Approximation Algorithms for 2-Stage Stochastic Optimization Problems

Chaitanya Swamy[1,⋆] and David B. Shmoys[2,⋆⋆]

[1] Department of Combinatorics & Optimization, University of Waterloo, Waterloo, ON N2L 3G1
cswamy@math.uwaterloo.ca
[2] School of ORIE and Department of Computer Science, Cornell University Ithaca, NY 14853
shmoys@cs.cornell.edu

Abstract. Stochastic optimization is a leading approach to model optimization problems in which there is uncertainty in the input data, whether from measurement noise or an inability to know the future. In this survey, we outline some recent progress in the design of polynomial-time algorithms with performance guarantees on the quality of the solutions found for an important class of stochastic programming problems — 2-stage problems with recourse. In particular, we show that for a number of concrete problems, algorithmic approaches that have been applied for their deterministic analogues are also effective in this more challenging domain. More specifically, this work highlights the role of tools from linear programming, rounding techniques, primal-dual algorithms, and the role of randomization more generally.

1 Introduction

Uncertainty is a facet of many decision environments and might arise due to various reasons, such as unpredictable information revealed in the future, or inherent fluctuations caused by noise. Stochastic optimization provides a means to handle uncertainty by modeling it by a probability distribution over possible realizations of the actual data called scenarios. The field of stochastic optimization or stochastic programming, has its roots in the work of Dantzig [4] and Beale [1] in the 1950s, and has increasingly found application in a wide variety of areas, including transportation models, logistics, financial instruments, and network design.

An important and widely used model in stochastic programming is the *2-stage recourse model*: first, given only distributional information about (some of) the data, one commits on initial (*first-stage*) actions. Then, once the actual data is realized according to the distribution, further *recourse actions* can be taken (in the *second stage*) to augment the earlier solution and satisfy the revealed requirements. The aim is to choose the initial actions so as to minimize the

⋆ Research supported partially by NSERC grant 327620-06.
⋆⋆ Research supported partially by NSF grant CCR-0430682.

expected total cost incurred. Typically the recourse actions entail making decisions in rapid reaction to the observed scenario, and are therefore more costly than decisions made ahead of time. Thus there is a trade-off between committing initially having only imprecise information while incurring a lower cost, and deferring decisions to the second–stage when we know the input precisely but the costs are higher. Many applications come under this setting, and much of the textbook of Birge and Louveaux [2] is devoted to models and algorithms for this class of problems.

A commonly cited example involves a setting where a company has to decide where to set up facilities to serve client demands. Typically the demand pattern is not known precisely at the outset, but one might be able to obtain, through simulation models or surveys, statistical information about the demands. This motivates the following 2-step decision process: in the first-stage, given only distributional information about the demands (and deterministic data for the facility opening costs), one must decide which facilities to open initially. Once the actual input (the client demands) is realized according to this distribution, we can extend the solution by opening more facilities, incurring a recourse cost, and we have to assign the realized demands to open facilities. This is the *2-stage stochastic uncapacitated facility location* problem. The recourse costs are usually higher than the original ones (because opening a facility later would involve deploying resources with a small lead time), could be different for the different facilities, and could even depend on the realized scenario.

The formal model. The 2-stage recourse model can be formalized as follows: we are given a probability distribution over possible realizations of the data called scenarios and we construct a solution in two stages. First, we may take some decisions to construct an anticipatory part of the solution, x, incurring a cost of $c(x)$. Then a scenario A is realized according to the distribution, and in the second-stage we may augment the initial decisions by taking recourse actions y_A, (if necessary) incurring a certain cost $f_A(x, y_A)$. The goal is to choose the initial decisions so as to minimize the expected total cost, $c(x) + \mathrm{E}_A\big[f_A(x, y_A)\big]$, where the expectation is taken over all scenarios according to the given probability distribution.

An important issue that we have left unspecified above is the question of how the scenario-distribution is represented. One simple approach is to assume that we are given as part of the input description a list that explicitly enumerates each scenario (occurring with non-zero probability) and its probability of occurrence. However, this causes a significant blow-up in the input size, since the distribution can easily have support size that is exponential in the other input parameters, that is, the non-stochastic portion of the input; for example, in stochastic facility location, consider the case where the demand of each client is set independently. Thus, to ensure that a "polynomial-time" algorithm in this model has running time polynomial in the other input parameters, one must restrict oneself to distributions with a polynomial-size support, which is a severe restriction; we shall call this the *polynomial-scenario model* to reflect this fact. The distribution mentioned above is captured by the *independent-activation model* introduced by Immorlica et al. [11], where the scenario-distribution is a product of independent distributions (described in the input). Typically, there is an underlying set of elements (clients) and a scenario is generated by independent choices (setting the demands) made for each element. Independent distributions allow one to succinctly specify a class of distributions with exponentially many scenarios, and have been used in the Computer Science community to model uncertainty in various settings [13,18,5]. However, many of the underlying stochastic applications often involve *correlated data* (e.g., in stochastic facility location the client demands are expected to be correlated due to economic and/or geographic factors), which the independent-activation model clearly does not capture. A more general way of specifying the distribution is the *black-box model*, where the distribution is specified *only* via a procedure (a "black box") that one can use to independently sample scenarios from the distribution. In this model, each procedure call is treated as an elementary operation, and the running time of an algorithm is measured in terms of the number of procedure calls. The black-box model incorporates the desirable aspects of both the previous models: it allows one to specify distributions with exponentially many scenarios and correlation in a compact way that makes it reasonable to talk about polynomial-time algorithms.

Stochastic optimization problems are often computationally quite difficult, and often more difficult than their deterministic counterparts, both from the viewpoint of complexity theory, as well as from a practical perspective. In many settings the computational difficulty stems from the fact that the distribution might assign a non-zero probability to an exponential number of scenarios, leading to considerable increase in the problem complexity, a phenomenon often called the "curse of dimensionality." Thus, many stochastic problems that are easy to solve in the polynomial-scenario model due to the expansive input encoding become *NP*-hard in the black-box model. For example, *stochastic linear programming* problems (i.e., stochastic problems that can be formulated as linear programs) are polynomial-time solvable in the polynomial-scenario model but become #*P*-hard in the black-box model [8]. In other settings, even with polynomially many scenarios, the stochastic problem gives rise to a more complex problem than its deterministic counterpart and is *NP*-hard, whereas the deterministic problem is solvable in polynomial time.

In this survey, we focus on the design of approximation algorithms for stochastic optimization problems. Throughout, we use a *ρ-approximation algorithm* to denote a polynomial-time algorithm that always returns a feasible solution with objective function value within a factor ρ of the optimum; ρ is called the approximation ratio or performance guarantee of the algorithm.

There is an abundance of literature in the stochastic programming community that deals with computational aspects of solving 2-stage stochastic programs, especially 2-stage linear programs (LPs), which we shall not cover here; the reader is referred to [2,22] for more information. Many of these methods are only suitable in the polynomial-scenario model and cannot handle the burden of an exponential number of scenarios. One appealing approach in the black-box model is to sample a certain number of times from the scenario-distribution, estimate the probability of a scenario by its frequency in the sampled set, and solve the 2-stage problem determined by this approximate distribution. This is known as the *sample average approximation* (SAA) method. The SAA method is a widely used heuristic in practice and has been empirically shown to work well in various settings (see, e.g., [15,28]). The main question here is: how many samples does one need to ensure that an optimal solution to the *sample-average problem* is a near-optimal solution to the original problem (with high probability)? While there are results that prove asymptotic convergence to the optimal solution (to the original problem) in the limit *as the number of samples goes to infinity*, fewer results are known about the rate of convergence, or equivalently, about *worst-case bounds* on the sample size required to obtain a near-optimal solution. Ideally one would like to show that a polynomial number of samples always suffice. Such a result would show that the SAA method gives a reduction from the black-box problem to a polynomial-scenario problem, thereby reducing the complexity of the stochastic problem, while losing a factor in the approximation guarantee. In particular, this would immediately give an approximation algorithm for stochastic linear programming problems in the black-box model. The work that most closely considers the aspect of worst-case bounds is a paper of Kleywegt, Shapiro and Homem-De-Mello [14] (see also [23]). Kleywegt et al. prove a sample-size bound for 2-stage programs that is independent of the number of scenarios, but depends on the variance of a certain quantity (calculated using the scenario-distribution) which need not be polynomially bounded, even for very structured programs. We shall return to this question of proving polynomial sample-size bounds for the SAA method in Section 4.

There are other sampling-based approaches where instead of sampling just once initially, the algorithm used to solve the stochastic problem contains a sampling subroutine that is called whenever one needs to estimate some quantity, such as the function value or the gradient. Dyer, Kannan and Stougie [7] use such an approach for a stochastic maximization LP, where samples are used to estimate the objective function value at a given point. This yields a sample size that is only polynomial in the maximum value attained by any scenario (due to the high variance in the values of the different scenarios). Nesterov and Vial [20] employ stochastic subgradients, estimated via sampling, in a subgradient-descent algorithm, and require a sample size that is polynomial in the maximum variation in the objective function value in the feasible region.

The design and analysis of algorithms with provable worst-case guarantees for 2-stage stochastic integer programs is a relatively recent research direction. The first such result appears to be due to Dye, Stougie and Tomasgard [6] who give a constant-factor approximation algorithm for a resource provisioning problem in the polynomial-scenario model. Subsequently, a series of papers [21,11,10,25] appeared on this topic in the Computer Science literature, and showed that one can obtain guarantees for a variety of stochastic combinatorial optimization problems by adapting the techniques developed for the deterministic analogue. Gupta, Pál, Ravi and Sinha [10], who were the first to consider the black-box model (under a certain cost assumption), make such a connection explicit. Shmoys and

Swamy [25], who give algorithms in the black-box model with arbitrary costs, show an even more explicit correspondence. They showed that one could derive approximation algorithms for most of the problems considered in [21,11,10] by adopting a natural LP rounding approach that, in effect, converted an LP-based approximation guarantee for the deterministic analogue into a guarantee for the stochastic generalization with a small loss in the approximation factor. Thus, if we can solve the stochastic LP (even approximately), which is a $\#P$-hard problem, then we will have essentially reduced the stochastic problem to its deterministic analogue.

This survey is organized as follows: in Section 2 we describe an approximation scheme for solving a large class of 2-stage stochastic LPs. In Section 3 we describe some techniques for devising approximation algorithms for stochastic integer programming problems. We focus mainly on the black-box model, but also sometimes consider the polynomial-scenario model; in Section 4 we consider the SAA method and establish a concrete connection between these two models.

2 Stochastic Linear Programs

We now describe the fully polynomial approximation scheme (FPAS) of Shmoys and Swamy [25] that can be applied to a rich class of 2-stage stochastic LPs. The algorithm returns a solution of value within $(1 + \kappa)$ times the optimum (with high probability), for any $\kappa > 0$, in time polynomial in the input size, $\frac{1}{\kappa}$, and a parameter λ, which is the maximum *ratio* between the second- and first-stage costs. As we show in Section 3, this provides us with a powerful and versatile tool for designing approximation algorithms for stochastic integer optimization problems in much the same way that linear programming has proved to be immensely useful in the design of approximation algorithms for deterministic optimization problems.

We shall consider a stochastic generalization of the set cover problem as an illustrative problem to explain the main ideas. In the *2-stage stochastic set cover* (SSC) problem, we are given a ground set U of n elements, a collection of subsets of U, $S_1, \ldots, S_m$, and a distribution over subsets of U that specifies the target set of elements to cover. In stage I, we can pick some sets paying a cost of w^{I}_S for each set S. Then, a scenario materializes which specifies a target set $A \subseteq U$ of elements to be covered and the costs $\{w^A_S\}$ of picking sets in that scenario, and one can pick additional sets to ensure that A is contained in the union of the sets selected in the two stages. The aim is to minimize the expected cost of the sets picked. Denoting the probability of scenario A by p_A (which we do not know explicitly, and could be 0), we can formulate the problem as an integer program and relax the integrality constraints to obtain the following linear program: minimize

$$\Big\{\sum_{S} w^{\mathrm{I}}_S x_S + \sum_{A \subseteq U, S} p_A w^A_S r_{A,S} : \sum_{S: e \in S} (x_S + r_{A,S}) \geq 1 \quad \forall A, e \in A; \quad x_S, r_{A,S} \geq 0 \quad \forall A, S.\Big\} \tag{SSC-P1}$$

Variable x_S indicates whether set S is chosen in stage I, and $r_{A,S}$ indicates if set S is chosen in scenario A. The constraint says that in every scenario A, every element in that scenario has to be covered by a set chosen either in stage I or in stage II. Notice that (SSC-P1) is an LP with an exponential number of variables *and* constraints, and it seems difficult to efficiently compute an (near-) optimal solution to (SSC-P1), since even writing out a solution can take exponential space (and time). However, if we fix the first-stage decisions, i.e., the x_S variables, then the scenarios become separable, so we can reformulate (SSC-P1) as follows: minimize

$$h(x) \;:=\; \sum_S w^{\mathrm{I}}_S x_S + \sum_{A\subseteq U} p_A f_A(x) \quad \text{subject to} \quad 0 \le x_S \le 1 \quad \forall S, \tag{SSC-P2}$$

$$\text{where} \tag{1}$$

$$f_A(x) \;:=\; \min\Big\{\sum_S w^A_S r_{A,S} : \sum_{S:e\in S} r_{A,S} \ge 1 - \sum_{S:e\in S} x_S \,\forall e \in A; \quad r_{A,S} \ge 0 \;\forall S.\Big\}$$

Here the second-stage decisions only appear in the minimization problem $f_A(x)$, which denotes the recourse problem that one needs to solve for scenario A. It is easy to show that (SSC-P2) is equivalent to (SSC-P1), and that its objective function is convex. Although we now have a compact *convex program*, the complexity of the problem resurfaces as follows: in general, it is $\#P$-hard to even evaluate the objective function $h(.)$ at a given point [8]. Nevertheless, we can leverage convexity and adapt the *ellipsoid method* to solve (SSC-P2).

In the ellipsoid method, we start by containing the feasible region within a ball and then generate a sequence of ellipsoids, each of successively smaller volume. In each iteration, one examines the center of the current ellipsoid and obtains a specific half-space defined by a hyperplane passing through the current ellipsoid center. If the current ellipsoid center is infeasible, then one uses a violated inequality as the hyperplane, otherwise, one uses an *objective function cut* to eliminate (some or all) feasible points whose objective function value is no better than the current center, and thus make progress. A new ellipsoid is then generated by finding the minimum-volume ellipsoid containing the half-ellipsoid obtained by the intersection of the current one with this half-space. Continuing in this way, using the fact that the volume of the successive ellipsoids decreases by a significant factor, one can show that after a polynomial number of iterations, the feasible point generated with the best objective function value is a near-optimal solution.

Let $\mathcal{P} = \mathcal{P}_0$ denote the polytope $\big\{x \in \mathbb{R}^m : 0 \le x_S \le 1 \text{ for all } S\big\}$, and x_i be the current iterate. Define $\lambda = \max(1, \max_{A,S} w^A_S/w^{\mathrm{I}}_S)$, which we assume is known. It is trivial to determine if x_i is feasible, so we only need to describe how to obtain an objective function cut. One option is to simply add the constraint $h(x) \le h(x_i)$, which is not a "linear" cut, but would preserve the convexity of the feasible region. But then in subsequent iterations, without the ability to evaluate (or even estimate) $h(.)$ at a given point, we would not even be able to decide if the current point is feasible (or even almost-feasible), which poses a formidable difficulty. Alternatively, one could use cuts generated by a *subgradient*, which is the analogue of gradient for a non-differentiable function: $d \in \mathbb{R}^m$ is a subgradient of a function $g : \mathbb{R}^m \mapsto \mathbb{R}$ at point u, if $g(v) - g(u) \geq d \cdot (v - u)$ for every $v \in \mathbb{R}^m$. If d_i is a subgradient at point x_i, one can add the *subgradient cut* $d_i \cdot (x - x_i) \leq 0$ and proceed with the (smaller) polytope $\mathcal{P}_{i+1} = \{x \in \mathcal{P}_i : d_i \cdot (x - x_i) \leq 0\}$. Unfortunately, even computing a subgradient is hard to do in polynomial time for the objective functions that arise in stochastic programs. We circumvent this obstacle by using an *approximate subgradient*:

Definition 1. *We say that $\hat{d} \in \mathbb{R}^m$ is an $(\omega, \mathcal{D})$-subgradient of a function $g : \mathbb{R}^m \mapsto \mathbb{R}$ at point $u \in \mathcal{D}$, if for every $v \in \mathcal{D}$, we have $g(v) - g(u) \geq \hat{d} \cdot (v - u) - \omega g(u)$.*

We abbreviate $(\omega, \mathcal{P})$-subgradient to ω-subgradient. An extremely useful property of ω-subgradients is that one can compute them efficiently by sampling. If $\hat{d}_i$ is an ω-subgradient at x_i, one can add the inequality $\hat{d}_i \cdot (x - x_i) \leq 0$ and obtain the polytope $\mathcal{P}_{i+1} = \{x \in \mathcal{P}_i : \hat{d}_i \cdot (x - x_i) \leq 0\}$. Since we use an approximate subgradient, this might discard points with $h(.)$ value less than $h(x_i)$. But for any point $y \in \mathcal{P}_i \setminus \mathcal{P}_{i+1}$, we have that $h(y) \geq (1 - \omega)h(x_i)$, so no such point has $h(.)$ value much smaller than $h(x_i)$. Continuing this way, we obtain a polynomial number of points $x_0, x_1, \ldots, x_k$ such that $x_i \in \mathcal{P}_i \subseteq \mathcal{P}_{i-1}$ for each i, and the volume of the ellipsoid centered at x_k containing $\mathcal{P}_k$, and hence that of $\mathcal{P}_k$ is small. Now if $h(.)$ has a bounded Lipschitz constant (h has Lipschitz constant at most K if $|h(v) - h(u)| \leq \|v - u\|_2 \ \forall u, v \in \mathbb{R}^m$) then one can show that $\min_i h(x_i)$ is close to the optimal value OPT with high probability. The entire procedure is summarized below.

FindOpt(γ, ϵ) [Returns $\bar{x} \in \mathcal{P}$ such that $h(\bar{x}) \leq OPT/(1 - \gamma) + \epsilon$. Assume $\gamma \leq \frac{1}{2}$. K is the Lipschitz constant.]

```
O1. Set $k \leftarrow 0$, $y_0 \leftarrow \mathbf{0}$, $N \leftarrow \lceil 2m^2 \ln(\frac{16KR^2}{V\epsilon}) \rceil$, $n \leftarrow N \log(\frac{8NKR}{\epsilon})$, and $\omega \leftarrow \gamma/2n$.
    Let $E_0 \leftarrow B(\mathbf{0}, R)$ and $\mathcal{P}_0 \leftarrow \mathcal{P}$.
O2. For $i = 0, \ldots, N$ do the following.
    a) If $y_i \in \mathcal{P}_k$, set $x_k \leftarrow y_i$. Let $\hat{d}_k$ be an $\omega$-subgradient of $h(.)$ at $x_k$. Let $H$ denote
       the half space $\{x \in \mathbb{R}^m : \hat{d}_k \cdot (x - x_k) \leq 0\}$. Set $\mathcal{P}_{k+1} \leftarrow \mathcal{P}_k \cap H$ and $k \leftarrow k + 1$.
    b) If $y_i \notin \mathcal{P}_k$, let $a \cdot x \leq b$ be a violated inequality, that is, $a \cdot y_i > b$, whereas
       $a \cdot x \leq b$ for all $x \in \mathcal{P}_k$. Let $H$ be the half space $\{x \in \mathbb{R}^m : a \cdot (x - y_i) \leq 0\}$.
    c) Set $E_{i+1}$ to be the ellipsoid of minimum volume containing the half-ellipsoid
       $E_i \cap H$.
O3. Set $k \leftarrow k - 1$. Return the point in $\{x_0, \ldots, x_k\}$ with minimum $h(.)$ value.
```

There are a few details needed to complete the algorithm description. First, since we cannot compute $h(x)$ we will not be able to compute the point $\arg\min_i h(x_i)$ in step O3. Instead, by using ω-subgradients we will find a point $\bar{x}$ in the convex hull of $x_0, \ldots, x_k$, such that $h(\bar{x})$ is close to $\min_i h(x_i)$. We repeatedly perform a bisection search on the line segment joining $\bar{x}$ (initialized to x_0) and x_i for $i = 1, \ldots, k$, using an ω-subgradient to infer which direction to move along the segment. Each time the search returns a point y such that $h(y)$ is close to $\min(h(\bar{x}), h(x_i))$, and we update $\bar{x}$ to y. Second, to convert the performance guarantee of procedure $\mathsf{FindOpt}$ into a purely multiplicative $(1+\kappa)$-guarantee, we need to obtain a lower bound on OPT (and set ϵ, γ accordingly). Under the mild assumption that the cost of every set S, in stage I and in every stage II scenario, is at least 1, one can do this by sampling initially $O(\lambda)$ times. Essentially, one can detect by sampling $O(\lambda)$ times, whether the probability that some scenario $A \neq \emptyset$ occurs is at least $\frac{1}{\lambda}$; if so, then $OPT \geq \frac{1}{\lambda}$, otherwise $x = \mathbf{0}$ is an optimal solution. Finally, we specify how to compute an ω-subgradient at a point $x \in \mathcal{P}$ efficiently. Let z^*_A be an optimal solution to the *dual* of $f_A(x)$.

Lemma 1. *(i) the vector d with components $d_S = \sum_A p_A(w^{\mathrm{I}}_S - \sum_{e \in A \cap S} z^*_{A,e})$ is a subgradient of $h(.)$ at x; (ii) for every scenario A and set S, $|w^{\mathrm{I}}_S - \sum_{e \in A \cap S} z^*_{A,e}| \leq \lambda w^{\mathrm{I}}_S$; and (iii) if $\hat{d} \in \mathbb{R}^m$ is such that $d_S - \omega w^{\mathrm{I}}_S \leq \hat{d}_S \leq d_S$ for every S, then $\hat{d}$ is an ω-subgradient of $h(.)$ at x.*

Parts (i) and (ii) of Lemma 1 show that each component of the subgradient vector is the expectation of a random variable (according to the scenario-distribution) with bounded variance. (Part (ii) also yields a bound on the Lipschitz constant of h.) So, with probability at least $1 - \delta$, one can estimate this expectation to within an additive error of ωw^{I}_S simultaneously for each S, using sample size $\mathsf{poly}\big(\text{input size}, \frac{\lambda}{\omega}, \ln(\frac{1}{\delta})\big)$. This yields an ω-subgradient, by part (iii) of Lemma 1. We compute an ω-subgradient at a polynomial number of points, with a polynomially small ω, so overall we get a sample size that is polynomial in the input size, λ, and $\frac{1}{\kappa}$. This sample-size bound is tight up to polynomial factors in the black-box model: one can construct examples where $\Omega(\lambda/\rho)$ samples are needed in the black-box model to obtain a performance guarantee of ρ [25], and the dependence on κ is also unavoidable due to the $\#P$-hardness result.

Shmoys and Swamy showed that the arguments above, especially Lemma 1, can be generalized to yield an approximation scheme for a broad class of 2-stage stochastic LPs which includes the fractional versions of a variety of stochastic combinatorial optimization problems such as (stochastic) covering problems (e.g., set cover, network design, multicut), facility location problems, multicommodity flow.

3 Stochastic Integer Programs

We now consider some stochastic combinatorial optimization problems, modeled as stochastic integer programs, and describe some methods that can be used to design approximation algorithms for these problems.

A general rounding technique. We first describe a simple, but powerful rounding framework due to [25], using stochastic set cover (SSC) as an illustrative example. Recall the relaxation (SSC-P2) for SSC. We will show that an LP-based approximation guarantee for the deterministic set cover (DSC) problem yields a corresponding guarantee for the stochastic problem. Given a DSC instance with a universe U of n elements, a family of subsets $S_1, \ldots, S_m$ with set S having weight w_S, consider the following LP relaxation of the integer problem of picking a minimum weight collection of sets to cover U.

$$OPT_{Det} := \min \sum_{S \in \mathcal{S}} w_S x_S \quad \text{subject to} \quad \sum_{S \in \mathcal{S}: e \in S} x_S \geq 1\ \forall e; \quad x_S \geq 0\ \forall S. \tag{SC-P}$$

Theorem 2. *Given an algorithm that for every DSC instance produces a solution of cost at most $\rho \cdot OPT_{Det}$, one can convert any solution x to* (SSC-P2) *to an integer solution of cost at most $2\rho \cdot h(x)$.*

Proof. Let r^*_A be an optimal solution to the recourse problem $f_A(x)$, so $f_A(x) = \sum_S w^A_S r^*_{A,S}$. Observe the following simple fact: an element e is covered to an extent of at least $\frac{1}{2}$ either by the variables x_S, or by the variables $r^*_{A,S}$ in *every scenario A* containing e. Let $E = \{e : \sum_{S: e \in S} x_S \geq \frac{1}{2}\}$. Then $(2x)$ is a fractional set cover solution for the instance with universe E, so one can obtain an integer set cover $\tilde{x}$ for E of cost at most $2\rho \cdot \sum_S w^{\mathrm{I}}_S x_S$. These are our stage I sets. Similarly, for any scenario A, $(2r^*_A)$ is a fractional set cover for $A \setminus E$, since for each such element e we have $\sum_{S: e \in S} r^*_{A,S} \geq \frac{1}{2}$. Therefore, one can cover these elements at a cost of at most $2\rho \cdot \sum_S w^A_S r^*_{A,S}$. So the cost of the solution $\tilde{x}$ is at most $2\rho \cdot h(x)$. □

Combined with the FPAS of Section 2, this yields approximation guarantees for various stochastic covering problems, e.g., we obtain guarantees of $2 \log n + \epsilon$ for SSC, and $4 + \epsilon$ for stochastic vertex cover.

Stochastic facility location. In the deterministic uncapacitated facility location (UFL) problem, given a set of candidate facilities $\mathcal{F}$ and a set of clients $\mathcal{D}$, we have to select a subset of facilities to open and assign each client to an open facility. Each facility i has an opening cost of f_i and each client j has demand d_j, and the cost of assigning client j to facility i is given by $d_j c_{ij}$, where c_{ij} is the distance between i and j and these distances form a metric. The goal is to minimize the sum of the facility opening and client assignment costs. In the 2-stage stochastic version of the problem, abbreviated SUFL, the demand of a client is a random variable (the demands may be correlated), and we can open facilities either in stage I, or after the scenario A with demands d^A_j is revealed, paying a cost of f^{I}_i or f^A_i respectively for opening facility i. We first consider SUFL in the polynomial-scenario model and show that one can design an approximation algorithm by dovetailing an approach used for UFL. Then we show that the above rounding technique can be adapted to derive an approximation algorithm for SUFL in the black-box model. For simplicity, we will assume that $d^A_j \in \{0, 1\}$ for every j, A, so a scenario now specifies a set of clients that need to be assigned to facilities.

Let $\mathcal{A}$ denote the collection of all scenarios, which is explicitly described in the input in the polynomial-scenario model. Consider the following LP relaxation for

SUFL. We use i to index the facilities, j to index the clients, and A to index the scenarios. Variables y_i and $y_{A,i}$ indicate whether facility i is opened in stage I or in scenario A respectively, and $x_{A,ij}$ indicates if client j is assigned to facility i in scenario A.

$$\min \sum_i f_i^{\mathrm{I}} y_i + \sum_A p_A\Big(\sum_i f_i^A y_{A,i} + \sum_{j\in A,i} c_{ij}x_{A,ij}\Big) \qquad \text{(P)}$$

$$\begin{aligned} \text{s.t.} \quad & \sum_i x_{A,ij} \geq 1 && \forall A, j \in A \\ & x_{A,ij} \leq y_i + y_{A,i} && \forall i, A, j \in A \\ & y_i, x_{A,ij}, y_{A,i} \geq 0 && \forall i, A, j \in A. \end{aligned}$$

$$\max \sum_{A,j\in A} p_A \alpha_{A,j} \qquad \text{(D)}$$

$$\begin{aligned} \text{s.t.} \quad & \alpha_{A,j} \leq c_{ij} + \beta_{A,ij} && \forall i, A, j \in A && (2) \\ & \sum_{j\in A} \beta_{A,ij} \leq f_i^A && \forall A, i && (3) \\ & \sum_{A,j\in A} p_A \beta_{A,ij} \leq f_i^{\mathrm{I}} && \forall i && (4) \\ & \alpha_{A,j}, \beta_{A,ij} \geq 0 && \forall i, A, j \in A. \end{aligned}$$

(D) is the dual program. We briefly sketch a primal-dual 3-approximation algorithm due to Mahdian [16], which closely resembles the Jain-Vazirani (JV) algorithm for UFL [12]. All dual variables are initially set to 0. It is easy to imagine the dual-ascent process: we uniformly increase all $\alpha_{A,j}$ variables at rate 1 until one of the constraints becomes tight. If constraint (2) goes tight for some (j, A) and facility i, we also start increasing $\beta_{A,ij}$ at rate 1. If constraint (3) goes tight for some A, i, then we tentatively open facility i for scenario A and freeze (i.e., stop increasing) all $\alpha_{A,j}, \beta_{A,ij}$ variables for which $\alpha_{A,j} \geq c_{ij}$. If (4) goes tight for a facility i, we tentatively open i for stage I, and for every scenario A, we freeze the $\alpha_{A,j}, \beta_{A,ij}$ variables for which $\alpha_{A,ij} \geq c_{ij}$. The process ends when all $\alpha_{A,j}$ variables are frozen. Now we perform a clean-up step for stage I, and for each scenario, to decide which facilities to open. For stage I, we open a maximal subset F of the tentatively open stage I facilities, such that for every (j, A), there is at most one facility $i \in F$ with $\beta_{A,ij} > 0$. In every scenario A, we open a maximal subset F_A of the tentatively open facilities for scenario A, such that for every $j \in A$, there is at most one facility $i \in F \cup F_A$ with $\beta_{A,ij} > 0$. The analysis proceeds as in the JV algorithm, by showing that for every (j, A), if the facility that caused $\alpha_{A,j}$ to freeze is not open, then there must be a facility opened in stage I or in scenario A that is at most $3\alpha_{A,j}$ distance away from j. This proves an approximation ratio of 3.

We now consider SUFL in the black-box model. We compactify (P) to obtain the convex program: minimize $h(y) := \sum_i f_i^{\mathrm{I}} y_i + \sum_{A\in\mathcal{A}} p_A g_A(y)$, where $g_A(y)$ is the minimum of $\sum_i f_i^A y_{A,i} + \sum_{j\in A,i} c_{ij}x_{A,ij}$ subject to the constraints $\sum_i x_{A,ij} \geq 1$ for all $j \in A$, $x_{A,ij} \leq y_i, y_{A,i}$ for all $i, j \in A$, and $x_{A,ij}, y_{A,i} \geq 0$ for all $i, j \in A$. Note that this is not a stochastic covering program. While UFL admits a star-covering relaxation (clients have to be covered by stars, a star is a facility and a set of clients assigned to it), the corresponding stochastic covering program does not model SUFL, because in SUFL when we open a facility in stage I we do not fix then the set of clients it will serve; this is decided in stage II, and will typically be scenario-dependent. Yet, the above rounding technique can be adapted here, by applying decoupling to the covering constraint

$\sum_i x_{A,ij} \geq 1$. Let ρ_{UFL} denote the integrality gap of UFL, which is at most 1.52 [17].

Theorem 3. *The integrality gap of (P) is at most* $2\rho_{\mathsf{UFL}}$.

Proof. Let y be any feasible solution to the convex program and let (x_A^*, y_A^*) be an optimal solution to $g_A(y)$. We write $x_{A,ij}^* = x_{A,ij}^{\mathrm{I}} + x_{A,ij}^{\mathrm{II}}$ for each scenario A and client $j \in A$, where $x_{A,ij}^{\mathrm{I}} \leq y_i^*$ and $x_{A,ij}^{\mathrm{II}} \leq y_{A,i}^*$. This is always possible since $x_{A,ij}^* \leq y_i^* + y_{A,i}^*$. So either $\sum_i x_{A,ij}^{\mathrm{I}} \geq \frac{1}{2}$ or $x_{A,ij}^{\mathrm{II}} \geq \frac{1}{2}$. For a client j, define $\mathcal{S}_j = \{A \ni j : \sum_i x_{A,ij}^{\mathrm{I}} \geq \frac{1}{2}\}$. For the stage I decisions, we construct a feasible fractional solution for a UFL instance where the facility costs are f_i^{I}, the assignment costs are c_{ij}, and the "demand" of client j is set to $\sum_{A \in \mathcal{S}_j} p_A$, and then round this using an algorithm for UFL. If we treat each (j, A) where $A \in \mathcal{S}_j$ as a separate client with demand p_A, we obtain a feasible solution by setting $\hat{y}_i = \min(1, 2y_i^*)$ and $\hat{x}_{A,ij} = \min(1, 2x_{A,ij}^{\mathrm{I}})$. yields a feasible solution for this instance. But since the $\hat{y}_i$ facility variables do not depend on the scenario, we can re-optimize the assignment for each (j, A) to obtain an assignment that does not depend on A. Thus, we can coalesce all the (j, A) clients into one, with demand $\sum_{A \in \mathcal{S}_j} p_A$. $2(\sum_i f_i^{\mathrm{I}} y_i^* + \sum_{j,i,A \in \mathcal{S}_j} p_A c_{ij} x_{A,ij}^{\mathrm{I}})$. Since the integrality gap is ρ_{UFL}, there is an integer solution $(\tilde{x}, \tilde{y})$ of cost at most $2\rho_{\mathsf{UFL}}(\sum_i f_i^{\mathrm{I}} y_i^* + \sum_{j,i,A \in \mathcal{S}_j} p_A c_{ij} x_{A,ij}^{\mathrm{I}})$; this determines which facilities to open in stage I. In any scenario A, each client j such that $A \in \mathcal{S}_j$ is assigned to the stage I facility given by the assignment $\tilde{x}$. For each remaining client j, since $\sum_i x_{A,ij}^{\mathrm{II}} \geq \frac{1}{2}$, the solution $\hat{y}_{A,i} = \min\bigl(1, 2y_{A,i}^*\bigr)$, $\hat{x}_{A,ij} = \min\bigl(1, 2x_{A,ij}^{\mathrm{II}}\bigr)$ yields a feasible solution for the UFL instance with client set $\{j \in A : A \notin \mathcal{S}_j\}$. Again the ρ_{UFL} integrality gap shows that there is an integer solution with "low" cost. Overall, we get that the total cost of the solution $\tilde{y}$ is at most $2\rho_{\mathsf{UFL}} \cdot h(y)$. This shows that the integrality gap is at most $2\rho_{\mathsf{UFL}}$ (and gives a 3.04-approximation algorithm in the polynomial-scenario model (taking $\rho_{\mathsf{UFL}} = 1.52$).) $\square$

The rounding approach does not yet yield the strongest performance guarantee currently known. We will return to this problem in Section 4.

Stochastic Steiner tree. We now describe the boosted sampling technique of Gupta et al. [10] that shows that for certain stochastic problems, an approximation algorithm for the deterministic problem that satisfies some *cost-sharing* properties, can be used to derive performance guarantees for the stochastic problem. We focus on the stochastic rooted Steiner tree (SST) problem: we have a graph $G = (V, E)$, a fixed root $r \in V$, and a distribution that specifies a random set of terminals to connect to the root. We can buy edges either in stage I or after the terminal set $A \subseteq V$ has been revealed, paying a cost of c_e or c_e^A respectively for edge e, so as to connect all the nodes in A to r. It is worth noting that we can formulate a *fractional version* of SST as a stochastic covering problem, where each cut separating a terminal from the root must be covered by edges bought in the two stages. One can therefore obtain a $(1 + \epsilon)$-optimal fractional solution in polynomial time. However the rounding procedure detailed above does not work, because the cut-covering problems obtained after decoupling the two stages need not correspond to Steiner tree instances (and may not even fall into the Goemans-Williamson framework [9]). We can use boosted sampling to devise a 4-approximation algorithm, under the cost restriction $c_e^A = \lambda^A c_e$ for every edge e in every scenario A. This reflects a limitation of the boosted sampling approach. In the case of SST, without such a restriction the problem becomes Group-Steiner-tree-hard [21], but for other problems such as stochastic {vertex cover, facility location}, one can obtain good guarantees *without imposing any cost restrictions* by using other techniques.

Here we assume for simplicity that $c_e^A = \lambda c_e$ for every A, e. Let $\mathsf{ST}(S)$ denote the cost of an optimal Steiner tree on $S \cup \{r\}$ wrt. costs $\{c_e\}$. We say that an algorithm $\mathcal{A}$ for the Steiner tree problem admits a *β-strict cost sharing* if there is a function $\xi : 2^V \times V \mapsto \mathbb{R}_{\geq 0}$ such that for every $S, T \subseteq V$ with $S \cap T = \emptyset$, (i) $\xi(S, u) = 0$ for $u \notin S$; (ii) $\sum_{u \in S} \xi(S, u) \leq \mathsf{ST}(S)$; and (iii) there is a procedure $\mathsf{Aug}_{\mathcal{A}}$ that augments the tree $\mathcal{A}(S)$ constructed by $\mathcal{A}$ on input S to a tree on $S \cup T \cup \{r\}$ incurring cost $c\big(\mathsf{Aug}_{\mathcal{A}}(S, T)\big) \leq \beta \sum_{u \in T} \xi(S \cup T, u)$. Intuitively $\xi(S, u)$ stands for u's share in the cost of a Steiner tree on S.

We may assume that G is complete and the edge costs form a metric. We use the MST heuristic as algorithm $\mathcal{A}$. This is a 2-approximation algorithm that admits a 2-strict cost sharing. Procedure $\mathsf{Aug}_{\mathcal{A}}$ consists of contracting S into the root, and building an MST on $T \cup \{r\}$ in the contracted graph. Rooting the MST on $S \cup \{r\}$ at r, we set $\xi(S, u) = \frac{1}{2}$(cost of the edge joining u to its parent). This satisfies properties (i) and (ii) above, and it is not hard to show that it satisfies (iii) with $\beta = 2$. The algorithm for SST is quite simple and extremely elegant: we draw λ samples $A_1, \ldots, A_\lambda$ from the distribution and build the tree $\mathcal{A}(S)$ where $S = \bigcup_i A_i$, as our first-stage solution. Intuitively, this tries to account for the λ inflation factor by sampling each scenario A, in expectation, λp_A times. In the second-stage, if scenario A is realized, we use $\mathsf{Aug}_{\mathcal{A}}$ to augment $\mathcal{A}(S)$ and connect $A \setminus S$ to the root. A nice feature of the algorithm is that only λ samples are required.

Let E_{I}^* and E_A^* be the edges purchased in stage I and in scenario A by an optimal (integer) solution to SST, and let $OPT = c(E_{\mathrm{I}}^*) + \lambda \mathrm{E}_A\big[c(E_A^*)\big]$ be the cost incurred. Let $\xi(X, Y)$ denote $\sum_{u \in Y} \xi(X, u)$. The first-stage cost can be bounded by noting that $\mathsf{ST}(S)$ is at most the cost of $Z_S = E_{\mathrm{I}}^* \cup \big(\bigcup_{i=1}^{\lambda} E_{A_i}^*\big)$ since Z_S connects S to r. The expected first-stage cost is at most $2\mathrm{E}_S\big[\mathsf{ST}(S)\big] \leq 2\mathrm{E}_S\big[c(Z_S)\big]$ which is at most $2 \cdot OPT$, since each scenario A is sampled λp_A times in expectation. The expected second-stage cost is given by $\lambda \mathrm{E}_{S,A}\big[c(\mathsf{Aug}_{\mathcal{A}}(S, A \setminus S))\big]$ which is at most $2\lambda \mathrm{E}_{S,A}\big[\xi(S \cup A, A \setminus S)\big]$ by property (iii). We can treat scenario A as an extra sample $A_{\lambda+1}$, and since the A_i's are identically distributed, we have that $\mathrm{E}_{S,A}\big[\xi(S \cup A, A \setminus S)\big] \leq \frac{1}{\lambda+1} \mathrm{E}_{S,A}\big[\xi(S \cup A, S \cup A)\big] \leq \frac{1}{\lambda+1} \mathrm{E}_{S,A}\big[\mathsf{ST}(S \cup A)\big]$. Finally, by arguing as we did for stage I, one can bound $\mathrm{E}_{S,A}\big[\mathsf{ST}(S \cup A)\big]$ by $\frac{\lambda+1}{\lambda} \cdot OPT$. Thus the expected second-stage cost is at most $2 \cdot OPT$, and the total cost is at most $4 \cdot OPT$.

Gupta et al. showed that boosted sampling can be applied to any stochastic problem satisfying a certain sub-additivity condition, if we have an approximation algorithm for the deterministic version that admits a β-strict cost-sharing (which is now defined more abstractly). They show that an α-approximation algorithm with a β-strict cost sharing gives an $(\alpha + \beta)$-approximation algorithm for the stochastic problem. In all known cases, such an approximation algorithm is obtained via the primal-dual schema and the cost shares are derived from the dual variables. Thus, boosted sampling can be viewed as a primal-dual approach for designing approximation algorithms for stochastic problems.

4 The Sample Average Approximation Method

The sample average approximation (SAA) method is a natural approach for computing solutions in the black-box model. Here we replace the original stochastic problem by a sample-average problem obtained by sampling scenarios some $\mathcal{N}$ times and estimating the scenario probabilities by their frequencies of occurrence in the sampled set, and solve this problem. If one can show that a polynomially bounded $\mathcal{N}$ suffices to obtain a $(1+\epsilon)$-optimal solution (to the original problem), then one would obtain a reduction from the black-box problem to a polynomial-scenario problem while losing a $(1 + \epsilon)$ factor. As mentioned earlier, Kleywegt et al. [14] prove a sample-size bound for general 2-stage programs that depends on the variance of a certain quantity, which need not be polynomially bounded. Although this bound is tight in the black-box model for general 2-stage programs [24], for structured programs such as the class of 2-stage LPs considered in [25], one can prove better bounds that do not follow directly from the bound in [14]. Swamy and Shmoys [27] gave a polynomial bound for this class by building upon ideas used in the ellipsoid-based FPAS of Section 2. More recently, Nemirovskii & Shapiro [19] showed that for the stochastic set cover problem, additional analytical insights yield similar bounds as a further consequence of the results of [14]. Thus, the SAA method yields a simpler, more efficient scheme for this class of programs.

The proof in [27] uses (approximate) subgradients to identify a notion of closeness between the sample-average and true objective functions. Loosely speaking, this notion captures the property that the ellipsoid-based FPAS can be made to run identically on both the sample-average and the true problems, which intuitively suggests that optimizing the sample-average function is nearly equivalent to optimizing the true function. Subsequently Charikar, Chekuri and Pál [3] gave a different proof for roughly the same class of programs. While [27] only shows that any *optimal* solution to the sample-average LP is a $(1 + \epsilon)$-optimal solution to the true LP (with high probability), Charikar et al. argue that by slightly modifying the "standard" SAA approach, one can prove that any α-optimal solution to the sampled problem is an $(\alpha + \epsilon)$-optimal solution to the true problem. This implies the remarkable consequence that one can, in effect, reduce the black box model (for a class of 2-stage recourse minimization problems) to the polynomial-scenario model. In Section 3, we gave a 3-approximation algorithm for SUFL in the polynomial-scenario model. Likewise, by mimicking the primal-dual algorithm for vertex cover one can obtain the same guarantee of

2 for the stochastic problem in the polynomial-scenario model [21]. The above result shows that these guarantees also extend to the black-box model.

We have described a variety of techniques for the design of approximation algorithms for 2-stage stochastic linear and integer programs. This thread of algorithmic analysis of stochastic optimization approximation algorithms has the potential to bring together the insights and approaches from several disjoint research communities: (traditional) stochastic programming, theoretical computer science, and machine learning (where the flavor of learning a distribution based on a limited number of samples plays a central role). There is much more work remaining than has already been done, since the bulk of the work done thus far in such black box settings does not extend to a variable number of stages, or to settings beyond the simple expectation minimization objective.

References

1. E. M. L. Beale. On minimizing a convex function subject to linear inequalities. *J. Royal Stat. Soc., Series B*, 17:173–184; discussion 194–203, 1955.
2. J. R. Birge and F. V. Louveaux. *Introduction to Stochastic Programming*. Springer-Verlag, NY, 1997.
3. M. Charikar, C. Chekuri, and M. Pál. Sampling bounds for stochastic optimization. *Proc. 9th RANDOM*, 257–269, 2005.
4. G. B. Dantzig. Linear programming under uncertainty. *Management Sci.*, 1:197–206, 1955.
5. B. Dean, M. X. Goemans, and J. Vondrak. Approximating the stochastic knapsack problem: the benefit of adaptivity. *Proc. 45th Annual IEEE Symposium on Foundations of Computer Science*, 208–217, 2004.
6. S. Dye, L. Stougie, and A. Tomasgard. The stochastic single resource service-provision problem. *Naval Res. Logistics*, 50:869–887, 2003. Also appeared as “The stochastic single node service provision problem”, COSOR-Memorandum 99-13, Dept. Math. & Comp. Sc., Eindhoven Tech. Univ., Eindhoven, 1999.
7. M. Dyer, R. Kannan, and L. Stougie. A simple randomised algorithm for convex optimisation. SPOR-Report 2002-05, Dept. Math. & Comp. Sc., Eindhoven Tech. Univ., Eindhoven, 2002.
8. M. Dyer and L. Stougie. Computational complexity of stochastic programming problems. SPOR-Report 2005-11, Dept. Math. & Comp. Sc., Eindhoven Tech. Univ., Eindhoven, 2005.
9. M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24:296–317, 1995.
10. A. Gupta, M. Pál, R. Ravi, and A. Sinha. Boosted sampling: approximation algorithms for stochastic optimization. *Proc. 36th ACM STOC*, 417–426, 2004.
11. N. Immorlica, D. Karger, M. Minkoff, and V. Mirrokni. On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems. *Proc. 15th SODA*, 684–693, 2004.
12. K. Jain and V.V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *J. ACM*, 48:274–296, 2001.
13. J. Kleinberg, Y. Rabani, and É. Tardos. Allocating bandwidth for bursty connections. *SIAM J. Comput.*, 30:191–217, 2000.

14. A. J. Kleywegt, A. Shapiro, and T. Homem-De-Mello. The sample average approximation method for stochastic discrete optimization. *SIAM J. Optimization*, 12:479–502, 2001.
15. J. Linderoth, A. Shapiro, and S. Wright. The empirical behavior of sampling methods for stochastic programming. *Annals Oper. Res.*, to appear.
16. M. Mahdian. *Facility Location and the Analysis of Algorithms through Factor-revealing Programs.* Ph.D. thesis, MIT, Cambridge, MA, 2004.
17. M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location. *Proc. 5th APPROX*, 229–242, 2002.
18. R. Möhring, A. Schulz, and M. Uetz. Approximation in stochastic scheduling: the power of LP based priority policies. *JACM*, 46:924–942, 1999.
19. A. Nemirovski and A. Shapiro. On complexity of Shmoys–Swamy class of two-stage linear stochastic programming problems. *Optimization Online,* 2006. http://www.optimization-online.org/DB_FILE/2006/07/1434.pdf.
20. Y. Nesterov and J.-Ph. Vial. Confidence level solutions for stochastic programming. CORE Discussion Papers, 2000. http://www.core.ucl.ac.be/services/psfiles/dp00/dp2000-13.pdf.
21. R. Ravi and A. Sinha. Hedging uncertainty: approximation algorithms for stochastic optimization problems. *Proceedings, 10th IPCO*, 101–115, 2004.
22. A. Ruszczynski and A. Shapiro. Editors, *Stochastic Programming*, Volume 10 of *Handbooks in Oper. Res. & Mgmt. Sc.*, North-Holland, Amsterdam, 2003.
23. A. Shapiro. Monte Carlo sampling methods. In A. Ruszczynski and A. Shapiro, editors, *Stochastic Programming*, volume 10 of *Handbooks in Oper. Res. & Mgmt. Sc.*, North-Holland, Amsterdam, 2003.
24. A. Shapiro and A. Nemirovski. On complexity of stochastic programming problems. *Optimization Online*, 2004. http://www.optimization-online.org/DB_FILE/2004/10/978.pdf.
25. D. B. Shmoys and C. Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *J. ACM*, to appear. Preliminary version appeared as "Stochastic optimization is (almost) as easy as deterministic optimization" in *Proc. 45th Annual IEEE FOCS*, 228–237, 2004.
26. C. Swamy. *Approximation Algorithms for Clustering Problems.* Ph.D. thesis, Cornell Univ., Ithaca, NY, May 2004. http://www.math.uwaterloo.ca/~cswamy/theses/master.pdf.
27. C. Swamy and D. B. Shmoys. The sample average approximation method for 2-stage stochastic optimization. November 2004. http://www.math.uwaterloo.ca/~cswamy/papers/SAAproof.pdf.
28. B. Verweij, S. Ahmed, A. J. Kleywegt, G. L. Nemhauser, and A. Shapiro. The sample average approximation method applied to stochastic routing problems: a computational study. *Comp. Opt. Appl.*, 24:289–333, 2003.