What about Wednesday? Approximation Algorithms for Multistage Stochastic Optimization

Anupam Gupta[1]*, Martin Pál[2]**, R. Ravi[3]***, and Amitabh Sinha[4]

[1] Dept. of Computer Science, Carnegie Mellon University, Pittsburgh PA 15213. `anupamg@cs.cmu.edu`

[2] DIMACS Center, Rutgers University, Piscataway, NJ. `mpal@acm.org`

[3] Tepper School of Business, Carnegie Mellon University, Pittsburgh PA 15213. `ravi@cmu.edu`

[4] Ross School of Business, University of Michigan, Ann Arbor MI 48109. `amitabh@umich.edu`

Abstract. The field of stochastic optimization studies decision making under uncertainty, when only probabilistic information about the future is available. Finding approximate solutions to well-studied optimization problems (such as Steiner tree, Vertex Cover, and Facility Location, to name but a few) presents new challenges when investigated in this framework, which has promoted much research in approximation algorithms.

There has been much interest in optimization problems in the setting of *two-stage stochastic optimization with recourse*, which can be paraphrased as follows: On the first day (Monday), we know a probability distribution π from which client demands will be drawn on Tuesday, and are allowed to make preliminary investments (e.g., installing links, opening facilities) towards meeting this future demand. On Tuesday, the actual requirements are revealed (drawn from the same distribution π) and we must purchase enough additional equipment to satisfy these demands; however, these purchases are now made at an inflated cost. In a recent paper [8], we proposed the *Boosted Sampling* framework which converted an approximation algorithm $\mathcal{A}$ for an optimization problem Π into one for the stochastic version of Π (provided $\mathcal{A}$ satisfied certain technical conditions).

In this paper, we give two generalizations of this *Boosted Sampling* framework: Firstly, we show that a natural extension of the framework works in a general k-stage setting, where information about the future is gradually revealed in several stages and we are allowed to take (increasingly expensive) corrective actions in each stage. We use these to give approximation algorithms for k-stage Steiner Tree, Facility Location and Vertex Cover.

* Supported in part by NSF CAREER award CCF-0448095 and an Alfred P. Sloan Fellowship.

** Supported by NSF grant EIA 02-05116.

*** Supported in part by NSF grant CCR-0105548 and ITR grant CCR-0122581.

Furthermore, the *Boosted Sampling* framework of [8] requires the *inflation parameter* specifying the increase in cost of future actions be independent of the distribution π. In this paper, we show how to extend the framework to the case where this inflation parameter is arbitrarily correlated with the client set S.

1 Introduction

Many problems in planning involve making decisions under uncertainity that unravels in several stages. Demand for new services such as cable television and broadband Internet access originate over time, leading to interesting questions in the general area of installing infrastructure to support demand that evolves over time. For instance, a communications company may want to solve the problem of constructing a network to serve demands as they arise but also keep potential future growth in hot-spots in mind while making investments in costly optic fiber cables. While traditional ways to solve such problems involve casting them as network loading and network expansion problems in each stage and solving for them sequentially, advances in forecasting methods have made a more integrated approach feasible: with the availability of forecasts about how future demands evolve, it is now preferable to use the framework of *multistage stochastic optimization with recourse* to model such problems.

Before we talk about the multistage optimization, let us describe the basic ideas via the example of the *two-stage* stochastic Steiner tree problem: Given a metric space with a root node, the Steiner tree problem is to find a minimum length tree that connects the root to a set of specified terminals S. In the two-stage stochastic version considered recently by several authors [8, 11, 20, 9], information about the set of terminals S is revealed only in the second stage while the probability distribution π from which this set is drawn known in the first stage (either explicitly [11, 20, 9], or as a black box from which one can sample efficiently [8]). One can now purchase some edges F_1 in the first stage—and once the set of terminals $S \subseteq V$ is revealed, one can buy some more edges $F_2(S)$ so that $F_1 \cup F_2(S)$ contains a rooted tree spanning S. The goal is to minimize the cost of F_1 plus the expected cost of the edges F_2. Note however that the edges purchased in the second stage F_2 are costlier by a factor of $\sigma > 1$; it is this that motivates the purchase of some anticipatory edges F_1 in an optimal solution. This is precisely the model that has been studied in the papers [8, 11, 20, 9]. In this paper, we will consider the following k-stage problem, and will give a $2k$-approximation for this problem. (Details of results for other problems appear in Section 1.1.)

The k-stage Problem. In the *k-stage* problem, we are allowed to buy a set of edges in each stage to augment our current solution in reaction to the updated information received in that stage in the form of a *signal*; however, the cost of buying any edge increases with each stage. Let us use σ_i to denote the inflation factor of stage i; i.e., how much more expensive each edge is in comparison to stage $i-1$. Assume $\sigma_1 = 1$; hence purchasing an edge e in stage i costs $c_e \prod_{j=1}^{i} \sigma_i$.

We work with the assumption that costs are non-decreasing, which corresponds to $\sigma_i \geq 1$.

- At the beginning of the i-th stage (where $1 \leq i \leq k-1$), we receive a *signal* s_i that represents the information gained about future terminals that will arise. After this observation s_i, we know that future signals, as well as the set $\mathbf{S}$ of eventual terminals, will come from a revised distribution conditioned on seeing this signal. After observing the signal s_i, we can purchase some more edges F_i at cost $\prod_{j=1}^{i} \sigma_j c(F_i)$.
- Finally, in the k-th stage we observe the realization of the random variable $\mathbf{S} = S$ of terminals, and have to buy the final set F_k so that $\cup_{i=1}^{k} F_k$ is a Steiner tree spanning S. Our goal is to minimize the expected cost incurred in all stages together, namely

$$\mathbf{E}[\sum_{i=1}^{k} (\prod_{j \leq i} \sigma_j)\ c(F_i)\,]. \qquad (1)$$

This multistage framework can be naturally extended to model problems like Vertex Cover and Facility Location as well; we give the formal definitions in Section 2. We then extend the *Boosted Sampling* framework from [8] to the multistage situation, and use this to give approximation algorithms for Stochastic Steiner Tree, Facility Location, and Vertex Cover.

1.1 Informal Description of Results

In this paper we extend the *Boosted Sampling* framework for two-stage stochastic optimization problems with recourse. The framework and terminology is defined in Section 2. This framework had been proposed in our earlier paper [8]; given an inflation parameter σ, and a distribution π over second-stage scenarios, the following procedure was used to translate optimization algorithms for deterministic problems to their two-stage stochastic variants:

```
1: Boosted Sampling: Sample $\sigma$ times from the distribution $\pi$ to get sets of clients $D_1, \ldots, D_\sigma$.
2: Building First Stage Solution: Build an $\alpha$-approximate solution for the clients $D = \cup_i D_i$.
3: Building Recourse: When actual future in the form of a set $S$ of clients appears (with probability $\pi(S)$), augment the solution of Step 2 to a feasible solution for $S$.
```

Fig. 1. Algorithm Boost-and-Sample(Π)

In [8], we proved that given an α-approximation algorithm for the deterministic version $\mathsf{Det}(\Pi)$ of any problem Π, boosted sampling would yield an approximation algorithm for the two-stage stochastic version $\mathsf{Stoc}(\Pi)$, as long as the deterministic algorithm satisfied certain technical cost-sharing conditions; moreover, the sampling framework worked even if the probability distribution π over the second-stage client sets was specified using a *black-box* from which one could draw samples efficiently.

Multistage Results. We first show (in Section 3) how to extend the boosted sampling framework to handle k-stage stochastic variants of problems (for all $k \geq 2$), and give the technical conditions under which effective approximations can be obtained. (See Theorems 6 and 7 for the precise statements). As in [8], these technical conditions are phrased in terms of the existence of certain "good" cost-sharing functions related to the approximation algorithms. In particular, we want the cost-shares to satisfy both *strictness* and *cross-monotonicity*; details and definitions appear in Section 2.

In Section 5, we show the existence of these "good" cost-shares for some problems, thus giving us constant-factor approximation algorithms when the number of stages is a constant. In particular, we look at the Steiner tree problem (where we get a $2k$-approximation for the k-stage problem), Facility Location (an approximation of $3 \cdot 2^k$), and Vertex Cover (at most 4^k). A more precise summary of our results for the k-stage versions of these problems is in the last column of Figure 2.

Correlated Inflation. As outlined in Figure 1, boosted sampling assumes a fixed *deterministic* inflation parameter σ. If this inflation parameter $\boldsymbol{\sigma}$ is random but *independent* of the distribution π over scenarios, one can just use $\mathbf{E}[\boldsymbol{\sigma}]$ in the place of σ. This independence assumption is somewhat restrictive, as the equipment prices often correlate with demands. Here, using the expected value can lead to very poor approximations.

In Section 4, we give a simple way to extend the Boosted Sampling framework to the case when $\boldsymbol{\sigma}$ is arbitrarily correlated with the distribution π without losing anything in the appproximation ratios.

1.2 Related Work

There is a huge body of work in the Operations Research community on multi-stage stochastic optimization with recourse; the study of stochastic optimization [3, 15] dates back to the work of Dantzig [4] and Beale [2] in 1955. Stochastic linear programming was defined in these papers, and have been very widely studied since, with gradient-based and decomposition-based approaches being known for some versions of stochastic linear programming. On the other hand, only moderate progress has been reported for stochastic integer (and mixed-integer) programming in both theoretical and computational domains; see [21, 16] for details.

The study of stochastic versions of NP-hard problems has received some attention lately in the theoretical computer science community, and approximation algorithms for *two-stage* stochastic programming versions of a variety of combinatorial optimization problems have been devised actively in several recent papers [8, 11, 20, 9, 22, 5].

Shmoys and Swamy [23] have recently shown that for a broad class of multistage stochastic linear programs, a $(1+\epsilon)$-approximate solution can be found in polynomial time using a Sampled-Average-Approximation approach. They show that for several problems (including Facility Location, Set Cover and Vertex Cover), the LP solution for each stage can be rounded to an integer solution *independently of other stages*. In contrast, our technique requires strict cost shares, but does not depend on the existence of a suitable LP relaxation, or the ability to round each stage independently.

Independently of our work, Hayrapetyan et al. [10] have also devised approximation algorithms for the multistage version of the Stochastic Steiner tree problem that we consider, using a reduction to a variant of an information network gathering problem which they address in their paper. They also provide an $O(k)$-approximation algorithm for multistage Stochastic Steiner tree (our approximation ratio is $2k$). However, their techniques do not seem to extend to the other covering problems that we address in this paper.

2 Basic Model and Notation

Let us define an abstract combinatorial optimization problem Π that we will adapt to a stochastic setting. The optimization problem Π is defined by U, the universe of *clients* (or demands), and the set X of *elements* we can purchase. For a subset $F \subseteq X$ of elements, let $c(F) = \sum_{e \in F} c_e$ denote the *cost* of F. Given a set S of clients, a solution F that *satisfies* each client $j \in S$ is labeled *feasible* for S. The definition of satisfaction naturally depends on the problem; e.g., in the (rooted) Steiner tree problem on a graph $G = (V, E)$, the universe of clients is the set of possible terminals ($U = V$), the element set X is the set of edges E, and a terminal $j \in S$ is satisfied by $F \subseteq X$ if F contains a path from j to the root vertex r. The cost of a set of edges $F \subseteq X$ is $c(F) = \sum_{e \in F} c_e$.

Given a set $S \subseteq U$ of clients, we let $\mathsf{Sols}(S) \subseteq 2^X$ be the set of *feasible solutions* for S. Given a client set $S \subseteq U$, the *deterministic version* $\mathsf{Det}(\Pi)$ of Π asks us to find a solution $F \in \mathsf{Sols}(S)$ of minimum cost. We denote by $\mathsf{OPT}(S)$ the cost of this minimum cost solution.

Definition 1. *A problem Π is* sub-additive *if for any S and S' being two sets of clients with solutions $F \in \mathsf{Sols}(S)$ and $F' \in \mathsf{Sols}(S')$, we have that (i) $S \cup S'$ is a legal set of clients for Π, and (ii) $F \cup F' \in \mathsf{Sols}(S \cup S')$.*

As in previous papers which give approximation algorithms for two-stage stochastic optimization problems [8, 11, 20], we restrict our attention to sub-additive problems. (Note that the sub-additivity in the rooted Steiner tree problem is ensured by the presence of the root r.)

Given any problem Π, we study the variant when the set of clients (or requirements) is not known in advance, but is revealed gradually. We proceed to build the solution in stages; in each stage, we gain a more precise estimate of the requirements of clients, and then can buy or extend a partial solution (at gradually increasing cost) in response to this updated information. Ultimately, we learn the entire set S of clients or requirements, and then must complete the existing partial solution to a feasible solution $F \in \mathsf{Sols}(S)$.

Multi-stage Stochastic Optimization Problems. We can now define stochastic variants $\mathsf{Stoc}(\Pi)$ of the problem Π. In this model, we obtain increasingly precise forecasts about user demands over several stages as in the Steiner tree example. In the $\underline{k\text{-stage problem}}$, we are allowed to buy a set of elements in each stage to augment our current solution in reaction to the updated information received in that stage; however, the cost of buying an element $e \in X$ is increasing with each stage. Extending the existing terminology, we use σ_i to denote the inflation factor of stage i; i.e., how much more expensive each element is in comparison to stage $i-1$. For completeness, we define $\sigma_1 = 1$. Hence purchasing an element $e \in X$ in stage i costs $c_e \prod_{j=1}^{i} \sigma_i$. We assume that costs are non-decreasing, which corresponds to $\sigma_i \geq 1$.

- At the beginning of the $\underline{i\text{-th stage}}$ (where $1 \leq i \leq k-1$), we receive a *signal* s_i that represents the information gained that we can use to correct our anticipation of the demands. Formally, the signal $\mathbf{s}_i$ is a random variable correlated with $\mathbf{S}$ and in stage i we observe s_i, a realization of $\mathbf{s}_i$. (Note that the signal $\mathbf{s}_1$ is a dummy signal, but we use it to simplify notation.) After this observation s_i, we know that future signals, as well as the set $\mathbf{S}$ of demands will come from the conditional distribution $[\pi | \mathbf{s}_1 = s_1, \mathbf{s}_2 = s_3, \ldots, \mathbf{s}_i = s_i]$. After observing the signal s_i, we can purchase some more elements $F_i \subseteq X$ at cost $\prod_{j=1}^{i} \sigma_j c(F_i)$.
- Finally, in the $\underline{k\text{-th stage}}$ we observe the realization of the random variable $\mathbf{S} = S$, and have to buy the final set F_k so that $\cup_{i=1}^{k} F_k \in \mathsf{Sols}(S)$. Again, our goal is to minimize the expected cost incurred in all stages together, that is,

  $$Z = \mathbf{E}\big[\ \textstyle\sum_{i=1}^{k} \left(\prod_{j \leq i} \sigma_j\right)\, c(F_i)\ \big].$$

  Note that each of the partial solutions $F_i = F_i(s_1, s_2, \ldots, s_i)$ may depend on signals up to stage i, but not on signals observed in the subsequent stages.

2.1 Cost sharing functions

We now define the notion of *cost shares* that we will crucially use to analyze our approximation algorithms. Loosely, a cost-sharing function ξ divides the cost of a solution $F \in \mathsf{Sols}(S)$ among the clients in S. Cost-sharing functions have long been used in game-theory (see, e.g., [13, 14, 18, 19, 24]).

Cost-shares with a closer algorithmic connection were recently defined by Gupta et al. [7] to analyze a randomized algorithm for the Rent-or-Buy network design problem. In this paper, we have to redefine *strict* cost-sharing functions slightly: in contrast to previously used definitions, our cost-shares are defined by, and relative to, an approximation algorithm $\mathcal{A}$ for the problem Π.

Definition 2 (Cost-shares). *A* cost-sharing algorithm $\mathcal{A}$ *for a problem* Π *takes an instance* (X, S) *of* Π *and outputs* **(a)** *a solution* $F \subseteq X$ *with* $F \in \mathsf{Sols}(S)$*, and* **(b)** *a real value* $\xi(X, S, j) \geq 0$ *for each client* $j \in S$*. This value* $\xi(X, S, j)$ *is called the* cost-share *of client* j*, and the function* $\xi(\cdot, \cdot, \cdot)$ *computed by* $\mathcal{A}$ *is the* cost sharing function *associated with* $\mathcal{A}$*.*

Cross-monotonicity of cost-sharing functions is a useful property often used in the game-theoretic as well as algorithmic literature:

Definition 3 (Cross-monotonicity). *A cost-sharing function* ξ *is* cross-monotone *if for every pair of client sets* $S \subseteq T$ *and client* $j \in S$*, we have* $\xi(X, T, j) \leq \xi(X, S, j)$*.*

We also require all cost-sharing functions to provide a lower bound on the cost of the optimal solution, in order to use the cost-sharing function to provide bounds on the cost of our solution.

Definition 4 (Competitiveness). *A cost-sharing function* ξ *is* competitive *if for every client set* S*, it holds that*

$$\textstyle\sum_{j \in S} \xi(X, S, j) \leq \mathsf{OPT}(X, S). \tag{2}$$

For a subset of clients $S' \subseteq S$, let $\xi(X, S, S')$ denote the sum $\sum_{j \in S'} \xi(X, S, j)$; thus competitiveness is the property that $\xi(X, S, S) \leq \mathsf{OPT}(X, S)$. In this paper, we will focus solely on competitive ξ.

Crucial to our proofs is the notion of *strictness* that relates the cost of extending a solution on S so as to serve more clients T to the cost shares of T. Formally, given a set X of elements and S of clients, let $F = \mathcal{A}(X, S)$ denote the solution found by algorithm $\mathcal{A}$. We create a new *reduced* instance of the problem Π by zeroing out the cost of all elements in F; this instance is denoted by X/F.

Definition 5 (Strictness [8]). *A cost-sharing algorithm* $\mathcal{A}$ *is* β-strict *if for any sets of clients* S, T *there exists a solution* $F_T \subseteq X$ *constructible in polynomial time such that* $\mathcal{A}(X, S) \cup F_T \in \mathsf{Sols}(T)$ *and*

$$c(F_T) \leq \beta \times \xi(X, S \cup T, T). \tag{3}$$

A subtly different notion of strictness is **c-strictness**; here the “c” is supposed to emphasize the enhanced role of the cost-shares.

Definition 6 (c-strictness of ξ). *Let* $S, T \subseteq U$ *be sets of clients, and let* X *be an instance of* Π*. The cost-sharing function* ξ *given by an algorithm* $\mathcal{A}$ *is* β*-c-strict if*

$$\xi(X/\mathcal{A}(X, S), T, T) \leq \beta \times \xi(X, S \cup T, T). \tag{4}$$

In other words, the total cost shares for the set T of clients in the reduced instance $X/\mathcal{A}(X, S)$ is at most β times the cost-shares for T if the clients in S were present as well. Note that if ξ is a competitive cost sharing function, then (3) implies (4), and ξ is β-c-strict if it is β-strict.

Recall that $\mathcal{A}$ is an α-approximation algorithm if $c(\mathcal{A}(X,S)) \leq \alpha \ \mathsf{OPT}(X,S)$. In this paper, we will need the following stronger guarantee (which goes hand-in-hand with c-strictness):

Definition 7 (Approximation w.r.t. ξ). *An algorithm $\mathcal{A}$ (with cost-sharing function ξ) is* an α-approximation algorithm with respect to ξ *if*

$$c(\mathcal{A}(X,S)) \leq \alpha \ \xi(X,S,S). \tag{5}$$

If ξ is competitive, then $\xi(X,S,S) \leq c(\mathsf{OPT}(S))$, and thus an α-approximation algorithm w.r.t. ξ is also simply an α-approximation algorithm. Moreover, (5) and (4) together implies that an α-approximation algorithm w.r.t. β-c-strict ξ is $(\alpha\beta)$-strict in the sense of Definition 5.

2.2 New Results on c-Strictness and Multi-stage Approximations

Having laid down the crucial definitions, we can finally state the main theorems for multi-stage stochastic covering problems.

Theorem 1. *There is an $\alpha \cdot \sum_{i=0}^{k-1} \beta^i$-approximation algorithm for the k-stage stochastic problem $\mathsf{Stoc_k}(\Pi)$ if the corresponding problem Π has an α-approximation algorithm $\mathcal{A}$ with respect to a β-c-strict cost-sharing function ξ, and this ξ is cross-monotone.*

A slightly weaker version of the above theorem can be proved without cross-monotone cost-shares: this is useful for problems like Vertex Cover for which good cross-monotone cost-shares do not exist [12].

Theorem 2. *There is an $\alpha \cdot \sum_{i=0}^{k-1} \beta_1 {\beta_2}^i$-approximation algorithm for the k-stage stochastic problem $\mathsf{Stoc_k}(\Pi)$ if the corresponding problem Π has an α-approximation algorithm $\mathcal{A}$ with respect to a β_1-c-strict cost-sharing function ξ, and $\mathcal{A}$ is also β_2-strict with respect to this ξ.*

Finally, since the previous results on strictness in [8] were not given in terms of c-strict cost-sharing functions, we adapt, restate (and in most cases, also improve) the guarantees here. Figure 2 summarizes the results in this paper.

Theorem 3. *There is a 2-approximation algorithm for the Minimum Steiner Tree problem w.r.t. a 1-c-strict cost sharing function ξ. Furthermore, this ξ is also cross-monotone.*

Theorem 4. *The Uncapacitated Facility Location problem admits a 3-approximation algorithm w.r.t. a 2-c-strict ξ. This cost-sharing function ξ is also cross-monotone.*

Theorem 5. *The Vertex Cover problem has 2-approximation algorithm w.r.t. an associated 2-strict (and hence 2-c-strict) cost-sharing function ξ.*

Problem	Approximation ratio α w.r.t. ξ	c-Strictness of ξ	Is ξ X-mono?	k-Stage Stochastic Approx.
Steiner Tree	2	1	Yes	$2k$
Facility Location	3	2	Yes	$3(2^k - 1)$
Vertex Cover	2	2	No	$\frac{2}{3}(4^k - 1)$

Fig. 2. A summary of the results in this paper

3 Multiple Stage Stochastic Optimization

For ease of exposition, let us state the algorithm assuming that the inflation factors σ_i are deterministically known in advance. With some extra work as in Section 4, randomly varying inflation factors can be handled—the details are deferred to a full version of this paper.

Let us first outline the key idea of the algorithm. In the two-stage Boosted Sampling framework with inflation being σ (see Figure 1), the first stage involved simulating σ independent runs of the second stage and building a solution that satisfied the union of these simulations. We use the same basic idea for the k-stage problem: in each stage i, we simulate σ_{i+1} "copies" of the remaining $(k-i)$-stage stochastic process; each such "copy" provides us with a (random) set of clients to satisfy, and we build a solution that satisfies the union of all these clients. The "base case" of this recursive idea is the final stage, where we get a set S of clients, and just build the required solution for S.

```
1: (Base case.) If $i = k$, draw one sample set of clients $S_k$ from the conditional
   distribution $[\pi | s_1, \dots, s_k]$. Return the set $S_k$.
2: (New samples.) If $i < k$, draw $\lfloor \sigma_{i+1} \rfloor$ samples of the signal $\mathbf{s}_{i+1}$ from the conditional
   distribution $[\pi | s_1, \dots, s_i]$. Let $s^1, \dots, s^n$ be the sampled signals (where $n = \lfloor \sigma_{i+1} \rfloor$).
3: (Recursive calls.) For each sample signal $s^j$, $j = 1, \dots, n$, recursively call Recur-
   Sample($\Pi, i+1, s_1, \dots, s_i, s^j$) to obtain a sample set of clients $S^j$. Return the set
   $S_i = \cup_{j=1}^{n} S^j$.
```

Fig. 3. Procedure Recur-Sample(Π, stage $i, s_1, \dots, s_i$)

Our algorithm, in each stage i (except the final, k-th stage), uses a very natural recursive sampling procedure that emulates σ_{i+1} executions of itself on the remaining $(k-i)$ stages. (This sampler is specified in Figure 3.) Having obtained a collection of sampled sets of clients, it then augments the current partial solution to a feasible solution for these sampled sets. Finally, in the k-th stage it performs the ultimate augmentation to obtain a feasible solution for the revealed set of demands. The expected number of calls to the black box required in stage i will be $\prod_{j=i+1}^{k} \sigma_j$.

The definition of the sampling routine is given in Figure 3, and the procedure Multi-Boost-and-Sample(Π, i) to be executed in round i is specified in Figure 4. Note that if we set $k = 2$, we get back precisely the Boosted Sampling framework from our previous paper [8].

```
1: (External signal.) If the current stage $i < k$, observe the signal $s_i$. If this is the final
   stage $i = k$, observe the required set of clients $S$ instead.
2: (Sample.) If $i = k$, let $D_k := S$. Else $i < k$, and then use procedure Recur-
   Sample($\Pi, i, s_1, \ldots, s_i$) to obtain a sample set of clients $D_i$.
3: (Augment solution.) Let $B_i = \cup_{j=1}^{i-1} F_j$ be the elements that were bought in earlier
   rounds. Set the costs of elements $e \in B_i$ to zero. Using algorithm $\mathcal{A}$, find a set of
   elements $F_i \subseteq X \setminus B_i$ to buy so that $(F_i \cup B_i) \in$ Sols($D_i$).
```

Fig. 4. Algorithm Multi-Boost-and-Sample(Π, i)

3.1 The Analysis

We will now show that this extended framework can be used to translate an approximation algorithm $\mathcal{A}$ for the deterministic version Det(Π) of a problem Π to its k-stage stochastic version $\mathsf{Stoc_k}(\Pi)$. The quality of this translation depends on the approximation guarantee of $\mathcal{A}$ with respect to some c-strict cost-sharing function. The main result of this section is the following:

Theorem 6. *Given a problem Π, if $\mathcal{A}$ is an α-approximation algorithm w.r.t. a β-c-strict cost-sharing function ξ, and if ξ is cross-monotone, then* Multi-Boost-and-Sample*(Π) is an $\alpha \cdot \sum_{i=0}^{k-1} \beta^i$-approximation algorithm for the k-stage stochastic problem* $\mathsf{Stoc_k}(\Pi)$.

Remark 1. We would like to note that the additional assumption of cross-monotonicity in Theorem 6 is not completely satisfactory. As we show in Theorem 7 in Section 3.2, this assumption can be removed at the expense of somewhat worse approximation ratio. This is useful for problems like Vertex Cover, for which a cross-monotone cost sharing function with good guarantees is not available [12].

Before we prove Theorem 6, we set the stage for the proof by providing a brief overview of the proof technique, and proving a couple of lemmas which provide useful bounds. A naïve attempt to prove this result along the lines of our previous paper[8] does not succced, since we have to move between the cost-shares and the cost of the solutions F_i, which causes us to lose factors of $\approx \alpha$ at each step. Instead, we bound all costs incurred in terms of the ξ's: we first argue that the expected sum of cost-shares paid in the first stage is no more than the optimum total expected cost Z^*, and then bound the sum of cost-shares in each consecutive stage in terms of the expected cost shares from the previous stage (with a loss of a factor of β at each stage). Finally, we bound the actual cost of the partial solution constructed at stage i by α times the expected cost shares for that stage, which gives us the geometric sum claimed in the theorem.

Let F^* be an optimal solution to the given instance of $\mathsf{Stoc_k}(\Pi)$. We denote by F_i^* the partial solution built in stage i; recall that $F_i^* = F_i^*(s_1, s_2, \ldots, s_i)$ is a function of the set of all possible i-tuples of signals that could be observed before stage i. The expected cost of this solution can be expressed as

$$Z^* = \sigma_1 \mathbf{E}[c(F_1^*(\mathbf{s}_1))] + \sigma_1\sigma_2 \mathbf{E}[c(F_2^*(\mathbf{s}_1, \mathbf{s}_2))] + \cdots + \sigma_1 \ldots \sigma_n \mathbf{E}[c(F_k^*(\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k))].$$

Lemma 1. *The expected cost share $\mathbf{E}[\xi(X, D_1, D_1)]$ is at most the total optimum cost Z^*.*

Proof. Consider D_1, the sample set of clients returned by $\mathsf{Recur\text{-}Sample}(\mathit{\Pi}, 1, s_1)$. We claim there is a solution $\widehat{F}(D_1)$, such that $\mathbf{E}[c(\widehat{F}(D_1))] \leq Z^*$ (the expectation is over the execution of the procedure Recur-Sample). To construct the solution $\widehat{F}(D_1)$, we consider the tree of recursive calls of the procedure Recur-Sample. For each recursive call $\mathsf{Recur\text{-}Sample}(\mathit{\Pi}, i, s_1, \ldots, s_i)$, we add the set of elements $F_i^*(s_1, \ldots, s_i)$ to $\widehat{F}(D_1)$. It is relatively straightforward to establish that (1) $\widehat{F}(D_1)$ is a feasible solution for the set D_1 and (2) the expected cost

$$\mathbf{E}[c(\widehat{F}(D_1))] \leq \mathbf{E}\left[\sum_{i=1}^{k} \Big(\prod_{j \leq i} \sigma_j\Big) c(F_i^*)\right] = Z^*. \tag{6}$$

Since the expected cost of a feasible solution for D_1 is bounded above by Z^*, the competitiveness of ξ implies that this bound must hold for the sum of cost shares as well.

Lemma 2. *Let $\hat{F} = F_1 \cup \cdots \cup F_{i-1}$ be the solution constructed in a particular execution of the first $i-1$ stages, and let s_i be the signal observed in stage i. Let D_i and D_{i+1} be the random variables denoting the samples returned by the procedure* Recur-Sample *in Stages i and $i+1$, and let F_i be the (random) solution constructed by $\mathcal{A}$ for the set of clients D_i. Then,*

$$\mathbf{E}[\xi(X/(\widehat{F} \cup F_i), D_{i+1}, D_{i+1})] \leq \frac{\beta}{\sigma_{i+1}} \cdot \mathbf{E}[\xi(X/\widehat{F}, D_i, D_i)]. \tag{7}$$

Proof. Recall that the sampling procedure $\mathsf{Recur\text{-}Sample}(\mathit{\Pi}, i, s_1, \ldots, s_i)$ gets $n = \lfloor \sigma_{i+1} \rfloor$ independent samples $s^1, s^2, \ldots s^n$ of the signal $\mathbf{s}_{i+1}$ from the distribution π conditioned on $s_1, \ldots, s_i$, and then for each sampled signal calls itself recursively to obtain the n sets $S^1, \ldots, S^n$. Note that the set $D_i = \bigcup_{j=1}^{n} S^j$ is simply the union of these n sets. On the other hand, the set D_{i+1} is obtained by observing the signal s_{i+1} (which is assumed to come from the same distribution $[\pi | s_1, \ldots, s_i]$), and then calling Recur-Sample with the observed value of s_{i+1}.

We now consider an alternate, probabilistically equivalent view of this process. First take $n+1$ samples $s^1, \ldots, s^{n+1}$ of the signal $\mathbf{s}_{i+1}$ from the distribution

$[\pi|s_1,\ldots,s_i]$. Call the procedure $\textsc{Sample}(\pi, i+1, s_1, \ldots, s_i, s^j)$ for each s^j to obtain sets $S^1, \ldots, S^{n+1}$. Pick an index j uniformly at random from the set of integers $1, \ldots, n+1$. Let $D_{i+1} = S^j$, and let D_i be the union of the remaining n sets. This process of randomly constructing the pair of sets (D_i, D_{i+1}) is clearly equivalent to the original process. Note that $D_i \cup D_{i+1} = \bigcup_{l=1}^{n+1} S^l$.

To simplify notation, let us denote $X/\widehat{F}$ by $\widehat{X}$. By the definition of β-c-strictness, we first get

$$\xi(\widehat{X}/F_i, D_{i+1}, D_{i+1}) \leq \beta \cdot \xi(\widehat{X}, D_i \cup D_{i+1}, D_{i+1}), \tag{8}$$

By the relation that D_{i+1} is equivalent to S^j sampled uniformly from $n+1$ alternates in the equivalent process above and that D_i is the union of the remaining n sets, we have

$$\mathbf{E}[\xi(\widehat{X}, D_i \cup D_{i+1}, S^j)] \leq \mathbf{E}\left[\frac{1}{n} \times \xi(\widehat{X}, D_i \cup D_{i+1}, D_i)\right]. \tag{9}$$

Now we use cross-monotonicity of the cost shares (which says that the cost shares of D_i should not increase when the elements of $D_{i+1} \setminus D_i$ join the fray), and finally get

$$\mathbf{E}\Big[\xi(\widehat{X}, D_i \cup D_{i+1}, D_i)\Big] \leq \mathbf{E}\Big[\xi(\widehat{X}, D_i, D_i)\Big]. \tag{10}$$

Chaining the above inequalities (8–10) proves the lemma.

Proof. (of Theorem 6) Recall that the expected cost of the solution given by Algorithm Multi-Boost-and-Sample(Π) is:

$$E[Z] = E\Big[\sum_{i=1}^{k} \big(\prod_{j=1}^{i} \sigma_j\big) c(F_i)\Big]$$

Using cross-monotonicity and the fact that $\mathcal{A}$ is an α-c-approximation algorithm with respect to the β-c-strict cost-shares ξ, we have:

$$E[Z] \leq \alpha E\Big[\sum_{i=1}^{k} \big(\prod_{j=1}^{i} \sigma_j\big) \xi(X/B_i, F_i, F_i)\Big]$$

Using Lemma 2 inductively on $\xi(X/B_i, F_i, F_i)$, we find that $\xi(X/B_i, F_i, F_i) \leq \frac{\beta^i}{\prod_{j=1}^{i} \sigma_j} \xi(X, D_1, D_1)$. Using this inequality in the bound for $E[Z]$ above:

$$E[Z] \leq \alpha E\Big[\sum_{i=1}^{k} \beta^i \xi(X, D_1, D_1)\Big]$$

Lemma 1 bounds $\xi(X, D_1, D_1)$ from above by Z^*. Using this bound in the inequality above completes the proof of Theorem 6.

3.2 Guarantees without cross-monotone cost-shares

We prove the following weaker theorem about the performance of Multi-Boost-and-Sample(Π) for problems where the cost shares are not cross-monotone.

Theorem 7. *If $\mathcal{A}$ is an α-c-approximation algorithm for Π with respect to β_1-c-strict cost-shares ξ that are also β_2-strict, then* Multi-Boost-and-Sample*(Π) is a $\alpha \cdot \sum_{i=0}^{k-1}(\beta_1\beta_2)^i$ approximation algorithm for the k-stage stochastic problem* $\mathsf{Stoc_k}(\Pi)$.

The proof of Theorem 7 proceeds along the lines of the proof of Theorem 6, except that the induction lemma (Lemma 2) needs to be adapted for cost-shares which are no longer cross-monotone. The new lemma is stated and proved below. We omit the proof of Theorem 7 for brevity, since it can be proved by combining Lemmas 1 and the following lemma exactly as in the proof of Theorem 6.

Lemma 3. *Let $\hat{F} = F_1 \cup \cdots \cup F_{i-1}$ be the solution constructed in a particular execution of the first $i-1$ stages, and let s_i be the signal observed in stage i. Let D_i and D_{i+1} be the random variables denoting the samples returned by the procedure* Recur-Sample *in Stages i and $i+1$, and let F_i be the (random) solution constructed by $\mathcal{A}$ for the set of clients D_i. Then,*

$$\mathbf{E}[\xi(X/(\widehat{F} \cup F_i), D_{i+1}, D_{i+1})] \leq \frac{\beta_1\beta_2}{\sigma_{i+1}} \cdot \mathbf{E}[\xi(X/\widehat{F}, D_i, D_i)]. \tag{11}$$

Proof. We proceed just as in the proof of Lemma 2, using the same notation. As in the previous lemma, we use the definition of β_1-c-strictness and the equivalent sampling process defined in the proof of Lemma 2 to obtain the following two inequalities:

$$\xi(\widehat{X}/F_i, D_{i+1}, D_{i+1}) \leq \beta_1 \cdot \xi(\widehat{X}, D_i \cup D_{i+1}, D_{i+1}), \tag{12}$$

$$\mathbf{E}[\xi(\widehat{X}, D_i \cup D_{i+1}, S^j)] \leq \mathbf{E}[\frac{1}{n+1}\xi(\widehat{X}, D_i \cup D_{i+1}, D_i \cup D_{i+1})]. \tag{13}$$

Now, by the competitiveness of the cost shares, we have

$$\mathbf{E}[\xi(\widehat{X}, D_i \cup D_{i+1}, D_i \cup D_{i+1})] \leq \mathbf{E}\Big[\mathsf{OPT}(\widehat{X}, D_i \cup D_{i+1})\Big]. \tag{14}$$

From symmetry and subadditivity, we get

$$\mathbf{E}\Big[\mathsf{OPT}(\widehat{X}, D_i \cup D_{i+1})\Big] \leq \mathbf{E}\left[\frac{n+1}{n}\mathsf{OPT}(\widehat{X}, D_i)\right]. \tag{15}$$

Finally, by (regular) β_2-strictness, we get

$$\mathbf{E}\Big[\mathsf{OPT}(\widehat{X}, D_i)\Big] \leq \mathbf{E}[\beta_2 \cdot \xi(\widehat{X}, D_i, D_i)] \tag{16}$$

Putting all the above inequalities together gives the lemma.

4 Correlated Inflation Factors

In this section, we show how to extend the basic Boosted Sampling framework to work in the case where the inflation factor σ is a random variable arbitrarily correlated with the random scenarios. For brevity, we describe the idea only for the two-stage setting, though the same idea can be used for the multi-stage framework of Section 3.

Formally, let us assume that we have access to a distribution π' over $\mathbb{R}_{\geq 1} \times 2^U$, where $\pi'(\sigma, S)$ is the probability that the set S arrives and the inflation factor is σ. We assume that we know an integer $M \in \mathbb{Z}$ which is an upper bound on the value of the inflation parameter σ; i.e., with probability 1, it should be the case that $\sigma \leq M$ holds. (Note that choosing a pessimistic value of M will only increase the running time, but not degrade the approximation guarantee of the framework.)

```
1: Boosted Sampling: Draw $M$ independent samples from the joint distribution $\pi'$ of
   $(\boldsymbol{\sigma}, \mathbf{S})$. Let $(\sigma_1, S_1)$, $(\sigma_2, S_2)$,..., $(\sigma_M, S_M)$ denote this collection of samples.
2: Rejection Stage: For $i = 1, \ldots, M$, accept the sample $S_i$ with probability $\sigma_i/M$. Let
   $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$ be the accepted samples, and let $S = \bigcup_j S_{i_j}$.
3: First-stage Solution: Using the algorithm $\mathcal{A}$, construct an $\alpha$-approximate first-stage
   solution $F^1 \in \mathsf{Sols}(S)$.
4: Second-stage Recourse: Let $T$ be the set of clients realized in the second stage.
   Use an augmenting algorithm to compute the second-stage solution $F^2$ such that
   $F^1 \cup F^2 \in \mathsf{Sols}(T)$.
```

Fig. 5. Algorithm General-Boost-and-Sample(Π)

The algorithm General-Boost-and-Sample(Π) is given in Figure 5. Note that if $\boldsymbol{\sigma}$ is a constant, then we would behave identically to the original Boosted Sampling framework. To get some intuition for the new steps, note that if the sampled inflation factor σ_i is large, this indicates that we want to handle the associated S_i in the first stage; on the other hand, if the σ_i is small, we can afford to wait until the second-stage to handle the associated S_i—and this is indeed what the algorithm does, albeit in a probabilistic way.

The following is an extension of the main structural theorem proved in our earlier paper [8]:

Theorem 8. *Consider a sub-additive combinatorial optimization problem Π, and let $\mathcal{A}$ be an α-approximation algorithm for its deterministic version* Det(Π)*. If $\mathcal{A}$ admits a β-strict cost sharing function, then* General-Boost-and-Sample(Π) *is an $(\alpha + \beta)$-approximation algorithm for* Stoc(Π)*.*

Proof. Let us transform the "random inflation" stochastic problem instance (X, π') to one with a fixed inflation factor thus: the distribution

$$\widehat{\pi}(\sigma, S) = \pi'(\sigma, S) \times (\sigma/M); \tag{17}$$

note that this ensures that $\sum_{\sigma,S} \widehat{\pi}(\sigma, S) \leq 1$, and hence we can increase the probability $\widehat{\pi}(1, \emptyset)$ so that the sum becomes exactly 1 and $\widehat{\pi}$ is a well-defined probability distribution. The inflation factor for this new instance is set to M, and hence the σ output by $\widehat{\pi}$ is only for expositional ease.

Now the objective for this new problem is to minimize the expected cost under this new distribution, which is

$$c(F_0) + \textstyle\sum_{\sigma,S} \widehat{\pi}(\sigma, S)\ M\ c(F_S) \ = \ c(F_0) + \textstyle\sum_{\sigma,S} \pi'(\sigma, S)\ (\sigma/M)\ M\ c(F_S),$$

which is the same as the original objective function; hence the two problems are identical, and running Boost-and-Sample on this new distribution $\widehat{\pi}$ with inflation parameter M would give us an $(\alpha + \beta)$-approximation.

Finally, note that one can implement $\widehat{\pi}$ given black-box access to π' by just rejecting any sample (σ, S) with probability σ/M. Including this implementation within Boost-and-Sample gives us precisely the above General-Boost-and-Sample, which completes the proof of the theorem.

This immediately implies a 3.55-approximation for Stochastic Steiner tree, a 4-approximation for Stochastic Vertex Cover, and a 5.45-approximation for Stochastic Facility Location even when the second-stage inflation factors are drawn from a distribution that may be arbitrarily correlated to the set of clients materializing in the second stage.

A naïve attempt to extend the Boosted Sampling algorithm of [8] might proceed by obtaining $E[\boldsymbol{\sigma}]$ samples and invoking the algorithm. Unfortunately, this approach is doomed to fail, as the following example shows. Indeed, consider the case where with probability $\frac{1}{2}$, the inflation $\sigma = M \gg 1$ but $S = \emptyset$, and with probability $\frac{1}{2}$, the inflation $\sigma \approx 1$ but $S \neq \emptyset$; the average value of σ is $\approx M/2$, but in fact we should be ignoring the high σ's.

5 New and Improved Strictness Results

In this section, we prove our results about approximation algorithms for the problems under consideration; in order to use Theorem 6, we have to give our approximation guarantees *with respect to the associated c-strict cost-sharing functions.*

5.1 Steiner Tree

Theorem 3. *There is a* 2*-approximation algorithm for the Minimum Steiner Tree problem w.r.t. a* 1*-c-strict cost sharing function* ξ*. Furthermore, this* ξ *is also cross-monotone.*

Proof. The cost sharing function ξ for Steiner tree given by Jain and Vazirani [13] is cross-monotonic, and the minimum spanning tree heuristic is a 2-approximation algorithm with respect to it. The fact that ξ is 1-c-strict can be verified by comparing the executions of the primal-dual algorithm of [13] (which is really an instance of the algorithms of [1, 6]) on the inputs (X, T) and $(X/\mathbb{T}, T)$, where $\mathbb{T}$ is *any* tree connecting the set S to the root (in fact, a slightly more complex argument works with $\mathbb{T}$ being an arbitrary set of edges). It can be observed that any non-root cluster in the latter execution behaves identically to its matching cluster in the former execution, hence its members accrue the same cost shares in both runs. A non-root component can only attach to the root component earlier in the latter execution, since the root component in the execution on $(X/\mathbb{T}, T)$ is larger (i.e. it contains all the vertices of $\mathbb{T}$), stopping the growth of cost shares of terminals in the affected cluster, while their shares in the former run may still continue to grow.

For a thorough discussion of the cost sharing scheme of Jain and Vazirani and its monotonicity properties, the reader is refered to [13].

5.2 Uncapacitated Facility Location

Theorem 4. *The Uncapacitated Facility Location (UFL) problem admits a* 3*-approximation algorithm w.r.t. a* 2*-c-strict* ξ*. This* ξ *is also cross-monotone.*

Proof. We use a slight variant of the cost sharing function ξ defined by Pál and Tardos [19]. We shall refer heavily to our paper [8], where we proved that the cost sharing function ξ satisfies $(3 + \sqrt{6})$-strictness.

First, let us define the cost shares. Let F be a set of facilities, and $S \subseteq U$ a set of clients. For each facility $p \in F$, [19] defines the *opening time* $t_p(S)$ of a facility p to be the radius r of the smallest ball $B(p, r)$ centered at p such that if each client in the ball is charged r, this would be enough to pay for the opening cost of p as well as connecting each client inside the ball to p.

$$t_p(S) = \min\{r \mid \sum_{j \in B(p,r)} r - c(j, p) \geq f_p\}$$

In [19], the fee charged to a client j for connecting to a facility p was defined as $\alpha_{jp} = \max(t_p(S), c(jp))$, and the cost share of j to be the minimum fee it could pay to get connected: $\xi(S, j) = \min_{p \in F} \alpha_{jp}$. The papers [17, 19] gave an algorithm $\mathcal{A}$ that is a 3-approximation w.r.t. ξ.

To obtain a better c-strictness, we slightly modify the cost sharing function by introducing a special rule for zero-cost facilities. Such facilities arise in later stages of the multistage problem: recall that we set the opening cost of a facility opened in an earlier stage to zero. We set the connection fee of a client j to a zero-cost facility to be one third of their distance; that is, we define $\alpha'_{jp} = c(j, p)/3$ if $f_p = 0$ and $\alpha'_{jp} = \alpha_{jp}$ otherwise. We define the new cost shares to be $\xi'(S, j) = \min_p \alpha'_{jp}$. It is not difficult to verify that the algorithm $\mathcal{A}$ from [19] is still a 3-approximation algorithm w.r.t. ξ'.

We need to consider executions of the algorithm $\mathcal{A}$ on two instances: the instance with original facility costs f_p, and the instance with costs $f'_p = 0$ if p was opened by the algorithm $\mathcal{A}(f, S)$, and $f'_p = f_p$ otherwise. We show that for every client j, its cost share in the latter instance is at most twice its cost share in the former, thus proving 2-c-strictness of ξ'.

Consider a client $j \in T$ whose cost share is $\xi(f, S \cup T, j)$ in the algorithm $\mathcal{A}(F, S \cup T)$. By the properties of the algorithm $\mathcal{A}$, there is a *primary facility* $p = p(j)$ for client j so that either $\xi(f, S \cup T, j) = \max(t_p(S \cup T), c(j, p))$ and $f_p > 0$, or $\xi(f, S \cup T, j) = c(j, p)/3$ and $f_p = 0$. In the latter case we have $f'_p = 0$ as well, and hence $\xi(f', T, j) \le c(j, p)/3 \le \xi(f, S \cup T, j)$. Thus from now on we can focus on the case $f_p > 0$.

A facility p is T-heavy, if $|B(p, t_p(S \cup T)) \cap T| \ge |B(p, t_p(S \cup T)) \cap S|$. If p is T-heavy, then we argue (along the lines of Claim 5.2 from [8]) that $t_p(T) \le \frac{1}{b} t_p(S \cup T)$. On the other hand, if p is T-light, we have that $t_p(S) \le \frac{1}{1-b} t_p(S \cup T)$, and it is a known fact that in the execution $A(f, S)$, there must be an open facility q such that $c(p, q) \le 2t_p(S)$.

Hence, the cost share $\xi(f', T, j) \le \max(t_p(T), c(j, p)) \le 2\xi(f, S \cup T, j)$ if p is T-heavy and $\xi(f', T, j) \le c(j, q)/3 \le \xi(f, S \cup T, j)$ if p is T-light.

5.3 Vertex Cover

Theorem 5. *The Vertex Cover problem has* 2*-approximation algorithm w.r.t. an associated* 2*-strict (and hence* 2*-c-strict) cost-sharing function* ξ*.*

Proof. This is a minor strengthening of Theorem 5.5 of [8]. We shall refer heavily to the notation and symbols from Section 5.2 of [8], which the reader is urged to use as a reference.

According to Equation (5.7) of [8], we have that

$$\|p^1_S + p^1_T - p^2\|_1 \le \|p^1_T\|_1, \tag{18}$$

where $\|\cdot\|_1$ denotes the l_1 norm.

Now, consider the run R_3 of the algorithm $\mathcal{A}$ on the set of edges T with vertex prices $c^3 = c - p^2$. We claim that the vector $\hat{p}$ defined by $\hat{p}(v) = \max(0, p^1(v) - p^2(v))$ is a feasible solution to the relaxed Vertex Cover instance (c^3, T). Indeed, for any vertex $v \in V$, we have $c_v - p^2(v) - \hat{p}(v) \le c_v - p^1(v)$, and since p^1 is feasible for the instance $(c, S \cup T)$ (and hence also for (c, T)), it follows that $\hat{p}$ is a feasible solution for the instance (c^3, T).

Note that $\mathsf{cost}(\hat{p}) = \|\hat{p}\|_1$ which is no more than $\|p^1_T\|_1 = 2\xi(S \cup T, T)$ by Equation (18), hence the theorem follows.

6 Concluding Remarks

While the algorithms described in this paper can provide approximation algorithms with performance guarantee linear in the number of stages for some problems (Stochastic Steiner Tree), this requires 1-c-strict cost-shares, which may not always exist. The question of which k-stage stochastic optimization problems can be approximated within ratios linear in k and which require exponential dependence on k is an intriguing one. We also note that the running time of our algorithm is exponential in k due to the recursive sampling procedure; we leave open the question whether a sub-exponential collection of samples can be used to construct the partial solutions in earlier stages while still resulting in an algorithm with a provably good approximation ratio.

References

1. Ajit Agrawal, Philip Klein, and R. Ravi. When trees collide: an approximation algorithm for the generalized steiner problem on networks. *SIAM J. Comput.*, 24(3):440–456, 1995. (Preliminary version in *23rd STOC*, 1991).
2. E. M. L. Beale. On minimizing a convex function subject to linear inequalities. *J. Roy. Statist. Soc. Ser. B.*, 17:173–184; discussion, 194–203, 1955. (Symposium on linear programming.).
3. John R. Birge and François Louveaux. *Introduction to stochastic programming.* Springer Series in Operations Research. Springer-Verlag, New York, 1997.
4. George B. Dantzig. Linear programming under uncertainty. *Management Sci.*, 1:197–206, 1955.
5. Kedar Dhamdhere, R. Ravi, and Mohit Singh. On two-stage stochastic minimum spanning trees. In *Proceedings of the 11th Integer Programming and Combinatorial Optimization Conference*, pages 321–334, 2005.
6. Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995. (Preliminary version in *5th SODA*, 1992).
7. Anupam Gupta, Amit Kumar, Martin Pál, and Tim Roughgarden. Approximations via cost-sharing. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 606–615, 2003.
8. Anupam Gupta, Martin Pál, R. Ravi, and Amitabh Sinha. Boosted sampling: Approximation algorithms for stochastic optimization problems. In *Proceedings of the 36th ACM Symposium on the Theory of Computing (STOC)*, pages 417–426, 2004.
9. Anupam Gupta, R. Ravi, and Amitabh Sinha. An edge in time saves nine: LP rounding approximation algorithms for stochastic network design. In *Proceedings of the 45th Symposium on the Foundations of Computer Science (FOCS)*, pages 218–227, 2004.
10. A. Hayrapetyan, C. Swamy, and E. Tardos. Network design for information networks. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 933–942, 2005.
11. Nicole Immorlica, David Karger, Maria Minkoff, and Vahab 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 691–700, 2004.
12. Nicole Immorlica, Mohammad Mahdian, and Vahab Mirrokni. Limitations of cross-monotonic cost-sharing schemes. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2005. to appear.

13. Kamal Jain and Vijay Vazirani. Applications of approximation algorithms to cooperative games. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing (STOC)*, pages 364–372, 2001.
14. Kamal Jain and Vijay V. Vazirani. Equitable cost allocations via primal-dual-type algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 313–321. ACM Press, 2002.
15. Peter Kall and Stein W. Wallace. *Stochastic programming.* Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons Ltd., Chichester, 1994.
16. Willem K. Klein Haneveld and Maarten H. van der Vlerk. *Stochastic Programming.* Department of Econometrics and OR, University of Groningen, Netherlands, 2003.
17. Ramgopal R. Mettu and C. Greg Plaxton. The online median problem. In *41st Annual Symposium on Foundations of Computer Science (Redondo Beach, CA, 2000)*, pages 339–348. IEEE Comput. Soc. Press, Los Alamitos, CA, 2000.
18. Hervé Moulin and Scott Shenker. Strategyproof sharing of submodular costs: budget balance versus efficiency. *Econom. Theory*, 18(3):511–533, 2001.
19. Martin Pál and Éva Tardos. Group strategyproof mechanisms via primal-dual algorithms. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 584–593, 2003.
20. R. Ravi and Amitabh Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. In *Proceedings of the 10th Integer Programming and Combinatorial Optimization Conference*, pages 101–115, 2004.
21. R. Schultz, L. Stougie, and M. H. van der Vlerk. Two-stage stochastic integer programming: a survey. *Statist. Neerlandica*, 50(3):404–416, 1996.
22. David Shmoys and Chaitanya Swamy. Stochastic optimization is (almost) as easy as deterministic optimization. In *Proceedings of the 45th Symposium on the Foundations of Computer Science (FOCS)*, pages 228–237, 2004.
23. David Shmoys and Chaitanya Swamy. Sampling-based approximation algorithms for multi-stage stochastic optimization. Manuscript, 2005.
24. H. P. Young. Cost allocation. In R. J. Aumann and S. Hart, editors, *Handbook of Game Theory*, volume 2, chapter 34, pages 1193–1235. North-Holland, 1994.