# An Approximation Scheme for Stochastic Linear Programming and its Application to Stochastic Integer Programs*

David B. Shmoys† Chaitanya Swamy‡

**Abstract**

Stochastic optimization problems attempt to model uncertainty in the data by assuming that the input is specified by 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 show that for a broad class of 2-stage linear models with recourse, one can, for any $\epsilon > 0$, in time polynomial in $\frac{1}{\epsilon}$ and the size of the input, compute a solution of value within a factor $(1+\epsilon)$ of the optimum, in spite of the fact that exponentially many second-stage scenarios may occur. In conjunction with a suitable rounding scheme, this yields the first approximation algorithms for 2-stage stochastic integer optimization problems where the underlying random data is given by a "black box" and no restrictions are placed on the costs in the two stages. Our rounding approach for stochastic integer programs shows that an approximation algorithm for a deterministic analogue yields, with a small constant-factor loss, provably near-optimal solutions for the stochastic generalization. Among the range of applications we consider are stochastic versions of the multicommodity flow, set covering, and facility location problems.

## 1 Introduction

The study of stochastic optimization problems dates back to the 1950's and the work of Dantzig [7] and Beale [2], 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 from the point of view of computational complexity theory; even extremely specialized (sub)problems are $\#P$-complete.

We focus on an important and widely used model in stochastic programming, the *2-stage recourse model*: first, given only distributional information about (some of) the data, one commits on initial (first-stage) actions, and then once the actual data is realized, according to the distribution, further *recourse actions* can be taken, so that one can augment the earlier solution to satisfy the revealed requirements, if necessary. Typically the recourse actions entail making decisions in rapid reaction to the observed scenario, that is, at the "last minute," and are therefore costlier than decisions made ahead of time. Many applications come under this setting and much of the textbook of Birge and Louveaux [4] is devoted to models and algorithms for this class of problems. Consider for example the problem of opening facilities to serve a

*A preliminary version [25] appeared in the Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, 2004.

†shmoys@orie.cornell.edu. School of ORIE, Cornell University, Ithaca, NY 14853. Research supported partially by NSF grant CCR-9912422.

‡cswamy@ist.caltech.edu. Work done while the author was a student at the Department of Computer Science, Cornell University, Ithaca, NY 14853. Research supported partially by NSF grant CCR-9912422.

set of clients. Initially given only distributional information (likelihood estimates) about the client demands (in addition to deterministic data for the facility and assignment costs), the first-stage decisions consist of deciding which facilities to open initially; but then once the actual input (the client demands) is realized according to this distribution one can extend (in a second stage) the solution by opening more facilities (if necessary) incurring a certain recourse cost. These recourse costs are typically more expensive than the original ones, e.g., because opening a facility at the last minute to handle excess demand might involve deploying resources at a much smaller lead time, may be different for the different facilities, and could even depend on the scenario that materializes.

We shall initially focus on the following stochastic generalization of the set cover problem: we are given a family of sets $S_1, \ldots, S_m$ over a ground set $U$, where each set $S$ has an *a priori* weight $w_S^{\mathrm{I}}$, and an *a posteriori* weight $w_S^{\mathrm{II}}$ associated with it. In the first stage, one selects some of these sets, incurring each associated weight $w_S^{\mathrm{I}}$, 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 [8], and later, Ravi and Sinha [22] 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 [14] 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 $\lambda$ such that $w_S^{\mathrm{II}} = \lambda w_S^{\mathrm{I}}$ for each set $S$. Gupta, Pál, Ravi, and Sinha [13] 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 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 [22]). We obtain the *first approximation algorithms* for a variety of applications including the stochastic versions of the multicommodity flow, set covering, and facility location problems, without placing any restrictions on the underlying probability distribution or the cost structure of the input. Moreover, the performance guarantees we obtain, in some of these applications, are improvements on the results obtained by [22, 14, 13] in weaker models.

We show that, given a class of (deterministic) set cover instances (e.g., the vertex cover problem), for which we have a $\rho$-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 [13].

Our result has two principle 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. This allows us to decouple the two stages (and indeed each of the scenarios for the second stage), and 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. It is important to 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, allowing one to "reduce" the stochastic problem to the deterministic analogue. Furthermore, if we consider the case when there are only a polynomial number of scenarios, then this rounding approach is sufficient to yield polynomial time algorithms with 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 these stochastic linear programming problems in spite of the fact that they are $\#P$-hard. (Not surprisingly, 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. If the current ellipsoid center is infeasible, then one uses a violated inequality to generate the hyperplane; otherwise, the hyperplane is obtained by computing the subgradient of the objective function at the ellipsoid center. 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 the solution for the stochastic integer optimization problem.

Our algorithm returns a solution of objective function value within $(1+\epsilon)$ of the optimum for any $\epsilon > 0$, in running time bounded by a polynomial in the input size, $\frac{1}{\epsilon}$, and the maximum *ratio* between the second- and first-stage costs. The algorithm works for both discrete and continuous distributions, and does not require any assumptions about the probability distribution (or the cost structure of the input). This result should be viewed as indicative of the fact that an exponential number of scenarios is *not* an insurmountable impediment to the design of efficient algorithms for these problems. (For example, in [3], this viewpoint was used as a motivation for considering so-called "robust" versions of deterministic optimization problems, as opposed to their stochastic versions.) We believe that this work could lead to both, the development of algorithms with provable guarantees for more general stochastic optimization models, such as multistage problems, and better computational procedures for solving 2-stage stochastic LPs.

**Related Work** Two-stage stochastic programs, both linear and integer programs, have been extensively studied in the stochastic programming literature, but relatively little is known about polynomial-time algorithms that deliver solutions that are provably good approximations to the optimum stochastic linear or integer program objective value.

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. Note that the estimated distribution has support of size at most the number of samples. While there are results that prove asymptotic convergence to the (true) optimal solution as the number of samples goes to infinity (see [23] and the references therein), and it has been reported that some of these

algorithms converge quickly in practice [18, 29], fewer results are known that give bounds on the rate of convergence and the sample size required to obtain a near-optimal solution (with high probability). Kleywegt, Shapiro, and Homem-De-Mello [16] (see also [23]) prove a bound on the sample size required to obtain a near-optimal solution that is polynomial in the dimension, but depends on the variance of a certain quantity (calculated using the scenario distribution) that might be exponential in the input size. Whereas the running time of our algorithm depends on the parameter $\lambda$, it does not depend on the underlying probability distribution. The algorithm of [13] also has the same dependence on $\lambda$. In fact, this dependence on $\lambda$ is unavoidable; we show that a performance guarantee of $\rho$ 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, $\lambda$, and $\frac{1}{\epsilon}$.

Dyer, Kannan, and Stougie [9], and Nesterov and Vial [21] also give algorithms for stochastic optimization and we now compare our algorithm with their work. Dyer et al. 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. In contrast, our algorithm is based on approximating the subgradient at a given point, and consequently the running time depends on the variation in the subgradient vector components which we show is bounded by $\lambda$. The algorithm of Nesterov and Vial also employs subgradients and uses a subgradient-descent approach; they obtain a running time that depends on the maximum variation in the objective function value in the feasible region, which in general is not polynomially bounded.

The first worst-case analysis of approximation algorithms for 2-stage stochastic integer programming problems appears to be due to Dye et al. [8], who give a constant performance guarantee for a resource provisioning problem (a maximization problem) in the polynomial scenarios setting based on rounding a linear program. Ravi and Sinha [22], and independently Immorlica et al. [14], and subsequently Gupta et al. [13] 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, [13] give an 8-approximation algorithm in the black-box model with proportional costs, and [22] 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 [13] give a guarantee of 8.45 in the black-box model with proportional costs, and [22] 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 [22], our results also extend to the case with scenario-dependent second-stage costs.

The rest of the paper is organized as follows. In Section 2, we focus on a stochastic generalization of the set cover problem to illustrate our rounding approach and motivate a compact convex programming relaxation of the problem. In Sections 3,4 we describe the algorithm to solve the resulting convex program and prove that it computes a near-optimal fractional solution to the stochastic set cover problem; Section 5 extends this analysis to show that the algorithm can be used to solve a rich class of 2-stage programs to near-optimality. In Section 6, we consider various applications and present approximation algorithms for 2-stage stochastic integer problems. Section 7 proves a lower bound on the sample size required in the black-box model, and we conclude in Section 8.

## 2 An illustrative example: the stochastic set cover problem

The deterministic weighted set-cover problem (DSC) is the following: given a universe $U$ of elements $e_1, \ldots, e_n$ and a collection of subsets of $U$, $S_1, \ldots, S_m$ with set $S_i$ having weight $w_i$, we want to choose a minimum weight collection of sets so that every element $e_j, j = 1, \ldots, n$, is included in some chosen set. The problem can be formulated as an integer program and the integrality constraints can be relaxed to yield the following linear program:

$$\min \quad \sum_S w_S x_S \quad \text{subject to} \quad \sum_{S:e\in S} x_S \geq 1 \ \text{ for all } e; \quad x_S \geq 0 \ \text{ for all } S. \tag{SC-P}$$

Let $OPT_{Det}$ denote the optimal value of (SC-P).

In the 2-stage stochastic generalization of the 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 set of elements $A \subseteq U$ to be covered. For our purposes, a scenario is just some subset of the elements $A \subseteq U$. We will assume without loss of generality that the set of all possible scenarios is 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 the quantities $p_A$ in the algorithm.* Throughout we will use $A$ to index the scenarios.

Each set $S_i$ has two weights associated with it, an *a priori* weight $w_i^{\mathrm{I}}$, and an *a posteriori* weight $w_i^{\mathrm{II}}$. In the first stage, one selects some of these sets, incurring a cost of $w_S^{\mathrm{I}}$ for choosing set $S$, then a scenario $A \subseteq U$ 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 sets selected (in both stages). The aim is to minimize the expected total cost of the solution, that is, the sum of the cost incurred in stage I, and the expectation over all stage II scenarios $A$ of the stage II cost of scenario $A$. The two-stage problem can also be formulated as an integer program and the integrality constraints can be relaxed to yield a linear program.

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

Variable $x_S$ indicates whether set $S$ is chosen in stage I, and $r_{A,S}$ indicates if set $S$ is chosen in scenario $A$. 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 $2\rho$. Thus, an optimal solution to* (SSC-P2) *gives a $2\rho$-approximation algorithm.*

**Proof :** Let $h(.)$ denote 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 (i.e., a solution to (SC-P)) 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 $\rho \cdot \sum_S 2w^{\mathrm{I}}_S x_S$. 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 $\rho \cdot \sum_S w^{\mathrm{II}}_S r_{A,S}$. 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* (SC-P) *is $\rho$, then the integrality gap of* (SSC-P1) *is at most $2\rho$.*

It is well known [6] that "the greedy algorithm" returns a solution to the deterministic set cover problem of weight at most $\ln n \cdot OPT_{Det}$. Thus, 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 [22]), 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 $2\rho$-approximation algorithm. This motivates the following formulation with only the stage I variables $x_S$.

$$\min \quad \sum_S w^{\mathrm{I}}_S x_S + f(x) \quad \text{subject to} \quad x_S \geq 0 \qquad \text{for all } S, \tag{SSC-P2}$$

$$\text{where } f(x) = \sum_{A\subseteq U} p_A f_A(x),$$

$$\text{and} \quad f_A(x) = \min \sum_S w^{\mathrm{II}}_S r_{A,S}$$

$$\text{s.t.} \sum_{S:e\in S} r_{A,S} \geq 1 - \sum_{S:e\in S} x_S \qquad \text{for all } e \in A,$$

$$r_{A,S} \geq 0 \qquad \text{for all } S.$$

It is straightforward to show that (SSC-P2) and (SSC-P1) are equivalent mathematical programs, and that the objective function of the latter is convex.

# 3 Solving the convex program: algorithm overview

We now leverage the fact that the objective function of (SSC-P2) is convex to show that the ellipsoid method can be adapted to find a near-optimal solution to (SSC-P2) in polynomial time. In doing so, a significant difficulty that we need to overcome, is the fact *that evaluating $f(x)$, and hence the objective function, may in general be #P-hard.* Section 5 generalizes the arguments to show that the algorithm can be applied to a more general class of 2-stage stochastic programs.

The ellipsoid method starts by containing the feasible region within a ball and generates 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 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 $\left\{x \in \mathbb{R}^m : 0 \leq x_S \leq 1 \text{ for all } S\right\}$, and let $h(x)$ be the (convex) objective function $w^{\mathrm{I}} \cdot x + f(x)$. If the current iterate $x_i$ is feasible, then one could add the constraint $h(x) \leq h(x_i)$ 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, we cannot even decide whether the current point is feasible (or even almost-feasible), and finding a separating hyperplane 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.

**Definition 3.1** *Let $g : \mathbb{R}^m \mapsto \mathbb{R}$ be a function. We say that $d$ is a subgradient of $g$ at the point $u$ if the inequality $g(v) - g(u) \geq d \cdot (v - u)$ holds for every $v \in \mathbb{R}^m$.*

Note that the subgradient at a given point need not be unique. It is known (see [5]) that if a function is convex then it has a subgradient at every point. If $d_i$ is the 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 $\mathcal{P}_{i+1} = \mathcal{P}_i \cap \{x : d_i \cdot (x - x_i) \leq 0\}$. 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 following notion of an approximate subgradient which is crucial to the working of our algorithm.

**Definition 3.2** *We say that $\hat{d}$ is a $(\omega, \mathcal{D})$-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 $g(v) - g(u) \geq \hat{d} \cdot (v - u) - \omega g(u)$.*

We will only use $(\omega, \mathcal{P})$-subgradients in the algorithm, which we abbreviate and denote as $\omega$-subgradients from now on. We show that one can compute, with high probability, an $\omega$-subgradient of $h(.)$ at any point $x$, by sampling from the black box on scenarios. At a feasible point $x_i$, we compute an $\omega$-subgradient $\hat{d}_i$ and add the inequality $\hat{d}_i \cdot (x - x_i) \leq 0$ to chop off a region of $\mathcal{P}_i$ and get the polytope $\mathcal{P}_{i+1}$. Since we use an approximate subgradient to generate the cut, we might discard points with objective function value better than that at the current iterate $x_i$; but one can show that for each point $y$ in $\mathcal{P}_{i+1} \setminus \mathcal{P}_i$, $h(y) \geq (1 - \omega)h(x_i)$, implying that a discarded point has function value not much better 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 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 a feasible point $x$ (and not the function value $f(x)$), 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.

One last hurdle remains however. Since we cannot compute $h(x)$ we will not be able to compute 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 $\min_i h(x_i)$ (without, however, computing these values). 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 $x_i$ for $i = 1, \ldots, k$, 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 $\min_i h(x_i)$.

# 4 Algorithm details and analysis

Let $OPT = \min\{h(x) : x \in \mathcal{P}\}$ denote the optimal solution value. We describe the algorithm for an arbitrary convex function $h(.)$ and an arbitrary (rational) polytope $\mathcal{P}$ (the feasible region is bounded). We use $\|u\|$ to denote the $\ell_2$ norm of $u$, i.e., $\left(\sum_{i=1}^{m} u_i^2\right)^{\frac{1}{2}}$. The following definition makes precise the notion of bounded variation.

**Definition 4.1 (Lipschitz Condition)** *Given a function $g : \mathbb{R}^m \mapsto \mathbb{R}$, we say that $g$ has Lipschitz constant (at most) $K$ if $|g(v) - g(u)| \leq K\|v - u\|$ for all $u, v \in \mathbb{R}^m$.*

Let the objective function $h : \mathbb{R}^m \mapsto \mathbb{R}$ have Lipschitz constant $K$. We shall assume that $x \geq \mathbf{0}$ is a defining inequality of $\mathcal{P}$. We also assume that the polytope $\mathcal{P}$ is contained in the ball $B(\mathbf{0}, R) = \{x : \|x\| \leq R\}$, and contains a ball of radius $r$ such that $\ln R$ and $\ln\left(\frac{1}{r}\right)$ are polynomially bounded. (For all the optimization problems considered, it is trivial to set $R$ and $r$ so that they satisfy all these properties; moreover Lemmas 6.2.4–6.2.6 in [12] show that one can always get such $R$ and $r$.) Set $V = \min(1, r)$ and define $\lambda = \max\left(1, \max_S \frac{w_S^{\text{II}}}{w_S^{\text{I}}}\right)$. We assume that $\lambda$ (or an upper bounds on it) is known to the algorithm.

The bulk of the work is performed by a procedure FindOpt (see Fig. 1). FindOpt takes two parameters $\gamma$ and $\epsilon$ and returns a feasible solution $\bar{x}$ such that $h(\bar{x}) \leq OPT/(1-\gamma) + \epsilon$, where $\gamma \leq \frac{1}{2}$ without loss of generality, assuming that one can compute $\omega$-subgradients for a sufficiently small $\omega$, in time polynomial in the dimension $m$, and $\ln\left(\frac{KRm}{V\epsilon}\right)$ (excluding the time to compute the $\omega$-subgradients). This is the main procedure that uses the ellipsoid method and the notion of $\omega$-subgradients to get close to an optimal solution as discussed earlier. It uses a subroutine FindMin which takes a set of feasible points $x_0, \ldots, x_k$, and returns a feasible point having function value close to $\min_i h(x_i)$ using $\omega$-subgradients. To convert this to a purely multiplicative guarantee we use a procedure ConvOpt to bootstrap algorithm FindOpt. 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 and return this solution, or obtain a lower bound on $OPT$ and then call FindOpt setting $\gamma$ and $\epsilon$ appropriately. By wrapping FindOpt within procedure ConvOpt, we may assume that FindOpt executes only if $OPT$ is "large," and thus set $\gamma$ and $\epsilon$ so that FindOpt returns a solution of cost at most $(1 + \kappa) \cdot OPT$.

For ease of understanding, we divide the analysis into two parts. In Section 4.1 and prove that procedure FindOpt returns a solution of objective value at most $OPT/(1-\gamma)+\epsilon$ in time polynomial in the dimension $m$, and $\ln\left(\frac{KRm}{V\epsilon}\right)$. We also show that, with high probability, ConvOpt correctly determines if $OPT$ is large enough and if FindOpt should be called. By our earlier discussion, one can always choose $R$ and $V$ so that $\ln\left(\frac{R}{V}\right)$ is polynomial in the input size. In Section 4.2 we show that for the stochastic set cover problem, one can efficiently compute $\omega$-subgradients (with a sufficiently high probability), and bound the parameter $K$, so that the entire procedure runs in polynomial time and delivers a solution of cost at most $(1 + \kappa) \cdot OPT$ with high probability. In Section 5 we generalize these arguments and show that for a large class of 2-stage stochastic programs, one can efficiently compute $\omega$-subgradients and bound the Lipschitz constant $K$, to argue that procedures FindOpt and ConvOpt can be used to find a $(1 + \kappa)$-optimal solution in polynomial time.

## 4.1 Analysis of the Generic Algorithm

We first analyze procedure FindOpt. Clearly we maintain the stated invariant. We will need the following well known facts (see for example, [12]).

**ConvOpt($\kappa, \delta$)** [Returns $\bar{x}$ such that $h(\bar{x}) \leq (1+\kappa) \cdot OPT$ with probability at least $1-\delta$. Assume $\delta \leq \frac{1}{2}$.]

C1. Define $\lambda = \max\bigl(1, \max_S w_S^{\mathrm{II}}/w_S^{\mathrm{I}}\bigr)$. Sample $M = \lambda \ln\bigl(\frac{1}{\delta}\bigr)$ 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), $OPT \geq \varrho/\lambda$, where $\varrho = \frac{\delta}{\ln(1/\delta)}$. Set $\epsilon = \kappa\varrho/(2\lambda)$, $\gamma = \kappa/3$. Return $\mathsf{Findopt}(\gamma, \epsilon)$.

**FindOpt($\gamma, \epsilon$)** [Returns a point $\bar{x}$ such that $h(\bar{x}) \leq OPT/(1-\gamma) + \epsilon$. Assume $\gamma \leq \frac{1}{2}$.]

O1. Set $k \leftarrow 0$, $y_0 \leftarrow \mathbf{0}$, $N \leftarrow 2m^2 \ln\bigl(\frac{16KR^2}{V\epsilon}\bigr)$, $n \leftarrow N \log\bigl(\frac{8NKR}{\epsilon}\bigr)$, 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 $\{x \in \mathbb{R}^m : \hat{d}_k \cdot (x - x_k) \leq 0\}$. 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. 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 $\mathsf{Findmin}(\omega; x_0, \ldots, x_k)$.

**FindMin($\omega; x_0, \ldots, x_k$)**

M1. Set $\rho \leftarrow \epsilon/4k$, $\bar{x} \leftarrow x_0$, $N' \leftarrow \log\bigl(\frac{8kKR}{\epsilon}\bigr)$.

M2. For $i = 1, \ldots, k$ do the following.
[We maintain the invariant that $h(\bar{x}) \leq \bigl(\min_{l=0}^{i-1} h(x_l) + (i-1)\rho\bigr)/(1-\omega)^{(i-1)N'}$.]

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 $h(y_1) \leq h(\bar{x})/(1-\omega)^{j-1}$, $h(y_2) \leq h(x_i)/(1-\omega)^{j-1}$.]

– 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.

**Fact 4.2** *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 4.3** *Let $E \subseteq \mathbb{R}^m$ be an ellipsoid and $H \subseteq \mathbb{R}^m$ be a half space passing through the center of $E$. Then there is a unique ellipsoid $E'$ of minimum volume containing the half-ellipsoid $E \cap H$ and $\frac{\mathsf{vol}E'}{\mathsf{vol}E} \leq e^{-1/(2m)}$.*

**Fact 4.4** *Let $T : \mathbb{R}^m \mapsto \mathbb{R}^m$ be an affine transformation with $T(x) = Qx + t$, where $\det Q \neq 0$. Then for any set $S \subseteq \mathbb{R}^m$ we have $\mathsf{vol}(T(S)) = |\det Q| \mathsf{vol}(S)$.*

**Lemma 4.5** *The points $x_0, \ldots, x_k$ generated by FindOpt at the end of step O2 satisfy $\min_{i=0}^{k} h(x_i) \leq \bigl(OPT + \frac{\epsilon}{4}\bigr)/(1-\omega)$.*

**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 a $\omega$-subgradient at $x_l$. Otherwise let $r = \frac{\epsilon}{8KR}$. Consider the affine transformation $T$ defined by $T(x) = rI_m(x - x^*) + x^* = rx + (1-r)x^*$ where $I_m$ is the $m \times m$ identity matrix, and let $W = T(\mathcal{P})$, so $W$ is a shrunken version of $\mathcal{P}$ "centered" around $x^*$. Observe that, (1) $W \subseteq \mathcal{P}$ because $\mathcal{P}$ is convex, and any point $T(x) \in W$ is a convex combination of $x \in \mathcal{P}$ and $x^* \in \mathcal{P}$, so $T(x) \in \mathcal{P}$; (2) $\mathsf{vol}(W) = r^m \mathsf{vol}(\mathcal{P}) \geq (rV)^m \mathsf{vol}(B(\mathbf{0}, 1))$ using Facts 4.4 and 4.3 and since $\mathcal{P}$ contains a ball of radius $V$ by assumption; and (3) for any $y = Tx \in W$, $\|y - x^*\| = r\|x - x^*\| \leq \frac{\epsilon}{4K}$ since $x, x^* \in B(\mathbf{0}, R)$, so $h(y) \leq h(x^*) + \frac{\epsilon}{4}$ since $h(.)$ has Lipschitz constant $K$. Since $\frac{\mathsf{vol}(E_{i+1})}{\mathsf{vol}(E_i)} \leq e^{-1/(2m)}$ for every $i$, and the volume of the ball $E_0 = B(\mathbf{0}, R)$ is $R^m \mathsf{vol}\big(B(\mathbf{0}, 1)\big)$, plugging things together we obtain

$$\mathsf{vol}(\mathcal{P}_k) \leq \mathsf{vol}(E_N) \leq e^{-N/(2m)}\, \mathsf{vol}(E_0) = \left(\tfrac{rV}{2}\right)^m \mathsf{vol}(B(0,1)) < \mathsf{vol}(W),$$

so there must be a point $y \in W$ that lies on a boundary of $\mathcal{P}_k$ generated by a hyperplane $\hat{d}_l \cdot (x - x_l) = 0$. This implies that $h(x_l) \leq h(y)/(1-\omega) \leq \big(h(x^*) + \frac{\epsilon}{4}\big)/(1-\omega)$. ■

**Lemma 4.6** *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 $i = k+1$, so we show the invariant. The invariant clearly holds when $i = 1$. Suppose the invariant holds at the beginning of iteration $i$. We will show that 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 after we set $\bar{x} \leftarrow y$ in step M2c) at the end of iteration $i$, we get that $h(\bar{x}) \leq (\min_{l=0}^{i} h(x_l) + i\rho)/(1-\omega)^{iN'}$, which satisfies the invariant at the beginning of iteration $i+1$.

To prove the claim about the inner loop, first notice that if at any point we have $\hat{d} \cdot (y_1 - y_2) = 0$, then since $y_1, y_2$ and $y$ all lie on the $\bar{x} - x_i$ line segment, we also have $\hat{d} \cdot (\bar{x} - y) = \hat{d} \cdot (x_i - y) = 0$. This implies that $h(y) \leq \min(h(\bar{x}), h(x_i))/(1-\omega)$ and in this case the claim holds. So assume that this is not the case. We will show by induction that $h(y_1) \leq h(\bar{x})/(1-\omega)^{j-1}$ and $h(y_2) \leq h(x_i)/(1-\omega)^{j-1}$ at the start of the $j^{th}$ iteration of the inner loop. This is true at the beginning of the inner loop when $j = 1$. Suppose that this is true for iterations $1, \ldots, j-1$. So we have, $h(y_1) \leq h(\bar{x})/(1-\omega)^{j-2}$ and $h(y_2) \leq h(x_i)/(1-\omega)^{j-2}$ at the start of the $(j-1)^{th}$ iteration. In iteration $j-1$, we set $y = \frac{y_1+y_2}{2}$ and either $\hat{d} \cdot (y_1 - y) > 0$ or $\hat{d} \cdot (y_2 - y) > 0$. In the former case, we have $h(y) \leq h(y_1)/(1-\omega) \leq h(\bar{x})/(1-\omega)^{j-1}$ and we update $y_1 \leftarrow y$; similarly, in the latter case we have $h(y) \leq h(x_i)/(1-\omega)^{j-1}$ and we update $y_2 \leftarrow y$. So in either case, at the beginning of the $j^{th}$ iteration we maintain that $h(y_1) \leq h(\bar{x})/(1-\omega)^{j-1}$ and $h(y_2) \leq h(x_i)/(1-\omega)^{j-1}$, and by induction the invariant holds through all iterations. At the end of iteration $N'$, we have $\|y - y_1\|,\ \|y - y_2\|$ both at most $\frac{\|\bar{x} - x_i\|}{2^{N'}} \leq \rho/K$, since $\|\bar{x} - x_i\| \leq 2R$ as $\bar{x}$ and $x_i$ both lie in $\mathcal{P} \subseteq B(\mathbf{0}, R)$, which implies that $h(y) \leq \min(h(y_1), h(y_2)) + \rho \leq \min(h(\bar{x}), h(x_i))/(1-\omega)^{N'} + \rho$. This proves the claim about the inner loop on $j$, and hence the lemma. ■

**Theorem 4.7** *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 $T(\omega)$ denotes the time taken to compute an $\omega$-subgradient and $\omega = \Theta\big(\gamma/m^2 \ln^2(\frac{KRm}{V\epsilon})\big)$.*

**Proof :** By Lemmas 4.5 and 4.6, we get that $h(\bar{x}) \leq \big(OPT + \frac{\epsilon}{2}\big)/(1-\omega)^{kN'+1}$. Since $kN' \leq N \log\big(\frac{8NKR}{\epsilon}\big) = n$ and $\omega = \gamma/2n$, we have $(1-\omega)^{kN'+1} \geq (1-\omega)^{n+1} \geq (1-\gamma) \geq \frac{1}{2}$ (since we assumed $\gamma \leq \frac{1}{2}$) which proves the performance guarantee, and shows that $\omega = \Theta\big(\gamma/m^2 \ln^2(\frac{KRm}{V\epsilon})\big)$. The running time is $O\big((N+n)T(\omega)\big)$ which is $O\big(T(\omega) \cdot m^2 \ln^2(\frac{KRm}{V\epsilon})\big)$. ■

Now we show that procedure ConvOpt works correctly with high probability. We make the very mild assumption that at any point $x \in \mathcal{P}$, in any non-null scenario, the total stage I cost + stage II cost of the scenario is at least 1, that is, $w^{\mathrm{I}} \cdot x + f_A(x) \geq 1$ for every scenario $A \neq \emptyset$. Note that with integer costs, this is simply saying that we incur a non-zero total cost in any non-null scenario. (The constant 1 may be replaced by any other constant by adjusting the number of samples required by ConvOpt accordingly.)

**Lemma 4.8** *Procedure ConvOpt determines (correctly) with probability at least* $1-\delta$*, that* $OPT \geq \varrho/\lambda$*, 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 $q \geq \ln\left(\frac{1}{\delta}\right)/M$, 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 $q \cdot w_S^{\mathrm{II}} \leq w_S^{\mathrm{I}}$. If $\delta/M < q < \ln\left(\frac{1}{\delta}\right)/M$, then we always return a correct answer since it is both true that $x = \mathbf{0}$ is an optimal solution, and that $OPT \geq q \geq \varrho/\lambda$. ■

## 4.2 Computing $\omega$-subgradients and bounding the Lipschitz constant $K$

We now focus on showing that algorithm ConvOpt returns a $(1+\kappa)$-optimal solution to relaxation (SSC-P2) of the stochastic set cover problem in polynomial time. Recall that our objective function is $h(x) = w^{\mathrm{I}} \cdot x + f(x)$ where $f(x) = \sum_{A \subseteq U} p_A f_A(x)$, and

$$f_A(x) = \min\Big\{\sum_S w_S^{\mathrm{II}} r_{A,S} : \sum_{S: e \in S} r_{A,S} \geq 1 - \sum_{S: e \in S} x_S \quad \text{for all } e \in A; \quad r_{A,S} \geq 0 \quad \text{for all } S\Big\}.$$

By taking the dual, we can write $f_A(x) = \max\big\{\sum_e (1 - \sum_{S: e \in S} x_S) z_{A,e} : z_A \in \mathcal{Q}_A\big\}$ where

$$\mathcal{Q}_A = \Big\{z \in \mathbb{R}^{|U|} : \sum_{e \in S} z_e \leq w_S^{\mathrm{II}} \text{ for all } S; \quad z_e = 0 \text{ for all } e \notin A; \quad z \geq \mathbf{0}\Big\}.$$

Recall that $\lambda = \max\big(1, \max_S \frac{w_S^{\mathrm{II}}}{w_S^{\mathrm{I}}}\big)$. We argue that for the function $h(.)$ one can efficiently compute $\omega$-subgradients and bound its Lipschitz constant $K$, as follows.

1. In Lemma 4.9 we show that any vector $\hat{d}$ that component-wise approximates a subgradient at $x$ to within a certain accuracy is an $\omega$-subgradient at $x$..
2. Next we show that at any point $x$, there is a "nice" subgradient $d$ with components $d_S \in [-w_S^{\mathrm{II}}, w_S^{\mathrm{I}}]$ (in Lemma 4.10). This gives a bound on the Lipschitz constant $K$, and will allow us to compute an $\omega$-subgradient by using a sampling procedure.
3. Lemma 4.12 and Corollary 4.13 show that since the components $d_S$ lie in a range bounded multiplicatively by $\lambda$, $\mathsf{poly}\big(m, \lambda, \frac{1}{\omega}\big)$ samples from the scenario distribution suffice to compute a vector $\hat{d}$ that component-wise approximates this subgradient $d$ to within the desired accuracy with high probability, and thus obtain a $\omega$-subgradient.

Using the above procedure to compute $\omega$-subgradients in procedure FindOpt with a small enough error probability, and putting the various pieces together we show in Theorem 4.15 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 4.9** *Let $d$ be a subgradient of $h(.)$ at the point $x \in \mathcal{P}$, and suppose that $\hat{d}$ 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 $\omega$-subgradient of $h(.)$ at $x$.*

**Proof :** Let $y$ be any point in $\mathcal{P}$. Since the polytope $\mathcal{P}$ has $x \geq \mathbf{0}$ as a defining constraint, it follows that $x_S, y_S \geq 0$ for all $S$. We have $h(y) - h(x) \geq d \cdot (y - x) = \hat{d} \cdot (y - x) + (d - \hat{d}) \cdot (y - x)$, so we need to lower bound the second term by $-\omega h(x)$. Since $d_S - \hat{d}_S \geq 0$ and $x_S, y_S \geq 0$ for every $S$, $(d - \hat{d}) \cdot (y - x) \geq -(d - \hat{d}) \cdot x \geq -\sum_S \omega w^{\mathrm{I}}_S x_S \geq -\omega h(x)$ (since $f(x) \geq 0$). ■

**Lemma 4.10** *Consider any point $x \in \mathbb{R}^m$, and let $z^*_A$ be an optimal dual solution for scenario $A$ with $x$ as the stage I decision vector. Then 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 $y$ be any point in $\mathbb{R}^m$. We have to show that $h(y) - h(x) \geq d \cdot (y - x)$. We know that $f_A(x) = \sum_e (1 - \sum_{S: e \in S} x_S) z^*_{A,e}$ for every scenario $A$. Also, since $z^*_A \in \mathcal{Q}_A$, at point $y$, we have $f_A(y) \geq \sum_e (1 - \sum_{S: e \in S} y_S) z^*_{A,e}$ for every scenario $A$. So $h(y) \geq w^{\mathrm{I}}_S \cdot y + \sum_{A \subseteq U} p_A \big(\sum_e (1 - \sum_{S: e \in S} y_S) z^*_{A,e}\big)$. The last term can be rewritten as

$$\sum_{A \subseteq U, e} p_A z^*_{A,e} - \sum_{A \subseteq U} p_A \sum_e \sum_{S: e \in S} y_S z^*_{A,e} = \sum_{A \subseteq U, e} p_A z^*_{A,e} - \sum_{A \subseteq U} p_A \sum_S y_S \left( \sum_{e \in S} z^*_{A,e} \right),$$

therefore we get that $h(y) \geq \sum_S y_S \big(w^{\mathrm{I}}_S - \sum_{A \subseteq U} p_A \sum_{e \in S} z^*_{A,e}\big) + \sum_{A \subseteq U, e} p_A z^*_{A,e}$. We can express $h(x)$ in a similar way with an equality instead of the inequality, and replacing $y_S$ with $x_S$. Subtracting the two terms, we get that $h(y) - h(x) \geq \sum_S (y_S - x_S) d_S$ where $d_S = w^{\mathrm{I}}_S - \sum_{A \subseteq U} p_A \sum_{e \in S} z^*_{A,e}$. To bound $\|d\|$, since $z^*_{A,e} \geq 0$ for all $A, e$, we have $d_S \leq w^{\mathrm{I}}_S$. Also, observe that $w^{\mathrm{I}}_S - w^{\mathrm{II}}_S \leq d_S$, since $\sum_{e \in S} z^*_{A,e} \leq w^{\mathrm{II}}_S$ for every scenario $A$, and $\sum_{A \subseteq U} p_A = 1$ because some scenario has to materialize (recall that we include the empty set also as a scenario). Therefore $|d_S| \leq \lambda w^{\mathrm{I}}_S$, and hence $\|d\| \leq \lambda \|w^{\mathrm{I}}\|$. ■

**Claim 4.11** *Suppose $\|d(x)\| \leq D$ for every $x$, where $d(x)$ is a subgradient of $h(.)$ at point $x$. Then $h(.)$ has Lipschitz constant (at most) $D$.*

**Proof :** Consider any two points $u, v \in \mathbb{R}^m$ and let $d, d'$ denote the subgradients at $u, v$ respectively, with $\|d\|, \|d'\| \leq D$, then we have $h(v) - h(u) \geq d \cdot (v - u) \geq -\|d\| \, \|v - u\| \geq -D\|v - u\|$, and similarly $h(u) - h(v) \geq -\|d'\| \, \|u - v\| \geq -D\|u - v\|$. ■

Claim 4.11 and Lemma 4.10 show that we can bound the Lipschitz constant $K$ by $\lambda \|w^{\mathrm{I}}\|$. Note that $\ln K$ is polynomially bounded. We will use the following sampling lemma to show that one can efficiently compute an $\omega$-subgradient of $h(.)$ at any point $x$.

**Lemma 4.12** *Let $X \in [-a, b]$ be a random variable, $a, b > 0$, computed by sampling from a probability distribution $\pi$. 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 $\pi$, one can compute an estimate $Y$ such that $\mu - 2c \cdot b \leq Y \leq \mu$ with probability at least $1 - \delta$.*

**Proof :** Let $q = \max(a, b)$. The variance of $X$ is $\sigma^2 = \mathrm{E}\big[X^2\big] - \mu^2 \leq q^2$. We divide the samples into $s_1 = \frac{20}{3} \ln\big(\frac{1}{\delta}\big)$ groups, each group containing $s_2 = 5\alpha^2/c^2$ samples. Let $X_{ij}$ be the value of $X$ computed from the $j^{th}$ sample of group $i$, $i = 1, \ldots, s_1$, $j = 1, \ldots, s_2$. Let $Y_i$ be the average of the $X_{ij}$ values. We set $Y = \text{median}(Y_1, \ldots, Y_{s_1}) - c \cdot b$. The variables $X_{ij}$ are iid with mean $\mu$ and variance $\sigma^2$. So we have $\mathrm{E}\big[Y_i\big] = \mu$ and $\mathrm{Var}\big[\mathrm{Y_i}\big] = \sigma^2/s_2$. By Chebyshev's inequality, we get $\Pr[|Y_i - \mu| > c \cdot b] \leq \frac{\sigma^2}{s_2 (cb)^2} \leq$

$\frac{\alpha^2}{s_2 c^2} \leq \frac{1}{5}$. Let $Z_i = 1$ if $|Y_i - \mu| > c \cdot b$, and 0 otherwise, and $Z = \sum_{i=1}^{s_1} Z_i$. Then $\mathrm{E}\big[Z\big] \leq s_1/5$ and the variables $Z_i$ are independent. If $Y > \mu$ or $Y < \mu - 2c \cdot b$, then at least $s_1/2$ variables $Z_i$ must be set to 1. Therefore by Chernoff bounds we have $\Pr\big[Y \notin [\mu - 2c \cdot b, \mu]\big] \leq \exp(-\frac{3s_1}{20}) \leq \delta$. ■

**Corollary 4.13** *At any point $x \in \mathcal{P}$, one can compute an $\omega$-subgradient with probability at least $1-\delta$ using at most $T(\omega) = \frac{400\lambda^2}{3\omega^2} \ln\big(\frac{m}{\delta}\big)$ independent samples from the probability distribution on scenarios.*

**Proof :** The proof is an easy corollary of Lemmas 4.10, 4.9 and 4.12. We use the sampling process described in Lemma 4.12. Each time we sample and get a scenario $A$, we compute the quantities $X_S = w^{\mathrm{I}}_S - \sum_{e \in S} z^*_{A,e}$ where $z^*_A$ is an optimal dual solution for scenario $A$ with $x$ as the first-stage vector. Observe that if $d_S = \mathrm{E}\big[X_S\big]$ then $d_S = w^{\mathrm{I}}_S - \sum_A p_A \sum_{e \in S} z^*_{A,e}$, so the vector $d$ with components $d_S$ is a subgradient at $x$ by Lemma 4.10. Since $X_S \in [-w^{\mathrm{II}}_S, w^{\mathrm{I}}_S]$ for each $S$, using Lemma 4.12 with error probability $\delta/m$ and $c = \omega/2$, we can estimate the expectation $\mathrm{E}\big[X_S\big] = d_S$ by $\hat{d}_S$ using the claimed number of samples, so that for each $S$ individually, we have $d_S - \omega w^{\mathrm{I}}_S \leq \hat{d}_S \leq d_S$ with probability at least $1 - \delta/m$. So the error probability over all sets $S$ is at most $\delta$, that is, $\Pr[\forall S,\ d_S - \omega w^{\mathrm{I}}_S \leq \hat{d}_S \leq d_S] \geq 1 - \delta$. So by Lemma 4.9, the vector $\hat{d} = \{\hat{d}_S\}$ is an $\omega$-subgradient at $x$ with probability at least $1 - \delta$. ■

Plugging in the time bound $T(\omega)$ (with a sufficiently small error probability) in Theorem 4.7, we get the following.

**Lemma 4.14** *Using the above procedure for computing $\omega$-subgradients, FindOpt finds a feasible solution $\bar{x}$ such that $h(\bar{x}) \leq OPT/(1-\gamma) + \epsilon$ with probability at least $1 - \delta$ in time* poly*(input size, $\frac{1}{\gamma}$, $\ln(\frac{1}{\epsilon})$, $\ln(\frac{1}{\delta})$).*

**Proof :** Theorem 4.7 gives the performance guarantee and accounts for the time taken excluding the time taken to compute $\omega$-subgradients. We need to show that with high probability *every* vector the algorithm computes is an $\omega$-subgradient for $\omega = \gamma/2n$ where $n = N \log\big(\frac{8NKR}{\epsilon}\big)$ and $N = 2m^2 \ln\big(\frac{16KR^2}{V\epsilon}\big)$. The total number of times we need to compute a $\omega$-subgradient is at most $N + n$. Setting the error probability to $\delta/(N+n)$, and $\omega = \gamma/2n$ in Corollary 4.13, we get that $T(\omega) = O\big(\frac{\lambda^2 n^2}{\gamma^2} \ln(\frac{m(N+n)}{\delta})\big)$ samples suffice to ensure that each individual vector computed is an $\omega$-subgradient with probability at least $1 - 1/((N+n)\delta)$. So the net error probability over all $\omega$-subgradient computations is at most $\delta$. 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 polynomial in the input size, $\lambda$, $\frac{1}{\gamma}$, $\ln\big(\frac{1}{\epsilon}\big)$ and $\ln\big(\frac{1}{\delta}\big)$. ■

**Theorem 4.15** *Procedure 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 polynomial in the input size, $\frac{1}{\kappa}$, and $\ln\big(\frac{1}{\delta}\big)$.*

**Proof :** By Lemma 4.8, we know that if ConvOpt calls FindOpt then with probability at least $1 - \delta$ we have $OPT \geq \varrho/\lambda$ where $\varrho = \frac{\delta}{\ln(1/\delta)}$. The performance guarantee and the time bound now follow from Lemma 4.14 since we set $\gamma = \kappa/3$ and $\epsilon = \kappa\varrho/(2\lambda) = \frac{\kappa\delta}{2\lambda \ln(1/\delta)}$. Since FindOpt may err with probability at most $\delta$, the net error probability is at most $2\delta$. ■

# 5 A General Class of Solvable Stochastic Programs

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

$$\begin{aligned}
&\min \quad w^{\mathrm{I}} \cdot x \;+\; f(x) \qquad \text{subject to} \quad x \in \mathcal{P} \subseteq \mathbb{R}^m_{\geq 0}, && \text{(Stoc-P)}\\
&\text{where} \quad f(x) \;=\; \sum_{A \in \mathcal{A}} p_A f_A(x),\\
&\text{and} \quad f_A(x) \;=\; \min \quad w^A \cdot r_A + q^A \cdot s_A\\
&\qquad\qquad \text{s.t.} \quad B^A s_A \;\geq h^A && (2)\\
&\qquad\qquad D^A s_A + T^A r_A \;\geq j^A - T^A x && (3)\\
&\qquad\qquad\qquad r_A, s_A \;\geq \mathbf{0}, r_A \in \mathbb{R}^m, s_A \in \mathbb{R}^\ell.
\end{aligned}$$

Here $\mathcal{A}$ denotes the set of all possible scenarios, and $\mathcal{P}$ is the feasible region polytope. We require that (a) $T^A \geq \mathbf{0}$ for every scenario $A$, and (b) at every point $x \in \mathcal{P}$, $f(x) \geq 0$ and that the primal and dual problems corresponding to $f_A(x)$ be 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 on the second-stage decisions $r_A$, under certain conditions. We assume that $x = \mathbf{0}$ lies in $\mathcal{P}$, since we would like to be able to express the option where one does nothing in the first stage and defers all decisions to stage II.

The essential property of this class of programs is that constraints (3) have the same matrix $T^A$ multiplying the recourse vector $r_A$ and the first-stage vector $x$ in every scenario $A$. This implies that the stage I decisions given by the vector $x$ and the stage II decisions for scenario $A$ given by the vector $r_A$ act in the same capacity. All of the stochastic optimization problems we will 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. Observe that this class of stochastic programs is rich enough to model stochastic problems with scenario-dependent recourse (that is, stage II) costs. To prevent an exponential blowup in the input, we consider an oracle model where an oracle supplied with scenario $A$ reveals the scenario-dependent data $(w^A, q^A, h^A, j^A, B^A, D^A, T^A)$; procedure ConvOpt will need to query this oracle only a polynomial number of times.

We now show that the analysis in Section 4.2 can be extended to show that algorithm ConvOpt computes a near-optimal solution to (Stoc-P). Let $h(.)$ denote the objective function which is easily shown to be convex. Define $\lambda = \max\bigl(1, \max_{A \in \mathcal{A}, S} \frac{w^A_S}{w^{\mathrm{I}}_S}\bigr)$. As before, we assume that the algorithm knows the value of $\lambda$. To extend the analysis in Section 4.2 we need to show the following three things: (1) one can compute a $\omega$-subgradient in polynomial time, (2) the Lipschitz constant $K$ can be set so that $\ln K$ is polynomially bounded, and (3) one can detect with high probability that $OPT$ is large. The third requirement is easily handled by Lemma 4.8. Under the mild assumption that at any $x \in \mathcal{P}$, every "non-null" scenario[1] $A \in \mathcal{A}$ incurs a total cost of at least 1, Lemma 4.8 holds, and shows that by sampling $\lambda \ln\bigl(\frac{1}{\delta}\bigr)$ times one can determine, with probability at least $1 - \delta$, whether $OPT \geq \frac{\delta}{\ln(1/\delta)\lambda}$, or if $x = \mathbf{0}$ is an optimal solution. To show (1) and (2) we proceed exactly as in Section 4.2. We show that at any point, there is a subgradient with a nice structure and bounded $\ell_2$ norm. By approximating this subgradient component-wise one obtains an $\omega$-subgradient, and the bound on $\ell_2$ norm gives a bound on the Lipschitz constant. The following lemma shows that the components of the subgradient vector have variation that is bounded multiplicatively by $\lambda$, so that one can use the sampling lemma, Lemma 4.12, to compute an $\omega$-subgradient with high probability by repeated sampling.

[1]The meaning of a non-null scenario will be intuitively clear from the application; for concreteness, we could define a null scenario as a scenario $A$ for which $f_A(\mathbf{0}) = \min_{x \in \mathcal{P}} f_A(x)$.

**Lemma 5.1** *Consider any point $x \in \mathbb{R}^m$, and let $(u_A^*, z_A^*)$ 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)*. Then, (i) the vector $d = w^{\mathrm{I}} - \sum_A p_A (T^A)^{\mathrm{T}} z_A^*$ is a subgradient at $x$, (ii) $\|d\| \leq \lambda \|w^{\mathrm{I}}\|$ and (iii) if $\hat{d}$ is a vector such that $d - \omega w^{\mathrm{I}} \leq \hat{d} \leq d$, then $\hat{d}$ is a $\omega$-subgradient at $x$.*

**Proof :** By taking the dual, we can write $f_A(x) = h^A \cdot u_A^* + (j^A - T^A x) \cdot z_A^*$. For any other point $y$, $(u_A^*, z_A^*)$ is a feasible dual solution for scenario $A$, given the stage I vector $y$. So $f_A(y) \geq h^A \cdot u_A^* + (j^A - T^A y) \cdot z_A^*$ and we have $h(y) \geq w^{\mathrm{I}} \cdot y + \sum_A p_A (h^A \cdot u_A^* + j^A \cdot z_A^* - y^{\mathrm{T}} (T^A)^{\mathrm{T}} z_A^*)$. As $y^{\mathrm{T}} (T^A)^{\mathrm{T}} z_A^*$ is a scalar, we can replace it by its transpose $\big((T^A)^{\mathrm{T}} z_A^*\big)^{\mathrm{T}} y = \big((T^A)^{\mathrm{T}} z_A^*\big) \cdot y$. Substituting above, and combining the terms with $y$, we get that

$$h(y) \geq \big(w^{\mathrm{I}} - \sum_A p_A (T^A)^{\mathrm{T}} z_A^*\big) \cdot y + \sum_A p_A (h^A \cdot u_A^* + j^A \cdot z_A^*). \tag{4}$$

Similarly, we have $h(x) = \big(w^{\mathrm{I}} - \sum_A p_A (T^A)^{\mathrm{T}} z_A^*\big) \cdot x + \sum_A p_A (h^A \cdot u_A^* + j^A \cdot z_A^*)$. Subtracting, we get that $h(y) - h(x) \geq d \cdot (y - x)$ where $d = w^{\mathrm{I}} - \sum_A p_A (T^A)^{\mathrm{T}} z_A^*$, showing that $d$ is a subgradient at $x$.

For every scenario $A \in \mathcal{A}$ we have $z_A^* \geq \mathbf{0}$, so $d \leq w^{\mathrm{I}}$ since $T^A \geq \mathbf{0}$. Observe that the dual of the scenario $A$ (primal) optimization problem has the constraint $(T^A)^{\mathrm{T}} z_A \leq w^A$. Since $z_A^*$ is a feasible dual solution, we have $(T^A)^{\mathrm{T}} z_A^* \leq w^A \leq \lambda w^{\mathrm{I}}$, and since $\sum_A p_A = 1$, this shows that, $d \geq w^{\mathrm{I}} - \lambda w^{\mathrm{I}}$. So we get that $\|d\| \leq \lambda \|w^{\mathrm{I}}\|$. Now by Claim 4.11 (which holds regardless of the function $h(.)$), one can set $K = \lambda \|w^{\mathrm{I}}\|$, so that $\ln K$ is polynomially bounded.

Finally to show part (iii), we proceed exactly as in Lemma 4.9. $h(y) - h(x) \geq d \cdot (y - x) = \hat{d} \cdot (y - x) + (d - \hat{d}) \cdot (y - x)$. Since $x, y \geq \mathbf{0}$, we have $(d - \hat{d}) \cdot (y - x) \geq -(d - \hat{d}) \cdot x \geq -\omega w^{\mathrm{I}} \cdot x \geq -\omega h(x)$, where the last inequality follows since $f(x) \geq 0$. ■

So as before, using Lemma 4.12, one can compute an $\omega$-subgradient at any point $x$ using $T(\omega) = O\big(\frac{\lambda^2}{\omega^2} \ln(\frac{m}{\delta})\big)$ samples.

**Theorem 5.2** *Procedure* ConvOpt *can be used to obtain a feasible solution to* (Stoc-P) *of objective function value at most $(1 + \kappa) \cdot OPT$ with probability at least $1 - 2\delta$, in time* $\mathsf{poly}\big($*input size*$, \lambda, \frac{1}{\kappa}, \ln(\frac{1}{\delta})\big)$.

## 5.1 2-Stage Programs with a Continuous Distribution

We now consider the class of 2-stage programs specified by (Stoc-P) where the second stage scenario is specified by a parameter $\xi$ that is continuously distributed with probability density function $p(\xi)$, and show that procedure ConvOpt can be used to obtain a $(1 + \kappa)$-optimal solution to this class of programs. Our objective function is $h(x) = w^{\mathrm{I}} \cdot x + \mathrm{E}_\xi\big[f(x, \xi)\big]$, where $\mathrm{E}_\xi\big[f(x, \xi)\big] = \int p(\xi) f(x, \xi)\, d\xi$ and $f(x, \xi)$ is the cost of scenario $\xi$ determined by the minimization problem in (Stoc-P) with parameters $w(\xi)$, $q(\xi)$, $h(\xi)$, $j(\xi)$, $B(\xi)$, $D(\xi)$ and $T(\xi)$. As before we assume that at every feasible point $x$ and scenario $\xi$, (a) $T(\xi) \geq \mathbb{0}$, (b) $\int p(\xi) f(x, \xi)\, d\xi \geq 0$, and that the primal and dual problems corresponding to $f(x, \xi)$ are feasible.

Notice that the proof of Lemma 5.1 (and hence that of Theorem 5.2) does not rely on the fact that the probability distribution is discrete. Consequently, the statement and proof of Lemma 5.1 extend easily to the continuous setting by replacing each occurrence of the summation $\sum_A p_A(\ldots)$ by the integral $\int d\xi\, p(\xi)(\ldots)$. At any point $x$, if $\big(u^*(\xi), z^*(\xi)\big)$ is an optimal solution for the dual problem corresponding to $f(x, \xi)$ with $z^*(\xi)$ being the dual multipliers for inequalities (3), then the vector $d = w^{\mathrm{I}} - \int d\xi\, p(\xi) T(\xi)^{\mathrm{T}} z^*(\xi)$ is a subgradient at $x$, and parts (ii) and (iii) of Lemma 5.1 hold as is where we define $\lambda$ to be $\max\big(1, \sup_{\xi, S} \frac{w(\xi)_S}{w_S^{\mathrm{I}}}\big)$, which we assume is known. One thus obtains a bound on the Lipschitz constant, and the fact that $\omega$-subgradients can be computed by sampling. Finally, under the assumption that at every $x \in \mathcal{P}$ and every $\xi$, either $w^{\mathrm{I}} \cdot x + f(x, \xi) \geq 1$ or $f(\mathbf{0}, \xi) = \min_{x \in \mathcal{P}} f(x, \xi)$, we can detect that $OPT$ is large using Lemma 4.8.

**Theorem 5.3** *Procedure ConvOpt returns a feasible solution to* (Stoc-P) *with a continuous distribution, of value at most* $(1+\kappa)\cdot OPT$ *in polynomial time.*

# 6 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. Our guarantees generalize and, in many cases, improve upon previous results that were obtained by placing restrictions either on the underlying distribution, or on the costs of the two stages.

## 6.1 Multicommodity Flow Problems

We consider a stochastic version of the concurrent multicommodity flow problem where we are given a set of commodities represented by source-sink pairs $(s_i, t_i)$, and we need to buy capacity to install on the edges so that in every scenario $A$, one can concurrently ship $d_i^A$ units of each commodity $i$ from its source $s_i$ to its sink $t_i$, where the scenario is generated by some probability distribution. We can either purchase capacity on an edge in stage I paying a lower price, or wait until the exact demands are known and buy capacity at a higher price; 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. Let $c^{\mathrm{I}}$ and $c^A$ denote the cost vectors for buying capacity in stage I and in scenario $A$ in stage II. The 2-stage stochastic multicommodity flow problem can be formulated as follows: minimize $\sum_e c_e^{\mathrm{I}} x_e + \sum_{A\subseteq\mathcal{A}} p_A g_A(x)$ ($\mathcal{A}$ is the set of all scenarios) subject to $0 \le x_e \le u_e$ for each $e$, $g_A(x)$ is the minimum value of $\sum_e c_e^A y_{A,e}$ subject to the constraints that for each $i$, the total flow for $(s_i, t_i)$ is at least $d_i^A$, for each edge $e$, the total flow on $e$ is at most $x_e + y_{A,e}$, and also at most $u_e$ (this ensures that $x_e + y_{A,e} \le u_e$).

Immorlica et al. [14] 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, (.e., a scenario specifies a set of source-sink pairs have to be connected to each other) there are still an exponential number of scenarios to enumerate. Note that there are no integrality constraints, that is, one we can install fractional amounts of capacity. Since we have formulated the problem as a convex program of the type handled by Theorem 5.2, our fully polynomial approximation scheme can be applied. Whereas our running time depends on $\lambda$, the ratio of stage II and stage I costs, it does not depend on $D$.

**Theorem 6.1** *For any* $\epsilon > 0$*, the stochastic concurrent multicommodity flow problem can be approximated to within a factor of* $(1+\epsilon)$ *in polynomial time.*

The algorithm in Figure 1 can be applied to solve other stochastic multicommodity flow variants in polynomial time as well. For example, one could consider a maximum multicommodity flow variant, where a scenario $A$ specifies a set of *active* $(s_i, t_i)$ pairs and we want to install capacity so as to ensure that the *total* flow routed between the active pairs is at least some threshold $b^A$. The only change here is that in the convex program, the scenario $A$ minimization problem contains a constraint stating that the net flow routed between the active pairs in $A$ is at least $b^A$ which replaces the concurrent flow constraints. This convex program can be solved to near-optimality, yielding a $(1+\epsilon)$-approximation algorithm for the problem, for any $\epsilon > 0$.

## 6.2 Covering problems

**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 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 $w_v^{\mathrm{I}}$, or in stage II at a cost of $w_v^A$ in scenario $A$. The previous results known for this problem were an 8-approximation algorithm in the black-box model, a 3-approximation algorithm in the setting where each edge is independently activated, both under the restriction that $w_v^A = \lambda w_v^{\mathrm{I}}$ for each $v$ and scenario $A$, due to Gupta et al. [13]; Ravi and Sinha [22] gave a 2-approximation algorithm when there are only polynomially many scenarios (but the second-stage costs may be scenario dependent).

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 probability distributions and scenario-dependent second-stage costs. This is the first approximation algorithm in this more general model with black box probability distributions.

**Theorem 6.2** *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.*

**Minimum multicut problem 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 the second stage, and we can choose either to "cut" an edge in stage I or in stage II, paying a cost of $w_e^{\mathrm{I}}$ or $w_e^A$ in scenario $A$, respectively. The multicut problem is an instance of the case of the set-cover problem, where we want to cover each $(s_i, t_i)$ path. Garg, Vazirani & Yannakakis [11] gave a primal-dual approximation algorithm for the deterministic problem that showed that the natural covering LP relaxation of this problem has an integrality gap of 2. Using their 2-approximation algorithm, and applying Corollary 2.2, we get the following result.

**Theorem 6.3** *For any $\epsilon > 0$, there is a $(4+\epsilon)$-approximation algorithm for the stochastic minimum multicut problem on trees.*

**General covering problems.** Kolliopoulos and Young [17] consider general deterministic covering problems with multiplicity constraints, of the form $\min w \cdot x$ subject to $Mx \geq r, x \leq b,\ x \in \mathbb{Z}_+^m$, where the entries of $w, M$ and $r$ are all non-negative. An example of such a problem is the multiset multicover problem with multiplicity constraints, which is an extension of the set-cover problem where each element $e$ is required to be covered $r_e$ times by the chosen sets; a set $S$ can cover element $e$ $M_{e,S}$ times and may be chosen at most $b_S$ times. Kolliopoulos and Young give bicriteria approximation algorithms for these problems. We obtain generalizations of their results for the corresponding stochastic covering problem where *copies* of a set $S$ may be purchased either in stage I or in stage II at a price of $w_s^{\mathrm{I}}$ in stage I or $w_s^A$ in scenario $A$ in stage II. We remark that due to the multiplicity constraints the convex programming relaxation of the stochastic problem is not of the form SSC-P2 since the matrix $T^A$ has negative entries; nevertheless one can show that algorithm ConvOpt in Figure 1 can be used to obtain a $(1+\kappa)$-optimal solution, which can then be rounded using the procedure detailed in Theorem 2.1.

## 6.3 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 $d_j c_{ij}$ where $d_j$ is the *demand* of client $j$, $c_{ij}$ is the distance between $i$ and $j$, and the distances $c_{ij}$ form a metric. The goal is to minimize the total facility opening costs and client assignment costs.

In the deterministic problem one assumes that the client demands are precisely known in advance; the 2-stage stochastic uncapacitated facility location (SUFL) problem handles settings where there is uncertainty in the demand, for example, due to macro-economic factors such as competition, technology, customer purchasing power etc. We are given a probability distribution on tuples $(d_1, \ldots, d_{|\mathcal{D}|})$ where $d_j \in \{0, 1, \ldots, D\}$ specifies the demand of client $j$ and $D$ is some known upper bound on the demand. We can open some facilities in stage I paying a cost of $f_i^{\mathrm{I}}$ for opening facility $i$, then the actual scenario $A$ with demands $d_j^A$ is revealed, and we may choose to open some more facilities in stage II, incurring a cost of $f_i^A$ for each facility $i$ that we open in scenario $A$. As indicated by the notation, the recourse costs $f_i^A$ may in general be scenario-dependent.

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. [13] 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 [22] gave an LP-rounding based 8-approximation algorithm in 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 $\mathcal{F}$, $j$ to index the clients in $\mathcal{D}$, and $\mathcal{A}$ to denote the set of all possible scenarios.

$$\min \quad \sum_i f_i^{\mathrm{I}} y_i + \sum_{A \in \mathcal{A}} p_A g_A(y) \qquad \text{subject to} \qquad 0 \le y_i \le 1 \qquad \text{for all } i, \qquad \text{(SUFL-P)}$$

$$\text{where} \quad g_A(y) = \min \quad \sum_i f_i^A y_{A,i} + \sum_j d_j^A \sum_i c_{ij} x_{A,ij}$$

$$\text{s.t.} \quad \sum_i x_{A,ij} \ge 1 \qquad \text{for all } j \text{ such that } d_j^A > 0,$$

$$x_{A,ij} \le y_i + y_{A,i} \qquad \text{for all } i \in \mathcal{F}, j \text{ such that } d_j^A > 0,$$

$$x_{A,ij}, y_{A,i} \ge 0 \qquad \text{for all } i, j.$$

Here $y_i$ indicates if facility $i$ is opened in stage I and $y_{A,i}$ indicates if facility $i$ is opened in the stage II scenario $A$. The variables $x_{A,ij}$ are the usual assignment variables indicating whether client $j$ is assigned to facility $i$. The minimization problem for a scenario $A$ determines the cost, $g_A(y)$, incurred for scenario $A$ and has constraints that enforce that each client $j$ with positive demand $d_j^A$ has to be assigned to a facility that is opened either in stage I or in scenario A. The term $\sum_{A \in \mathcal{A}} p_A g_A(y)$ is therefore the expected second stage cost. Observe that (SUFL-P) lies in the class of 2-stage stochastic programs handled by Theorem 5.2, and therefore one can use the algorithm in Figure 1 to obtain a solution $y$ of cost at most $(1 + \epsilon)$ times the optimal in time polynomial in the size of the input and $\frac{1}{\epsilon}$. Let $\rho_{\mathsf{DUFL}}$ denote the integrality gap of the deterministic problem which is bounded by 1.52 [20].

**Theorem 6.4** *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 :** For notational simplicity, we shall focus on the case in which the demands $d_j^A$ in any scenario $A$ are either 0 or 1, that is, a scenario $A$ is now just a subset of the clients $\mathcal{D}$ that need to be assigned to facilities. The extension to the setting with arbitrary demands requires only cosmetic notational changes.

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 $f^{\mathrm{I}}_i$, 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.

In fact, 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 $\hat{y}_i$ facility variables 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 $\hat{x}_{A,ij} = 0$ and then considering the facilities in non-decreasing order of the assignment cost $c_{ij}$, set $\hat{x}_{A,i'j} = \min\bigl(\hat{y}_{i'}, 1 - \sum_i \hat{x}_{A,ij}\bigr)$ for each facility $i'$ Note that this new fractional assignment is completely determined by the values of the fractional facility variables 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 $\rho_{\mathsf{DUFL}}$, given this DUFL instance with a 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 $j$ 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 $\{j \in A : A \notin \mathcal{S}_j\}$. 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 $2\rho_{\mathsf{DUFL}}\bigl(\sum_i f^A_i y_{A,i} + \sum_{i,j \in A : A \notin \mathcal{S}_j} c_{ij} x_{A,ij}\bigr)$. This solution tells us which facilities to open in scenario $A$ and how to assign the clients $j$ in $A$ with $A \notin \mathcal{S}_j$. Hence, the overall cost of the solution with first-stage facilities $\tilde{y}$ is at most $2\rho_{\mathsf{DUFL}} \cdot OPT$, which implies 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 [20] 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 [27] (see Section 2.4) gives an algorithm with this property, improving upon the algorithm of [26] (which also has this property); the algorithm converts any fractional solution to an integer solution increasing the cost by a factor of at most 1.705.

We use this algorithm to obtain the approximation algorithm. We modify the definition of $\mathcal{S}_j$ slightly, so as to balance the contribution from stages I and II. Let $\theta = \frac{1.705}{1.705+1.52}$. Let $\mathcal{S}_j = \{A \subseteq \mathcal{D} : \sum_i x^{\mathrm{I}}_{A,ij} \geq \theta\}$. So now we have a fractional solution in which we set $\hat{y}_i = \min\bigl(1, y_i/\theta\bigr)$, and using the re-optimization procedure described earlier, we can find the optimal fractional assignment $\hat{x}$ corresponding to the $\hat{y}$ values. We round this using the algorithm of Swamy [27] to get a solution $(\tilde{x}, \tilde{y})$ of cost at most $\frac{1.705}{\theta} \cdot \bigl(\sum_i f^{\mathrm{I}}_i y_i + \sum_{i,j} \sum_{A \in \mathcal{S}_j} p_A c_{ij} x_{A,ij}\bigr)$. This determines the facilities to open in stage I. In any scenario $A$, each client $j \in A$ such that $A \in \mathcal{S}_j$ is taken care of by a stage I facility. Next, we determine which facilities to open in scenario $A$ and how to assign the remaining clients in $A$ by constructing a feasible fractional solution for a deterministic subproblem with client set $\{j \in A : A \notin \mathcal{S}_j\}$ and "rounding" this solution. We set $\hat{y}_{A,i} = \min\bigl(1, y_{A,i}/(1-\theta)\bigr)$ and for each client $j \in A$ such that $A \notin \mathcal{S}_j$, set $\hat{x}_{A,ij} = \min\bigl(1, x^{\mathrm{II}}_{A,ij}/(1-\theta)\bigr)$. We "round" this solution using the algorithm of Mahdian et al. [20] (which is not an LP rounding algorithm) since the issue of the demands does not apply to this stage, to get an integer solution of cost at most $\frac{1.52}{1-\theta} \cdot \bigl(\sum_i f^A_i y_{A,i} + \sum_{i,j \in A: A \notin \mathcal{S}_j} c_{ij} x_{A,ij}\bigr)$. So the total cost incurred if we open the facilities given by $\tilde{y}$ in stage I is at most $3.225 \cdot OPT$. ■

The algorithm remains unchanged if we have arbitrary demands $d^A_j$, and/or scenario-dependent assignment costs $c^A_{ij} = \gamma^A c_{ij}$. The only change in the analysis, is that in the feasible fractional solution we exhibit to bound the cost of the stage I decisions computed, we set the demand of a client $j$ equal to $\sum_{A \in \mathcal{S}_j} p_A d^A_j \gamma^A$, and in the fractional solution constructed for a stage II scenario $A$, each client $j \in A$ such that $A \notin \mathcal{S}_j$ has demand $d^A_j \gamma^A$.

**Remark 6.5** It is possible to prove an integrality gap of at most 3 by adapting the primal-dual algorithm of Jain & Vazirani [15]. This was also observed by Mahdian [19] and Devanur (personal communication). But this requires explicit knowledge of the probability of *every* scenario $p_A$, and it seems difficult to obtain a polynomial-time algorithm this way.

**Extensions** Our approach yields constant-factor approximation algorithms for various other stochastic facility location problems. 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. We obtain constant performance guarantees for the stochastic versions of the facility location problem with penalties, or soft capacities, or service installation costs. The details may be found in [27].

# 7 The dependence of the running time on $\lambda$

We have remarked previously that the running time of our algorithm (and that of Gupta et al. [13]) depends on the parameter $\lambda$, 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.

We show the lower bound by considering a rather simple instance of SSC with a single set and a single element. The example exposes the inherent limitation of using a black box to infer knowledge about the probability distribution on scenarios; it is straightforward to generalize the example to construct lower bound instances for other stochastic problems. 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 that samples from the distribution on scenarios). The only decision here is whether to buy set $S$ in stage I or to defer buying the set to stage II. Let $\mathcal{A}_N$ denote an algorithm that draws exactly $N$ samples. Let $O^*$ denote the value of the *integer* optimum solution.

**Theorem 7.1** *If $\mathcal{A}_N$ returns a (fractional) solution of cost at most $c \cdot O^*$ with probability at least $1-\delta$ where $1 \leq c < \frac{\lambda}{2}$, then it must be that $N \geq \left(\lambda \ln(\frac{1}{\delta} - 1)\right)/2c$. The bound applies even if $\mathcal{A}_N$ returns only a fractional solution of cost at most $c \cdot O^*$.*

**Proof :** Let $X$ be a random variable that denotes the number of times scenario $\{e\}$ occurs in the $N$ samples. If $X = 0$, then $\mathcal{A}_N$ must choose to defer to stage II with probability at least $1-\delta$ (the algorithm may flip coins), that is, it must return the *integer solution* $x = 0$ with probability at least $1-\delta$. Otherwise, with $p = 0$, and hence, $O^* = 0$, $\mathcal{A}_N$ will pick (a non-zero fraction of) set $S$ in stage I with probability at least $\delta$, and thus incur a non-zero cost, that is, a cost greater than $c \cdot O^*$ with probability at least $\delta$. Choose any $\epsilon > 0$ such that $c \leq \lambda/\big(2(1+\epsilon)\big)$ and consider any $\epsilon' > 0$ where $\epsilon' \leq \epsilon$. Set $p = (1+\epsilon')c/\lambda \leq \frac{1}{2}$ and define $N_0(\epsilon') = \left(\lambda \ln(\frac{1}{\delta} - 1)\right)/\big(2(1+\epsilon')c\big)$. Let $r = \Pr[X = 0] = (1-p)^N > e^{-2pN}$ (since $p \leq \frac{1}{2}$). The optimal solution is to pick $S$ in stage I, and incur a cost of 1. But if $N < N_0(\epsilon')$, then $r > e^{-2pN_0(\epsilon')} = \frac{\delta}{1-\delta}$, so with probability at least $(1-\delta)r > \delta$, $\mathcal{A}_N$ will choose the solution $x = 0$ and incur a cost of $(1+\epsilon')c > c \cdot O^*$. Therefore for $\mathcal{A}_N$ to satisfy the required performance guarantee we must have $N \geq N_0(\epsilon')$ for every $\epsilon' \in (0, \epsilon]$ which implies that $N \geq \left(\lambda \ln(\frac{1}{\delta} - 1)\right)/2c$. ■

**Corollary 7.2** *If algorithm $\mathcal{A}_N$ returns a (fractional) solution of expected cost at most $c \cdot O^*$ where $1 \leq c < \frac{\lambda}{6}$, then it must be that $N \geq (\lambda \ln 2)/6c$.*

**Proof :** By Markov's inequality, $\mathcal{A}_N$ returns a solution of cost at most $3c \cdot O^*$ with probability at least $\frac{2}{3}$. The claim now follows from Theorem 7.1. ■

Now suppose that for the stochastic set cover problem, for every element $e$, a) we know its activation probability $p_e = \sum_{A \subseteq U : e \in A} p_A$, and b) we can sample scenarios conditioned on the fact that $e$ is activated, that is, scenarios from $\{A \subseteq UU : e \in A\}$ with scenario $A$ generated with probability $p_A/p_e$. It is easy to see if the elements have independent probabilities of being activated (as considered in [14, 13]), then these conditions are satisfied. More generally, consider the subclass of problems mentioned in Section 5 where $B^A \geq \mathbf{0}$ for every scenario $A$, the parameters $w^A, q^A, D^A, T^A$ do not depend on the scenario, and $D^A = D \geq \mathbf{0}$. Fix an indexing of the rows of $T$ (and $D, j$). The columns of $T$ play the role of sets in the set cover problem, while the rows play the role of elements, and following the notation used in the set cover problem, we will use $S$ to index the columns of $T$ (and the components of $x, r_A$), and $e$ to index the rows of $T$ (and the components of $j^A$ and the dual vector $z_A$). We also require that in every scenario $A$, $j_e^A$ is either 0 or a fixed quantity $j_e$. Suppose that a) we know the "activation" probability $p_e = \sum_{A \in \mathcal{A} : j_e^A > 0} p_A$, and b) we can sample scenarios conditioned on the event that $j_e^A > 0$. As mentioned above, the set cover example fits into this framework with $T$ as the element-set incidence matrix, rows representing elements and columns representing sets, and vector $j^A$ given by $j_e^A = 1$ if $e \in A$, 0 otherwise. Using this additional structure we obtain the following result.

**Lemma 7.3** *At any point $x \in \mathcal{P}$, an $\omega$-subgradient of $h(.)$ can be computed with probability at least $1-\delta$ in time polynomial in the input size, $\frac{1}{\omega}$, and $\ln\big(\frac{1}{\delta}\big)$.*

**Proof :** Let $n$ denote the number of rows of $T$, so $j^A \in \mathbb{R}^n$ for each scenario $A$. The dual of the scenario $A$ minimization problem is the following:

$$\begin{aligned} \max \quad & h^A \cdot u_A + (j^A - Tx) \cdot z_A && \text{(DP)} \\ \text{s.t.} \quad & (B^A)^{\mathrm{T}} u_A + D^{\mathrm{T}} z_A \leq q && (5) \\ & T^{\mathrm{T}} z_A \leq w^{\mathrm{II}} && (6) \\ & u_A, z_A \geq \mathbf{0}, \end{aligned}$$

where $j^A, z_A \in \mathbb{R}^n, q \in \mathbb{R}^n, T \in \mathbb{R}^{n\times m}$, and $D \in \mathbb{R}^{n\times \ell}$. Let $\mathcal{A}_e$ denote the set of scenarios $\{A \in \mathcal{A} : j^A_e > 0\}$, that is, there are the scenarios where element $e$ is activated. Let $(u^*_A, z^*_A)$ be an optimal solution to (DP). Note that since $B^A, D, T \geq \mathbf{0}$, one may assume without loss of generality that if $(j^A - Tx)_e \leq 0$ then $z^*_{A,e} = 0$, and that $h^A \cdot u^*_A \geq 0$. Let $T = (t_{e,S})$. By Lemma 5.1 we know that the vector $d$ with components $d_S = w^{\mathrm{I}}_S - \sum_A p_A \sum_e t_{e,S} z^*_{A,e}$ is a subgradient at $x$. Since $j^A_e \leq 0$ for $A \notin \mathcal{A}_e$, we can write $d_S = w^{\mathrm{I}}_S - \sum_e p_e t_{e,S} \sum_{A\in\mathcal{A}_e} \frac{p_A}{p_e} \cdot z^*_{A,e}$.

The proof is in three parts. First, we show by adapting the proof of part (i) of Lemma 5.1, that if $(u'_A, z'_A)$ is a $(1-\epsilon)$-optimal solution to (DP), then the vector $d'$ with components $d'_S = w^{\mathrm{I}}_S - \sum_e p_e t_{e,S} \sum_{A\in\mathcal{A}_e} \frac{p_A}{p_e} \cdot z'_{A,e}$ is an $\epsilon$-subgradient at $x$. Next, our goal will be to get structured near-optimal solutions $(\hat{u}_A, \hat{z}_A)$ for each scenario $A$, so that one can estimate the quantity $\sum_{A\in\mathcal{A}_e} \frac{p_A}{p_e} \cdot \hat{z}_{A,e}$ for each element $e$ to within a certain accuracy. We will show that one can get $(1-\epsilon)$-optimal solutions $(\hat{u}_A, \hat{z}_A)$ to (DP) (for a suitable $\epsilon$) such that for every element $e$, the variation in the $\hat{z}_{A,e}$ values for $A \in \mathcal{A}_e$ is polynomially bounded. Finally, we will estimate $\sum_{A\in\mathcal{A}_e} \frac{p_A}{p_e} \cdot \hat{z}_{A,e}$ for each element $e$ to within a factor of $(1 \pm \epsilon)$ with high probability, and argue that these estimates induce a near-optimal solution to (DP) for each scenario $A$, and thereby yield an $\omega$-subgradient of $h(.)$ at $x$.

Let $(u'_A, z'_A)$ be any $(1-\epsilon)$-optimal solution to (DP). Consider any point $y \in \mathcal{P}$. Since $(u'_A, z'_A)$ is a feasible dual solution for scenario $A$ with $y$ as the first-stage vector, as in Lemma 5.1 we have

$$h(y) \geq \Big(w^{\mathrm{I}} - \sum_A p_A T^{\mathrm{T}} z'_A\Big) \cdot y + \sum_A p_A (h^A \cdot u'_A + j^A \cdot z'_A). \tag{7}$$

Also since the value of $(u'_A, z'_A)$ is at least $(1-\epsilon)$ times that of $(u^*_A, z^*_A)$,

$$\begin{aligned} h(x) \leq & w^{\mathrm{I}} \cdot x + \sum_A p_A \Big(h^A \cdot u'_A + (j^A - Tx) \cdot z'_A + \epsilon\big(h^A \cdot u^*_A + (j^A - Tx) \cdot z^*_A\big)\Big) \\ \leq & \Big(w^{\mathrm{I}} - \sum_A p_A T^{\mathrm{T}} z'_A\Big) \cdot x + \sum_A p_A (h^A \cdot u'_A + j^A \cdot z'_A) + \epsilon h(x). \end{aligned} \tag{8}$$

Subtracting (8) from (7), we get that $h(y) - h(x) \geq d' \cdot (y - x) - \epsilon h(x)$ where $d' = w^{\mathrm{I}} - \sum_A p_A T^{\mathrm{T}} z'_A$ with components $d'_S = w^{\mathrm{I}}_S - \sum_e p_e t_{e,S} \sum_{A\in\mathcal{A}_e} \frac{p_A}{p_e} \cdot z'_{A,e}$. So $d'$ is an $\epsilon$-subgradient of $h(.)$ at $x$.

Now we obtain a specific structured $(1-\epsilon)$-optimal solution to (DP) for each scenario $A$. We use $i$ to index the columns of $D$. Let $D = (d_{e,i})$. For each element $e$, define $c_e = \min\big(\min_i \frac{q_i}{d_{e,i}}, \min_S \frac{w^{\mathrm{II}}_S}{t_{e,S}}\big)$. Since $B^A \geq \mathbf{0}$, constraints (5) imply that $z^*_{A,e} \leq \min_i q_i/d_{e,i}$. Similarly constraints (6) imply that $z^*_{A,e} \leq \min_S w^{\mathrm{II}}_S/t_{e,S}$, so we have $z^*_{A,e} \leq c_e$ for every element $e$. Let $\epsilon = \omega/3$. We assume that $\epsilon \leq \frac{1}{2}$ without loss of generality. Set $\hat{z}_{A,e} = (1-\epsilon)\big(z^*_{A,e} + \frac{\epsilon c_e}{n}\big)$ if $(j^A - Tx)_e = j^A_e - \sum_S t_{e,S} x_S > 0$ and 0 otherwise, and $\hat{u}_A = (1-\epsilon)u^*_A$. Note that $\hat{z}_A \geq (1-\epsilon) z^*_A$ since $z^*_{A,e} = 0$ if $(j^A - Tx)_e \leq 0$.

Since $j^A_e = j_e$ for every scenario $A \in \mathcal{A}_e$, either $\hat{z}_{A,e} = 0$ for every scenario in $\mathcal{A}_e$ (if $j_e - \sum_S t_{e,S} x_S \leq 0$), or $\hat{z}_{A,e} > 0$ for every scenario in $\mathcal{A}_e$. In the latter case, for any scenario $A \in \mathcal{A}_e$, $\hat{z}_{A,e} \geq \frac{(1-\epsilon)\epsilon c_e}{n} \geq \frac{\epsilon c_e}{2n}$ and $\hat{z}_{A,e} \leq c_e$ since $z^*_{A,e} \leq c_e$, so that the variation in $\hat{z}_{A,e}$ is polynomially bounded. It is clear that the value of $(\hat{u}_A, \hat{z}_A)$ is at least $(1-\epsilon)$ times the optimum value of (DP); we show that $(\hat{u}_A, \hat{z}_A)$ is a feasible solution to (DP). Constraints (6) are satisfied since

$$\sum_e t_{e,S} \hat{z}_{A,e} \leq (1-\epsilon)\Big(\sum_e t_{e,S} z^*_{A,e} + \frac{\epsilon}{n} \cdot \sum_e t_{e,S} c_e\Big) \leq (1-\epsilon)\Big(w^{\mathrm{II}}_S + \frac{\epsilon}{n} \cdot \sum_e w^{\mathrm{II}}_S\Big) \leq w^{\mathrm{II}}_S,$$

where the second inequality follows since $c_e \leq \frac{w^{\mathrm{II}}_S}{t_{e,S}}$. Similarly, one can show that inequalities (5) hold, so $(\hat{u}_A, \hat{z}_A)$ is a feasible solution.

Set $\mu_e = \sum_{A \in \mathcal{A}_e} \frac{p_A}{p_e} \cdot \hat{z}_{A,e}$. We estimate $\mu_e$ by $Y_e$ as follows. If $j_e - \sum_S t_{e,S} x_S \leq 0$, then we know that $\mu_e = 0$ and we set $Y_e = 0$. Otherwise, we sample $N = \frac{6n}{\epsilon^3} \ln\left(\frac{2}{\delta}\right) = O\left(\frac{n}{\omega^3} \ln(\frac{1}{\delta})\right)$ times from the conditional distribution on $\mathcal{A}_e$, and for each sampled scenario $A$ compute $\hat{z}_{A,e}$ as above. Let $Y_{e,k}$ for $k = 1, \ldots, N$ denote the value of the $k^{th}$ sample. Let $Y'_e = \left(\sum_{k=1}^{N} Y_k\right)/N$; we set $Y_e = Y'_e(1-\epsilon)$. Since scenario $A \in \mathcal{A}_e$ is generated with probability $p_A/p_e$, $\mathrm{E}\left[Y'_e\right] = \mathrm{E}\left[Y_{e,k}\right]$ is precisely $\mu_e$. Note that each $Y_{e,k} \in \left[\frac{\epsilon c_e}{2n}, c_e\right]$ and so $\mu_e > \epsilon c_e/2n$. So by standard Chernoff bounds we have $\Pr[|Y'_e - \mu_e| > \epsilon\mu_e] < 2 \cdot \exp\left(-\frac{N\mu_e\epsilon^2}{3c_e}\right) \leq \delta/n$ which implies that $Y_e \geq \mu_e(1-\epsilon)^2 \geq \mu_e(1-2\epsilon)$ and $Y_e \leq (1-\epsilon^2)\mu_e < \mu_e$ with probability at least $1 - \delta/n$.

Finally, we show that vector $\tilde{d}$ with components given by $\tilde{d}_S = w_S^{\mathrm{I}} - \sum_e p_e t_{e,S} Y_e$ is an $\omega$-subgradient with probability at least $1-\delta$. From the above analysis, we know that $\Pr\left[\forall e, Y_e \in [\mu_e(1-2\epsilon), \mu_e]\right] \geq 1-\delta$, so we may assume that this event happens. We will show that the $Y_e$ values induce a $(1-3\epsilon)$-optimal solution $(\tilde{u}_A, \tilde{z}_A)$ to (DP) for each scenario $A$, such that $\sum_{A \in \mathcal{A}_e} \frac{p_A}{p_e} \cdot \tilde{z}_{A,e} = Y_e$. Therefore, as shown before, we get that $\tilde{d}$ is an $\omega$-subgradient of $h(.)$ at the point $x$.

To prove the claim, consider the solution $\tilde{u}_A = \hat{u}_A$ and $\tilde{z}_{A,e} = \frac{Y_e}{\mu_e}\hat{z}_{A,e}$ for every scenario $A$, element $e$. Clearly $\hat{z}_{A,e}(1-2\epsilon) \leq \tilde{z}_{A,e} \leq \hat{z}_{A,e}$, and so for every scenario $A$, $(\tilde{u}_A, \tilde{z}_A)$ is a feasible solution to (DP) and its value is at least $(1-2\epsilon)$ times the value of $(\hat{u}_A, \hat{z}_A)$ (since $h^A \cdot \tilde{u}_A \geq 0$) and hence at least $(1-3\epsilon)$ times the value of $(u^*_A, z^*_A)$. ■

Using Lemma 7.3 to compute the subgradient at any point $x$, we can implement procedure FindOpt in Figure 1 to run in time that does not depend on $\lambda$, and return a point $\bar{x}$ of objective function value $OPT/(1-\gamma) + \epsilon$ with probability at least $1-\delta$. Combining this with a modified ConvOpt procedure, we obtain the following theorem.

**Theorem 7.4** *For the above subclass of problems, given this extra information about the distribution, one can compute a $(1+\kappa)$-optimal solution with probability at least $1-\delta$ in time* $\mathsf{poly}\left(\textit{input size}, \frac{1}{\kappa}, \ln(\frac{1}{\delta})\right)$.

**Proof :** Using the additional information about the probability distribution, we can now detect with probability 1, if $OPT$ is "large", without any sampling in step C1 of ConvOpt. Define a "null scenario" as a scenario $A$ with $j_e^A = 0$ for all elements $e$. Let $\mathcal{S}$ be the set of all non-null scenarios. Assuming that we incur a total cost of at least 1 for every non-null scenario, $OPT \geq \sum_{A \in \mathcal{S}} p_A$. Observe that $\sum_{A \in \mathcal{S}} p_A$ is at least $\sum_{A \in \mathcal{A}_e} p_A = p_e$ for any element $e$, and is at most $\sum_e p_e$, since event $\{\text{some } A \in \mathcal{S} \text{ is realized}\}$ implies that some element is activated. So if $\max_e p_e > \frac{1}{n\lambda}$ then $OPT > \frac{1}{n\lambda}$, and we can set $\varrho = \frac{1}{n}$ in step C3 of ConvOpt, and call procedure FindOpt setting $\gamma, \epsilon$ appropriately; otherwise $\Pr\left[\{\text{some } A \in \mathcal{S} \text{ is realized}\}\right] \leq \frac{1}{\lambda}$ which implies that $x = 0$ is an optimal solution and we return this solution. Thus the entire algorithm runs in time that does not depend on $\lambda$. ■

# 8 Conclusions and discussion

We presented an algorithm to solve a large class of 2-stage stochastic linear programs to within a factor of $(1+\epsilon)$ of the optimum, for any $\epsilon > 0$, in time polynomial in $\frac{1}{\epsilon}$, the size of the input, and the ratio $\lambda$ between the second- and first-stage costs. The algorithm works for both discrete and continuous distributions and requires only a black box to draw independent samples from the underlying probability distribution on scenarios. We show that $\lambda$ is an inherent lower bound on the number of samples required in the black-box model; to the best of our knowledge, this is is the first result that shows that a broad class of 2-stage stochastic LPs can be solved in time polynomial in the input size and $\lambda$. We used our algorithm to devise the first approximation algorithms for a variety of 2-stage stochastic integer optimization problems in the black-box model and without any assumptions about the cost structure of the input. The performance guarantees

are obtained by first solving a fractional relaxation of the problem using our algorithm to solve stochastic linear programs, and then rounding the near-optimal fractional solution. We show that this rounding step can be performed by utilizing existing algorithms for the deterministic analogue of the problem, thereby reducing, in some sense, the stochastic problem to its deterministic counterpart.

Our algorithm for solving stochastic LPs is based on solving a convex-programming relaxation of the problem by adapting the ellipsoid method. One obvious question is whether one can obtain more efficient algorithms, in theory and/or practice, to solve 2-stage stochastic programs by adapting other techniques used for deterministic convex optimization, for example, interior-point methods, projection methods. From a practical perspective, it would be useful to investigate whether one can use the notion of approximate subgradients as defined in this paper, which one can compute efficiently via sampling, within the framework of a cutting plane algorithm, possibly along with a column generation procedure (since we have an exponential number of variables), to get an efficient heuristic for solving stochastic linear programs.

A very appealiing approach, often used in practice, is the sample average approximation (SAA) method which consists of replacing the original stochastic program, where the underlying scenario distribution may have an exponentially large support, by the sample average problem which is a (smaller) linear program of size polynomial in the number of samples, that one can then solve efficiently using one's favorite LP solver. The issue here is the number of samples required to ensure that an optimal (or near-optimal) solution to the sample average problem is, with high probability, a provably near-optimal solution to the original stochastic problem. As mentioned in the Introduction, Kleywegt et al. [16] show that for general stochastic programs the sample size required can be bounded by a polynomial in the dimension of the problem, and the variance of a certain quantity, however this variance need not be polynomially bounded, even for our structured class of LPs. Following our work, various researchers have suggested that it might be possible to argue that the SAA method actually yields an approximation scheme; in forthcoming work [28] we show that this indeed is true. We show that the convergence proof of our algorithm can be adapted to prove a convergence theorem for the SAA method. We show that for the class of problems considered in this paper, the SAA method converges to a $(1+\epsilon)$-optimal solution to the true problem within a number of samples that is polynomial in the input size, $\frac{1}{\epsilon}$, and $\lambda$.

Independent of our work, Shapiro and Nemirovski [24] have shown that in the black-box model, the bounds on the sample size proved in [16] for the SAA method, are tight, up to polynomial factors, for arbitrary 2-stage stochastic programs. In this paper (and in [28]), we consider a *structured* class (Stoc-P) of 2-stage stochastic LPs that is rich enough to capture the fractional versions of a variety of combinatorial optimization problems, and by exploiting this structure we prove a bound on the sample size that is significantly better than the bound in [16]. In particular, even for the class (Stoc-P) and with small values of $\lambda$, the sample size bound in [16] can be *exponentially large*, whereas our bound is polynomial in $\lambda$. Most recently, and subsequent to the publication of a preliminary version of our work [25], Nemirovski and Shapiro (personal communication) showed that for the 2-stage stochastic set cover problem with non-scenario-dependent costs, if one preprocesses the input to eliminate certain first-stage decisions, then the bound of [16] for the SAA method applied to this modified problem, becomes polynomial in $\lambda$.

# 9 Acknowledgments

We are grateful to Mike Todd and Shane Henderson for useful discussions and very helpful suggestions.

# References

[1] M. Akgül. *Topics in relaxation and ellipsoidal methods*. Pitman Advanced Publishing Program, London, 1997.

[2] E. M. L. Beale. On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society, Series B*, 17:173–184; discussion 194–203, 1955.

[3] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. Mathematical Programming, Series B, 98:49–71, 2003.

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

[5] J. Borwein and A. S.Lewis. *Convex Analysis and Nonlinear Optimization* Springer-Verlag, NY, 2000.

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

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

[8] 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 "The stochastic single node service provision problem", COSOR-Memorandum 99-13, Dept. of Mathematics and Computer Science, Eindhoven, Technical University, Eindhoven, 1999.

[9] 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.

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

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

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

[13] A. Gupta, M. Pál, R. Ravi, and A. Sinha. Boosted sampling: approximation algorithms for stochastic optimization. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 417–426, 2004.

[14] 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 the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 684–693, 2004.

[15] K. Jain and V.V. Vazirani. Approximation algorithms for metric facility location and $k$-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM* 48(2):274–296, 2001.

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

[17] S. Kolliopoulos and N. Young. Tight approximation results for general covering integer programs. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 522–528, 2001.

[18] J. Linderoth, A. Shapiro, and R. K. Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*. To appear.

[19] M. Mahdian. *Facility location and the analysis of algorithms through factor-revealing programs*. Ph.D. thesis, MIT, Cambridge, MA, 2004.

[20] M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 229–242, 2002.

[21] 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.

[22] R. Ravi and A. Sinha. Hedging uncertainty: approximation algorithms for stochastic optimization problems. In *Proceedings of the 10th International Conference on Integer Programming and Combinatorial Optimization*, pages 101–115, 2004.

[23] A. Shapiro. Monte Carlo sampling methods. In A. Ruszczynski and A. Shapiro, editors, *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*, North-Holland, Amsterdam, 2003.

[24] A. Shapiro and A. Nemirovski. On complexity of stochastic programming problems. Published electronically in *Optimization Online*, 2004. http://www.optimization-online.org/DB_FILE/2004/10/978.pdf.

[25] D. B. Shmoys and C. Swamy. Stochastic optimization is (almost) as easy as deterministic optimization. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 228–237, 2004.

[26] D. B. Shmoys, É. Tardos, and K. I. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.

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

[28] C. Swamy and D. B. Shmoys. The sample average approximation method for 2-stage stochastic optimization. *Unpublished manuscript*, 2004.

[29] 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. *Computational Optimization and Applications*, 24:289–333, 2003.