On the Costs and Benefits of Procrastination: Approximation Algorithms for Stochastic Combinatorial Optimization Problems

Nicole Immorlica* David Karger* Maria Minkoff* Vahab S. Mirrokni*

Abstract

Combinatorial optimization is often used to "plan ahead," purchasing and allocating resources for demands that are not precisely known at the time of solution. This advance planning may be done because resources become very expensive to purchase or difficult to allocate at the last minute when the demands are known. In this work we study the tradeoffs involved in making some purchase/allocation decisions early to reduce cost while deferring others at greater expense to take advantage of additional, late-arriving information. We consider a number of combinatorial optimization problems in which the problem instance is uncertain—modeled by a probability distribution—and in which solution elements can be purchased cheaply now or at greater expense after the distribution is sampled. We show how to approximately optimize the choice of what to purchase in advance and what to defer.

1 Introduction

Combinatorial optimization is often used to "plan ahead," purchasing and allocating resources for needs that are not precisely known at the time of solution. For example, a network designer might have to make a best guess about the future demands in a network and purchase capacity accordingly. At other times, however, it is possible to "wait and see," deferring decisions about resource allocation until demands or constraints become clear. This allows one to plan an optimal solution for each potential outcome. There is often a trade-off involved, in that allocations made later may be more expensive. For example, the network designer may be able to arrange cheap long-term contracts for capacity purchased ahead of time, but may need to purchase extra capacity at the last minute on a more expensive "spot market."

Beyond the basic optimization problem, then, there is the problem of deciding which part of the solution should be set early and cheaply based on limited information about the problem input, and which part should be deferred and solved more expensively with the arrival of additional information about the input.

In this paper we study a particular framework, derived from stochastic programming, for dealing with this time-information trade-off in the presence of uncertainty. Formally, we postulate a probability distribution $\Pr[\mathcal{I}]$ over problem instances $\mathcal{I}$. We consider a collection of variables x_i and y_j that describe candidate solutions to the problem, where different settings of the variables are feasible for different inputs $\mathcal{I}$. We are required to set the variables x_i, then sample a problem instance $\mathcal{I}$ from the distribution, and finally, with knowledge of the instance, to set the variables y_j so that (x, y) is feasible for $\mathcal{I}$. Given a cost function $c(x, y)$ on solutions, our goal is to minimize the expected cost $E[c(x, y)]$ subject to the feasibility constraints of the random instance.

In the standard two-stage stochastic programming with recourse model [1, 11], the problem instances are polytopes over x_i and y_j (representing linear or integer linear programs), and the cost function is linear. When the distribution involves only a small number of distinct instances $\{\mathcal{I}\}$, the problem can be formulated as large scale linear (or integer linear) program and solved using standard methods from mathematical programming.

1.1 Our Results. We explore the stochastic optimization framework in the context of problems with more combinatorial structure, and apply new techniques that exploit this structure. Specifically, we study stochastic versions of min-cost flow, bin packing, vertex cover, shortest path, and the Steiner tree problem. In each of these problems, a ground set of elements $e \in E$ is specified (vertices in the vertex cover problem, edges in the Steiner problems, bins in bin packing). A (randomly selected) problem instance $\mathcal{I}$ defines a set of feasible solutions, each corresponding to a subset $\mathcal{F}_{\mathcal{I}} \subseteq 2^E$. We can buy certain elements "in advance" at cost c_e, then sample a problem instance, and must then buy other elements at "last-minute" costs λc_e so as to produce a feasible set for our problem instance. Our goal is to minimize the expected total cost.

*MIT Computer Science and AI Laboratory. {nickle,karger,mariam,mirrokni}@theory.lcs.mit.edu Supported by the IRIS project under National Science Foundation cooperative agreement ANI-0225660. Dr. Minkoff was supported by an AT&T Laboratories Graduate Fellowship

It is noteworthy that all of our problems are covering problems and thus "monotone," in that any superset of a feasible solution is feasible. This is convenient because it means that purchasing elements in advance never "invalidates" a potentially feasible solution—the advance purchases may be wasted, but at worst they can be ignored and any desired feasible solution constructed from the post-sampling purchases. Thus, we can focus all of our attention on optimizing cost without worrying about making feasibility-missteps.

We study two main types of instance probability distributions. A *bounded support* distribution gives nonzero probability to only a polynomial number of distinct problem instances. And *independent* distribution makes each element/constraint of the problem instance active independently with some probability.

Our results for specific problems are as follows:

Min Cost Flow. Given a source and sink and a probability distribution on demand, buy some edges in advance and some after sampling (at greater cost) such that the given amount of demand can be routed from source to sink. This problem can be solved exactly via linear programming.

Bin packing. A collection of items is given, each of which will need to be packed into a bin with some probability. Bins can be purchased in advance at cost 1; after the determination of which items need to be packed, additional bins can be purchased at cost $\lambda > 1$. How many bins should be purchased in advance to minimize the expected total cost? We show that this problem can be efficiently approximated arbitrarily close to optimal.

Vertex Cover. A graph is given, along with a probability distribution over sets of edges that may need to be covered. Vertices can be purchased in advance at cost 1; after determination of which edges need to be covered, additional vertices can be purchased at cost λ. Which vertices should be purchased in advance? We give a 4-approximation based on a linear programming relaxation for the case where the probability distribution involves only polynomially many distinct edge sets, and a combinatorial 6.3-approximation for the case when each edge must be covered independently with fixed probability p.

Cheap Path. We are given a graph and told that a randomly selected pair of vertices (or one fixed vertex and one random vertex) will need to be connected by a path. We can purchase edge e at cost c_e before the pair is known or at cost λc_e after and wish to minimize the expected total edge cost. We show that this problem is equivalent to the multicommodity rent-or-buy problem, so that previously known approximation algorithms apply.

Steiner Tree. A graph is given, along with a probability distribution over sets of terminals that need to be connected by a Steiner tree. Edge e can be purchased at cost c_e in advance or at cost λc_e after the set of terminals is known. We give a constant factor approximation for the case where the expected number of terminals is constant (generalizing the Cheap Path result). We also give a constant factor approximation for the case where the edges form an ultrametric and an $O(\log n)$ approximation for general edge costs.

1.2 Related work.

Stochastic programming is a tremendous field with a vast literature [14]. It is applicable whenever probability distributions of inputs are known or can be estimated. One of the most widely used models in stochastic programming is the two-stage recourse model mentioned earlier. It involves an initial deterministic decision, an opportunity to observe additional information, and then a recourse action in response to each random outcome. The two-stage model can be naturally generalized by adding additional recourse stages, each consisting of an observation and a decision responding to it. Stochastic linear programs are generally tackled via a combination of mathematical programming and advanced probabilistic techniques. A key difficulty in solving these problems is dealing with a very large uncertainty space, as one gets a separate set of constraints for each potential outcome.

Stochastic multicommodity flow is a particularly well studied problem in this area (cf. [12]). When it is the costs that are stochastic the problem is relatively easy as the expectations propagate through, but the case of stochastic capacities or demands has not yet been fully solved. Stochastic multicommodity flow has been used extensively as a model for a variety of real-world applications. Recently, Mitra and Wang [10] derived a flow-based framework for stochastic traffic engineering in which the objective is to maximize revenue from serving demands that are specified by probability distributions. Their model is similar to ours, in that it uses a two-tier cost function and explores the trade-off between deterministic allocations and probabilistic provisioning. They present conditions under which the problem can be reduced to an instance of convex programming.

A rather different stochastic optimization framework assumes random instances but requires only that constraints be satisfied *with certain probability.* This framework is sometimes known as "chance constrained programs." For example, Kleinberg, Rabani and Tardos [8] consider chance-constrained knapsack, bin-packing and load-balancing problems. In particular, in bin-packing, given a probability distribution on the sizes of items, one is concerned with packing all of the items into a minimum number of bins so that it is unlikely that any one of them overflows. Kleinberg et al. provide an approximation guarantee that is a function of $\log p^{-1}$ (where p is the probability with which bin capacity is allowed to be violated).

2 Preliminaries

In this section we give a formal definition of the preplanning framework for stochastic combinatorial optimization problems, discuss a number of basic properties shared by the problems in this framework, and present some generally applicable techniques.

2.1 Formal Problem Statement. Formally, a preplanning version of a combinatorial optimization problem is specified in our framework by a ground set of elements $e \in E$, a probability distribution on instances $\{I\}$, a cost function $c : E \rightarrow \mathbb{R}$, and a penalty factor $\lambda \geq 1$. Each instance $\mathcal{I}$ has a corresponding set of feasible solutions $\mathcal{F}_{\mathcal{I}} \subseteq 2^E$ associated with it. Suppose a set of elements $A \subseteq E$ is purchased before sampling the probability distribution. Let c_A denote the *posterior cost function*, i.e.

$$c_A(e) = \begin{cases} 0 & \text{if } e \in A \\ \lambda c(e) & \text{otherwise} \end{cases}$$

The objective of a preplanning combinatorial optimization problem is to choose a subset of elements A to be purchased in advance so as to minimize the total expected cost of a feasible solution

$$c(A) + \mathrm{E}\left[\min_{S \in \mathcal{F}_{\mathcal{I}}} c_A(S)\right]$$

over a random choice of an instance $\mathcal{I}$.

2.2 The Threshold Property. Our first observation is that optimal preplanning solutions exhibit a natural local-optimality property.

Consider a solution that purchases some set of elements $A \subseteq E$ in advance and then, on sampling problem instance $\mathcal{I}$, buys additional elements $L_{\mathcal{I}}$. Note that $A \cup L_{\mathcal{I}}$ is a feasible solution for the instance $\mathcal{I}$. Conversely, knowing the complete feasible solution $F_{\mathcal{I}} \subseteq E$ used when each instance $\mathcal{I}$ is sampled, it is easy to determine which elements are purchased in advance:

THEOREM 2.1. *An element should be purchased in advance if and only if the probability it is used in the solution for a randomly chosen instance exceeds $1/\lambda$.*

The theorem follows immediately from the fact that the cost contribution of an element e is $\lambda \Pr[e \text{ used}]$ if it is not purchased in advance. We refer to this theorem as the *Threshold Property*.

2.3 Approximation Algorithms as Subroutines. With the exception of min-cost flow and cheap path, every problem we study is $\mathcal{NP}$-hard even in the traditional, non-stochastic setting [3]. This immediately implies that the preplanning versions of these problems are also $\mathcal{NP}$-hard (one can simply define a distribution that assigns probability one to the unique instance of interest). Additionally, it complicates our task since we do not know how to find an optimal solution even to a particular instance. However, we observe that using an approximation algorithm instead has limited consequences:

THEOREM 2.2. *Given an α-approximation algorithm* ALG*, let A_0 be a subset of elements that minimizes the expected cost of a solution obtained with* ALG *over a random choice of a instance $\mathcal{I}$, i.e. $c(A) + \mathrm{E}\left[c_A^{\mathsf{ALG}}(\mathcal{I})\right]$. Then the cost of a preplanning solution that purchases elements in A_0 in advance is at most α times the minimum possible cost whether one uses an exact or an approximation algorithm to complete the solution for the randomly sampled instance.*

The above theorem implies that we can reduce the preplanning version of an $\mathcal{NP}$-hard problem to solving a preplanning instance of another optimization problem that has a polynomial-time algorithm. For example, since in a metric space an MST over a subset of nodes S provides a 2-approximation for min-cost Steiner tree on S, we can use preplanning to optimize the cost of an MST (instead of a steiner tree) on the terminals and lose only a factor of 2 in approximating the optimum preplanning cost.

3 Network Predesign for Min-Cost Flow

Consider a stochastic min-cost flow problem. We wish to provide capacity on a network sufficient to carry a (random) amount of flow demand D from a source s to a sink t. We have an option of pre-installing some amount of capacity in advance at some cost per unit. We are also allowed to rent additional capacity once the demands become known, but at cost a factor of λ larger per unit. The sum of capacity installed in advance and capacity rented must also satisfy a given upper bound (total capacity) for each edge. The goal is to minimize (over the given probability distribution on demands) the expected cost of installing sufficient capacity in the network so as to be able to satisfy the demand.

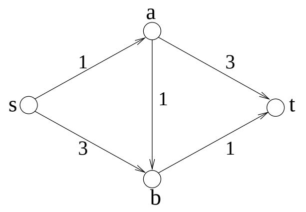

Figure 1: Example demonstrating anomalies in min-cost flow network predesign.

When the probability distribution on D has polynomial support (i.e., only polynomially many distinct demands are possible) then the min-cost flow network predesign problem is polynomial-time solvable using the standard "merger of linear programs" approach from stochastic programming described in the introduction. We define a variable a_{ij} for the capacity purchased in advance on edge (i, j), and then write down, for each possible demand value, a min-cost flow linear program on a graph where there are a_{ij} units of cost-0 capacity on edge (i, j). We then take a weighted (by the probability distribution) combination of the individual linear programs' objectives to get our objective function.

Although this solution is well known, we discuss min-cost flow because it demonstrates several interesting behaviors in stochastic optimization.

Provisioning for expected demand is unwise. The plausible approach of provisioning for the expected amount of demand, i.e. buying capacity sufficient to route the expected demand in the network, can be far from optimal. Consider a network consisting of a single edge of base cost 1 between terminal nodes s and t. Suppose the demand D that has to be sent from s to t is d with probability $p = (1+\lambda)/2\lambda$ and zero otherwise. Provisioning for expected demand means purchasing capacity pd in advance and buying the rest of it later if the actual demand turns out to be d. The expected cost of this solution is $pd + \lambda p(1-p)d$. On the other hand, a solution that buys capacity d in advance incurs a cost of d. The ratio between these two solutions is $(1+\lambda)^2/4\lambda$, which grows arbitrarily bad with large λ.

Flow may be sent on non-shortest paths. Consider the graph G on 4 nodes shown in Figure 1. Suppose that each edge has maximum allowed capacity 1 and that $\lambda = 2$. Let $\Pr[D = 0] = 1/4$, $\Pr[D = 1] = 1/4$, and $\Pr[D = 2] = 1/2$. Note that the only way to route 2 units of demand is by saturating the edges $(s, a), (a, t), (s, b), (b, t)$. Thus, the probability of using each of these edges is at least $1/2 \geq 1/\lambda$, so by the Threshold Property introduced in Section 2.2 any optimal solution has to buy 1 unit of capacity on each of those arcs in advance. However, once they are prepurchased, it is unnecessary to use arc (a, b). Hence, the shortest s-a-b-t path is never used to route flow, even if only 1 unit of demand needs to be sent.

Prepurchases need not form paths. Using the same graph G in Figure 1, take $\lambda = 2, \Pr[D = 0] = 5/12$, $\Pr[D = 1] = 1/4$, $\Pr[D = 2] = 1/3$. In this case an optimal solution ends up prepurchasing 1 unit of capacity only on arcs (s, a) and (b, t).

4 Bin packing with preplanning

In the classical bin-packing problem, one is given a set of n items of different sizes. The goal is to pack all items into unit-size bins so as to minimize the cost of the packing, namely the total number of bins used. Now suppose that the set of items to be packed is chosen at random. Bins may be purchased in advance at cost 1 or after the set of items is known at cost λ. Our objective is to minimize the cost of bins required for packing in expectation over a random selection of a set of items. Since classical bin-packing is NP-hard, we can of course not expect to find an optimal solution. But we show how to match the approximation ratio achievable for the standard bin-packing problem.

Bin packing can be brought into our framework by thinking of each bin as an "element" that may need to be purchased. But the problem is particularly easy because all elements are the same. Thus, we need only specify *how many* of these elements are to be purchased in advance. For convenience, we can assume that the bins are numbered $b_1, \ldots, b_n$ and that any solution buys bins in order—i.e., it buys some prefix $b_1, \ldots, b_k$ of the bins. The problem is then essentially solved by invoking the Threshold Property, which says that we should buy bins in advance only if their probability of use exceeds $1/\lambda$. Since the probability of using b_i is no greater than that of using b_{i-1}, there is some maximum i for which the probability of using b_i exceeds $1/\lambda$, implying we should buy i bins in advance. Put more directly:

THEOREM 4.1. *Let B be a random variable denoting the optimum number of bins used for a particular (random) set of items. Then the optimum preplanning algorithm is to buy, in advance, the number of bins k such that* $\Pr[B \geq k] \geq 1/\lambda$ *while* $\Pr[B \geq k+1] < 1/\lambda$.

This essentially solves our problem, modulo two details: that we cannot compute B (the optimum bin packing size) and that we apparently need to compute the probability distribution over B. Both these problems are surmountable. In short, we can use an approximation algorithm instead of an exact algorithm to pack the items, inheriting the same approximation bound, and we can use Monte Carlo sampling (repeatedly choose a random set of items and pack them) to get an accurate estimate for $\Pr[B \geq k]$ and identify an approximately optimum value of k to buy in advance. Using Monte Carlo estimation adds another $(1+\epsilon)$ factor to the approximation bound (taking time polynomial in $1/\epsilon$). Details are omitted.

Notice that our approach to bin-packing did not make explicit use of the probability distribution over items to be packed. It applies to any probability distribution on items and sizes from which we can sample efficiently.

5 Vertex Cover with preplanning.

In the (unweighted) vertex cover problem, given an undirected graph, the goal is to find a subset of vertices of minimum cardinality such that at least one endpoint of each graph edge is in the subset. Now suppose that only a subset of edges needs to be covered, but that we do not know in advance exactly which ones. Given a probability distribution on the sets of edges to be covered, the goal of the preplanning vertex cover problem is to determine an optimum set of vertices to buy in advance (at cost 1), so as to minimize the expected cost of a vertex cover for a random subset of edges, provided that additional vertices can be added at cost λ each.

5.1 Bounded-Support Distributions. In this section we show how to obtain a constant-factor approximation for the case when the probability distribution over problem instances has bounded support, i.e. the number of possible subsets of edges to be covered is polynomially bounded. As with min-cost flow, we start with the standard stochastic programming of combining linear programs for individual problem instances. We observe that this technique can be extended to combinatorial optimization problems that are solved by rounding fractional solutions to linear programming relaxations, and apply it to the vertex cover problem.

Suppose we are given a graph $G = (V, E)$ and a probability distribution on the polynomial number of "active" edge sets from $\mathcal{F} \subseteq 2^E$ that might to be covered by the vertices. In other words, for each $F \in \mathcal{F}$, we have access to p^F, the probability that we have to construct a vertex cover for exactly the edges in F. For $F \notin \mathcal{F}$, $p^F = 0$.

We can model the preplanning version of the vertex cover problem with an integer program. For each vertex i, let $x_i = 1$ if i is bought in advance, and let $y_i^F = 1$ if i is added to the vertex cover once it is revealed that F is the set of active edges that have to be covered. For edge $(i,j) \in F$, either i or j has to be in a vertex cover for edge set F, i.e. at least one of the variables x_i, x_j, y_i^F, y_j^F has to be 1. Writing this constraint for all edges in each of the potential edge sets, we obtain the following integer program:

$$\begin{array}{lll} \text{Min} & \sum_{i \in V} x_i + \lambda \sum_{i \in V, F \in \mathcal{F}} p^F y_i^F & \\ \text{s.t.} & x_i + x_j + y_i^F + y_j^F \geq 1 & \forall\ (i,j) \in F \in \mathcal{F} \\ & x_i \in \{0,1\} & \forall\ i \in V \\ & y_i^F \in \{0,1\} & \forall\ F \in \mathcal{F},\ i \in V \end{array}$$

Since the number of edge sets $F \in \mathcal{F}$ is polynomial, the corresponding linear programming relaxation can be solved efficiently. The value of an optimal solution to this LP relaxation provides a lower bound on the optimal integer solution.

Next, we construct a solution to the vertex cover preplanning problem by rounding an optimal solution to the LP. Let us buy vertices i such that $x_i \geq 1/4$, i.e., we round such x_i to 1. Once the set F of edges to be covered is revealed, we purchase the additional vertices i such that $y_i^F \geq 1/4$, i.e., we round such y_i^F to 1. Let $\hat{x}, \hat{y}$ be the corresponding integral solution. The analysis goes exactly as for the standard vertex cover problem: at least one of the four variables associated with a particular sampled edge must have value at least 1/4, so the rounded solution is feasible. At the same time, we have multiplied each fractional value by at most 4 (from 1/4 to 1) so our solution is a 4-approximation to the fractional solution, and thus to the true optimum.

5.2 Independent edge set. In this section we consider the version of the problem in which each edge is active (e.g. has to be covered) independently with probability p. As before, given a graph $G = (V, E)$, we would like to determine an optimum set of vertices $A \subset V$ to buy in advance (at cost 1), so as to minimize the expected cost of a vertex cover for a random subset of edges $F \subseteq E$, provided that additional vertices can be added at cost λ each. Note that once vertices in the set A are specified, extending it to a cover of the edge set F is equivalent to finding a vertex cover in the $V \setminus A$-induced subgraph of $G_F = (V, F)$. We assume that $\lambda \geq \lambda_0$ where $\lambda_0 = 3.15$. In the event $\lambda < \lambda_0$, we can obtain a trivial 6.3-approximation algorithm by not purchasing any vertices in advance and using a 2-approximation algorithm for vertex cover.

This problem demonstrates one key idea for tackling stochastic problems: that of concentrating sufficient probability at one spot to make the Threshold Property apply so that we can justify purchasing certain elements in advance. In particular, we focus our attention on certain high-degree vertices (in the original graph) and argue that the fact that they are very likely to have an incident edge sampled justifies purchasing them in advance. We will use much the same idea to tackle Steiner tree problems in the next section.

DEFINITION 5.1. *Given a graph $G = (V, E)$, define a* k-matching *to be a subset of edges that induces degree at most k on every vertex. Call a vertex $v \in V$* tight *if its degree is exactly k.*

Our approximation algorithm is as follows. Define $k = \log_{1-p}(1 - 1/\lambda)$, construct some *maximal* k-matching (greedily), and purchase in advance the set A_t of tight vertices in the k-matching.

To show that this algorithm yields a constant-factor approximation, we prove two things. First, that the total number of vertices prepurchased (at cost 1) by our algorithm is proportional to the optimum solution's (expected) cost (in both stages). Second, we prove that in the graph which remains, it is optimal to prepurchase no additional vertices. Thus, our algorithm's second-stage purchase cost is optimal and, in particular, less than the both-stages cost of the optimum solution on the original graph.

LEMMA 5.1. $|A_t| \leq 4.3\mathsf{OPT}$.

Proof. Due to space limits, we will be sloppy with constants. In particular we use the fact that $k = \Theta(1/\lambda p)$, since $\frac{1}{\lambda} \leq \frac{1}{\lambda_0}$.

We consider an instance of the preplanning problem in which only edges of the chosen k-matching have to be covered (e.g. the edge set of the graph from which some edges are sampled consists just of the edges of the k-matching). Clearly, the cost of an optimal solution to this instance is no more than OPT.

We prove our bound using a surcharging scheme. For each dollar spent by the optimum, our charging scheme spends at most one additional dollar. At the same time, it spends $\Omega(1)$ dollars per tight vertex. It follows that the number of tight vertices is $O(1)$ times the expected number of dollars spent in our charging scheme, which in turn is twice the optimum expected cost.

The charging scheme goes as follows. For each vertex u purchased in advance by OPT, we spend one dollar on u and an extra $1/k = \Theta(\lambda p)$ dollars on each neighbor of u. Since u has degree at most k, this costs at most one additional dollar. In the second stage, consider the (random) set of edges actually included. If (u, v) is an *isolated edge*, i.e. there are no other edges incident on u or v, and neither u nor v has been purchased in advance, then OPT must spend λ dollars purchasing u or v. In this case, we charge an additional λ dollars to the *other* endpoint. In this case as well, we end up spending at most twice what OPT spends.

Now consider any tight vertex v. If OPT purchases it in advance we are done as a dollar was spent on it. If not, consider each neighbor u of v. If u is purchased in advance then v receives $1/k$ dollars from u. Now suppose u is not purchased in advance. Consider the event I_u that (u, v) is included as an isolated edge in the random problem instance. This event happens when (u, v) is chosen and no other edges incident on u or v are chosen; since u and v have degree at most k and $\lambda > \lambda_0$,

$$\begin{aligned} \Pr[I_u] &\geq p(1-p)^{2k} \\ &= p(1-p)^{\Theta(1/\lambda p)} \\ &= \Omega(p). \end{aligned}$$

Since the events I_u are disjoint, and since v receives λ dollars each time some event I_u occurs, we conclude that vertex v receives $\Omega(\lambda p)$ dollars in expectation for each neighbor u not bought in advance.

In summary, we have shown that v receives $\Omega(\lambda p)$ dollars from each neighbor—deterministically when that neighbor is bought in advance, in expectation otherwise. Since v is tight, it has $k = \Theta(1/\lambda p)$ neighbors and the lemma follows.

LEMMA 5.2. *Purchasing the tight vertices in advance (and deleting their incident edges as covered) leaves a graph in which it is optimal to prepurchase no vertices.*

Proof. In the original graph, every edge not in the k-matching must have at least one tight endpoint (otherwise it could have been added to the k-matching, contradicting its maximality), and is therefore covered by the tight vertices. The remaining (k-matching) edges induce degree at most k in every vertex. The probability that any vertex has any incident edge (and is thus useful in the solution) is then less than $1/\lambda$. Thus by the Threshold Property it is optimal to buy no vertices in advance.

THEOREM 5.1. *Purchasing in advance a set of all tight vertices induced by a maximal k-matching, where $k = \lceil \log_{1-p}(1 - \frac{1}{\lambda}) \rceil$ yields a solution of cost at most 6.3 times the optimum.*

Proof. The first lemma shows that purchasing all the tight vertices costs at most $4.3 \cdot \mathsf{OPT}$. It leaves a graph whose optimum solution is only cheaper, meaning it too costs at most OPT. And the second lemma proves that buying nothing is an optimal solution. Thus, we spend at most $4.3 \cdot \mathsf{OPT}$ in the first stage and at most $2\mathsf{OPT}$ in the second stage by using a 2-approximation for vertex cover, for a total of $6.3\mathsf{OPT}$.

6 Steiner Tree Predesign

In the network Steiner tree problem we are given an edge-weighted graph $G = (V, E)$ and a subset of nodes $S \subset V$ that need to be connected. The goal is to find a minimum cost tree spanning S. We consider preplanning versions of this problem, in which S is the set of active clients drawn from some distribution.

Formal problem statement. Let $G = (V, E)$ be an undirected edge-weighted graph. Let $c_e \geq 0$ denote the cost of an edge $e \in E$. We call a subset S of nodes (clients) active if all of the nodes in it wish to be connected. Given a probability distribution over active sets $\{S\}$, the objective is to minimize the expected cost of connecting up active clients. A subset of edges $A \subset E$ can be purchased in advance at cost c_e; additional edges can be purchased later on at cost λc_e, where $\lambda \geq 1$. Once a set S of active clients is revealed, the cheapest way to connect up all the clients in S is to build a min-cost Steiner tree over the vertices of S using the edge cost function c_A, where $c_A(e) = 0$ if $e \in A$, and $c_A(e) = \lambda c_e$ otherwise. Let $T_{\mathsf{ST}}(S)$ be an optimum Steiner tree over S for the edge cost function c_A. The objective of the network predesign problem is to choose A to minimize the cost of the solution $c(A)+c_A(T_{\mathsf{ST}}(S))$ in expectation over S.

6.1 Relation to previous problems.

A number of interesting special cases of this problem are equivalent to previously-studied combinatorial optimization problems; we therefore inherit constant factor approximations from those problems.

Cheap path to root. A root is specified in advance and a single, randomly chosen node wishes to be connected to that root. This problem is equivalent to connected facility location, with probability (scaled by λ) replacing demand, and hence can be approximated to within a constant [5, 13].

Cheap path. A randomly chosen pair of nodes wishes to be connected. This problem is equivalent to multicommodity rent-or-buy, with probability (scaled by λ) replacing demand, which has several constant factor approximation algorithms [9, 4].

When nodes become active independently, but the expected number of active clients is 1, there is a constant probability of having exactly one active client. This allows a simple reduction to the previous problems.

THEOREM 6.1. *If the expected number of active clients is at most 1, an optimal solution for cheap path-to-root yields a 2-approximation to the rooted Steiner tree predesign problem.*

Since the cheap path-to-root problem can be approximated to within a constant factor 3.55 [5] we get a 7.1-approximation algorithm for our special case. An obvious generalization yields a $7.1k$-approximation for the case where the expected number of clients is $k > 1$.

6.2 High probability nodes.

We turn to the case of independent client activations. In this section we show that if all node activation probabilities are lower-bounded by $1/\lambda$, then prepurchasing a minimum spanning of the entire node set is a 2-approximation to the optimum. This result is of limited interest but is an important component of our general solution for ultrametrics in the following section.

We can assume without loss of generality that our graph is *metric*—i.e., that the edge connecting any two vertices is a shortest path—since adding an edge of length equal to the shortest path does not change the optimum (one can always use the path instead of the imaginary edge). In such a graph, it is known that the optimum Steiner tree on any subset of the vertices is two approximated by the minimum spanning tree (MST) on the graph induced by those vertices.

LEMMA 6.1. *If every vertex in a graph is active with probability at least p, then the expected cost of the MST on the active vertices is at least p times the cost of the entire MST.*

Note that this lemma does not distinguish pre- and post-purchase costs.

Proof. (Sketch). The basic approach is to analyze Prim's algorithm with deferred decisions to build a minimum spanning tree on the active nodes. Details are similar to the Karger-Klein-Tarjan [7] analysis of minimum spanning tree value using a randomly sampled *edge* subset.

COROLLARY 6.1. *If for all clients i, $p_i \geq \frac{1}{f\lambda}$ (where $f \geq 1$), then prepurchasing a minimum spanning tree over all potentially active clients is a f-approximation to Steiner tree predesign.*

Proof. Let M be the prepurchase cost of the entire MST. Suppose the optimum buys nothing in advance. Then by the previous lemma, the expected prepurchase cost of the MST of the active clients is at least $M/f\lambda$. Since the optimum buys late, it would have to multiply this cost by λ, paying M/f—meaning M is an f-approximation to the optimum. The general proof follows by observing that the spending c in advance can only reduce the cost of the MST by c, implying that the optimum would still expect to pay at least $(M - c)/f$ in the second stage. Thus, the optimum is at least $c + (M - c)/f \geq M/f$.

6.3 Algorithm for Ultrametric Case. We now give a constant factor approximation algorithm for the Steiner tree predesign problem for the case when the underlying graph $G = (V, E)$ forms an *ultrametric* — an assignment of edge weights such that $\tilde{c}_{uv} \leq \max(\tilde{c}_{uw}, \tilde{c}_{wv})$. This ultrametric property implies that the shortest-path distance between any two vertices in the graph is no more than than the weight of the heaviest edge on a path between them, and is in fact equal to the heaviest edge on the path connecting them in the MST. It is also the case that in ultrametrics, Steiner points are never useful in a Steiner tree—i.e., the Steiner tree on some nodes is just the MST on those nodes.

The basic idea of our algorithm is to cluster nodes into components, each containing clients of total probability mass $\Theta(1/\lambda)$. On the one hand, we can think of each component as a "supernodet" which, since it has $\Theta(1/\lambda)$ probability mass, becomes active with probability $\Theta(1/\lambda)$. Corollary 6.1 lets us conclude that purchasing a minimum spanning tree on the supernodes is approximately optimal. This creates at least one "hub" in each supernode, connected to the root. But once hubs are built, since the probability mass within each superclient is not *too* large, we can show that there is no real benefit in prepurchasing edges *inside* any super-client to connect vertices to hubs.

As a first step toward demonstrating these ideas, suppose that the MST is a star with the root at the center, and that each vertex has activation probability less than $1/\lambda$. We prove that purchasing nothing in advance is approximately optimal.

Suppose first that the optimum solution purchases some non-MST edge (u, v) in advance. Vertices u and v are connected to the root by MST edges of cost c_u and c_v respectively. Suppose that instead of buying edge (u, v) we buy the two edges connecting u and v to the root. This can only help our second stage purchases: u and v are still connected to each other, through the root. From the definition of the MST, the weight w of the edge connecting u and v is no less than c_u or c_v; thus connecting both directly to the root costs at most $2w$. We can apply this replacement rule to every prepurchased non-MST edge. It follows that at cost twice the optimum, we get a prepurchase with the same second-stage cost as the optimum, made up entirely of edges connected directly to the root.

Given this prepurchase, consider the second stage. Again by the MST property, any vertex that becomes active can connect directly to the root more cheaply than it can connect to any other vertex. Thus, in the postpurchase phase as well, only edges incident to the root are purchased. Now, however, we can invoke the Threshold Property. Since only edges incident to the root are purchased, each such edge is only "useful" if its non-root endpoint becomes active. We assumed that each endpoint activation probability was less than $1/\lambda$. Thus, by the Threshold Property, it makes more sense not to buy that edge in advance.

In order to generalize this argument, we develop a clustering algorithm that builds a particular MST with a special "starlike" structure. This algorithm is very specific to ultrametrics. Any ultrametric space has a *bottleneck cut*: a partition of the points into two sides such that the distance between any two points on opposite sides is the same and is no less than the distance between any two points on the same side. The most obvious proof of this fact uses the MST. Consider the heaviest edge in the MST. Removing it produces the bottleneck cut, as can be seen from the fact that the distance between two points is equal to the maximum-weight edge on the MST path connecting those points.

One can use this bottleneck cut in a recursive MST algorithm for ultrametrics: find the bottleneck cut, recursively find MSTs of the two sides, and then connect them with an *arbitrary* edge across the bottleneck cut (as all have the same length). We will specialize this generic algorithm for our purposes. For the purposes of the algorithm, we define the *probability mass* of a set of vertices to be the probability that some vertex in the set becomes active. Our specialization produces a minimum spanning tree in which some of the vertices have been labelled "hubs." It has the following properties:

1. The hubs form a connected subgraph of the MST.

2. The total probability mass of vertices hanging off each hub (i.e., whose paths through the MST reach that hub before any other) is *at least* $1/\lambda$.

3. The total probability mass of *each* subtree hanging off a hub is *at most* $1/\lambda$.

The type of MST we wish to construct is shown in Figure 2.

LEMMA 6.2. *Algorithm* CLUSTER *produces an MST with the specified properties.*

Proof. When the algorithm is called on a graph with probability mass less than $1/\lambda$, it returns an MST with no hubs, vacuously fulfilling the properties.

If the algorithm is called on a graph of mass exceeding $1/\lambda$, it recurses on the two sides of the bottleneck cut. Each side returns without a hub if and only if it is *small*—i.e., has probability mass less than $1/\lambda$.

If both sides are small, we create one hub at the join point of the two sides' MSTs. Since removing this

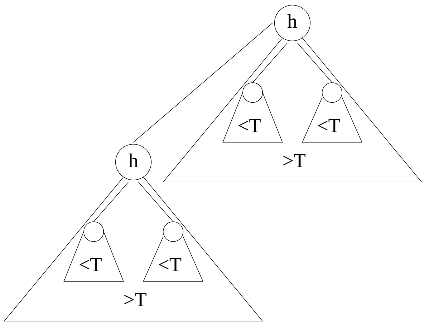

Figure 2: A hub tree. Nodes labelled h are hubs. $T = 1/\lambda$ is the upper bound for individual tree mass and lower bound for total mass hanging off a hub.

CLUSTER(G)
Input: ultrametric space G.

if G has probability mass less than $1/\lambda$ **then**
  return an MST of G with no hubs
else
  Compute the bottleneck cut (X, Y) of G
  CLUSTER(X)
  CLUSTER(Y)
  if neither X nor Y has a hub **then**
    connect X and Y with any edge e
    make one endpoint of e a hub
  else
    Connect X and Y with some edge, using hubs in X and Y as the endpoints if they exist

Figure 3: Clustering procedure.

join point separates the two MSTs (at least—it may further subdivide one side), no "hanging off" subtree is too large. However, in total at least $1/\lambda$ probability mass hangs off the hub (as that is the reason the hub was created).

If both sides are large, both returned MSTs will have hubs, so we can connect a hub on each side, again fulfilling the properties (as this attachment does not change the set of nodes hanging off any hub).

If one side is large and the other small, then the small side gets attached to (hangs off of) a hub on the large side. But since the attached side is small, hanging it off the hub does not violate the specified properties.

LEMMA 6.3. *The cost of the piece of the MST connecting the hubs is at most* OPT.

Proof. Imagine contracting all vertices hanging off each hub into their hub creating a "supernode". This is done by contracting MST edges, so the MST of the contracted graph consists of the uncontracted edges of the original MST. But these are precisely the MST edges connecting up the hubs (and the root). In this contracted graph, the probability that some vertex in a supernode becomes active is (by the definition of probability mass) at least $1/\lambda$. Thus, by Corollary 6.1, purchasing the (remaining) MST edges costs at most OPT. But these are precisely the edges connecting the hubs to the root.

It follows that by paying OPT in advance, we can prepurchase edges that connect all the representatives (including the root). Equivalently, we can contract all the representatives into the root. Since what we contracted was a connected portion of the MST, the MST of the contracted graph is just the remaining MST of the original graph. But Property 3 above tells us the form of this remaining MST: it will consist of a collection of subtrees hanging off the (contracted) root, each subtree of size at most $1/\lambda$. This is essentially the "starlike" structure we want to reduce to.

LEMMA 6.4. *In the starlike contracted MST just described, it is within a factor of 2 of optimal to buy nothing in advance.*

Proof. The analysis runs as for the star above: pre- and post-purchased intra-cluster edges can be replaced by edges connecting nodes directly to the root. Once we have done so, since each subtree of the MST has weight less than $1/\lambda$, none of the prepurchased edges can be used with probability greater than $1/\lambda$, so none is worth buying in advance.

Combining the above arguments, we see that by spending at most OPT, we can reduce to a graph in which purchasing nothing in advance is within a factor of 2 of optimum for that graph. Since that graph's optimum costs at most OPT, we see that purchasing nothing gives a solution of expected cost $2 \cdot \mathsf{OPT}$. Combining, we find that our solution has cost at most $3 \cdot \mathsf{OPT}$.

6.4 General Metrics. Finally, we remark that if edge costs form a tree metric, we can solve the problem optimally in polynomial time. In a tree metric minimum cost Steiner tree on any subset of nodes is simply a spanning tree on that subset. Thus, for each edge we can compute the exact probability of using that edge in the Steiner tree over a random set of active clients. In particular for a given edge e, we have $\Pr[e \text{ used}] = 1 - \prod_{i \in U_e}(1 - p_i)$, where U_e is the set of clients whose tree path to the root contains e. The edge should be purchased in advance if and only if the probability of it being used is at least $1/\lambda$.

Since any metric can be embedded into a tree metric with distances approximated by a factor of $O(\log n)$ in expectation [2], we obtain the following result:

THEOREM 6.2. *There is an $O(\log n)$ approximation for the metric Steiner tree network predesign problem.*

7 Conclusions and Open Problems

In this paper we presented a novel "preplanning" framework that allows to study the time-information tradeoff in solving problems with uncertainty in the inputs. We have examined a number of (generally $\mathcal{NP}$-hard) combinatorial optimization problems in which it makes to postulate a probability distribution over possible instances and to specify a portion of the solution in advance, and developed algorithms for computing approximately optimal pre- and post-sampling parts of a solution.

We leave open a number of questions concerning the problems we have considered. Another interesting open question is the integrality of the linear program for the min-cost flow with preplanning. If it could be in fact shown that there exists an optimal solution that pre-installs only integral amounts of capacity, then the next natural question to ask is whether we can solve the problem using a purely combinatorial algorithm.

We could also extend our framework to other combinatorial optimization problems. One natural problem to consider is facility location. In the preplanning version of the problem, given a probability distribution on the demand of each client, we would like to determine which facilities should be opened in advance, provided that one can add more facilities after the exact demands have been determined, albeit at a higher price. The goal is to minimize the overall expected cost plus the facility opening cost. Recently, it was brought to our attention that Gupta et al [6] came up with a constant factor approximation for the scenario based version of this problem in which the number of possible later scenarios is polynomail. They also designed a constant factor approximation for the Steiner network predesign problem [6].

We can also easily formulate a number of stochastic scheduling problems in the context of our framework. Taking job duration times to be probabilistically distributed, we may ask how many machines should be reserved in advance in order to complete all jobs by some deadline, or how much processing time to reserve in advance (with an option of extending it later) given a fixed number of machines.

References

[1] J.R. Birge. Stochastic programming and applications. *INFORMS Journal on Computing*, 9(2):111–133, spring 1997.

[2] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proc. ACM Symposium on Theory of Computing*, 2003.

[3] M.R. Garey and D.S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness.* W. H. Freeman and Company, 1979.

[4] A. Gupta, A. Kumar, M. Pal, and T. Roughgarden. Approximation via cost-sharing: a simple approximation algorithm for the multicommodity rent-or-buy problem. In *Proc. 44th Annual Symposium on Foundations of Computer Science*, Cambridge, MA, 2003.

[5] A. Gupta, A. Kumar, and T. Roughgarden. Simpler and better approximation algorithms for network design. In *Proc. ACM Symposium on Theory of Computing*, pages 365–372, 2003.

[6] A. Gupta, R. Ravi, and A. Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. Working paper, 2004.

[7] David R. Karger, Philip N. Klein, and Robert E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *Journal of the ACM*, 42(2):321–328, March 1995.

[8] J. Kleinberg, Y. Rabani, and É. Tardos. Allocating bandwidth for bursty connections. *SIAM J. Computing*, 30(1):191–217, 2000.

[9] A. Kumar, A. Gupta, and T. Roughgarden. A constant factor approximation algorithm for the multicommodity rent-or-buy problem. In *Proc. 43d Annual Symposium on Foundations of Computer Science*, pages 333–342, 2002.

[10] D. Mitra and Q. Wang. Stochastic traffic engineering, with applications to network revenue management. In *Proceedings of IEEE INFOCOM 2003*, 2003.

[11] N.V. Sahinidis. Optimization under uncertainty: state-of-the-art and opportunities. University of Illinois,Urbana, February 2003.

[12] H. Soroush and P. B. Mirchandani. The stochastic multicommodity flow problem. *Networks*, 20:121–155, 1990.

[13] C. Swamy and A. Kumar. Primal-dual algorithms for connected facility location problems. In *Proc. 5th APPROX*, volume 2462 of *LNCS*, pages 256–269, 2002.

[14] M. H. van der Vlerk. Stochastic programming bibliography. World Wide Web, `http://mally.eco.rug.nl/spbib.html`, 1996-2003.