Stochastic Optimization is (almost) as Easy as Deterministic Optimization[*]

David B. Shmoys[†]
shmoys@cs.cornell.edu

Chaitanya Swamy[†]
swamy@cs.cornell.edu

Abstract

Stochastic optimization problems attempt to model uncertainty in the data by assuming that (part of) the input is specified in terms of a probability distribution. We consider the well-studied paradigm of 2-stage models with recourse: first, given only distributional information about (some of) the data one commits on initial actions, and then once the actual data is realized (according to the distribution), further (recourse) actions can be taken. We give the first approximation algorithms for 2-stage discrete stochastic optimization problems with recourse for which the underlying random data is given by a "black box" and no restrictions are placed on the costs in the two stages, based on an FPRAS for the LP relaxation of the stochastic problem (which has exponentially many variables and constraints). Among the range of applications we consider are stochastic versions of the set cover, vertex cover, facility location, multicut (on trees), and multicommodity flow problems.

1. Introduction

The study of stochastic optimization problems dates back to the 1950's and the work of Dantzig [3] (among others) and attempts to model uncertainty in the data by assuming that (part of) the input is specified in terms of a probability distribution, rather than by deterministic data given in advance. Stochastic optimization techniques and models have become an important paradigm in a wide range of application areas, including transportation models, logistics, financial instruments, and network design. Stochastic models are often computationally quite difficult, both from a practical perspective, as well as from the viewpoint of complexity theory; even extremely specialized (sub)problems may be $\#P$-hard [6].

We shall focus on an important subclass of stochastic optimization problems: *2-stage problems with recourse*. Many applications come under this setting, such as the problem of installing facilities to serve a set of clients; one is initially given only a probability distribution on the demands of the clients (in addition to deterministic data for the facility and assignment costs) on which to base initial (first stage) decisions, but then one can extend (in a second stage) the solution once the actual input is realized according to this distribution, where these recourse costs may be different (and typically more expensive) than the original ones. Much of the textbook of Birge and Louveaux [1] is devoted to models and algorithms for this class of problems.

We shall initially focus on the following stochastic generalization of the set cover problem: we are given a family of sets $S_1, \dots, S_m$ over a ground set U, where each set S has an *a priori* weight w_S^{I}, and an *a posteriori* weight w_S^{II} associated with it. In the first stage, one selects some of these sets, incurring each associated weight w_S^{I}, then a subset $A \subseteq U$ is drawn according to a specified distribution, and then additional sets may be selected (incurring their second stage weights) so as to ensure that A is contained in the union of the selected sets (in both stages). The aim is to minimize the expected cost of the solution. Note that an explicit representation of a feasible solution would require specifying an exponential amount of information (since there are $2^{|U|}$ possible choices for A), and so it will be necessary to give more compact, algorithmic representations.

An important issue left ambiguous in the description above is the way in which the probability distribution is specified; several approaches have recently been considered in papers that address related 2-stage stochastic optimization problems. Dye, Stougie, and Tomasgard [4], and later, Ravi and Sinha [14] assume that there are only a polynomial number of scenarios, i.e., choices for A, that occur with positive probability. Independently, Immorlica, Karger, Minkoff, and Mirrokni [10] consider both this assumption, as well as the model where each element occurs with its own independent probability, and in so doing they enlarge the space of scenarios to be exponentially large. This is done with the rather severe restriction of assuming that the *costs in the two stages are proportional*, that is, there is a parameter λ such that $w_S^{\mathrm{II}} = \lambda w_S^{\mathrm{I}}$ for each set S. Gupta, Pál, Ravi, and Sinha [9] also require this assumption, but give a more general way to specify the distribution, which we shall call the *black-box model*: they assume that

[*] A full version is available at www.cs.cornell.edu/˜swamy/stochopt.ps
[†] Dept. of Computer Science, Cornell University, Ithaca, NY 14853. Research supported partially by NSF grant CCR-9912422.

the algorithm may make use of samples that are drawn according to the distribution of scenarios.

Our Results. We achieve the best qualities of all of these approaches: we work in the black-box model and obtain results in settings where the costs in the two stages need not be proportional, and the second-stage costs may even *depend on the particular scenario realized* (as in [14]).

We show that, given a class of (deterministic) set cover instances (e.g., the vertex cover problem), for which we have a ρ-approximation guarantee with respect to the natural linear programming relaxation, we can, for any $\epsilon > 0$, obtain a randomized $(2\rho + \epsilon)$-approximation algorithm for the stochastic generalization. This generalizes and improves upon performance guarantees of [9].

Our result has two principal components. First, we show that if we formulate the stochastic set cover problem as an (exponentially large) integer program and solve its linear programming (LP) relaxation, then a surprisingly simple rounding approach suffices to prove the guarantee claimed above. The essence of our approach is that the relaxation indicates for each element either that it is at least half covered in the first stage, or else it must be at least half covered in *each* scenario in which it occurs in the second stage. We have decoupled the two stages (and indeed each of the scenarios for the second stage), and can apply the deterministic result to each separately. Thus, the fact that we lose a factor of 2 is exactly tied into the fact that we are considering a 2-stage problem. Note that we need to examine only the first-stage variables to decouple the two stages, and not the entire (exponentially large) LP solution. In fact, this decomposition can be applied to a number of stochastic integer optimization problems. Furthermore, if we consider the case when there are only a polynomial number of scenarios, then this rounding approach is sufficient to yield strong performance guarantees for a wide range of applications.

Second, and this is the more technically difficult part of the paper, we give a fully polynomial randomized approximation scheme for solving a broad class of 2-stage stochastic LPs, in spite of the fact that this may be $\#P$-hard. (The LP has an exponential number of both variables and constraints.) We believe that this is a tool of independent interest, in particular, in the stochastic programming literature, and will find application in the design of approximation algorithms for other stochastic integer optimization problems.

We approximately solve this linear program by working with an equivalent (but compact) convex programming formulation, and show that the ellipsoid algorithm can be adapted to yield such a scheme. In the ellipsoid algorithm for convex programming, the algorithm generates a sequence of ellipsoids, starting with an ellipsoid that contains the entire feasible region; in each iteration, a hyperplane is generated that is intersected with the current ellipsoid, and the next ellipsoid generated is the minimum-volume ellipsoid containing this intersection. At a feasible point, the hyperplane is computed by computing the subgradient of the objective function at the center of the current ellipsoid. We introduce a notion of approximate subgradients that is sufficient to yield the same convergence of the algorithm. Furthermore, we show that this approximate subgradient can be computed in randomized polynomial time using samples from the distribution (obtained from a black box). Finally, the ellipsoid algorithm outputs the iterate with the best objective function value. However, evaluating the objective function value at a given point for our class of stochastic programs may be $\#P$-hard; nonetheless, approximate subgradient information is sufficient to efficiently compute a point of cost close to the cost of the minimum-cost iterate (without, however, computing these costs). Note that for the rounding algorithm, one only needs the convex program's (near-)optimal solution to compute a solution to the stochastic integer optimization problem.

Our algorithm returns a $(1 + \epsilon)$-optimal solution in time bounded by a polynomial in the input size, $1/\epsilon$, and the maximum *ratio* λ between the second- and first-stage costs, and suggests that contrary to conventional wisdom, an exponential number of scenarios is *not* an insurmountable impediment to the design of efficient algorithms for these problems.

It is useful to compare our result with some work in the stochastic programming literature on the sample average approximation (SAA) method for solving stochastic programs, where one samples from the distribution on scenarios and then solves an approximate problem, estimating the probability of a scenario by its frequency. Kleywegt, Shapiro, and Homem-De-Mello [11] prove a bound on the sample size required to obtain a near-optimal solution (with high probability) that is polynomial in the dimension, but depends on the variance of a certain quantity (calculated using the scenario distribution) that need not be polynomially bounded. Whereas the running time of our algorithm depends on the parameter λ, it does not depend on the underlying probability distribution. The algorithm of [9] also has the same dependence on λ. In fact, this dependence on λ is unavoidable; we show that a performance guarantee of ρ requires $\Omega(\lambda/\rho)$ samples. The dependence on $\frac{1}{\epsilon}$ is also necessary in light of the known $\#P$-hardness results. To the best of our knowledge, this is the first result to show that (a broad class of) 2-stage stochastic LPs can be solved in time polynomial in the input size, λ, and $\frac{1}{\epsilon}$. It is also useful to contrast our work with that of Dyer, Kannan, and Stougie [5], who focus on computing an estimate for the objective function value at a given point (for a maximization problem) by sampling from the distribution sufficiently many times. But this yields a running time that is only polynomial in the maximum *value* attained by *any* scenario, and does not yield a polynomial approximation scheme for our setting.

The first worst-case analysis of approximation algorithms for 2-stage stochastic integer programming problems appears to be due to Dye et al. [4], who give a constant performance guarantee for a resource provisioning problem (a maximization problem) in the polynomial scenarios setting based on rounding an LP. Ravi and Sinha [14], and independently Immorlica et al. [10], and subsequently Gupta et al. [9] consider various 2-stage problems including applications that follow from our general settings. We now contrast our results in these applications with previous results.

For the vertex cover problem, our technique yields a $(4+\epsilon)$-approximation algorithm in the black-box distribution model. In contrast, [9] give an 8-approximation algorithm in the black-box model with proportional costs, and [14] give a guarantee of 2 in the polynomial scenarios setting. We also give extensions to multi-covering generalizations of the set cover problem. For the stochastic uncapacitated facility location problem, we give a $(3.225+\epsilon)$-approximation algorithm, whereas [9] give a guarantee of 8.45 in the black-box model with proportional costs, and [14] give a guarantee of 8 in the polynomial scenarios setting. Our approach yields constant-factor performance guarantees for several facility location variants, including facility location with penalties, or soft capacities, or service installation costs. As in [14], our results also extend to the case with scenario-dependent second-stage costs.

2. A motivating example: stochastic set cover

The deterministic weighted set-cover problem (DSC) is the following: given a universe U of n elements and a family $\mathcal{S}$ of m subsets of U, with set $S \in \mathcal{S}$ having weight w_S, we want to choose a minimum-weight collection of sets so that each element e is contained in some chosen set. We can express the problem as an integer program and relax the integrality constraints to get a linear program.

$$\min \sum_S w_S x_S \quad \text{s.t.} \sum_{S:e\in S} x_S \geq 1 \;\; \forall e, \quad x_S \geq 0 \;\; \forall S. \qquad \text{(P)}$$

Let OPT_{Det} denote the optimal value of (P). It is well known [2] that "the greedy algorithm" returns a solution of weight at most $\ln n \cdot OPT_{Det}$.

In the 2-stage stochastic generalization of this problem, abbreviated SSC, the elements to be covered are not known in advance. There is a probability distribution over *scenarios*, and each scenario specifies the actual subset of elements to be covered. For our purposes, a scenario is just a subset of elements $A \subseteq U$. We will assume that the set of all possible scenarios is just the power set 2^U (including the empty set $\emptyset$). We use p_A to denote the probability of scenario A (p_A could be 0, if scenario A never occurs). *We will never explicitly use the quantities p_A in the algorithm.* Throughout, we use A to index the scenarios.

Each set S has two weights associated with it: an *a priori* weight w_S^{I}, and an *a posteriori* weight w_S^{II}. We may choose a set either in stage I paying a price of w_S^{I}, or in stage II after the scenario $A \subseteq U$ occurs paying a price of w_S^{II}, so that A is contained in the union of the sets selected (in both stages). The goal is to minimize the sum of the stage I cost and the expectation over all scenarios A of the stage II cost of scenario A. The two-stage problem can be formulated as an integer program with the following linear relaxation.

$$\begin{aligned} \min \quad & \sum_S w_S^{\mathrm{I}} x_S + \sum_{A,S} p_A w_S^{\mathrm{II}} r_{A,S} & & \text{(SSC-P1)} \\ \text{s.t.} \quad & \sum_{S:e\in S} x_S + \sum_{S:e\in S} r_{A,S} \geq 1 & \forall A, e \in A \quad & (1) \\ & x_S, r_{A,S} \geq 0 & \forall A, S. & \end{aligned}$$

Variable x_S indicates whether set S is chosen in stage I, and $r_{A,S}$ is 1 if and only if set S is chosen in scenario A in stage II. Constraint (1) 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. A $\{0,1\}$-solution, corresponds exactly to a solution to our problem. The following theorem forms the basis of our methodology for tackling various 2-stage stochastic optimization problems. Let OPT denote the optimal value of (SSC-P1).

Theorem 2.1 *Suppose that we have a procedure that for every instance of* DSC *produces a solution of cost at most $\rho \cdot OPT_{Det}$. Then, one can convert any solution (x,r) to* (SSC-P1) *to an integer solution losing a factor of at most 2ρ. Thus, an optimal solution to* (SSC-P2) *gives a 2ρ-approximation algorithm.*

Proof : Let $h(.)$ be the objective function. We will argue that we can obtain an integer solution $(\tilde{x},\tilde{r})$ of cost at most $2\rho \cdot h(x,r)$. Observe the following simple fact: an element e is either covered to an extent of at least $\frac{1}{2}$ in the first stage by the variables x_S, or it is covered to an extent of at least $\frac{1}{2}$ 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 set E and so, one can obtain an integer set cover $\tilde{x}$ for this instance of cost at most $\rho \cdot \sum_S 2w_S^{\mathrm{I}} x_S$. Similarly, for any scenario A, $(2r_A)$ is a fractional set cover for the elements in $A \setminus E$, 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 $\rho \cdot \sum_S w_S^{\mathrm{II}} r_{A,S}$. So if we output $\tilde{x}$ as the first-stage decisions, we get a solution of cost at most $2\rho \cdot h(x,r)$. ■

Corollary 2.2 *If the integrality gap of* (P) *is ρ, then the integrality gap of* (SSC-P1) *is at most 2ρ.*

Theorem 2.1 shows that if we could solve (SSC-P1), then we could get a $2\ln n$-approximation algorithm for SSC. In particular, when there are only a polynomial number of scenarios with non-zero probability, we obtain a $2\ln n$-approximation algorithm. In general, however, it seems difficult to find an optimal solution to (SSC-P1) in its present form, since it has both an exponential number of variables and an exponential number of constraints; even writing out an optimal solution might take exponential space. Observe that in the proof of Theorem 2.1, we needed to examine *only the stage I variables* x_S of the fractional solution, in order to round it to an integer solution. This is important, because if the rounding algorithm required information about each stage II scenario A, that is, an exponential amount of information (as in [14]), then one would *not* get a polynomial-time algorithm even if one could "solve" (SSC-P1) efficiently. In contrast, Theorem 2.1 shows that if we could somehow compactly express (SSC-P1), and solve the resulting program efficiently and find an (near-) optimal (fractional) *first-stage vector* x, then one can obtain a 2ρ-approximation algorithm. This motivates the following formulation with only the stage I variables x_S.

$$\min \sum_S w_S^{\mathrm{I}} x_S + f(x) \;\text{ s.t. }\; 0 \le x_S \le 1 \quad \forall S, \quad \text{(SSC-P2)}$$

where $f(x) = \sum_{A\subseteq U} p_A f_A(x)$, and $f_A(x) = \min\bigl\{\sum_S w_S^{\mathrm{II}} r_{A,S} : \sum_{S:e\in S} r_{A,S} \ge 1 - \sum_{S:e\in S} x_S \;\forall e \in A;\quad r_{A,S} \ge 0 \;\forall S\bigr\}$. It is straightforward to show that (SSC-P2) and (SSC-P1) are equivalent programs, and that the objective function of the latter is convex.

3. Solving the convex program

We now leverage the fact that the objective function of (SSC-P2) is convex, and show that the ellipsoid method can be adapted to find a near-optimal solution to (SSC-P2) in polynomial time, *despite the fact that evaluating $f(x)$, and hence the objective function, may in general be #P-hard*. Section 4 generalizes the arguments to show that the algorithm can be applied to a broad class of 2-stage programs.

The ellipsoid method starts by containing the feasible region within a ball; in each iteration, a new ellipsoid is generated by finding the minimum-volume ellipsoid that contains the intersection of the current one with a specific half-space defined by a hyperplane. If the current ellipsoid center is infeasible, then one uses an inequality violated by it; otherwise, one uses an *objective function cut*, to eliminate points whose objective function value is no better than the current center. 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 certain number of iterations, the feasible point generated with the best objective function value is a near-optimal solution.

The above description makes clear that the inability to evaluate $f(x)$ is an obstacle to applying the ellipsoid method in this case. Let $\mathcal{P} = \mathcal{P}_0$ denote the polytope $\bigl\{x \in \mathbb{R}^m : 0 \le x_S \le 1 \text{ for all } S\bigr\}$, and let $h(x)$ be the (convex) objective function $w^{\mathrm{I}} \cdot x + f(x)$. If the current iterate x_i is feasible, that one could add the constraint $h(x) \le h(x_i)$ while maintaining the convexity of the feasible region. But then, in subsequent iterations, one would need to check if the current iterate is feasible, and generate a separating hyperplane if not. Without the ability to evaluate (or even estimate) the objective function value, this appears to pose a formidable difficulty. An alternative possibility is to use cuts generated by a *subgradient*, which essentially plays the role of the gradient when the function is not differentiable. We say that d is a subgradient of a function $g : \mathbb{R}^m \mapsto \mathbb{R}$ at the point u if the inequality $g(v) - g(u) \ge d \cdot (v - u)$ holds for every $v \in \mathbb{R}^m$.

Note that the subgradient at a given point need not be unique. If d_i is the subgradient at point x_i, we can add the *subgradient cut* $d_i \cdot (x - x_i) \le 0$ and proceed with the (smaller) polytope $\mathcal{P}_{i+1} = \mathcal{P}_i \cap \{x : d_i \cdot (x - x_i) \le 0\}$. Unfortunately, even computing the subgradient at a point x seems hard to do in polynomial time for the objective functions that arise in stochastic programs. To circumvent this obstacle, we define the notion of an *approximate subgradient* which is crucial to the working of our algorithm.

Definition 3.1 *We say that $\hat{d}$ is an $(\omega, \mathcal{D})$-subgradient of a function $g : \mathbb{R}^m \mapsto \mathbb{R}$ at the point $u \in \mathcal{D}$ if for every $v \in \mathcal{D}$ we have, $g(v) - g(u) \ge \hat{d} \cdot (v - u) - \omega g(u)$.*

The algorithm only uses $(\omega, \mathcal{P})$-subgradients which we denote as ω-subgradients from now on. We show that one can compute with high probability an ω-subgradient of $h(.)$ at any point x, by sampling from the black box on scenarios. At a feasible point x_i, we compute an ω-subgradient $\hat{d}_i$ and add the inequality $\hat{d}_i \cdot (x - x_i) \le 0$ to chop off a region of $\mathcal{P}_i$ and get the polytope $\mathcal{P}_{i+1}$. 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 of $\mathcal{P}_k$) is "small" (this is made precise later). Now if the function $h(.)$ has bounded variation on nearby points, then one can show that $\min_i h(x_i)$ is close to the optimal value $h(x^*)$ with high probability. Since we approximate the subgradient at x, our running time depends only on the variation in the subgradient vector components, which we show is bounded by the maximum *ratio* of the stage II and stage I costs.

A final hurdle is that, since we cannot compute $h(x)$, we will not be able to determine the point $\bar{x} = \arg\min_i h(x_i)$. Nonetheless, by using approximate subgradients we will find a point $\bar{x}$ in the convex hull of $x_0, \ldots, x_k$ at which the objective function value is close to $\min_i h(x_i)$. At the heart of this procedure is a subroutine that given two points y_1, y_2

returns a point y on the line segment joining y_1 and y_2 such that $h(y)$ is close to $\min(h(y_1), h(y_2))$. We find y by performing a bisection search, using the subgradient to infer which direction to move along the line segment. By repeatedly calling the above subroutine with $\bar{x}$ (initialized to x_0) and $x_i, i = 1, \ldots, k$, each time updating $\bar{x}$ to the point returned by the subroutine, at the end we get a point $\bar{x}$ such that $h(\bar{x})$ is close to $\min_i h(x_i)$.

Algorithm details. Let $OPT = \min\{h(x) : x \in \mathcal{P}\}$ denote the optimal solution value. We give the algorithm for an arbitrary convex function $h(.)$ and an arbitrary (rational) polytope $\mathcal{P}$ (the feasible region is bounded). Let $\|u\|$ denote the ℓ_2 norm of u, i.e., $\left(\sum_{i=1}^m u_i^2\right)^{\frac{1}{2}}$. Given a function $g : \mathbb{R}^m \mapsto \mathbb{R}$, we say that g has *Lipschitz constant* (at most) K if $|g(v) - g(u)| \le K\|v - u\|$ for all $u, v \in \mathbb{R}^m$.

Let the objective function $h : \mathbb{R}^m \mapsto \mathbb{R}$ have Lipschitz constant K. We assume that $x \ge \mathbf{0}$ is a defining inequality of $\mathcal{P}$. We also assume that the polytope $\mathcal{P}$ is contained in the ball $B(\mathbf{0}, R) = \{x : \|x\| \le R\}$, and contains a ball of radius r such that $\ln R$ and $\ln\left(\frac{1}{r}\right)$ are polynomially bounded. (It is trivial to obtain such values of R and r for all the optimization problems considered; Lemmas 6.2.4–6.2.6 in [8] show that one can always get such R and r.) Set $V = \min(1, r)$ and define $\lambda = \max\left(1, \max_S \frac{w_S^{\mathrm{II}}}{w_S^{\mathrm{I}}}\right)$.

The bulk of the work is performed by a procedure FindOpt (see Fig. 1). FindOpt takes two parameters γ and ϵ and returns a feasible solution $\bar{x}$ such that $h(\bar{x}) \le OPT/(1-\gamma)+\epsilon$, where $\gamma \le \frac{1}{2}$ without loss of generality, in time polynomial in m and $\ln\left(\frac{KRm}{V\epsilon}\right)$, assuming that one can compute ω-subgradients for a sufficiently small ω. This is the main procedure that uses the ellipsoid method and the notion of ω-subgradients to get close to an optimal solution as discussed earlier. To convert this to a purely multiplicative guarantee, in procedure ConvOpt we first sample a certain number of times from the distribution on scenarios and determine with high probability that either, $x = \mathbf{0}$ is an optimal solution to (SSC-P2) and return this solution, or obtain a lower bound on OPT and then call FindOpt setting γ and ϵ appropriately. By using procedure ConvOpt to bootstrap FindOpt, we may assume FindOpt executes only if OPT is "large", and thus set γ and ϵ so that FindOpt returns a solution of cost at most $(1+\kappa) \cdot OPT$.

By our earlier discussion, one can choose R and V so that $\ln\left(\frac{R}{V}\right)$ is polynomial in the input size. In the analysis we show that for the stochastic set cover problem one can compute ω-subgradients (with a sufficiently high probability), and bound the Lipschitz constant K, so that the entire procedure runs in polynomial time. In Section 4 we generalize these arguments to argue that ω-subgradients can be a computed for a large class of 2-stage programs, so that FindOpt and ConvOpt can be used to find a $(1+\kappa)$-optimal solution in polynomial time.

```
ConvOpt(κ, δ)   [Returns a point $\bar{x}$ such that $h(\bar{x}) \le (1+\kappa)\cdot OPT$ with high probability. Assume $\delta \le \frac{1}{2}$.]
C1. Define $\lambda = \max\left(1, \max_S w_S^{\mathrm{II}}/w_S^{\mathrm{I}}\right)$. Sample $M = \lambda\ln\left(\frac{1}{\delta}\right)$
    times from the distribution on scenarios. Let $X$ = number of
    times a non-null scenario occurs.
C2. If $X = 0$, return $x = \mathbf{0}$ as an optimal solution.
C3. Otherwise (with high probability), $OPT \ge \varrho/\lambda$, where $\varrho = \frac{\delta}{\ln(1/\delta)}$.
    Set $\epsilon = \kappa\varrho/(2\lambda)$, $\gamma = \kappa/3$. Return FindOpt ($\gamma$,$\epsilon$).

FindOpt(γ, ϵ)   [Returns a point $\bar{x}$ such that $h(\bar{x}) \le OPT/(1-\gamma)+\epsilon$. Assume $\gamma \le \frac{1}{2}$.]
O1. Set $k \leftarrow 0$, $y_0 \leftarrow \mathbf{0}$, $N \leftarrow 2m^2\ln\left(\frac{16KR^2}{V\epsilon}\right)$, $n \leftarrow N\log\left(\frac{8NKR}{\epsilon}\right)$,
    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.
    [We maintain the invariant that $E_i$ is an ellipsoid centered at $y_i$ containing the current polytope $\mathcal{P}_k$.]
      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 $\{x \in \mathbb{R}^m : \hat{d}_k \cdot (x - x_k) \le 0\}$.
         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 $a \cdot x \le b$ be a violated inequality, that
         is, $a \cdot y_i > b$, whereas $a \cdot x \le b$ for all $x \in \mathcal{P}_k$. Let $H$
         be the half space $\{x \in \mathbb{R}^m : a \cdot (x - y_i) \le 0\}$.
      c) Set $E_{i+1}$ to be the ellipsoid of minimum volume containing the half-ellipsoid $E_i \cap H$.
O3. Let $k \leftarrow k-1$. We now have a collection of points
    $x_0, \ldots, x_k$ such that each $x_l \in \mathcal{P}_l \subseteq \mathcal{P}_{l-1}$. Return
    FindMin($\omega$; $x_0, \ldots, x_k$).

FindMin(ω; x_0, ..., x_k)
M1. Set $\rho \leftarrow \epsilon/4k$, $\bar{x} \leftarrow x_0$, $N' \leftarrow \log\left(\frac{8kKR}{\epsilon}\right)$.
M2. For $i = 1, \ldots, k$ do the following.
    [We maintain the invariant that $h(\bar{x}) \le \left(\min_{l=0}^{i-1} h(x_l) + (i-1)\rho\right)/(1-\omega)^{(i-1)N'}$.]
      a) We use binary search to find $y$ on the $\bar{x} - x_i$ line segment with value close to $\min(h(\bar{x}), h(x_i))$.
         Initialize $y_1 \leftarrow \bar{x}$, $y_2 \leftarrow x_i$.
      b) For $j = 1, \ldots, N'$ do the following.
         [We maintain that $h(y_1) \le h(\bar{x})/(1-\omega)^{j-1}$, $h(y_2) \le h(x_i)/(1-\omega)^{j-1}$.]
           – Let $y \leftarrow \frac{y_1+y_2}{2}$. Compute an $\omega$-subgradient $\hat{d}$ of
             $h$ at the point $y$. If $\hat{d} \cdot (y_1 - y_2) = 0$, then exit the
             loop. Otherwise exactly one of $\hat{d} \cdot (y_1 - y)$ and
             $\hat{d} \cdot (y_2 - y)$ is positive.
           – If $\hat{d} \cdot (y_1 - y) > 0$, set $y_1 \leftarrow y$, else set $y_2 \leftarrow y$.
      c) Set $\bar{x} \leftarrow y$.
M3. Return $\bar{x}$.
```

Figure 1. The convex optimization algorithm.

3.1. Analysis

We first analyze procedure FindOpt. The following facts are well known (see, e.g., [8]).

Fact 3.2 *The volume of the ball $B(u, D) = \{x \in \mathbb{R}^m : \|x - u\| \leq D\}$ where $u \in \mathbb{R}^m, D \geq 0$ is $D^m \mathsf{vol}(B(\mathbf{0}, 1))$.*

Fact 3.3 *Let $E \subseteq \mathbb{R}^m$ be an ellipsoid and $H \subseteq \mathbb{R}^m$ be a half space passing through the center of E. There is a unique ellipsoid E' of minimum volume containing the half-ellipsoid $E \cap H$ and $\frac{\mathsf{vol}(E')}{\mathsf{vol}(E)} \leq e^{-1/(2m)}$.*

Lemma 3.4 *The points $x_0, \ldots, x_k$ generated by FindOpt satisfy $\min_{i=0}^{k} h(x_i) \leq \big(OPT + \frac{\epsilon}{4}\big)/(1 - \omega)$.*

Proof : Let x^* be an optimal solution. If $\hat{d}_l \cdot (x^* - x_l) \geq 0$ for some l, then $h(x_l) \leq h(x^*)/(1 - \omega)$ since $\hat{d}_l$ is an ω-subgradient at x_l. Otherwise using a volume reduction argument, one can show that there must be a point y lying on some hyperplane $\hat{d}_l \cdot (x - x_l) = 0$ such that $\|y - x^*\| \leq \frac{\epsilon}{4K}$, so $h(x_l) \leq h(y)/(1 - \omega) \leq \big(h(x^*) + \frac{\epsilon}{4}\big)/(1 - \omega)$. ■

Lemma 3.5 *Procedure FindMin returns a point $\bar{x}$ such that $h(\bar{x}) \leq \big(\min_{i=0}^{k} h(x_i) + \frac{\epsilon}{4}\big)/(1 - \omega)^{kN'}$.*

Proof : The proof follows from the invariant stated in step M2 with $i = k + 1$. The inner "For j=…" loop returns a point y such that $h(y) \leq \min(h(\bar{x}), h(x_i))/(1 - \omega)^{N'} + \rho$. So if the invariant holds at the start of iteration i, setting $\bar{x} \leftarrow y$ in step M2c) at the end of iteration i ensures that it holds at the beginning of iteration $i + 1$. ■

Theorem 3.6 *Algorithm FindOpt returns a feasible point $\bar{x}$ satisfying $h(\bar{x}) \leq OPT/(1 - \gamma) + \epsilon$ in time $O\big(T(\omega) \cdot m^2 \ln^2(\frac{KRm}{V\epsilon})\big)$, where $T(\omega)$ is the time taken to compute an ω-subgradient and $\omega = \Theta\big(\gamma/m^2 \ln^2(\frac{KRm}{V\epsilon})\big)$.*

Now we show that ConvOpt works correctly with high probability. We make the very mild assumption that for an optimal solution x^*, in any scenario $A \neq \emptyset$, the total cost of scenario A is at least 1, that is, $w^{\mathrm{I}} \cdot x^* + f_A(x^*) \geq 1$.

Lemma 3.7 *ConvOpt determines with probability at least $1 - \delta$, that $OPT \geq \frac{\varrho}{\lambda}$, or that $x = \mathbf{0}$ is an optimal solution.*

Proof : Note that $\varrho \leq 1$ since $\delta \leq \frac{1}{2}$. Since in every non-null scenario, we incur a cost of at least 1, $OPT \geq q$, where $q = \sum_{A \subseteq U, A \neq \emptyset} p_A$ is the probability of occurrence of a non-null scenario. Let $r = \Pr[X = 0] = (1 - q)^M$. So $r \leq e^{-qM}$ and $r \geq 1 - qM$. If $q \geq \ln\big(\frac{1}{\delta}\big)/M$, then $\Pr[X = 0] \leq \delta$. So with probability at least $1 - \delta$ we will say that $OPT \geq \varrho/\lambda$ which is true since $OPT \geq q \geq \frac{1}{\lambda}$. If $q \leq \delta/M$, then $\Pr[X = 0] \geq 1 - \delta$. We return $x = \mathbf{0}$ as an optimal solution with probability at least $1 - \delta$ which is indeed an optimal solution, because $q \leq \frac{1}{\lambda}$ implies that it is always at least as good to defer to stage II since the expected stage II cost of a set S is at most $q \cdot w^{\mathrm{II}}_S \leq w^{\mathrm{I}}_S$. If $\delta/M < q < \ln\big(\frac{1}{\delta}\big)/M$, then we always return a correct answer since it is both true that $x = \mathbf{0}$ is an optimal solution, and that $OPT \geq q \geq \varrho/\lambda$. ■

We now focus on showing that the algorithm returns a $(1 + \kappa)$-optimal solution to (SSC-P2) with probability at least $1 - \delta$ in polynomial time. Recall that our objective function is $h(x) = \omega^{\mathrm{I}} \cdot x + f(x)$ where $f(x) = \sum_{A \subseteq U} p_A f_A(x)$ and $f_A(x) = \min\big\{\sum_S w^{\mathrm{II}}_S r_{A,S} : \sum_{S: e \in S} r_{A,S} \geq 1 - \sum_{S: e \in S} x_S \ \forall e \in A; \ \ r_{A,S} \geq 0 \ \forall S\big\}$. By taking the dual, we can write $f_A(x) = \max\{\sum_e (1 - \sum_{S: e \in S} x_S) z_{A,e} : z_A \in \mathcal{Q}_A\}$ where $\mathcal{Q}_A = \big\{z \in \mathbb{R}^{|U|} : \sum_{e \in A \cap S} z_e \leq w^{\mathrm{II}}_S$ for all S; $z_e = 0$ for all $e \notin A; \ \ z \geq \mathbf{0}\big\}$. (Here we include variables z_e for $e \notin A$ in the dual for convenience.) Recall that $\lambda = \max\big(1, \max_S \frac{w^{\mathrm{II}}_S}{w^{\mathrm{I}}_S}\big)$. We show that for the function $h(.)$, one can efficiently compute ω-subgradients, and bound the Lipschitz constant K. This follows from 3 facts.

1. Any vector $\hat{d}$ that component-wise approximates a subgradient at x to within a certain accuracy is an ω-subgradient at x (Lemma 3.8).
2. At any point x there is a "nice" subgradient d with components $d_S \in [-w^{\mathrm{II}}_S, w^{\mathrm{I}}_S]$ (Lemma 3.9). This gives a bound on the Lipschitz constant K.
3. Since the components d_S lie in a range bounded multiplicatively by λ, $\mathsf{poly}\big(m, \lambda, \frac{1}{w}\big)$ samples suffice to compute an estimate $\hat{d}$ that component-wise approximates the subgradient with high probability (Corollary 3.11), and thus obtain an ω-subgradient.

Using the above procedure to compute ω-subgradients (with a small enough error probability) in procedure FindOpt, and putting the various pieces together, we show in Theorem 3.12 that ConvOpt returns a point $\bar{x}$ such that $h(\bar{x}) \leq (1 + \kappa) \cdot OPT$ with probability at least $1 - 2\delta$ in time $\mathsf{poly}\big(\text{input size}, \lambda, \frac{1}{\kappa}, \ln(\frac{1}{\delta})\big)$.

Lemma 3.8 *Let d be a subgradient of $h(.)$ at the point $x \in \mathcal{P}$, and suppose $\hat{d}$ is a vector such that $d_S - \omega w^{\mathrm{I}}_S \leq \hat{d}_S \leq d_S$ for all S. Then $\hat{d}$ is an ω-subgradient of $h(.)$ at x.*

Proof : Let $y \in \mathcal{P}$. We have $h(y) - h(x) \geq d \cdot (y - x) = \hat{d} \cdot (y - x) + (d - \hat{d}) \cdot (y - x)$. Since $0 \leq d_S - \hat{d}_S \leq \omega w^{\mathrm{I}}_S$ and $x_S, y_S \geq 0$ for every S we have, $(d - \hat{d}) \cdot (y - x) \geq -(d - \hat{d}) \cdot x \geq -\sum_S \omega w^{\mathrm{I}}_S x_S \geq -\omega h(x)$. ■

Lemma 3.9 *Let $x \in \mathbb{R}^m$, and let z^*_A be an optimal dual solution for scenario A with x as the stage I vector. The vector d with components $d_S = w^{\mathrm{I}}_S - \sum_A p_A \sum_{e \in S} z^*_{A,e}$ is a subgradient at x, and $\|d\| \leq \lambda \|w^{\mathrm{I}}\|$.*

Proof : Let $y \in \mathbb{R}^m$. We have to show that $h(y) - h(x) \geq d \cdot (y - x)$. The dual solution $z_A^* \in \mathcal{Q}_A$ provides a lower bound on $f_A(y)$ for every scenario A, using which one obtains that $h(y) - h(x) \geq \sum_S d_S(y_S - x_S)$. To bound $\|d\|$, observe that $w_S^{\mathrm{I}} - w_S^{\mathrm{II}} \leq d_S \leq w_S^{\mathrm{I}}$, since for every A, $z_{A,e}^* \geq 0 \ \forall e$, $\sum_{e \in S} z_{A,e}^* \leq w_S^{\mathrm{II}}$, and $\sum_{A \subseteq U} p_A = 1$. So $|d_S| \leq \lambda w_S^{\mathrm{I}}$, and hence $\|d\| \leq \lambda \|w^{\mathrm{I}}\|$. ■

Since at any point x there is a subgradient $d(x)$ with $\|d(x)\| \leq \lambda \|w^{\mathrm{I}}\|$, the definition of a subgradient implies that $K \leq \lambda \|w^{\mathrm{I}}\|$ so that $\ln K$ is polynomially bounded.

Using standard Chebyshev and Chernoff bounds one can show the following.

Lemma 3.10 *Let $X \in [-a, b]$ be a random variable, $a, b > 0$, computed by sampling from a distribution π. Let $\mu = \mathrm{E}\big[X\big]$ and $\alpha = \max(1, a/b)$. Then for any $c > 0$, by taking $\frac{100\alpha^2}{3c^2} \ln\big(\frac{1}{\delta}\big)$ independent samples from π, one can compute an estimate $\hat{X}$ such that $\mu - 2cb \leq \hat{X} \leq \mu$ with probability at least $1 - \delta$.*

Corollary 3.11 *At any point $x \in \mathcal{P}$, one can compute an ω-subgradient with probability at least $1 - \delta$ using at most $\frac{400\lambda^2}{3\omega^2} \ln\big(\frac{m}{\delta}\big)$ independent samples from the probability distribution on scenarios.*

Theorem 3.12 *Using the above sampling method to compute ω-subgradients, ConvOpt computes a feasible solution to (SSC-P2) of cost at most $(1 + \kappa) \cdot OPT$ with probability at least $1 - 2\delta$, in time $\mathsf{poly}\big(\text{input size}, \lambda, \frac{1}{\kappa}, \ln(\frac{1}{\delta})\big)$.*

Proof : By Lemma 3.7, we know that if ConvOpt calls FindOpt then with probability at least $1 - \delta$ we have $OPT \geq \frac{\varrho}{\lambda}$. We compute a ω-subgradient at most $N + n$ times where $n = N \log\big(\frac{8NKR}{\epsilon}\big)$, $N = 2m^2 \ln\big(\frac{16KR^2}{V\epsilon}\big)$. With $\omega = \frac{\gamma}{2n}$ and error probability $\frac{\delta}{N+n}$ in Corollary 3.11, we get that $T(\omega) = O\big(\frac{\lambda^2 n^2}{\gamma^2} \ln(\frac{m(N+n)}{\delta})\big)$ samples ensure that each individual vector computed is an ω-subgradient with probability at least $1 - \frac{1}{(N+n)\delta}$. So the net error probability of FindOpt is at most δ, and the error probability of ConvOpt is at most 2δ. Theorem 3.6 shows the performance guarantee, and the time taken is $O\big((N+n)T(\omega)\big) = O\big(n^3\lambda^2(\ln N + \ln(\frac{1}{\delta}))/\gamma^2\big)$, which is $\mathsf{poly}\big(\text{input size}, \lambda, \frac{1}{\kappa}, \ln(\frac{1}{\delta})\big)$. ■

4. A class of solvable stochastic programs

We now show that ConvOpt can be used to solve the following broad class of 2-stage stochastic programs.

$$\min \quad w^{\mathrm{I}} \cdot x + \sum_{A \in \mathcal{A}} p_A f_A(x) \quad \text{s.t.} \quad x \geq \mathbf{0}, \ x \in \mathcal{P} \subseteq \mathbb{R}^m, \tag{Stoc-P}$$

$$\begin{aligned} f_A(x) \ = \ \min \quad & w^A \cdot r_A + q^A \cdot s_A \\ \text{s.t.} \quad & B^A s_A \geq h^A && (2) \\ & D^A s_A + T^A r_A \geq j^A - T^A x && (3) \\ & r_A, s_A \geq \mathbf{0}, \ r_A \in \mathbb{R}^m, \ s_A \in \mathbb{R}^n. \end{aligned}$$

Here $\mathcal{A}$ denotes the set of all possible scenarios. We require that (a) $T^A \geq \mathbf{0}$ for every scenario A, and (b) for every $x \in \mathcal{P}$, $f(x) \geq 0$ and that the primal and dual problems corresponding to $f_A(x)$ are feasible for every scenario A. A sufficient condition for (b) is to insist that $0 \leq f_A(x) < +\infty$ at every point $x \in \mathcal{P}$ and scenario $A \in \mathcal{A}$. We can relax condition (a) somewhat and solve a more general class of programs that allow one to incorporate upper bounds, in certain cases. Observe that this class of stochastic programs is rich enough to model stochastic problems with scenario-dependent recourse (that is, stage II) costs. All the stochastic optimization problems we consider can be expressed as convex programs in the above form, and one can therefore obtain a near-optimal fractional solution for each of these problems in polynomial time. To prevent an exponential blowup in the input, we consider a model where an oracle supplied with scenario A reveals the scenario-dependent data $\big(w^A, q^A, h^A, j^A, B^A, D^A, T^A\big)$. The analysis in Section 3.1 can be extended to show that our algorithm can compute a near-optimal solution to (Stoc-P). Let $h(.)$ be the (convex) objective function. Define $\lambda = \max\big(1, \max_{A \in \mathcal{A}, S} \frac{w_S^A}{w_S^{\mathrm{I}}}\big)$, which we assume is known to the algorithm. The key property we require is the ability to compute ω-subgradients. This will follow from Lemma 4.1 which shows that at any point there is a subgradient whose components have variation bounded by λ (this also bounds the Lipschitz constant), and that one can obtain an ω-subgradient by component-wise approximating this vector.

Lemma 4.1 *Let $x \in \mathcal{P}$ and (u_A^*, z_A^*) be an optimal dual solution for scenario A with x as the stage I vector, where z_A^* is the dual multiplier corresponding to inequalities (3). The vector $d = w^{\mathrm{I}} - \sum_A p_A (T^A)^{\mathrm{T}} z_A^*$ is a subgradient at x, with $\|d\| \leq \lambda \|w^{\mathrm{I}}\|$. If $\hat{d}$ is a vector such that $d - \omega w^{\mathrm{I}} \leq \hat{d} \leq d$, then $\hat{d}$ is an ω-subgradient at x.*

Theorem 4.2 *We can obtain a feasible solution to (Stoc-P) of cost at most $(1 + \kappa) \cdot OPT$ with probability at least $1 - 2\delta$ in time polynomial in the input size, $\frac{1}{\kappa}$, and $\ln\big(\frac{1}{\delta}\big)$.*

In addition, we can use ConvOpt to solve the class of programs (Stoc-P) where the second-stage is specified by a continuous random variable ξ with density function $p(\xi)$.

5. Applications

We give a number of applications for which we prove the first known performance guarantees in the black-box model without any restrictions on the costs in the two stages.

Vertex cover. The stochastic vertex cover problem is a special case of the stochastic set cover problem where we want to cover the edges of a graph by its vertices. The edge set A (i.e., scenario) to be covered is chosen from a probability distribution and is revealed only in stage II; we may choose a vertex v either in stage I paying a cost of w_v^{I}, or in stage II at a cost of w_v^A in scenario A. The previous results for this problem were an 8-approximation algorithm in the black-box model, and a 3-approximation algorithm in the setting where each edge is independently activated, both for the case when $w_v^A = \lambda w_v^{\mathrm{I}}$ for each v and scenario A, due to Gupta et al. [9]; Ravi and Sinha [14] gave a guarantee of 2 when there are only polynomially many scenarios (but with scenario-dependent second-stage costs).

Since the stochastic vertex cover problem is a special case of the stochastic set cover problem, and the deterministic vertex cover LP is known to have an integrality gap of 2, by Corollary 2.2, we obtain, for any $\epsilon > 0$, a $(4+\epsilon)$-approximation algorithm for the stochastic version with black-box distributions and scenario-dependent second-stage costs. This is the first approximation algorithm in this more general model with black-box distributions.

Theorem 5.1 *For any $\epsilon > 0$, there is a $(4+\epsilon)$-approximation algorithm for the stochastic vertex cover problem with arbitrary probability distributions and scenario-dependent stage II costs.*

Using the results of Kolliopoulos & Young [12], we also get algorithms for general stochastic covering problems.

Minimum multicut on trees. In the deterministic minimum multicut problem on trees, we are given a tree with costs w_e on the edges, and pairs of vertices (s_i, t_i). The goal is to remove a minimum-cost set of edges so as to disconnect each (s_i, t_i) pair. In the stochastic variant, the pairs to be disconnected are revealed only in stage II, and we can choose either to "cut" an edge in stage I or in stage II, paying a cost of w_e^{I} or w_e^A in scenario A, respectively. The multicut problem is an instance of the set-cover problem, where we want to cover each (s_i, t_i) path. Using the algorithm of Garg, Vazirani, and Yannakakis [7] for the deterministic setting, and applying Corollary 2.2, we get the following.

Theorem 5.2 *For any $\epsilon > 0$, there is a $(4+\epsilon)$-approximation algorithm for the stochastic minimum multicut problem on trees.*

Facility location problems. In the deterministic uncapacitated facility location (DUFL) problem, given a set of candidate facility locations $\mathcal{F}$, and a set of clients $\mathcal{D}$, we want to open facilities at a subset of the locations in $\mathcal{F}$, and assign each client to an open facility. Opening a facility at location i incurs a cost of f_i, and the cost of assigning client j to facility i is (proportional to) the distance c_{ij} between i and j; the distances c_{ij} form a metric. The goal is to minimize the total facility opening costs and client assignment costs. In the 2-stage stochastic uncapacitated facility location (SUFL) problem, we are given a probability distribution on the clients that are activated and need to be assigned to facilities; a facility i may be opened in stage I or in stage II, incurring a cost of f_i^{I} or f_i^A in scenario A, respectively.

For the special case where $f_i^A = \lambda f_i^{\mathrm{I}}$ for each $i \in \mathcal{F}$ and each scenario A, Gupta et al. [9] gave an 8.45-approximation algorithm in the black-box model, and a 6-approximation algorithm in the setting where each client is activated independently. Ravi and Sinha [14] gave an LP-rounding based 8-approximation algorithm for the polynomial scenarios setting that can handle scenario-dependent facility opening and client assignment costs, where the assignment cost in scenario A is $c_{A,ij} = \gamma^A c_{ij}$ for all i, j. Their rounding algorithm needs to know the optimal fractional solution for *each stage II scenario* which renders it unsuitable when there are exponentially many scenarios.

We improve upon all of these results. We consider a convex programming relaxation of the problem and give a different rounding approach that decides which facilities to open in stage I based on *only the stage I fractional solution.* Combined with our algorithm to solve the convex program, this yields a 3.225-approximation algorithm in the black-box model with scenario-dependent costs. One can write the following convex program for SUFL. We use i to index the facilities in $\mathcal{F}$ and j to index the clients.

$$\min \quad \sum_i f_i^{\mathrm{I}} y_i + \sum_{A \subseteq \mathcal{D}} p_A g_A(y) \quad \text{s.t.} \quad 0 \le y_i \le 1 \quad \forall i, \tag{SUFL-P}$$

$$\begin{aligned} \text{where } g_A(y) = \min \; & \sum_i f_i^A y_{A,i} + \sum_{j \in A, i} c_{ij} x_{A,ij} \\ \text{s.t. } & \sum_i x_{A,ij} \ge 1 && \forall j \in A, \\ & x_{A,ij} \le y_i + y_{A,i} && \forall j \in A, i, \\ & x_{A,ij}, y_{A,i} \ge 0 && \forall j \in A, i. \end{aligned}$$

Let $\rho_{\mathsf{DUFL}} \le 1.52$ [13] denote the integrality gap of DUFL.

Theorem 5.3 *There is a $(3.225+\epsilon)$-approximation algorithm for SUFL based on rounding a near-optimal solution to* (SUFL-P)*. Moreover, the integrality gap of* (SUFL-P) *is at most $2\rho_{\mathsf{DUFL}} \le 3.04$. These results hold even with scenario-dependent assignment costs $c_{ij}^A = \gamma^A c_{ij}$.*

Proof : We first show that the integrality gap of (SUFL-P) is at most $2\rho_{\mathsf{DUFL}}$. The proof is along the lines of the proof of Theorem 2.1. Let y be an optimal solution to (SUFL-P) and (x_A, y_A) be the optimal solution for scenario A given the first-stage decision vector y. Let OPT be the optimal solution value. We will show that we can decouple the first-stage and second-stage decisions, so that one can get an integer solution by separately solving a DUFL problem for stage I and a DUFL problem for each stage II scenario. Fix a scenario A and a client $j \in A$. Let $F_{A,j} = \{i : x_{A,ij} > 0\}$. We write $x_{A,ij} = x^{\mathrm{I}}_{A,ij} + x^{\mathrm{II}}_{A,ij}$ where $x^{\mathrm{I}}_{A,ij} \leq y_i$ and $x^{\mathrm{II}}_{A,ij} \leq y_{A,i}$. Since $x_{A,ij} \leq y_i + y_{A,i}$ we can always split $x_{A,ij}$ in the above way. Observe that j must be assigned to an extent of at least $\frac{1}{2}$ either by the assignment $\{x^{\mathrm{I}}_{A,ij}\}$ or by the assignment $\{x^{\mathrm{II}}_{A,ij}\}$, that is, either $\sum_i x^{\mathrm{I}}_{A,ij} \geq \frac{1}{2}$ or $\sum_i x^{\mathrm{II}}_{A,ij} \geq \frac{1}{2}$. In the former case, we will assign j to a facility opened in stage I, and in the latter case we will assign j to a facility opened in stage II.

More precisely, for any client j, consider the set of scenarios $\mathcal{S}_j = \{A \subseteq \mathcal{D} : \sum_i x^{\mathrm{I}}_{A,ij} \geq \frac{1}{2}\}$. For our stage I decisions, we shall construct a feasible fractional solution for a DUFL instance in which the facility costs are f^{I}_i, the assignment costs are c_{ij}, and each client j has a *demand* equal to $\sum_{A \in \mathcal{S}_j} p_A$; we then round this fractional solution to an integer solution using known algorithms for DUFL. We first construct a feasible solution in which there is a client (j, A) for each scenario $A \in \mathcal{S}_j$, with demand p_A, and then coalesce these scenario-dependent clients into one. Consider (j, A) such that $A \in \mathcal{S}_j$. We can obtain a feasible solution by setting $\hat{x}_{A,ij} = \min\bigl(1, 2x^{\mathrm{I}}_{A,ij}\bigr)$ and $\hat{y}_i = \min\bigl(1, 2y_i\bigr)$ for each $i \in \mathcal{F}$. (Note that a client may be assigned to an extent greater than 1.) However, the $\hat{y}_i$ values do not depend on the scenario and given the $\hat{y}_i$ values, we can re-optimize the fractional assignment for each client j: first reset $\hat{x}_{A,ij} = 0$, then considering facilities in non-decreasing order of c_{ij}, set $\hat{x}_{A,i'j} = \min\bigl(\hat{y}_{i'}, 1 - \sum_i \hat{x}_{A,ij}\bigr)$ for each i'. But this new fractional assignment is completely determined by the $\hat{y}_i$ values and *does not depend on* A, and so we can now view all of these clients (j, A) as one client j with demand $\sum_{A \in \mathcal{S}_j} p_A$. The facility cost of this fractional solution is at most $2\sum_i f^{\mathrm{I}}_i y_i$, and the assignment cost is no more than the one for the scenario-dependent clients, $2\sum_{i,j}\sum_{A \in \mathcal{S}_j} p_A c_{ij} x^{\mathrm{I}}_{A,ij} \leq 2\sum_{i,j}\sum_{A \in \mathcal{S}_j} p_A c_{ij} x_{A,ij}$. Using the fact that the integrality gap of DUFL is ρ_{DUFL}, given this DUFL instance with fractional solution $(\hat{x}, \hat{y})$, we can now obtain an integer solution $(\tilde{x}, \tilde{y})$ of cost at most $2\rho_{\mathsf{DUFL}}\bigl(\sum_i f^{\mathrm{I}}_i y_i + \sum_{i,j}\sum_{A \in \mathcal{S}_j} p_A c_{ij} x_{A,ij}\bigr)$; this determines the set of facilities to open in stage I, and for each client j takes care of the scenarios in $\mathcal{S}_j$.

In any scenario A, each client $j \in A$ such that $A \in \mathcal{S}_j$ is assigned to the stage I facility given by the assignment $\tilde{x}$. To assign the remaining clients, we solve a DUFL instance with client set $D_A = \{j \in A : A \notin \mathcal{S}_j\}$. Since $A \notin \mathcal{S}_j$, we have that $\sum_i x^{\mathrm{II}}_{A,ij} \geq \frac{1}{2}$, and hence if we reset $\hat{x}_{A,ij} = \min\bigl(1, 2x^{\mathrm{II}}_{A,ij}\bigr)$, $\hat{y}_{A,i} = \min\bigl(1, 2y_{A,i}\bigr)$ for each $i \in \mathcal{F}$, we get a feasible solution for this set of clients. Again, we can get an integer solution of cost at most $2\rho_{\mathsf{DUFL}}\bigl(\sum_i f^A_i y_{A,i} + \sum_{i,j \in D_A} c_{ij} x_{A,ij}\bigr)$. This solution tells us which facilities to open in scenario A and how to assign the clients in D_A. Hence, the overall cost of the solution with first-stage facilities $\tilde{y}$ is at most $2\rho_{\mathsf{DUFL}} \cdot OPT$, implying that the integrality gap is at most $2\rho_{\mathsf{DUFL}}$.

To obtain the approximation algorithm, we first obtain a near-optimal solution y in polynomial time. The difficulty in converting the proof of the integrality gap into a rounding algorithm is that the algorithm that shows that $\rho_{\mathsf{DUFL}} \leq 1.52$ due to [13] requires knowledge of the client demands, whereas we do not know the demand $\sum_{A \in \mathcal{S}_j} p_A$ of a client j, and might not be able to even estimate it by sampling, since the probability p_A could be extremely small. We therefore need an approximation algorithm for DUFL that works without explicit knowledge of the client demands. Swamy [15] (Section 2.4) gives an algorithm with this property; the algorithm converts any fractional solution to an integer solution based on just the fractional facility variables, increasing the cost by a factor of at most 1.705. We use this algorithm to obtain a 3.225-approximation algorithm by modifying the definition of $\mathcal{S}_j$ slightly so as to balance the contribution from stages I and II. ■

The analysis extends with minor notational changes to the case where we have arbitrary scenario-dependent demands d^A_j, and/or assignment costs $c^A_{ij} = \gamma^A c_{ij}$. Our approach yields the first constant-factor approximation algorithms for other stochastic facility location problems, such as facility location with penalties, or soft capacities, or service installation costs. In each case, we solve the relaxation of the stochastic integer program using the algorithm in Figure 1, and round the near-optimal solution by using a rounding algorithm for the deterministic problem in conjunction with a variant of the rounding procedure detailed above.

Multicommodity flow. We consider a stochastic version of a multicommodity flow problem where have to buy capacity to install on the edges so that one can concurrently ship d^A_i units of commodity i from its source s_i to its sink t_i in each scenario A. We can either purchase capacity on an edge e in stage I paying cost c^{I}_e, or wait until the exact demands are known and buy capacity at cost c^A_e in scenario A in stage II; the total amount of capacity that we can install on an edge is limited by u_e. The goal is to minimize the total (expected) cost of installing capacity. The stochastic multicommodity flow problem can be formulated as follows: minimize $\sum_e c^{\mathrm{I}}_e x_e + \sum_{A \in \mathcal{A}} p_A g_A(x)$ ($\mathcal{A}$ is the set of all scenarios) subject to $0 \leq x_e \leq u_e$ for each e, and $g_A(x)$ is the minimum value of $\sum_e c^A_e y_{A,e}$ subject to the constraints that for each i, the total flow for (s_i, t_i) is at least d^A_i, for each edge e, the total flow on e is at most $x_e + y_{A,e}$, and also at most u_e (this encodes that $x_e + y_{A,e} \leq u_e$).

Immorlica et al. [10] considered the single-commodity version of this problem and gave an algorithm based on writing an LP that enumerates all scenarios, one for each possible demand value, and solving the LP to compute the optimal first-stage decisions. Consequently, their running time depends on the *maximum demand* D that may be realized. This approach suffers from the "curse of dimensionality" and does not work well in the multicommodity setting, since even if the maximum demand is 1, there are still an exponential number of scenarios to enumerate. Note that there are no integrality constraints, that is, one can install fractional amounts of capacity. We can solve the stochastic multicommodity flow program using the algorithm in Figure 1. Whereas our running time depends on λ, the ratio of stage II and stage I costs, it does not depend on D.

Theorem 5.4 *For any $\epsilon > 0$, the stochastic concurrent multicommodity flow problem can be approximated to within a factor of $(1+\epsilon)$ in polynomial time.*

6. The dependence of the running time on λ

We have remarked previously that the running time of our algorithm (and that of [9]) depends on λ, the maximum ratio between costs in the two stages. We first argue that in the black-box model, this is necessary, and then provide stronger conditions on the way in which the distribution is specified that allows this dependence to be avoided.

Consider an instance of SSC with universe $U = \{e\}$ and just one set $S = U$, where $w^{\mathrm{I}}_S = 1,\ w^{\mathrm{II}}_S = \lambda$. Let p denote the probability that scenario $\{e\}$ occurs (which is unknown to the algorithm). The only decision here is whether to buy set S in stage I or to buy it in stage II. Let $\mathcal{A}_N$ denote an algorithm that draws exactly N samples. Let O^* denote the value of the *integer* optimum solution. It is relatively straightforward to prove the following result.

Theorem 6.1 *If $\mathcal{A}_N$ returns a (fractional) solution of cost at most $c \cdot O^*$ with probability at least $1-\delta$ where $1 \le c < \frac{\lambda}{2}$, then it must be that $N \ge \big(\lambda \ln(\frac{1}{\delta} - 1)\big)/2c$. Hence, if algorithm $\mathcal{A}_N$ returns a solution of expected cost at most $c \cdot O^*$ where $1 \le c < \frac{\lambda}{6}$, then it must be that $N \ge (\lambda \ln 2)/6c$.*

Suppose that for the stochastic set cover problem, $w^A_S = w^{\mathrm{II}}_S$ for each S, A and for every element e, a) we know the activation probability p_e, and b) we can sample scenarios from $\{A \subseteq U : e \in A\}$, with the probability of generating $A \ni e$ being p_A/p_e. Note that if the elements are activated independently (as considered in [10, 9]), then these conditions are satisfied. More generally, consider the class of problems given in Section 4 where w^A, q^A, D^A, T^A do not depend on the scenario A and $D^A = D \ge \mathbf{0}$. Fix an indexing of the rows of T (and D); we require that for every scenario A, $B^A \ge \mathbf{0}$, and j^A_e is either 0 or j_e. Suppose we know $p_e = \sum_{A \in \mathcal{A}: j^A_e > 0} p_A$, and can sample scenarios conditioned on the fact that $j^A_e > 0$. Using this additional structure we obtain the following result.

Theorem 6.2 *At any point $x \in \mathcal{P}$, an ω-subgradient of $h(.)$ can be computed with probability at least $1-\delta$ in time polynomial in the input size, $\frac{1}{\omega}$, and $\ln\big(\frac{1}{\delta}\big)$. Thus, ConvOpt can be implemented to run in time that does not depend on λ.*

Acknowledgments We thank Mike Todd and Shane Henderson for useful discussions and very helpful suggestions.

References

[1] J. R. Birge and F. V. Louveaux. *Introduction to Stochastic Programming*. Springer-Verlag, New York, 1997.

[2] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233–235, 1979.

[3] G. B. Dantzig. Linear programming under uncertainty. *Management Science*, 1:197–206, 1955.

[4] S. Dye, L. Stougie, and A. Tomasgard. The stochastic single resource service-provision problem. *Naval Research Logistics*, 50(8):869–887, 2003. Also appeared as COSOR-Memorandum 99-13, Dept. of Mathematics and Computer Sc., Eindhoven, Tech. Univ., Eindhoven, 1999.

[5] M. Dyer, R. Kannan, and L. Stougie. A simple randomised algorithm for convex optimisation. SPOR-Report 2002-05, Dept. of Mathematics and Computer Science, Eindhoven Technical University, Eindhoven, 2002.

[6] M. Dyer and L. Stougie. Computational complexity of stochastic programming problems. SPOR-Report 2003-20, Dept. of Mathematics and Computer Science, Eindhoven Technical Univ., Eindhoven, 2003.

[7] N. Garg, V. Vazirani, and M. Yannakakis. Primal dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.

[8] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, New York, 1988.

[9] A. Gupta, M. Pál, R. Ravi, and A. Sinha. Boosted sampling: approximation algorithms for stochastic optimization. In *Proceedings of 36th STOC*, pages 417–426, 2004.

[10] N. Immorlica, D. Karger, M. Minkoff, and V. Mirrokni. On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems. In *Proceedings of 15th SODA*, pages 684–693, 2004.

[11] A. J. Kleywegt, A. Shapiro, and T. Homem-De-Mello. The sample average approximation method for stochastic discrete optimization. *SIAM J. of Optimization*, 12:479–502, 2001.

[12] S. Kolliopoulos and N. Young. Tight approximation results for general covering integer programs. In *Proceedings of 42nd FOCS*, pages 522–528, 2001.

[13] M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location. In *Proceedings of 5th APPROX*, pages 229–242, 2002.

[14] R. Ravi and A. Sinha. Hedging uncertainty: approximation algorithms for stochastic optimization problems. In *Proceedings of 10th IPCO*, pages 101–115, 2004.

[15] C. Swamy. *Approximation Algorithms for Clustering Problems*. Ph.D. thesis, Cornell University, Ithaca, NY, 2004.