that argument to the case of stochastic uniform convergence (see b) below).

a) As in the body of the paper, m denotes a replication, and f_ϵ^m denotes the asymptotic approximations generated during replication m. The notation $f_\epsilon^m(\mathbf{x}^m) \to f^*$ (wp1) will mean $\lim_{\epsilon\to 0} P(|f_\epsilon^m(\mathbf{x}^m) - f^*| < t) = 1,\ \forall t > 0$. Similarly, $f_\epsilon^m(\mathbf{x}) \to \bar{f}^m \geq f^*$ (wp1) will mean $\lim_{\epsilon\to 0} P(f_\epsilon^m(\mathbf{x}) > f^* - t) = 1,\ \forall t > 0$. In the following proof, we will use the same convention for the functions $\bar{F}_{\epsilon,M}$.

Because $\epsilon \to 0$, we have $K_m(\epsilon) \to \infty$, and $N \to \infty$, the SASA functions provide asymptotic lower bounds, and therefore $\bar{F}_{\epsilon,M}(\mathbf{x}^c) \to \bar{F}_M(\mathbf{x}^c) \leq f^*$ (wp1). From Theorem 1, $\mathbf{x}^m$ is the unique solution of the sequence generated in replication m and $f_\epsilon^m(\mathbf{x}^m) \to f^* \leq f^m(\mathbf{x}^c)$ with probability one. Since $\bar{F}_M = \frac{1}{M}\sum_{m=1}^M f^m$, it follows that $\bar{F}_{\epsilon,M}(\mathbf{x}^c) \to \bar{F}_M(\mathbf{x}^c) \geq f^*$ (wp1). Hence pointwise convergence holds.

b) In order to show stochastic uniform convergence, we appeal to Rubin (1956) for which we need to verify that (i) X is a compact set, (ii) iid sampling, (iii) the objective function $c^\top x + h(x, \omega)$ has a uniform Lipschitz constant due to the relatively complete recourse assumption hold. Assumptions 1-4 in section 3 ensures that the above conditions hold. Hence uniform convergence holds, and therefore $\bar{F}_{\epsilon,M}(\mathbf{x}^c(\epsilon)) \to \bar{F}_M(\mathbf{x}^c) \to f^*$ (wp1). ■

Appendix B: The SSN Instance - Back to the Future

The subtitle of this Appendix ("Back to the Future") is intended to reflect the fact that there are certain instances from the past which need to be fully explored before we can claim to have a thorough understanding of the two-stage SLP paradigm. SSN is one such instance. The subtitle conveys the fact that we went back two decades to revive our interest in the SSN model, and the results of this paper have shown us how to solve such problems fairly accurately, and without requiring computing platforms that are beyond the ordinary.

SSN is a network planning model which seeks to size links (i.e. choose link capacities) before demand is known with certainty. This model is limited to adding capacity to links that have been identified for expansion, and in this sense, it does not recommend what additional links may be useful. The model does not consider the possibility of adding new links, which would require stochastic integer programming. Thus given a collection of links, we wish to choose additional capacities under a budget constraint, while ensuring that the expected unserved demand is minimized. The model uses the following notation.
First stage decision variables:

$x_j \equiv$ the amount of capacity to be added to the j^{th} link.

First stage data/parameters:

$n \equiv$ the number of links that are to be considered for capacity expansion.

$b \equiv$ the total capacity budget for the entire network.

$\tilde{\omega} \equiv$ the m dimensional random variable that represents demands associated with the m point to point pairs served by the network.

The first stage problem can be summarized as follows:

$$\text{Min} \quad E[h(x,\tilde{\omega})] \tag{29a}$$

$$\text{s.t.} \quad \sum_j x_j \le b \tag{29b}$$

$$x \ge 0 \tag{29c}$$

The function $h(x,\tilde{\omega})$ is a random variable representing unserved demand. It is a function of the capacity expansion plan x and a random vector $\tilde{\omega}$. For each outcome ω following model is used to estimate unserved demand, and the generic name referring to this piece of the model is the second stage problem. The second stage decision variables are

$f_{ir} \equiv$ the number of calls associated with point to point pair i that are served via route $r \in R(i)$.

$s_i \equiv$ the number of unserved requests associated with point to point pair i.

Second stage data/parameters:

$m \equiv$ the number of point to point pairs served by the network.

$R(i) \equiv$ the set of routes that can be used to connect point to point pair i.

$A_{ir} \equiv$ is an incidence vector in $\mathbb{R}^n$ whose j^{th} element is 1 if link j belongs to route $r \in R(i)$, and is 0 otherwise.

$e \equiv$ is a vector in $\mathbb{R}^n$ of current link capacities.

And the second stage problem is the following:

$$h(x,\omega) \quad = \quad \text{Min} \quad \sum_{i=1}^{m} s_i \tag{30a}$$

$$\text{s.t.} \quad \sum_i \sum_{r \in R(i)} A_{ir} f_{ir} \leq x + e \tag{30b}$$

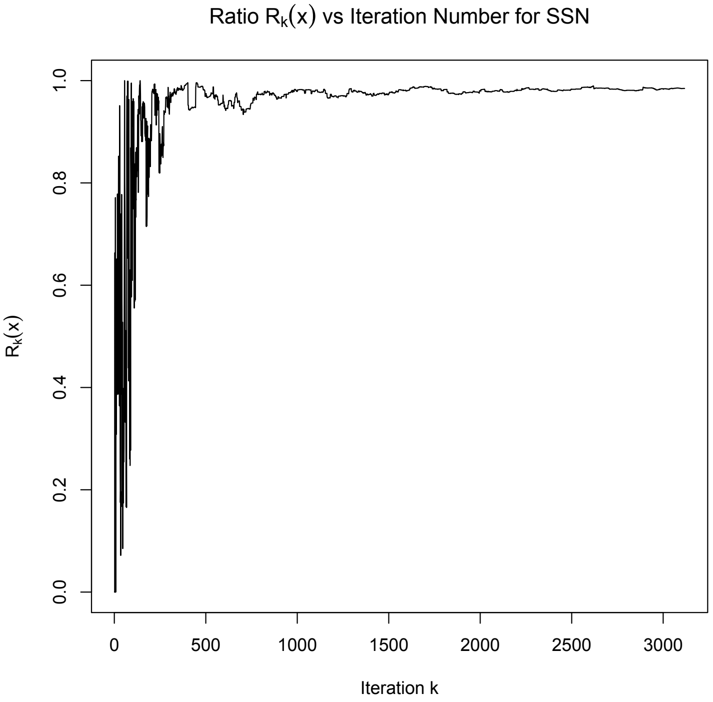
$$\sum_{r \in R(i)} f_{ir} + s_i = \omega_i \quad i = 1, ..., m \tag{30c}$$

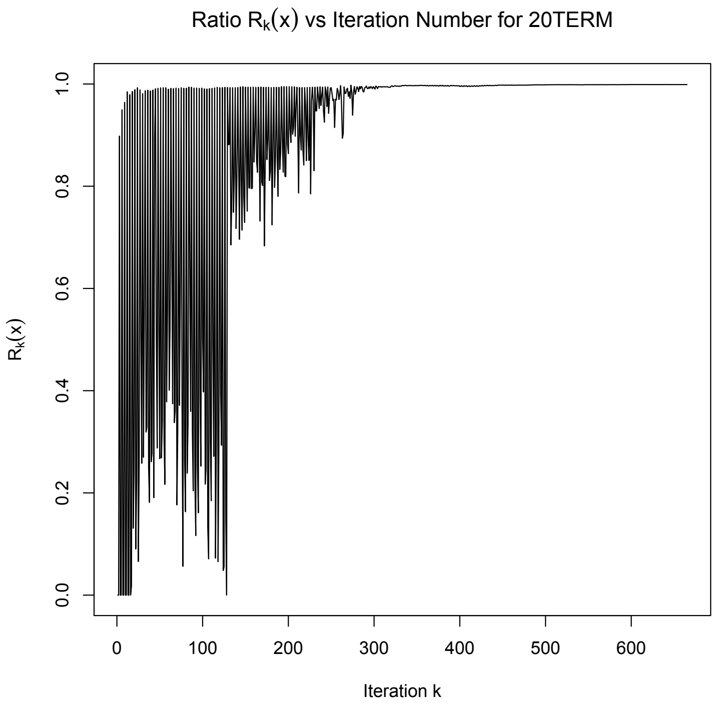
$$f, s \geq 0 \tag{30d}$$

Constraints (29b) limits the expansion plan under a total capacity level b while constraints (30b) ensure that routing is achieved without violating any link capacity. Constraints (30c) are flow balance equations of the network.

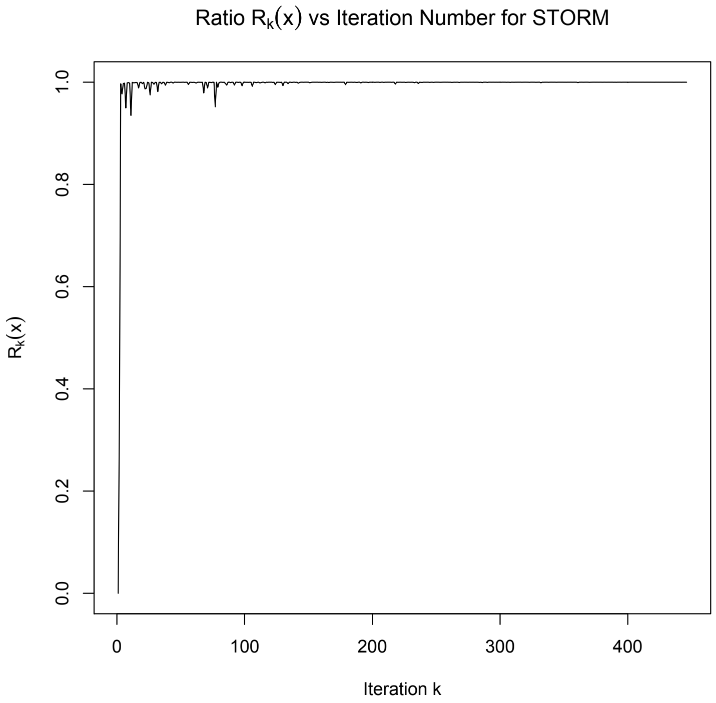
Appendix C: Figures of Shadow Price Stability

The following figures show that for the three "industrial-strength" instances included in this paper, the VF approximations are poor representations of the expected recourse function in early iterations. As the SD algorithm proceeds, the ratios R_k approach 1 (see Shadow Price Stability in subsection 3.2.1), indicating vastly improved approximations in the vicinity of candidate and incumbent solutions, and they are all steady by the time that the In-Sample rule accepts an incumbent. The number of data points in these graphs exceed the number of iterations because SD updates incumbent objective estimates periodically, and the ratios are calculated for those updates too.





Ratio $R_k(x)$ vs Iteration Number for 20TERM



Ratio $R_k(x)$ vs Iteration Number for STORM

References

Bassok, Y., R. Anupindi, R. Akella. 1999. Single-period multiproduct inventory models with substitution. *Operations Research* **47**(4) 632–642.

Bayraksan, G., D. P. Morton. 2011. A sequential sampling procedure for stochastic programming. *Operations Research* **59**(4) 898–913.

Birge, J. R., F. V. Louveaux. 1997. *Introduction to Stochastic Programming*. Springer.

Chen, X., M. Sim, P. Sun, J. Zhang. 2008. A linear decision-based approximation approach to stochastic programming. *Operations Research* **56**(2) 344–357.

Efron, B. 1979. Bootstrap methods: another look at the jackknife. *The Annals of Statistics* 1–26.

Fu, M.C. 2002. Optimization for simulation: theory vs. practice. *INFORMS J. on Computing* **14**(3) 192–215.

Gangammanavar, H, Suvrajeet Sen, V M Zavala. 2015. Stochastic Optimization of Sub-Hourly Economic Dispatch With Wind Energy. *IEEE Transactions on Power Systems* **PP**(99) 1–11. doi:10.1109/TPWRS.2015.2410301. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7065337`.

Glynn, P. W., G. Infanger. 2013. Simulation-based confidence bounds for two-stage stochastic programs. *Mathematical Programming* **138**(1-2) 15–42.

Grasso, A., W. LaPlante. 2011. Enhancing adaptability of US military forces. Washington, DC: Office of the Under Secretary of Defense for Acquisition, Technology and Logistics.

Hastie, T., R. Tibshirani, J. Friedman. 2009. *The Elements of Statistical Learning*, vol. 2. Springer.

Herer, Y. T., M. Tzur, E. Yücesan. 2006. The multilocation transshipment problem. *IIE Transactions* **38**(3) 185–200.

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

Higle, J. L., S. Sen. 1991b. Statistical verification of optimality conditions for stochastic programs with recourse. *Annals of Operations Research* **30**(1) 215–239.

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

Higle, J. L., S. Sen. 1996a. Duality and statistical tests of optimality for two stage stochastic programs. *Mathematical Programming* **75**(2) 257–275.

Higle, J. L., S. Sen. 1996b. *Stochastic decomposition: a statistical method for large scale stochastic linear programming*, vol. 8. Springer.

Higle, J. L., S. Sen. 1999. Statistical approximations for stochastic linear programming problems. *Annals of Operations Research* **85** 173–193.

Holdren, J., E. Lander, H. Varmus. 2010. Report to the president and congress designing a digital future: Federally funded research and development in networking and information technology. *Executive Office of the President and President's Council of Advisors on Science and Technology* .

Infanger, G. 1999. Private communication.

Kim, S., R. Pasupathy, S. G. Henderson. 2015. A guide to sample average approximation. *Handbook of Simulation Optimization*. Springer, 207–243.

Kleywegt, A. J., A. Shapiro, Homem de Mello. 2002. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization* **12**(2) 479–502.

Kuhn, D., W. Wiesemann, A. Georghiou. 2011. Primal and dual linear decision rules in stochastic and robust optimization. *Mathematical Programming* **130**(1) 177–209.

Linderoth, J., A. Shapiro, S. Wright. 2006. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research* **142**(1) 215–241.

Linderoth, J., S. Wright. 2003. Decomposition algorithms for stochastic programming on a computational grid. *Computational Optimization and Applications* **24**(2-3) 207–250.

Louveaux, F., Y. Smeers. 1988. Optimal investments for electricity generation: A stochastic model and a test problem. *Numerical Techniques for Stochastic Optimization* 33–64.

Mak, W., D. P. Morton, R. K. Wood. 1999. Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters* **24**(1) 47–56.

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

Nemirovski, A., A. Juditsky, G. Lan, A. Shapiro. 2009. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization* **19**(4) 1574–1609.

Nesterov, Y., J. Vial. 2008. Confidence level solutions for stochastic programming. *Automatica* **44**(6) 1559–1568.

Oliveira, W., C. Sagastizábal, S. Scheimberg. 2011. Inexact bundle methods for two-stage stochastic programming. *SIAM Journal on Optimization* **21**(2) 517–544.

Pasupathy, R. 2010. On choosing parameters in retrospective-approximation algorithms for stochastic root finding and simulation optimization. *Operations Research* **58**(4-Part-1) 889–901.

Polyak, B T, A B Juditsky. 1992. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization* .

Powell, W. B. 2007. *Approximate Dynamic Programming: Solving the curses of dimensionality*, vol. 703. John Wiley & Sons.

Römisch, W. 2003. Stability of stochastic programming problems. *Handbooks in Operation Research and Management Science* **10** 483–554.

Royset, J. O., R. Szechtman. 2013. Optimal budget allocation for sample average approximation. *Operations Research* **61** 762–776.

Rubin, H. 1956. Uniform-Convergence of Random Functions with Applications to Statistics. *Annals of Mathematical Statistics* **27**(1) 200–203.

Sen, S., R. D. Doverspike, S. Cosares. 1994. Network planning with random demand. *Telecommunication Systems* **3**(1) 11–30.

Shapiro, A., D. Dentcheva, A. Ruszczyński. 2009. *Lectures on stochastic programming: modeling and theory*, vol. 9. SIAM.

Shapiro, A., T. Homem de Mello. 1998. A simulation-based approach to two-stage stochastic programming with recourse. *Mathematical Programming* **81**(3) 301–325.

Shapiro, A., T. Homem de Mello, J. Kim. 2002. Conditioning of convex piecewise linear stochastic programs. *Mathematical Programming* **94**(1) 1–19.

Singh, K. 1981. On the asymptotic accuracy of Efron's bootstrap. *The Annals of Statistics* 1187–1195.

Vogel, S. 2008. Universal confidence sets for solutions of optimization problems. *SIAM Journal on Optimization* **19**(3) 1467–1488.

Zverovich, V., C. I. Fábián, E. F. Ellison, G. Mitra. 2012. A computational study of a solver system for processing two-stage stochastic LPs with enhanced Benders' decomposition. *Mathematical Programming Computation* **4**(3) 211–238.