Boosted Sampling: Approximation Algorithms for Stochastic Optimization

Anupam Gupta* Martin Pál † R. Ravi‡ Amitabh Sinha‡

ABSTRACT

Several combinatorial optimization problems choose elements to minimize the total cost of constructing a feasible solution that satisfies requirements of clients. In the STEINER TREE problem, for example, edges must be chosen to connect terminals (clients); in VERTEX COVER, vertices must be chosen to cover edges (clients); in FACILITY LOCATION, facilities must be chosen and demand vertices (clients) connected to these chosen facilities.

We consider a stochastic version of such a problem where the solution is constructed in two stages: Before the actual requirements materialize, we can choose elements in a *first* stage. The actual requirements are then revealed, drawn from a pre-specified probability distribution π; thereupon, some more elements may be chosen to obtain a feasible solution for the actual requirements. However, in this *second* (recourse) stage, choosing an element is costlier by a factor of $\sigma > 1$. The goal is to minimize the first stage cost plus the expected second stage cost.

We give a general yet simple technique to adapt approximation algorithms for several deterministic problems to their stochastic versions via the following method.

- *First stage:* Draw σ independent sets of clients from the distribution π and apply the approximation algorithm to construct a feasible solution for the union of these sets.
- *Second stage:* Since the actual requirements have now been revealed, augment the first-stage solution to be feasible for these requirements.

We use this framework to derive constant factor approximations for stochastic versions of VERTEX COVER, STEINER TREE and UNCAPACITATED FACILITY LOCATION for arbitrary distributions π in one fell swoop. For special (product) distributions, we obtain additional and improved results. Our techniques adapt and use the notion of strict cost-shares introduced in [5].

*Dept. of Computer Science, Carnegie Mellon University, Pittsburgh PA 15213. Email: anupamg@cs.cmu.edu

†Dept. of Computer Science, Cornell University, Ithaca NY 14853. Supported by ONR grant N00014-98-1-0589. Email: mpal@cs.cornell.edu.

‡GSIA, Carnegie Mellon University, Pittsburgh PA 15213. Supported in part by NSF grant CCR-0105548 and ITR grant CCR-0122581 (The ALADDIN project). Email: {ravi, asinha}@andrew.cmu.edu

STOC'04, June 13–15, 2004, Chicago, Illinois, USA.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Algorithms, Theory

Keywords

stochastic optimization, boosted sampling, cost sharing, approximation algorithms

1. INTRODUCTION

Infrastructure planning problems involve making decisions under uncertainty about future requirements; while more effective decisions can be made after the actual set of clients have materialized, the decision-making costs are inflated if deferred until then. The following simple two-stage model captures this tradeoff effectively. Future requirements are uncertain, but are assumed to be drawn from a known probability distribution (e.g., from demand forecasts, industry outlooks). In light of this information, an *anticipatory* part of the solution may be constructed in a first-stage at the current costs. Subsequently, the requirements facing the planner materialize in the form of a client set (drawn from the distribution), and the first-stage solution must be *augmented* to satisfy the revealed requirements. The elements chosen in this second stage are costlier than when chosen earlier, reflecting the need for careful (first-stage) planning. Given the uncertainty of the requirements, the traditional minimum-cost goal may be adapted to minimize the total *expected* cost of the solution.

As an example, consider the STOCHASTIC STEINER TREE problem that specifies an inflation parameter σ and a probability distribution π on the set of terminal nodes (which are clients) that have to be connected to the root in a given rooted discrete metric space. A subset of edges E_0 may be bought by paying the original lengths in the first stage. Once the actual set of terminals S is revealed, we must then buy the recourse edges E_S at σ times their lengths so that S is connected to the root by edges in $E_0 \cup E_S$. The objective is to minimize $c(E_0) + \mathbf{E}[\sigma\, c(E_S)]$. Here the expectation is over π, the randomness in the set of terminals revealed.

The framework is that of *two-stage stochastic optimization with recourse* [13, 12, 23] which may be paraphrased as "On Monday, we only know the input distribution on the clients, and we can buy some resources. On Tuesday, the client set is now completely specified, but things are now more expensive (in our case, by a factor σ); we must buy any additional resources needed to get a feasible solution to the instance."

Following this framework, in STOCHASTIC VERTEX COVER, the clients are edges to be covered, and we are given a probability distribution over sets of edges that will arrive; vertices become σ times more expensive after these edges are revealed. The STOCHASTIC FACILITY LOCATION problem on a metric space containing clients and facilities with opening costs defines a probability distribution over the set of clients that will require connection to open facilities. Opening facilities becomes σ times costlier in the second stage. The objective, in addition to expected cost of opening facilities, also includes expected connection costs of the revealed clients to their closest open facilities.

Our Results. In this paper, we give a simple yet general framework to translate approximation algorithms for deterministic optimization problems[1] into approximation algorithms for corresponding stochastic versions with second-stage inflation parameter σ. Given an α-approximation algorithm for the classical problem, one can use it in the following framework:

1. *Boosted Sampling:* Sample σ times from the distribution π to get sets of clients $D_1, \ldots, D_\sigma$.
2. *Building First Stage Solution:* Build an α-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.

Note that *we do not need to know the distribution π explicitly*; it could be a black-box from which we can draw samples. (In practice, these samples could be drawn from market predictions, or from Monte-Carlo simulations.) Thus we can sidestep the often-lethal problem of handling an exponential number of scenarios.

Informal Main Result 1.1 If the α-approximation algorithm $\mathcal{A}$ satisfies some technical properties (the problem is *sub-additive*, and $\mathcal{A}$ admits a β-strict *cost-sharing* function[2]), then the above framework yields an $\alpha + \beta$ approximation for the stochastic version of the problem.

The framework is laid out in Section 2 and the formal result is Theorem 3.1. Using this framework, we show that stochastic variants of STEINER TREE, FACILITY LOCATION, and VERTEX COVER have constant-factor approximation algorithms; the details are in Sections 4-5. The approximation ratios α and strictness of the corresponding cost-shares β, and the resulting guarantees for the stochastic variants are summarized in Figure 1.1.

We also consider the special case of *independent decisions*; in this, each client j has a probability π_j of arrival *independent* of other clients, and the probability $\pi(S)$ of the set S materializing is given by $\prod_{j \in S} \pi_j \prod_{j \notin S}(1 - \pi_j)$. For this model, we can also give a 8-approximation for the stochastic version of the STEINER FOREST problem and improve the approximation ratios of the corresponding versions of VERTEX COVER and FACILITY LOCATION to 3 and 6 respectively.

While a natural approach to utilizing an approximation algorithm for a deterministic problem is to set the client requirements at their expected value according to π, we note that this approach cannot yield bounded approximation ratios even in simple examples. Rather, using the full power of sampling in building the first stage solution gives a provably good solution as we demonstrate.

Related Work. The study of stochastic optimization [2, 10] dates back to the work of Dantzig [3] and Beale [1] in 1955; these papers defined the notions of *stochastic linear programming*. Stochastic LPs have been very widely studied since, and several gradient-based and decomposition-based approaches are known for two-stage 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 [23, 13] for details.

While stochastic scheduling problems have been studied extensively in the literature [19], the papers often try to identify cases where some standard scheduling policies yield optimal results, or the results hold for some special distributions (e.g., where job sizes are exponentially distributed), or focus on problems for which the deterministic versions are polynomial-time solvable. There are, of course, exceptions; see, e.g., [14, 4, 16, 24].

Very recently, there has been a surge of interest in stochastic versions of NP-hard problems, with papers on the topic by Ravi and Sinha [21], and independently, by Immorlica et al. [7]. Both these works look at versions of our model with some restrictions on the distribution π, while we consider *arbitrary distributions* π. In particular, they consider the following cases.

- the *scenario model*, where the distribution π has its support on a family $\mathcal{F}$ of possible subsets explicitly given as part of the input (and hence the algorithms are allowed to take time $\mathrm{poly}(|\mathcal{F}|, n)$), and
- the *independent decisions* model, where each element j has an associated probability π_j, and the probability of a set $\pi(S) = \prod_{j \in S} \pi_j \prod_{j \notin S}(1 - \pi_j)$. (I.e., the sets are chosen by flipping a coin independently for each element.)

Since our algorithms in Sections 4 and 5 work for *arbitrary distributions*, our theorems hold in both the above models as well. In particular, our 3.55- and 3-approximations for stochastic STEINER TREE and VERTEX COVER in the independent model improve upon the $O(\log n)$- and 6.3-approximations in [7] respectively.

One can define (as in [21]) other stochastic variants of the problems we define here: e.g., one can imagine that there are multiple inflation parameters, and that instead of all things getting dearer by σ, different parts of the problem change in different ways. This work leaves open the question of whether our framework can be extended to handle such multiple-parameter stochastic problems.

Stochastic Steiner Tree appears similar to the *maybecast* problem of Karger and Minkoff [11]; however, the latter is a single-stage optimization problem. Finally, though some of our techniques, including strict cost-shares come from the work of [6, 5], the problems considered there are deterministic optimization problems.

2. MODEL AND NOTATION

We define an abstract combinatorial optimization problem that we will adapt to a stochastic setting. To define Π, a combinatorial optimization problem, let U be the universe of *clients* (or requirements), and let X be the set of *elements* that we can purchase. For any $F \subseteq X$, let $c(F)$ denote the *cost* of the element set F. Given

[1]While the approximation algorithm solves the deterministic counterpart of the problem as opposed to the *stochastic* one, there is no requirement for this algorithm itself to be deterministic, e.g., randomized approximation algorithms can just as well be used in our framework.

[2]These terms will soon be defined, in Definitions 2.1 and 2.2 respectively.

Problem	Non-Stoc. Approximation Ratio α	Strictness β	Stochastic Approximation	
			General Distrib.	Indep. decisions
Steiner Tree	1.55 (Robins & Zelikovsky)	2	3.55	3.55
Vertex Cover	2 (Primal-dual)	6	8	3
Facility Location	3 (Mettu-Plaxton)	5.45	8.45	6
Steiner Network	4 (Gupta et al.)	$4^{\ddagger}$	-	8

Figure 1.1: ***Result Summary*** **.** ‡***Weaker strictness that gives approximations only in the independent decisions model.***

a set of clients $S \subseteq U$, let $\mathsf{Sols}(S) \subseteq 2^X$ be the set of *feasible solutions* for the client set S. The *deterministic version* $\mathsf{Det}(\Pi)$ specifies a fixed subset of clients $S \subseteq U$, and the objective is to return $F \in \mathsf{Sols}(S)$ with least cost. We denote by $\mathsf{OPT}(S)$ a solution in $\mathsf{Sols}(S)$ of minimum cost.

Definition 2.1 *We require that all the problems Π we consider satisfy* sub-additivity. *This property states that if S and S' are two sets of clients with solutions $F \in \mathsf{Sols}(S)$ and $F' \in \mathsf{Sols}(S')$, then we must have that (i) $S \cup S'$ is a legal set of clients for Π, and (ii) $F \cup F' \in \mathsf{Sols}(S \cup S')$.*

For example, consider the STEINER TREE problem on a graph $G = (V, E)$; the clients $U = V$ are the set of vertices, and the elements X are edges E of the graph. To ensure sub-additivity, we require a *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 $S \subseteq V$ of *terminals*, the solutions are $\mathsf{Sols}(S) = \{T \mid T \text{ connects all vertices of } S \cup \{r\}\}$.

Given any problem Π, we can define a variant adapted from the framework of *two-stage stochastic programming with recourse* as follows.

- There will be *two stages* of purchasing. Let $\sigma \geq 1$ be a given *inflation parameter*; every element $x \in X$ costs c_x in the first stage but costs $\sigma\, c_x$ in the second.
- In the first stage, the algorithm is only given access to an oracle that can draw from the probability distribution $\pi : 2^U \to [0, 1]$ in time $poly(|U|)$. It can then construct a *first-stage solution* by buying a set of elements F_0 at cost $c(F_0)$.
- In the second stage, one set $S \subseteq U$ of clients is *realized* according to the distribution π; i.e., the probability that S is realized is $\pi(S)$. We assume that this set S is conditionally independent of any of our actions in the first stage. Now the *second-stage solution* (also called the *recourse*) consists of a set of elements F_S purchased at the inflated cost $\sigma\, c(F_S)$. The union of the sets $F_0 \cup F_S$ must lie in $\mathsf{Sols}(S)$, i.e., the first and second stage solutions taken together give a feasible solution for the realized set of clients.

The objective of an algorithm for the stochastic problem $\mathsf{Stoc}(\Pi)$ is to select a set F_0, and then, given a set S from the distribution π, to select F_S to minimize the expected cost of the solution:

$$c(F_0) + \textstyle\sum_{S \subseteq U} \pi(S) \sigma c(F_S). \tag{2.1}$$

Hence the stochastic version of the Steiner Tree problem allows us to purchase some edges F_0 in the first stage, and once the set of realized clients $S \subseteq V$ is revealed, to buy some more edges F_S so that $F_0 \cup F_S$ contains a tree spanning S.

2.1 Cost sharing functions

We now define the *cost-shares* that are used crucially to bound the performance of our approximation algorithms. Loosely, a cost-sharing function is one that divides the cost of a solution $F \in \mathsf{Sols}(S)$ among the client set S. Cost-sharing functions have long been used in the context of game-theory [8, 9, 17, 18, 25]. We will use (a slight variant of) a cost-sharing function defined recently by Gupta et al. [5]: in contrast to previous cost-sharing mechanisms, this cost-sharing function is defined relative to a fixed (approximation) algorithm for the problem Π.

Definition 2.2 Given an α-approximation algorithm $\mathcal{A}$ for the deterministic problem Π, the function $\xi : 2^U \times U \to \mathbb{R}_{\geq 0}$ is a *β-strict cost-sharing function* if the following properties hold:

P1. For a set $S \subseteq U$, $\xi(S, j) > 0$ only for $j \in S$.

P2. For a set $S \subseteq U$, $\sum_{j \in S} \xi(S, j) \leq c(\mathsf{OPT}(S))$. **(fairness)**

P3. If $S' = S \uplus T$, then $\sum_{j \in T} \xi(S', j) \geq (1/\beta) \times$ cost of *augmenting* the solution $\mathcal{A}(S)$ to a solution in $\mathsf{Sols}(S')$. **(strictness)**

Formally, we require the existence of a polynomial time algorithm $\mathsf{Aug}_{\mathcal{A}}$ which can augment $\mathcal{A}(S)$ to a solution in $\mathsf{Sols}(S')$ at cost at most $\beta \sum_{j \in T} \xi(S', j)$.

Define the function $\xi(S, A)$ as the sum $\sum_{j \in A} \xi(S, j)$.

Remark 2.3 Note that one possible algorithm $\mathsf{Aug}_{\mathcal{A}}$ is obtained by just zeroing out the costs of elements already picked in $\mathcal{A}(S)$, and running $\mathcal{A}$ again. In all cases we consider, there are natural algorithms $\mathcal{A}$ for which this $\mathsf{Aug}_{\mathcal{A}}$ ensures strictness; however, in this paper, we choose algorithms $\mathcal{A}$ and $\mathsf{Aug}_{\mathcal{A}}$ that complement each other and give better approximation ratios.

3. APPROXIMATION VIA COST SHARING

In this section, we give a general technique for converting an approximation algorithm $\mathcal{A}$ for a deterministic problem $\mathsf{Det}(\Pi)$ into an approximation algorithm for the stochastic version $\mathsf{Stoc}(\Pi)$, provided the problem Π satisfies sub-additivity, and there is a cost-sharing function ξ that is strict w.r.t. $\mathcal{A}$.

3.1 Algorithm: Boosted Sampling

Given an instance of a stochastic problem $\mathsf{Stoc}(\Pi)$, the goal of the first stage is to buy the elements that will be useful for the unknown client set realized in the second stage. Since our algorithm is not clairvoyant and hence cannot see the future, the next best thing it can do is to sample from the distribution π, and use the samples as an indication of what the future holds. This simple idea is the basis of our method.

A naïve attempt would be to sample once from the distribution and use the set obtained as our prediction for the future: however, this is not aggressive enough in that it ignores the fact that the future is more expensive by a factor of σ. In fact, as $\sigma \to \infty$, the optimal solution would be to assume that every client in U will be realized and must be accounted for in the first stage itself. Motivated by these concerns, the algorithm for the problem $\mathsf{Stoc}(\Pi)$ is stated below, in terms of the α-approximation algorithm $\mathcal{A}$ for Π, which comes equipped with a β-strict cost-sharing function (and hence an associated augmentation algorithm $\mathsf{Aug}_{\mathcal{A}}$).

Our algorithm requires a number of samples that is linear in σ. Indeed, if all we have is a sampling access to the distribution π, it is not hard to see that $\Omega(\sigma)$ samples are needed. In many practical situations, this may not be a concern, unless the value of σ is exorbitant. In some cases (e.g., see Section 6), additional information about the distribution π can be used to make the running time independent of σ. Let $\lfloor\sigma\rfloor$ denote σ rounded down to the nearest integer.

Algorithm Boost-and-Sample(Π)

1. Draw $\lfloor\sigma\rfloor$ independent samples $D_1, D_2, \ldots, D_{\lfloor\sigma\rfloor}$ of the realized clients by sampling from the distribution π. Let $D = \cup_i D_i$.
2. Using the algorithm $\mathcal{A}$, construct an α-approximate *first-stage solution* $F_0 \in \mathsf{Sols}(D)$.
3. If the client set S is realized in the second stage, use the augmenting algorithm $\mathsf{Aug}_{\mathcal{A}}$ from (P3) to compute F_S such that $F_0 \cup F_S \in \mathsf{Sols}(S)$.

The following theorem is the main result of the paper:

Theorem 3.1 *Consider a combinatorial optimization problem* Π *that is sub-additive, and let* $\mathcal{A}$ *be an* α*-approximation algorithm for its deterministic version* $\mathsf{Det}(\Pi)$ *that admits a* β*-strict cost sharing function. Then* Boost-and-Sample(Π) *is an* $(\alpha+\beta)$*-approximation algorithm for* $\mathsf{Stoc}(\Pi)$.

We will prove Theorem 3.1 in the rest of the section. In the next section, we illustrate an application of this theorem to obtain an approximation algorithm for the STOCHASTIC STEINER TREE. In subsequent sections, we go on to consider several other problems, and show that their approximation algorithms and attendant cost sharing functions provide approximation algorithms for the corresponding stochastic versions.

Proof of Theorem 3.1: To simplify the notation, we shall assume that σ is an integer, i.e. $\sigma = \lfloor\sigma\rfloor$. It is straightforward to verify that the proof can be carried out for arbitrary real values of σ.

We will bound the expected costs of our first and second-stage solutions separately. Let F_0^* be the first-stage component of the optimal solution, and F_S^* be the second-stage component if the set of realized clients is S. Hence

$$\text{the optimal cost } Z^* = c(F_0^*) + \textstyle\sum_S \pi(S)\,\sigma\, c(F_S^*). \tag{3.2}$$

Let us denote $Z_0^* = c(F_0^*)$ and $Z_r^* = \sum_S \pi(S)\,\sigma\, c(F_S^*)$.

First stage: We claim that there is an element $\widehat{F}_1 \in \mathsf{Sols}(D)$ such that $\mathbf{E}[c(\widehat{F}_1)] \le Z^*$. Indeed, define $\widehat{F}_1 = F_0^* \cup F_{D_1}^* \cup F_{D_2}^* \cup \ldots \cup F_{D_\sigma}^*$. The fact that $\widehat{F}_1 \in \mathsf{Sols}(D)$ follows from sub-additivity of the problem Π. Therefore,

$$\begin{aligned}\mathbf{E}_D[c(\widehat{F}_1)] &\le c(F_0^*) + \mathbf{E}_D[\textstyle\sum_{i=1}^{\sigma} c(F_{D_i}^*)]\\ &= c(F_0^*) + \textstyle\sum_{i=1}^{\sigma} \mathbf{E}_{D_i}[c(F_{D_i}^*)]\\ &= c(F_0^*) + \sigma \textstyle\sum_S \pi(S)\, c(F_S^*) \quad = Z^*,\end{aligned}$$

the penultimate equality following from the fact that each of the D_i's are chosen from the probability distribution π. Since we have an α-approximation algorithm for $\mathsf{Det}(\Pi)$, our solution F_0 satisfies $\mathbf{E}[c(F_0)] \le \alpha c(\widehat{F}_1)$, which in turn is at most $\alpha\, Z^*$, bounding our first stage costs.

Second stage: Let S be the set of realized clients, and let F_S be the result of our algorithm $\mathsf{Aug}_{\mathcal{A}}$ such that $F_0 \cup F_S \in \mathsf{Sols}(S)$. We need to bound our expected second stage cost, which is $\sigma\,\mathbf{E}[c(F_S)]$. The β-strictness of the cost sharing function ξ implies that $c(F_S) \le \beta\,\xi(D\cup S, S)$. In fact, we even have $c(F_S) \le \beta\,\xi(D\cup S, S\setminus D)$.

Consider the following alternate probabilistic process to generate the sets D_i and the set S: Draw $\sigma+1$ independent samples $\widehat{D}_1, \widehat{D}_2, \ldots, \widehat{D}_{\sigma+1}$ from the distribution π. Now choose a random value K uniformly from $\{1, 2, \ldots, \sigma+1\}$, and set $S = \widehat{D}_K$ and $D = \cup_{i\ne K}\widehat{D}_i$. This process is identically distributed to the original process, since we are picking the sets independently. Let $\widehat{\mathbb{D}}$ be the union of *all* the $\widehat{D}_i$'s, and let $\widehat{\mathbb{D}}_{-i}$ be the union $\cup_{l\ne i}\widehat{D}_l$ of all the sets except $\widehat{D}_i$.

Since the cost sharing function is fair (Property P2), we have

$$\textstyle\sum_{i=1}^{\sigma+1} \xi(\widehat{\mathbb{D}}, \widehat{D}_i \setminus \widehat{\mathbb{D}}_{-i}) \le c(\mathsf{OPT}(\widehat{\mathbb{D}})).$$

By our random choice of K, we get

$$\mathbf{E}_K[\xi(\widehat{\mathbb{D}}, \widehat{D}_K \setminus \widehat{\mathbb{D}}_{-K})] \le \tfrac{1}{\sigma+1} c(\mathsf{OPT}(\widehat{\mathbb{D}}))$$

Since the alternate process is probabilistically identical to the one we used to pick D and S,

$$\begin{aligned}\mathbf{E}_{D,S}[\xi(D\cup S, S\setminus D)] &= \mathbf{E}_{\widehat{\mathbb{D}},K}[\xi(\widehat{\mathbb{D}}, \widehat{D}_K \setminus \widehat{\mathbb{D}}_{-K})]\\ &\le \tfrac{1}{\sigma+1}\,\mathbf{E}_{\widehat{\mathbb{D}}}[c(\mathsf{OPT}(\widehat{\mathbb{D}}))] \end{aligned} \tag{3.3}$$

To complete the argument, we now show that $\mathbf{E}_{\widehat{\mathbb{D}}}[c(\mathsf{OPT}(\widehat{\mathbb{D}}))] \le \frac{\sigma+1}{\sigma} Z^*$. To derive a feasible solution to $\widehat{\mathbb{D}}$, define $\widehat{F}_2 = F_0^* \cup F_{\widehat{D}_1}^* \cup F_{\widehat{D}_2}^* \cup \ldots \cup F_{\widehat{D}_{\sigma+1}}^*$. Again, the fact that $\widehat{F}_2 \in \mathsf{Sols}(\widehat{\mathbb{D}})$ follows from sub-additivity of Π. Thus we have

$$\begin{aligned}\mathbf{E}_{\widehat{\mathbb{D}}}[c(\mathsf{OPT}(\widehat{\mathbb{D}}))] &\le c(F_0^*) + \textstyle\sum_{i=1}^{\sigma+1} \mathbf{E}_{\widehat{D}_i}[c(F_{\widehat{D}_i}^*)]\\ &\le Z_0^* + (\sigma+1)\,\tfrac{Z_r^*}{\sigma}\\ &\le \tfrac{\sigma+1}{\sigma}\,(Z_0^* + Z_r^*) = \tfrac{\sigma+1}{\sigma}\, Z^*. \end{aligned} \tag{3.4}$$

Thus $\mathbf{E}_S[c(F_S)] \le \beta\,\mathbf{E}_{D,S}[\xi(D\cup S, S\setminus D)]$, which using (3.3) and (3.4), is bounded by $\frac{\beta}{\sigma} Z^*$. Finally, since our second stage cost is $\sigma\, c(F_S)$, our expected second stage cost is $\mathbf{E}_S[\sigma\, c(F_S)] \le \beta\, Z^*$.

Putting together the first and second stage costs gives the bound claimed in the theorem. ■

Remark 3.2 One might wonder whether tighter results can be obtained by choosing some number other than $\lfloor\sigma\rfloor$ for the number of samples used in constructing the first-stage solution. We believe that while small and problem-specific improvements might be possible, sampling $\lfloor\sigma\rfloor$ times is the best possible for Theorem 3.1. Sampling fewer than $\lfloor\sigma\rfloor$ times does not improve the bound on the first stage cost since the optimal solution might have a large first-stage component; it hurts the second-stage cost since the proportionate cost of the second-stage is now much larger. Sampling greater than σ times clearly hurts the first-stage cost, while offering no improvement in the second-stage cost guarantee (for example, when all scenarios require disjoint solutions).

4. STOCHASTIC STEINER TREES

In the classical deterministic STEINER TREE problem, we are given a set of vertices V, and the costs c_e on edges satisfy the triangle inequality. (This assumption is without loss of generality, since we can take the metric completion of the graph.) We assume there is a designated *root* vertex r. Given a set of terminals (i.e., the clients), the goal is to buy a set of edges (the elements) of minimum cost so that the terminals and the root r lie in a connected component. Note that the presence of the root ensures that the problem is sub-additive.

Now let us consider the problem Stoc(STEINER TREE): in the first stage, we can buy some edges F_0 at cost $\sum_{e \in F_0} c_e$. In the second stage, a set of terminals $S \subseteq V$ is realized with probability $\pi(S)$, after which we may buy some more edges to connect the terminals to the root; however, these edges must be bought at cost $\sigma\, c_e$ each.

Theorem 4.1 *There exists a* 2*-approximate algorithm* $\mathcal{A}$ *for* STEINER TREE*, along with a cost sharing function* ξ *that is* 2*-strict for* $\mathcal{A}$*.*

PROOF. The algorithm $\mathcal{A}$ is simply Prim's algorithm [20] for minimum spanning tree; given a set of terminals S, it ignores the vertices not in $S \cup \{r\}$, and builds an MST on $S \cup \{r\}$. It is well-known that the cost $c(\mathcal{A}(S))$ of any MST is within a factor of 2 of the cost of the optimal Steiner tree $\mathsf{OPT}(S)$.

Given an MST $\mathcal{A}(S)$ on the set of terminals S, let us imagine it to be rooted at r; for $j \in S$, set $\xi(S, j)$ to be *half* the cost of the edge connecting j to its parent in $\mathcal{A}(S)$, which we call j's *parental edge* in $\mathcal{A}(S)$. Clearly, if $j \notin S$, then $\xi(S, j) = 0$. By definition, $\sum_{j \in S} \xi(S, j) = \frac{1}{2}\, c(\mathcal{A}(S))$; since the MST is a 2-approximation to the Steiner tree problem, this implies that $\frac{1}{2}\, c(\mathcal{A}(S)) \le c(\mathsf{OPT}(S))$, and hence ξ is fair.

Finally, to prove the 2-strictness, consider a set $S' = S \uplus T$. The augmenting procedure $\mathsf{Aug}_{\mathcal{A}}$ basically zeroes out the edges of $\mathcal{A}(S)$ and runs Prim's algorithm; i.e., it takes the solution $\mathcal{A}(S)$, and for each $j \in T$, adds the parental edge of j in $\mathcal{A}(S \cup T)$. We claim this gives a solution in $\mathsf{Sols}(S \cup T)$. (Indeed, each vertex $j \in T$ whose parent in $\mathcal{A}(S \cup T)$ was in $S \cup \{r\}$ will now be connected to $\mathcal{A}(S)$, and hence to r; the general argument follows by a simple induction.) Since these edges cost $2 \times \sum_{j \in T} \xi(S \cup T, j)$, we have proved the theorem. $\square$

Note that the argument for strictness only required that each vertex in the solution $\mathcal{A}(S)$ was connected to the root, and hence the same cost shares are also 2-strict for *any* heuristic for Steiner Tree. Now, using the 1.55-approximation for STEINER TREE [22] and Theorem 3.1, we obtain the following improved theorem:

Theorem 4.2 *There exists a factor* 1.55 *approximation algorithm for* STEINER TREE*, along with a cost-sharing function* ξ *that is* 2*-strict for it. Hence, there is a* 3.55*-approximation algorithm for* Stoc(STEINER TREE).

This improves on the $O(\log n)$-approximation for the *independent decisions* version of Stoc(STEINER TREE) given by Immorlica et al. [7].

5. OTHER APPLICATIONS

This section will be devoted to looking at several other (deterministic) problems Π; for each problem, we will give an α-approximation algorithm $\mathcal{A}$ and its accompanying β-strict cost-share function. Our results have been summarized in Figure 1.1.

5.1 Facility Location

An instance of FACILITY LOCATION is given by a set of facilities F and a set of clients S. The distances c_{ij} between any pair of points i, j from $F \cup S$ form a metric. Each facility p has opening cost f_p; the goal is to open a subset of facilities F' to minimize the opening costs plus the sum of distances from each client to its closest open facility:

$$\textstyle\sum_{p \in F'} f_p + \sum_{j \in S} c(j, F').$$

The main result for Stoc(FACILITY LOCATION) is the following:

Theorem 5.1 *The cost sharing function* ξ *given by Pál and Tardos is* 5.45*-strict for the* 3*-approximation algorithm of Mettu and Plaxton. Hence, there is a* 8.45*-approximation algorithm for* Stoc(FACILITY LOCATION).

PROOF. Let us briefly review the algorithm of Mettu and Plaxton [15], and the cost sharing defined by Pál and Tardos [18]. Let S be a set of clients. For a facility p, let $B(p, \tau)$ be a ball with center p and radius τ. We define the *opening time* $t_p(S)$ of a facility p w.r.t. the set of clients S to be the unique radius τ such that

$$f_p = \textstyle\sum_{j \in B(p,\tau) \cap S} (\tau - c(j, p)). \tag{5.5}$$

Let the set $C_p(S) = \{j \in S \mid c(j,p) < t_p(S)\}$ be called the *contributing set* for p. Note that if we charge each client in C_p the amount $t_p(S)$, we exactly recover the facility cost of p plus the cost of assigning clients in C_p to p. We drop the set of clients from the notation, and say t_p instead of $t_p(S)$ when there is no danger of confusion. The cost shares of clients are then defined as

$$\xi(S, j) = \min_{p \in F} \{\ \max(t_p(S), c(j, p))\ \}. \tag{5.6}$$

Intuitively, the contribution of user j towards the facility p should be either t_p if $j \in C_p$, or the connection cost $c(j, p)$ if $j \notin C_p$. The client can (and does) choose to contribute only to the least demanding facility; the facility p for which this minimum is attained is called the *primary* facility of j (in the run on S). A facility p is said to be *well-funded* if $\xi(S, j) \ge t_p(S)/3$ for all $j \in C_p$.

The algorithm $\mathcal{A}$ we use is a slight modification of the algorithm of Mettu and Plaxton; given a set of clients S, $\mathcal{A}$ considers all the well-funded facilities p in order of increasing opening time $t_p(S)$. For each such (well-funded) facility p, the algorithm declares it *open* if there are no previously opened facilities within a radius $2\, t_p(S)$ around p.

For each open facility p, $\mathcal{A}$ assigns all clients in C_p to p. (By construction, the sets C_p for open facilities p are disjoint.) It then assigns each client not lying in any C_p to its closest open facility. The following facts can be derived from the arguments in [15] and [18]:

1. For each open facility p, the cost shares $\xi(S, C_p)$ of the clients in C_p pay $1/3$ of f_p plus their assignment cost. (See [18, Lemma 2.4] and the preceding discussion therein.)
2. For each facility p, there exists a *well-funded* facility q (possibly $p = q$) such that $c(p, q) \le 2\,(t_p - t_q)$. (Note that it must be that $t_q \le t_p$.)
3. For each facility p, there exists an *open* facility q within a distance of $2t_p$. Hence, if p is a primary facility for some client j, then $c(j, q) \le 3\xi(S, j)$.

To show strictness of ξ, we must specify the algorithm $\mathsf{Aug}_{\mathcal{A}}$ which augments a solution $\mathcal{A}(S)$ to cover a set of new clients T with $S \cap T = \emptyset$. In the following, let $C_p = C_p(S \cup T)$ denote the contributor set of a facility p in the run $\mathcal{A}(S \cup T)$. Similarly, when we say a facility p is well-funded, we mean that p is well-funded in the run $\mathcal{A}(S \cup T)$. A facility p is called T*-heavy* if $|C_p \cap T| \ge b|C_p|$ (where the parameter $b \in (0, 1)$ will be specified later), and is T*-light* otherwise. Note that a T-light facility must have $|C_p \cap S| \ge (1 - b)|C_p|$.

Claim 5.2 *If* p *is a* T*-light facility, then*

$$t_p(S) \le \tfrac{1}{1-b}\, t_p(S \cup T)\ -\ \tfrac{1}{|C_p \cap S|} \textstyle\sum_{j \in C_p \cap T} c(j, p).$$

PROOF. Consider the set $C_p = \{j \in S \cup T \mid c(j,p) < t_p\}$; by the definition (5.5),

$$f_p + \sum_{j \in C_p} c(j,p) = |C_p|\, t_p(S \cup T)$$

Since p is T-light, $|C_p \cap S| \geq (1-b)\,|C_p|$.

$$\begin{aligned} f_p + \sum_{j \in C_p \cap S} c(j,p) &= |C_p|\, t_p(S \cup T) - \sum_{j \in C_p \cap T} c(j,p) \\ &\leq |C_p \cap S|\, \frac{t_p(S \cup T)}{1-b} - \sum_{j \in C_p \cap T} c(j,p). \end{aligned}$$

which means facility p was already paid for at time $t_p(S \cup T)/(1-b) - \sum_{j \in C_p \cap T} c(j,p)/(|C_p \cap S|)$ in the run $\mathcal{A}(S)$, proving the claim. □

Augmentation procedure $\mathsf{Aug}_{\mathcal{A}}$**:** To augment $\mathcal{A}(S)$ to cover T as well, we pick a subset of well-funded T-heavy facilities to open greedily in a manner very similar to that in $\mathcal{A}(S \cup T)$: we consider all well-funded T-heavy facilities in order of increasing $t_p(S \cup T)$, and open a facility p if there is no facility q already open within a radius $2\, t_p(S \cup T)$ of p. (Note that q may have been opened in either $\mathcal{A}(S)$, or in the augmenting phase before p was considered.) We never open any T-light or non-well-funded facilities. At the end of this procedure, for a client $j \in C_p$ whose p is open, we assign j to p; else we assign j to the closest open facility.

Claim 5.3 *The augmentation cost for a set T is at most $(3+\sqrt{6}) \sum_{j \in T} \xi(S \cup T, j)$.*

PROOF. Firstly, consider any well-funded T-heavy facility p. Since p is T-heavy, the shares of clients in $C_p \cap T$ can pay for a $b/3$ fraction of the facility cost plus their own connection costs. Hence we must consider clients j whose primary facility p is either not well-funded or not T-heavy. We claim that in both cases there must be a facility close to p opened either by $\mathcal{A}(S)$, or in the augmenting phase. Note that by the properties of the algorithm $\mathcal{A}$, there is a well-funded facility q such that $t_q(S \cup T) \leq t_p(S \cup T)$ and $c(p,q) \leq 2\,(t_p(S \cup T) - t_q(S \cup T))$.

Now, if q is T-heavy, by the properties of our augmentation procedure, there must a facility r that was open in the augmentation step such that $c(q,r) \leq 2\, t_q(S \cup T)$. On the other hand, if q is T-light, we have that $t_q(S) \leq t_q(S \cup T)/(1-b)$ by Claim 5.2 above. Thus, in the run $\mathcal{A}(S)$, there must be an open facility r such that $c(q,r) \leq 2\, t_q(S) \leq (2/(1-b))\, t_q(S \cup T)$.

In both cases, the assignment cost of the client j is bounded by

$$\begin{aligned} c(j,r) &\leq c(j,p) + c(p,q) + c(q,r) \\ &\leq c(j,p) + 2(t_p(S \cup T) - t_q(S \cup T)) + \frac{2}{1-b}\, t_q(S \cup T) \\ &\leq c(j,p) + \frac{2}{1-b}\, t_p(S \cup T) \\ &\leq (1 + \frac{2}{1-b})\, \xi(S \cup T, j). \end{aligned}$$

To balance $3/b$ and $(1 + 2/(1-b))$, we can now pick $b = 3 - \sqrt{6}$ to get the desired result. □

Since $\beta = 3 + \sqrt{6} \leq 5.45$, this proves the theorem.

5.2 Vertex Cover

In the Vertex Cover problem, we are given a graph $G = (V, E)$ with costs c_v on vertices. The clients are the edges, and our goal is to choose a subset V' of the vertices so that each edge is *covered*; i.e., at least one of its adjacent vertices is chosen. In the stochastic version, we pay c_v for picking a vertex v in the first phase, and $\sigma\, c_v$ for picking it in the second phase. We will prove the following theorem:

Theorem 5.4 *There is a 8-approximation algorithm for* Stoc(VERTEX COVER).

Before we do this, let us define a version of the problem called *Relaxed Stochastic Vertex Cover*. In the relaxed version of the stochastic problem, we are allowed to make payments to a vertex in both stages, with $p^1(v)$ and $p^2(v)$ being payments made to v in the first and second stage respectively. A vertex v is chosen if and only if $p^1(v) + p^2(v)/\sigma \geq c_v$. Again, given a set of realized edges S, the set of chosen vertices must form a feasible vertex cover for S. The cost of our solution is just the sum of payments, i.e. $\sum_{v \in V} p^1(v) + p^2(v)$, and the goal is again to minimize the expected cost.

Note that by requiring $p^1(v) \in \{0, c_v\}$ and $p^2(v) \in \{0, \sigma c_v\}$, we get back to our usual stochastic framework, and hence the relaxed problem allows us to make partial commitments to vertices in the first stage. However, it turns out that we can convert any algorithm $\mathcal{A}$ for the relaxed problem into an algorithm $\mathcal{A}'$ for the unrelaxed version with the same expected cost. Indeed, if $p^1(v)$ is the amount of money placed on vertex v by $\mathcal{A}$ in the first stage, the algorithm $\mathcal{A}'$ picks the vertex v in its first stage with probability $\min\{1, p^1(v)/c_v\}$. In the second stage, $\mathcal{A}'$ selects the vertex v if v was selected by $\mathcal{A}$ (that is, $p_v^1 + p_v^2/\sigma \geq c_v$), and if $\mathcal{A}'$ has not already selected it in the first stage. By linearity of expectations, the expected cost incurred by $\mathcal{A}'$ in each phase is at most the cost incurred by $\mathcal{A}$ in that phase. Thus it suffices to give an algorithm and a cost sharing function for relaxed vertex cover.

The Algorithm $\mathcal{A}$: We use a standard primal-dual 2-approximation algorithm $\mathcal{A}$ for vertex cover. Let $S \subseteq E$ be the set of edges in the instance. For each edge e, we have a dual variable y_e, initially set to 0. We simultaneously raise all dual variables at a uniform rate. A vertex v becomes *tight* when the duals of edges adjacent to it can pay its cost, i.e. when $\sum_{e \in \delta(v)} y_e = c_v$. When a vertex v becomes tight, we *freeze* all edges adjacent to it, i.e., we stop raising their dual variables. We continue raising the dual variables of all unfrozen edges, until all edges become frozen.

Output: The algorithm places payments $p(v) = \sum_{e \in \delta_v} y_e$ on each vertex $v \in V$. Since each edge is adjacent to some tight vertex v, it has been paid c_v and hence bought outright; thus the solution is feasible for S.

The Cost Shares: Define $\xi(S, e) = y_e$; since each edge pays both its endpoints, it holds that $\sum_{v \in V} p(v) = 2 \sum_{e \in S} y_e$. Furthermore, $\sum_e \xi(S, e)$ is just the LP dual value, and hence at most $\mathsf{OPT}(S)$.

Clearly, the algorithm $\mathcal{A}$ is a 2-approximation for the vertex cover problem. To prove Theorem 5.4, it suffices to prove the strictness of ξ for $\mathcal{A}$.

Theorem 5.5 *The cost shares ξ defined above are 6-strict with respect to the algorithm $\mathcal{A}$.*

PROOF. Let S and T be two disjoint sets of edges. To augment the solution $\mathcal{A}(S)$ to handle T as well, the augmenting algorithm $\mathsf{Aug}_{\mathcal{A}}$ looks at the (relaxed) payment function $p : V \to \mathbb{R}_{\geq 0}$ for the set of edges S, and runs the algorithm $\mathcal{A}$ on the set of edges T

with the reduced costs $c'_v = c_v - p(v)$. To prove strictness, we need to compare this augmentation cost to the cost share $\xi(S \cup T, T)$. To this end, we shall compare several runs of $\mathcal{A}$ on different related inputs.

- **Run R_1:** This is the run of $\mathcal{A}$ with original costs c_v on the set $S \cup T$. Let y^1_e be the duals produced. Let us define payments $p^1_S(v) = \sum_{e \in \delta(v) \cap S} y_e$ and $p^1_T(v) = \sum_{e \in \delta(v) \cap T} y_e$ respectively.

  Note that $p^1 = p^1_{S \cup T} = p^1_S + p^1_T$ is exactly the payment function computed by $\mathcal{A}$. Furthermore, this is the run that computes the cost-shares $\xi(S \cup T, T)$.
- **Runs R_S and R_T:** The run R_S is the run of $\mathcal{A}$ on the set of edges S, but with costs $c^S = c - p^1_T$ (i.e., reduced by the payments of T in R_1). Similarly the run R_T is on the edges T, with reduced costs $c^T = c - p^1_S$.
- **Run R_2:** This is $\mathcal{A}$'s run on the edge set S, with original costs c, and hence corresponds to the actual run of the first stage. Let y^2 be the duals and p^2 the payments computed.
- **Run R_3:** This is on the edge set T, with reduced costs $c^3 = c - p^2$; hence, this corresponds to the augmentation step for T. Again, y^3 and p^3 are the duals and the payments.

By the definition of R_S, the freezing time of all edges $e \in S$ in the two runs R_S and R_1 is the same; hence the dual y^S is just the dual y^1 restricted to the set S, and $p^1_S = p^S$. Similarly, the dual y^T from the run R^T is identical to the dual y^1 restricted to T, and $p^1_T = p^T$. We claim that, to prove the theorem, it suffices to prove the following claim:

Claim 5.6 $\sum_{v \in V} p^3(v) \leq 3 \sum_{v \in V} p^1_T(v)$.

Before we prove this claim, let us see how it proves Theorem 5.5. Note that cost of the augmentation run R_3 is exactly $\sum_v p^3(v)$, while the cost shares

$$\xi(S \uplus T, T) = \sum_{e \in T} y_e = \frac{1}{2} \sum_{v \in V} p^1_T(v).$$

Hence $\xi(S \cup T, T)$ can defray at least one-sixth of the cost of the run R_3, proving the theorem. □

Proof of Claim 5.6: The proof relies on the following Lipschitz-type property of the algorithm $\mathcal{A}$: imagine the vertex costs to be vectors in $\mathbb{R}^{|V|}$, and suppose the costs change by an amount ϵ (in their L_1-distance), then we claim that the payments do not change by more than $2\,\epsilon$. Formally,

Claim 5.7 (Lipschitz continuity) *Consider two runs R and $\widehat{R}$ of $\mathcal{A}$ with the same edge set S on two different cost vectors c and $\widehat{c}$, and let p and $\widehat{p}$ be the two vectors of payments computed. If we define Δ so that $(p - \widehat{p}) = (c - \widehat{c}) + \Delta$, then $\|\Delta\|_1 \leq \|c - \widehat{c}\|_1$.*

The proof of the Lipschitz condition follows below; but let us use it to complete this proof. First, we use it to compare the runs R_S and R_2 (both being defined on the edge set S): define Δ_1 so that

$$p^2 - p^1_S = c - (c - p^1_T) + \Delta_1 = p^1_T + \Delta_1; \qquad (5.7)$$

then Claim 5.7 implies that $\|\Delta_1\|_1 \leq \|p^1_T\|_1$. The second application of Claim 5.7 is to the runs R_3 and R_T (both on the edge set T); it implies that if Δ_2 is such that

$$p^3 - p^1_T = (c - p^2) - (c - p^1_S) + \Delta_2 = p^1_S - p^2 + \Delta_2, \quad (5.8)$$

then $\|\Delta_2\|_1 \leq \|p^1_S - p^2\|_1$; this, by (5.7), is at most $\|p^1_T\|_1 + \|\Delta_1\|_1 \leq 2\|p^1_T\|_1$. Furthermore, plugging (5.7) into (5.8), we get

$$p^3 - p^1_T = -(p^1_T + \Delta_1) + \Delta_2.$$

Simplifying, this gives $\|p^3\|_1 = \|\Delta_1\|_1 + \|\Delta_2\|_1 \leq 3\|p^1_T\|_1$. ■

Proof of Claim 5.7: Consider the two runs R and $\widehat{R}$ of $\mathcal{A}$ on the two cost vectors c and $\widehat{c}$ being executed in parallel. Let $p_t(v)$ and $\widehat{p}_t(v)$ be the payments towards vertex v accumulated in the respective runs until time t. We claim that the quantity $\Phi(t) = \sum_{v \in V} |(c(v) - p_t(v)) - (\widehat{c}(v) - \widehat{p}_t(v))|$ never increases as a function of t. Since $\Phi(0) = \|c - \widehat{c}\|_1$ and $\Phi(\infty) = \|(p - \widehat{p}) - (c - \widehat{c})\|_1 = \|\Delta\|_1$, this will prove Claim 5.7.

Consider any edge $e = \{u, v\}$ at time t in both runs. If e is not frozen in either run, it causes both $p(u)$ and $\widehat{p}(u)$ to increase at unit rate; the same arguments hold for v. Since u is not tight in either run, $c(u) - p_t(u) > 0$ and $\widehat{c}(u) - \widehat{p}_t(u) > 0$, and edge e contributes to both terms equally; hence it is currently contributing at rate zero to the difference $(c(u) - p_t(u)) - (\widehat{c}(u) - \widehat{p}_t(u))$. If e is frozen in both runs, its current rate of contribution is zero as well.

Now suppose that e is frozen in only one of the runs; say, it is frozen in the run R but not in the run $\widehat{R}$ (the other case is symmetric). That means one of its endpoints must be tight in R; w.l.o.g., assume the tight vertex is u. Thus $c(u) - p_t(u) = 0$. In the run $\widehat{R}$, the contribution of e makes the term $\widehat{c}(u) - \widehat{p}_t(u) = |(c(u) - p_t(u)) - (\widehat{c}(u) - \widehat{p}_t(u))|$ *decrease* at unit rate. However, its contribution towards v, and hence towards the term $|c(v) - p_t(v) - (\widehat{c}(v) - \widehat{p}_t(v))|$ increases at a rate of at most 1 in the worst case. Hence, the quantity Φ never increases. ■

6. INDEPENDENT DECISIONS: STEINER FOREST & OTHER IMPROVEMENTS

The *independent decisions* model was defined in Section 1 as the model when each client $j \in U$ has a probability π_j of requiring service *independently* of all other clients. For this special case of our model, we show that a weaker version of strict cost-shares is sufficient to obtain algorithms for stochastic problems. This allows us to obtain approximations for some more problems (e.g., for STEINER NETWORK), and obtain stronger results for others (e.g., for VERTEX COVER and FACILITY LOCATION) in the independent decisions model.

Given a problem Π, we use $\mathsf{Ind}(\Pi)$ to denote the stochastic extension of Π in this independent decisions model. For this section, we will need the following weaker definition of strictness that holds only for additions of a *single* client.

Definition 6.1 Given an α-approximation algorithm $\mathcal{A}$ for the deterministic problem Π, the function $\xi : 2^U \times U \to \mathbb{R}_{\geq 0}$ is a *β-uni-strict cost-sharing function* if properties (P1), (P2) hold in conjunction with:

P3$'$. If $S' = S \uplus \{j\}$, then $\xi(S', j) \geq (1/\beta) \times$ cost of *augmenting* the solution $\mathcal{A}(S)$ to a solution in $\mathsf{Sols}(S')$. **(uni-strictness)**

Again, we need a *poly-time algorithm* $\mathsf{Aug}_{\mathcal{A}}$ that does the augmentation with cost at most $\beta\,\xi(S', j)$.

6.1 The (Even Simpler) Algorithm Ind-Boost

Let us define the (yet simpler) algorithm for the independent case:

Algorithm Ind-Boost(Π)

1. Choose a set D by picking each element $j \in U$ with probability $\sigma\,\pi_j$ independently.

2. Using the algorithm $\mathcal{A}$, construct an α-approximate solution $F_0 \in \mathsf{Sols}(D)$.
3. Let S be the set of clients realized in the second stage. For each client $j \in S$, use the augmentation algorithm $\mathsf{Aug}_{\mathcal{A}}$ of (P3′) to compute F_j such that $F_0 \cup F_j \in \mathsf{Sols}(D \cup \{j\})$. Output $F_S = \bigcup_{j \in S} F_j$ as the second stage solution. Note that by subadditivity, $F_0 \cup F_S \in \mathsf{Sols}(S)$.

Note that $\mathsf{Ind\text{-}Boost}(\Pi)$ can be implemented in polynomial time regardless of how large σ is. Here is a version of the main Theorem 3.1 for the independent decisions model. This version assumes only the weaker property of uni-strictness, hence it is useful when fully strict cost sharing is not known, or when uni-strictness leads to better approximation guarantees.

Theorem 6.2 *Consider a deterministic combinatorial optimization problem Π that is sub-additive, and let $\mathcal{A}$ be an α-approximation algorithm for Π with a β-uni-strict cost sharing function. Then* $\mathsf{Ind\text{-}Boost}(\Pi)$ *is an $(\alpha+\beta)$-approximation algorithm for* $\mathsf{Ind}(\Pi)$.

PROOF. While it is possible to prove this result closely following the lines of that for Theorem 3.1, we give a slightly different proof here.

First, some notation: let $\pi(S) = \prod_{j \in S} \pi_j \prod_{j \notin S}(1-\pi_j)$. Let F_0^* be the first-stage component of the optimal solution, and F_S^* be the second-stage component if the set of realized clients is S, and let Z^* be defined as in Equation (3.2).

First stage: Again, we claim that there is $\widehat{F}_1 \in \mathsf{Sols}(D)$ such that $\mathbf{E}[c(\widehat{F}_1)] \leq Z^*$; the actual proof is by a slightly different "coupling" argument. As a thought experiment, throw elements of D into σ sets $D_1, \ldots, D_\sigma$ independently and uniformly at random. Now, since D was picked by sampling U at rate $\sigma\,\pi_j$, each D_i is distributed as though we sampled element $j \in U$ with probability π_j. (The contents of different D_i's are correlated negatively, but we will only use linearity of expectations.)

Define $\widehat{F}_1 = F_0^* \cup F_{D_1}^* \cup F_{D_2}^* \cup \ldots \cup F_{D_\sigma}^*$. Again, $\widehat{F}_1 \in \mathsf{Sols}(D)$ from sub-additivity, and

$$\begin{aligned}\mathbf{E}_D[c(\widehat{F}_1)] &\leq c(F_0^*) + \mathbf{E}_D[\sum_{i=1}^{\sigma} c(F_{D_i}^*)] \\ &= c(F_0^*) + \sum_{i=1}^{\sigma} \mathbf{E}_{D_i}[c(F_{D_i}^*)] \\ &= c(F_0^*) + \sigma \sum_S \pi(S)\, c(F_S^*) \quad = Z^*.\end{aligned}$$

Now an α-approximation algorithm for $\mathsf{Det}(\Pi)$ gives us a solution F_0 with $\mathbf{E}[c(F_0)] \leq \alpha c(\widehat{F}_1) \leq \alpha Z^*$, bounding our first stage costs.

Second stage: Let S be the set of realized clients, and let $F_S = \bigcup_{j \in S} F_j$ be the result of our algorithm $\mathsf{Aug}_{\mathcal{A}}$. Note that for all $j \in S$, $F_0 \cup F_j \in \mathsf{Sols}(\{j\})$, thus by subadditivity $F_0 \cup F_S \in \mathsf{Sols}(S)$. We need to bound our expected second stage cost, which is $\sigma\,\mathbf{E}[c(F_S)]$, which we will bound by the expected *first stage* cost.

Define ϕ_j for an element $j \in U$ to be the random variable $\phi_j = \xi(D,j)$, and ψ_j to be the cost of augmenting a solution for D to include j as well, in the case that $j \in S$. (Hence, $\psi_j = c(F_j)$ if $j \in S$ and $\psi_j = 0$ if $j \notin S$.) Let $X_j = \sigma\psi_j - \beta\,\phi_j$.

Now let us condition on all the first-stage coin-tosses $\mathcal{T}$ in U except for j's toss. Let $D_{\mathcal{T}}$ be all the vertices picked according to $\mathcal{T}$ (which does not include j), and consider the expected value of X_j over the first-stage toss for j, and the tosses of the realized set S.

$$\mathbf{E}_{D,S}[\sigma\,\psi_j \,|\, \mathcal{T}\,] = \sigma \times \pi_j \times (1-\sigma\,\pi_j)\, c(F_j) \quad \text{and} \tag{6.9}$$

$$\mathbf{E}_D[\beta\,\phi_j \,|\, \mathcal{T}\,] = \beta \times \sigma \times \pi_j \times \xi(D_{\mathcal{T}} \uplus \{j\}, j). \tag{6.10}$$

By uni-strictness of $\mathcal{A}$, it follows that (6.10) is at least (6.9), and hence $\mathbf{E}_{D,S}[X_j \,|\, \mathcal{T}\,] \leq 0$. Since this holds for all $\mathcal{T}$, $\mathbf{E}_{D,S}[X_j] \leq 0$ unconditionally, and thus

$$\mathbf{E}_{D,S}[\psi_j] \leq \frac{\beta}{\sigma}\,\mathbf{E}_D[\phi_j]. \tag{6.11}$$

Note that Properties (P1) and (P2) of the cost shares ξ imply that

$$\begin{aligned}\sum_{j \in U} \mathbf{E}_D[\phi_j] = \sum_{j \in U} \mathbf{E}_D[\xi(D,j)] &= \mathbf{E}_D[\sum_{j \in D} \xi(D,j)] \\ &\leq \mathbf{E}_D[c(OPT(D))] \leq Z^*.\end{aligned} \tag{6.12}$$

Furthermore, $\mathbf{E}_{D,S}[F_S] \leq \sum_j \mathbf{E}_{D,S}[\psi_j]$ by sub-additivity; using this, (6.11) and (6.12), we get that expected second-stage cost $\sigma\mathbf{E}[c(F_S)] \leq \beta\, Z^*$, proving the result. $\square$

6.2 Steiner network

The Steiner network problem is a generalization of the Steiner tree problem, and is defined over an edge-weighted graph. A client u is now a pair (s_i, t_i) of vertices, and given a set of clients S, a feasible solution consists of a set of edges F such that for each $(s_i, t_i) \in S$, there is a path from s_i to t_i in F. The problem is easily verified to be sub-additive. The following result gives us the claimed 8-approximation for Ind(STEINER FOREST).

Theorem 6.3 ([5]) *There is a 4-approximation algorithm for the Steiner network problem which admits a 4-uni-strict cost-sharing function.*

The reader is referred to [5] for the algorithm description and a proof of Theorem 6.3. We note that algorithms with an approximation factor of 2 are known for the Steiner network problem, but none of them admits a uni-strict cost-sharing function. We believe that the techniques of [5] can lead to fully strict cost-sharing for Steiner Network, although we are still lacking a proof at this moment.

6.3 Facility Location

Improved results may be obtained for other problems which have already been studied in Section 5.

Theorem 6.4 *There is a 3-approximation algorithm $\mathcal{A}$ for the facility location problem, along with cost-shares ξ that are 3-uni-strict w.r.t. $\mathcal{A}$. Hence, there is a 6-approximation for* Ind(FACILITY LOCATION).

PROOF. The proof closely follows that of Theorem 5.1, which the reader is urged to peruse. Here, we will be concerned with the special case of the singleton set $T = \{j\}$.

Consider the run $\mathcal{A}(S \cup \{j\})$, and let p be the primary facility of j in this run. Here is the augmentation procedure $\mathsf{Aug}_{\mathcal{A}}$: if p is open in the run $\mathcal{A}(S)$, it simply assigns j to p. If p is closed, it has two options: if p is $\{j\}$-heavy, it opens p and assigns j to it. Otherwise, it assigns j to the closest facility opened in $\mathcal{A}(S)$.

We claim that the augmentation cost is at most $3\xi(S \cup \{j\}, j)$. Indeed, if we decide to open j's primary facility p, $\xi(S \cup \{j\}, j)$ can pay for a b-fraction of the facility cost of p plus assignment cost of j. If not, Claim 5.2 implies that $t_p(S) \leq t_p(S \cup \{j\}) \frac{|C_p|}{|C_p \cap S|} - \frac{c(j,p)}{|C_p \cap S|}$. We know that there is an open facility r within distance $2\,t_p(S)$ from p, and so reroute j to r. The connection cost in this case is at most

$$c(j,p) + 2\left(t_p(S \cup \{j\})\,\frac{|C_p|}{|C_p \cap S|} - \frac{c(j,p)}{|C_p \cap S|}\right)$$

which is at most $3\max(c(j,p), t_p(S \cup \{j\})) = 3\,\xi(S \cup \{j\}, j)$. Since we need to minimize $\max\{1/b, 3\}$, the best value is $b = 1/3$, finishing the proof. $\square$

6.4 Vertex Cover

We present the following improvement on the 6.3-approximation given for $\mathsf{Ind}(\text{VERTEX COVER})$ given by Immorlica et al. [7]. As discussed in Section 5.2, to obtain an approximation algorithm for $\mathsf{Stoc}(\text{VERTEX COVER})$, it is enough to consider the relaxed version of the problem, where we are allowed to make arbitrary payments p^1 and p^2 to vertices in the two stages, with the vertex v being bought if $p^1(v) + p^2(v)/\sigma \geq c_v$. As mentioned there, results for this relaxed problem can be easily transferred back to obtain an algorithm in the standard model: this is done by choosing a vertex with probability $p^1(v)/c_v$ in the first stage, and then picking it in the second stage if $p^1(v) + p^2(v)/\sigma \geq c_v$ and it was not already picked.

Theorem 6.5 *There is a 2-approximation algorithm $\mathcal{A}$ for the relaxed Vertex Cover problem, and cost-shares ξ that are 1-strict with respect to $\mathcal{A}$, thus giving us a 3-approximation algorithm for* $\mathsf{Ind}(\text{VERTEX COVER})$.

PROOF. The algorithm $\mathcal{A}$, as well as the cost shares ξ, are the same as in Section 5.2. To augment a solution $\mathcal{A}(S)$ on the addition of the edge $e = \{u, v\}$, the augmentation procedure $\mathsf{Aug}_{\mathcal{A}}$ opens the endpoint whose reduced cost is less. I.e., if the payments in $\mathcal{A}(S)$ are denoted by p, we pay $\delta = \min(c_u - p(u), c_v - p(v))$ to the vertex from $\{u, v\}$ that achieves this minimum and open it. Proving strictness is now equivalent to proving that $\delta \leq \xi(S \cup \{e\}, e)$.

Indeed, consider the runs $\mathcal{A}(S)$ and $\mathcal{A}(S \cup \{e\})$. Both runs behave identically till some endpoint of e, say u, goes tight in the latter run. At that point, the payment made by other edges to u in $\mathcal{A}(S \cup \{e\})$ is exactly $c_u - \xi(S \cup \{e\}, e)$. Since the two runs were identical till now, u has received this payment in $\mathcal{A}(S)$ as well, and hence $p(u) \geq c_u - \xi(S \cup \{e\}, e)$. Hence, $\xi(S \cup \{e\}, e) \geq c_u - p(u) \geq \delta$, proving the theorem. $\square$

6.5 k-Hypergraphs

A set system $G = (V, E)$ on a ground set V with $E \subseteq 2^V$ is called a k-hypergraph if the cardinality of each element in E is no more than k. The Hypergraph Vertex Cover problem is defined as a straightforward generalization of the Vertex Cover problem to hypergraphs. We briefly mention how we can obtain a $(k+1)$-approximation for $\mathsf{Ind}(k\text{-HYPERGRAPH VERTEX COVER})$.

The standard primal-dual algorithm for vertex cover discussed in Section 5.2 yields a k-approximation for Hypergraph Vertex Cover, since the dual variable of each edge is used to pay for at most all its k vertices. Defining reduced costs and cost shares as in the k-hypergraph extension of the definitions in Section 5.2, the augmentation procedure $\mathsf{Aug}_{\mathcal{A}}$ augments the solution to an edge e by opening the vertex in e whose reduced cost is minimum. This can be verified to be 1-strict along the same lines as Theorem 6.5.

Corollary 6.6 *There is a k-approximation algorithm $\mathcal{A}$ for the relaxed k-Hypergraph Vertex Cover problem, and cost-shares ξ that are 1-strict with respect to $\mathcal{A}$, thus giving us a $(k+1)$-approximation algorithm for* $\mathsf{Ind}(k\text{-HYPERGRAPH VERTEX COVER})$.

7. REFERENCES

[1] 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.).

[2] John R. Birge and François Louveaux. *Introduction to stochastic programming*. Springer Series in Operations Research. Springer-Verlag, New York, 1997.

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

[4] Ashish Goel and Piotr Indyk. Stochastic load balancing and related problems. In *40th Annual Symposium on Foundations of Computer Science (New York, 1999)*, pages 579–586. IEEE Computer Soc., Los Alamitos, CA, 1999.

[5] 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.

[6] Anupam Gupta, Amit Kumar, and Tim Roughgarden. Simpler and better approximation algorithms for network design. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 365–372, 2003.

[7] 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 684–693, 2004.

[8] 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.

[9] 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.

[10] Peter Kall and Stein W. Wallace. *Stochastic programming*. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons Ltd., Chichester, 1994.

[11] David R. Karger and Maria Minkoff. Building Steiner trees with incomplete global knowledge. In *Proceedings of the 41th Annual IEEE Symposium on Foundations of Computer Science*, pages 613–623, 2000.

[12] Willem K. Klein Haneveld and Maarten H. van der Vlerk. Stochastic integer programming: general models and algorithms. *Ann. Oper. Res.*, 85:39–57, 1999. Stochastic programming. State of the art, 1998 (Vancouver, BC).

[13] Willem K. Klein Haneveld and Maarten H. van der Vlerk. *Stochastic Programming*. Department of Econometrics and OR, University of Groningen, Netherlands, 2003.

[14] Jon Kleinberg, Yuval Rabani, and Éva Tardos. Allocating bandwidth for bursty connections. *SIAM J. Comput.*, 30(1):191–217 (electronic), 2000.

[15] 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.

[16] Rolf H. Möhring, Andreas S. Schulz, and Marc Uetz. Approximation in stochastic scheduling: the power of lp-based priority policies. *Journal of the ACM (JACM)*, 46(6):924–942, 1999.

[17] Hervé Moulin and Scott Shenker. Strategyproof sharing of submodular costs: budget balance versus efficiency. *Econom.*

Theory, 18(3):511–533, 2001.

[18] 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.

[19] Michael Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, 1995.

[20] Robert C. Prim. Shortest interconnection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[21] R. Ravi and Amitabh Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. In *Proceedings of the 10th Integer Programming and Combinatorial Optimization Conference*, 2004. To appear.

[22] Gabriel Robins and Alexander Zelikovsky. Improved Steiner tree approximation in graphs. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–779, 2000.

[23] R. Schultz, L. Stougie, and M. H. van der Vlerk. Two-stage stochastic integer programming: a survey. *Statist. Neerlandica*, 50(3):404–416, 1996.

[24] Martin Skutella and Marc Uetz. Scheduling precedence-constrained jobs with stochastic processing times on parallel machines. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 589–590. Society for Industrial and Applied Mathematics, 2001.

[25] 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.