Distributions on level-sets with applications to approximation algorithms

Aravind Srinivasan
Bell Laboratories, Lucent Technologies
600–700 Mountain Avenue
Murray Hill, NJ 07974-0636, USA

Abstract

We consider a family of distributions on fixed-weight vectors in $\{0,1\}^t$; these distributions enjoy certain negative correlation *properties and also satisfy pre-specified conditions on their marginal distributions. We show the existence of such families, and present a linear-time algorithm to sample from them. This yields improved approximation algorithms for the following problems: (a) low-congestion multi-path routing; (b) maximum coverage versions of set cover; (c) partial vertex cover problems for bounded-degree graphs; and (d) the Group Steiner Tree problem. For (a) and (b), the improvement is in the approximation ratio; for (c), we show how to speedup existing approximation algorithms while preserving the best-known approximation ratio; we also improve the approximation ratio for certain families of instances of unbounded degree. For (d), we derive an approximation algorithm whose approximation guarantee is at least as good as what is known; our algorithm is shown to have a better approximation guarantee for the worst known input families for existing algorithms.*

1. Introduction

We present a new family $\mathcal{F}_t$ of distributions on $\{0,1\}^t$, and develop a linear-time algorithm to generate a random sample from such distributions. This yields improved approximation algorithms for various NP-hard problems. Our starting point is the following problem. Given $p_1, p_2, \ldots, p_t \in [0,1]$ such that $\ell = \sum_i p_i$ is an integer, how do we randomly round the p_i to $X_1, X_2, \ldots, X_t \in \{0,1\}$ such that: **(i)** $\sum_i X_i = \ell$ with probability one, and **(ii)** for any given $a_1, a_2, \ldots, a_t \geq 0$, there is a Chernoff-type bound on the probability of $\sum_i a_i X_i$ deviating much from $\sum_i a_i p_i$? A concrete example of this is provided by application (a) below, where we need to simultaneously solve several instances of the following problem: given t paths in a graph with respective "preferences" $p_1, p_2, \ldots, p_t$, choose $\ell = \sum_i p_i$ paths "judiciously" in a sense captured by the above large-deviation bound requirement. Independently rounding the p_i will satisfy (ii); however, if we must simultaneously solve, say, k such instances, then the probability of satisfying all requirements of the form "$\sum_i X_i = \ell$" can be as small as $2^{-\Theta(k)}$ if we conduct independent rounding. Our family $\mathcal{F}_t$ provides a solution to this problem, and has other useful features–alluded to just before we describe application (b) below–which lead to certain improved approximation algorithms.

Recall that the *weight* of a vector $x \in \{0,1\}^t$ is the number of ones in it. Let $W_k(t)$ denote the set of elements of $\{0,1\}^t$ with weight equal to k, and $[s]$ denote the set $\{1,2,\ldots,s\}$. Consider any sequence $P = (p_1, p_2, \ldots, p_t)$ of t reals such that each p_i lies in $[0,1]$, and such that $\sum_i p_i$ is an *integer*. Each such P defines a distribution $D(t;P)$ that is a member of our family $\mathcal{F}_t$; $D(t;P)$ is defined to be any distribution[1] on $\{0,1\}^t$ such that if $\ell \doteq \sum_i p_i$ and $(X_1, X_2, \ldots, X_t)$ denotes a vector sampled from $D(t;P)$, then the following properties hold:

(A1) $\forall i, \Pr[X_i = 1] = p_i$;

(A2) $\Pr[|\{i : X_i = 1\}| = \ell] = 1$ (i.e., $D(t;P)$ is distributed on $W_\ell(t)$), and

(A3) the following "negative correlation" properties hold for all $S \subseteq [t]$:

$$\Pr[\bigwedge_{i\in S}(X_i = 0)] \leq \prod_{i\in S}\Pr[X_i = 0]; \quad (1)$$

$$\Pr[\bigwedge_{i\in S}(X_i = 1)] \leq \prod_{i\in S}\Pr[X_i = 1]. \quad (2)$$

From (A1), we see that $\mathbf{E}[|\{i : X_i = 1\}|] = \ell$; (A2) requires that the value $|\{i : X_i = 1\}|$ actually be *deterministic*. Also, although the X_i's are not independent, (A3) helps show that any non-negative linear combination of $X_1, X_2, \ldots, X_t$ is sharply concentrated around its mean. Informally, (A3) is the following type of negative correlation property: if $X_{i_1} = X_{i_2} = \cdots = X_{i_s} = b \in \{0,1\}$,

[1] If there is more than one such distribution, we choose one arbitrarily.

then the conditional probability of $X_{i_{s+1}}$ equaling b cannot be "too large". One may expect that (A2), which fixes the number of X_i that can equal b, will imply a negative correlation property such as (A3). However, we show a large class of distributions on $\{0,1\}^t$ that satisfy (A1) and (A2), but not (A3). Thus, extra care is needed in guaranteeing (A3).

We are not aware of any earlier proof of existence of distributions such as $D(t;P)$. We show this existence algorithmically: given P, we present a linear-time algorithm to generate a random sample from $D(t;P)$; the algorithm can be easily converted to a linear-work RNC algorithm running in $O(\log t)$ time. [Several natural candidates for such an algorithm do not work. For instance, consider generating the X_i independently with $\Pr[X_i = 1] = p_i$, and taking the distribution of $(X_1, X_2, \ldots, X_t)$ conditional on the event "$|\{i : X_i = 1\}| = \ell$". This can be seen to violate (A1) for $P = (0.75, 0.75, 0.5)$. See [27] for the relationship of such conditioning to some negative correlation–type results.] If $p_i = \ell/t$ for all i, then one choice for $D(t;P)$ is the hypergeometric distribution (sampling ℓ elements from $[t]$ *without* replacement). This, as well as the case where $\ell = 1$, are two instances of our problem (i.e., of showing that $D(t;P)$ exists, and sampling efficiently from it) with known solutions. Our sampling algorithm, when combined with some other new ideas, leads to a new type of randomized rounding scheme for linear programming (LP) relaxations of NP-hard problems, leading to the following applications.

(a). Low-congestion Multi-path Routing. Here, we are given a graph $G = (V, E)$ with a capacity $c_f > 0$ for each edge f, along with k pairs of vertices (s_i, t_i). For each $i \in [k]$, we are also given: (i) a demand $\rho_i > 0$, (ii) a collection $\mathcal{P}_i$ of (s_i, t_i)–paths, and (iii) an integer $1 \leq \ell_i \leq |\mathcal{P}_i|$. The objective is to choose ℓ_i paths from $\mathcal{P}_i$ for each i, in order to minimize the *relative congestion*: the maximum, over all edges f, of $(1/c_f)$ times the total demand of the chosen paths that pass through f. (We make the usual *balance assumption* [17]: if edge f lies in a path $P \in \mathcal{P}_i$, then $c_f \geq \rho_i$.) The case where $\ell_i = 1$ for all i is a classical problem, and is studied in [23, 17]. Our problem with arbitrary ℓ_i, in addition to its intrinsic interest, is motivated by the following optical networking problem. Given their high data rates (Gigabits/second and Terabits/second), a key requirement in optical networks is fast *restoration* from node/edge failures [18, 25, 10, 24, 9]. Many restoration strategies have been studied/deployed. A popular scheme that guarantees full recovery from single node/edge failures is variously called *1+1 Dedicated Protection*, *1+1 Restoration*, etc.: the same signal is transmitted over an active route and a *disjoint* backup route, hence being robust to single node/edge failures [10, 24, 9]. The obvious extension to protect against multiple node/edge failures has also been studied. To generate such paths in practice, min-cost max-flow is used to generate a large number of disjoint paths for each vertex pair, and a "suitable" subset of these paths is chosen in some heuristic way to minimize the congestion of the routing. Our formulation provides a rigorous approach to selecting these subsets.

We present an approximation algorithm for our problem with the same approximation ratio as for the case where $\ell_i = 1$ for all i [23, 17]; see (5) and (6). (In the above optical routing application, each $\mathcal{P}_i$ is a set of node/edge-disjoint paths; this does not seem to provide any improvement to the approximability, in our setting as well as in that of [23, 17].) Briefly, here is how distributions such as $D(t;P)$ help. We solve a natural LP relaxation of the problem. For each $i \in [k]$, this gives a vector $r_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,t_i})$ of weights $p_{i,j} \in [0,1]$ for the paths in $\mathcal{P}_i$, where $t_i = |\mathcal{P}_i|$; we have $\sum_j p_{i,j} = \ell_i$ for each i. Independently for each i, we sample from $D(t_i; r_i)$, and select the paths that are chosen by this process. By (A2), we have with probability 1 that ℓ_i paths are chosen for each i: if we had sampled the paths independently, then the probability of this happening can be as small as $2^{-\Theta(k)}$. Also, (A3) guarantees that we get a Chernoff-type bound on the probability of getting a "large" relative congestion. We do not know of any other methods that deliver the approximations we get.

Many randomized rounding schemes, defined say by binary random variables $X_1, X_2, \ldots, X_t$, can be analyzed such that it suffices to have "small" values for several non-negative linear combinations of terms of the form $\Pr[\bigwedge_{i \in S}(X_i = b)]$, where $b \in \{0,1\}$; see, e.g., (9), (13). Property (A3) shows that sampling from distributions such as $D(t;P)$ gives as good results in such cases, as does sampling the X_i independently, say. Moreover, sampling from $D(t;P)$ gives us extra properties such as (A2). In particular, one result in application (c) below exploits the "probability one" aspect of (A2) as follows: all relevant terms in an analysis become *deterministic*, except for a random variable W. W takes values in a poly-width integer range, and we aim to keep it "high"; $\mathbf{E}[W] \geq w$, but W may have poor lower-tail behavior, i.e., $\Pr[W \ll w]$ may be "large". However, since W takes values in a poly-width integer range and since all other relevant terms are deterministic, we can repeat our random process polynomially many times and get $W \geq \lfloor w \rfloor$ with high probability. This would be difficult with other related techniques which do not have an analog of (A2) [and hence cannot keep all other relevant terms deterministic]. These and other ideas lead to the following applications.

(b). Maximum Coverage Versions of Set Cover. A natural variant of set cover that is much-studied especially from a facility location viewpoint [26, 19, 3, 5, 28, 15, 4, 13, 16], is as follows. We are given the set $X = [n]$ with a weight $w_i \geq 0$ for each $i \in [n]$; also given is a family $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of subsets of X, with a cost c_j for each S_j. Given a budget L, the problem is to select a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ of total cost at most L, in order to maximize the total weight of the elements i covered by $\mathcal{S}'$. Define $\alpha_s = 1 - (1 - 1/s)^s$; as s increases from 1, α_s decreases, and tends to $1 - 1/e$ as $s \to \infty$. For the unit-cost case where $c_j = 1$ for all j, an α_L–approximation algorithm follows from the work of [20, 8, 28, 15, 13]. (From a theoretical perspective, this is essentially an $(1 - 1/e)$–approximation, since we can assume that L is "large": the problem can be solved exactly in $(n + m)^{O(1)} \cdot m^L$ time by enumeration.) For the general case with arbitrary c_j, the first constant-factor approximation algorithm is shown in [16]: the approximation ratio is $1 - 1/e \sim 0.632$, and is best-possible unless $NP \subseteq DTIME[n^{O(\log\log n)}]$ [16]. We improve these results as follows, for instances where each i has at most s sets S_j containing it. For the unit-cost case, we get an α_s–approximation; with further work, we get an $(\alpha_s - \epsilon)$–approximation for the case of general costs, where $\epsilon > 0$ is any given constant. An important special case is when $s = 2$: we get the variant of vertex cover where, given a cost for each vertex and a weight for each edge, we wish to cover a maximum-weight subset of the edges subject to a budget on the total cost of the vertices chosen [13]. Motivated by the recent interest in multi-criteria optimization, we also present constant-factor approximations for certain "multiple budget" versions of the problem.

(c). Partial Vertex Cover. A problem complementary to the above variant of vertex cover is the k-vertex cover problem: given an undirected graph $G = (V, E)$ and an integer $k \leq |E|$, we wish to select a minimum number of vertices in V such that at least k edges have at least one end-point selected. 2–approximation algorithms for this problem have been presented in [6, 14, 2]. As in the case of vertex cover, it is of interest to determine families of graphs for which $(2 - \Omega(1))$–approximations are possible. It has been shown recently in [11] that this is possible for graphs of bounded maximum degree d: a $(2 - \Omega(1/d))$–approximation algorithm running in time $n^{O(d^5)}$ is developed in [11], where $n = |V|$. We show how to reduce the running time to a polynomial that is independent of d; we also get a a $(2 - \Omega(1))$–approximation for any family of (G, k)–pairs for which the optimal objective function value is $\Omega(k)$.

(d). The Group Steiner Tree problem. This problem generalizes the classical Steiner tree problem [22]; its restriction to trees is a generalization of set cover. Given an undirected graph $G = (V, E)$, subsets $S_1, S_2, \ldots, S_k$ of V, and a cost for each edge, we need to construct a minimum-cost tree in G that intersects each S_i. We present an approximation algorithm whose approximation guarantee is at least as good as what is known [12]; also, our algorithm has a better approximation guarantee for the worst known input families for the algorithm of [12]. Please see §3.4.

We start by showing how to sample from distributions such as $D(t; P)$ in §2. The four applications discussed above are then considered in §3.

2. The sampling algorithm

We will call a finite sequence $r = (s_1, s_2, \ldots)$ of reals *valid* if $s_i \in [0, 1]$ for all i; for an integer a, we will define r to be *a-valid* iff r is valid and $\sum_i s_i = a$. Our problem is to generate a sample $(X_1, X_2, \ldots, X_t)$ from $D(t; P)$ for some given ℓ-valid vector P, where $\ell \geq 1$.

2.1. The basic recipe

Call X_i *fixed* if, at some point, we round it to 0 or 1. Our basic idea is to repeat the following step: fix some yet-unfixed variable(s) in some probabilistic manner, and suitably alter, for the yet-unfixed variables, their probabilities of being 1. One way of doing this is as follows. We construct an $(\ell - 1)$-valid vector $Q' = (q'_2, q'_3, \ldots, q'_t)$ and an ℓ-valid vector $Q = (q_2, q_3, \ldots, q_t)$ such that $p_i = p_1 q'_i + (1 - p_1) q_i$ for $i = 2, 3, \ldots, t$. Independently of all random choices made so far, fix X_1 at 1 with probability p_1, and at 0 with probability $1 - p_1$. If $X_1 = 0$, we recursively sample $(X_2, X_3, \ldots, X_t)$ from $D(t - 1; Q)$; else if $X_1 = 1$, $(X_2, X_3, \ldots, X_t)$ is recursively sampled from $D(t - 1; Q')$. It is easy to prove by induction that this basic scheme satisfies (A1) and (A2). Unfortunately, if we are not careful in choosing Q and Q', (A3) can be violated. For instance, if $p_1 \leq \min\{p_i, 1 - p_i\}$ for all $i \geq 2$, it can be shown that for *any* $(\ell - 1)$-valid Q', there is a feasible choice for Q above. However, (A3) dictates that $q'_i \leq p_i$ for each i. Thus, (A1) and (A2) do not always imply (A3); such counterexamples can be shown even if the condition "$p_1 \leq \min\{p_i, 1 - p_i\}$ for all $i \geq 2$" does not hold.

Our solution, presented in §2.2, is somewhat different; in particular, when we fix a variable, we will alter the probability of being 1 for *at most one other* variable.

2.2. Algorithm and analysis

We start by describing a procedure called `simplify`(X, Y, α, β), which is the main subroutine of our algorithm. The purpose of this procedure is to randomly fix one of the variables X and Y. More formally, α and β, both of which lie in $[0, 1]$, are input parameters to the procedure; random variables $X, Y \in \{0, 1\}$ are output parameters. [Even if the random variables X and Y have some existing semantics, these are disregarded by the procedure; as will be seen below, the output distributions of X and Y are (random) functions of only α and β, and do not depend on how X and Y were defined before we ran the procedure.] The procedure *probabilistically* sets the distributions of X and Y. Therefore, the output values of $\Pr[X = 1]$ and $\Pr[Y = 1]$ are themselves random variables, and so we can speak of $\mathbf{E}[\Pr[X = 1]]$ and $\mathbf{E}[\Pr[Y = 1]]$. The two main properties that must hold are:

(B1) $\mathbf{E}[\Pr[X = 1]] = \alpha$ and $\mathbf{E}[\Pr[Y = 1]] = \beta$;

(B2) at least one of X and Y gets fixed.

The procedure simplify(X, Y, α, β) is as follows. If $\alpha = \beta = 0$, then X and Y are both fixed at 0; else if $\alpha = \beta = 1$, then X and Y are both fixed at 1. Otherwise, there are three cases.

Case I: $0 < \alpha + \beta < 1$**.** In this case, with probability $\alpha/(\alpha+\beta)$ we fix variable Y at 0 and set $\Pr[X = 1] = \alpha+\beta$; with the remaining probability of $\beta/(\alpha+\beta)$, we fix X at 0 and set $\Pr[Y = 1] = \alpha + \beta$.

Case II: $\alpha + \beta = 1$**.** Here, we fix both variables. With probability α we fix variable Y at 0 and variable X at 1; with probability β we fix X at 0 and Y at 1.

Case III: $1 < \alpha + \beta < 2$**.** Here, with probability $(1-\beta)/(2-\alpha-\beta)$ we fix X at 1 and set $\Pr[Y = 1] = \alpha+\beta-1$; with the remaining probability of $(1-\alpha)/(2-\alpha-\beta)$ we fix Y at 1 and set $\Pr[X = 1] = \alpha + \beta - 1$.

It is easy to check that (B1) and (B2) are satisfied by the above description of simplify. Before specifying our algorithm, we need one more definition. Given a set L of labeled nodes, define a *pairing tree* of L to be any tree whose leaf-set is L, and in which each internal node has exactly two children. A *short pairing tree* of L is any pairing tree of L constructed as follows. Let U denote the current set of nodes that have no parent (initially, $U = L$). If $|U| = 1$, we are done; else if $|U| \geq 2$, group U into $\lfloor |U|/2 \rfloor$ pairs plus at most one more element, in an arbitrary way. For each of the $\lfloor |U|/2 \rfloor$ pairs, create a new node, and make the two elements in the pair be children of this new node. Note that after this step, the set of nodes with no parent has size $\lceil |U|/2 \rceil$. Thus, if we repeat until $|U| = 1$, we get that any short pairing tree of L has height $\lceil \log_2 |L| \rceil$.

We now come to our main algorithm. To facilitate ease of discussion later, we present it in the context of sampling a given sub-sequence of $(X_1, X_2, \ldots, X_t)$. More precisely, let $Q = (q_1, q_2, \ldots, q_k)$ be an r-valid vector for some integer r. Suppose we want to generate a random sample $(X_{i_1}, X_{i_2}, \ldots, X_{i_k})$ from $D(k; Q)$, where the i_j are distinct elements of the set $[t]$. If $k \leq 1$, the problem is trivial, so suppose $k \geq 2$. Start with the leaf-set $L = \{(i_1, q_1), (i_2, q_2), \ldots, (i_k, q_k)\}$, and construct an arbitrary pairing tree T of L. Informally, our algorithm is as follows. The leaf labeled (i_j, q_j) corresponds to random variable X_{i_j}. Each internal node u at height 1 runs simplify on the random variables represented by its two children; one of these random variables gets fixed, and the other is sent to u's parent. In general, each internal node receives one random variable each from its two children; it fixes one of these, and sends the other to its parent. Formally, values propogate up the tree and the random variables $X_{i_1}, X_{i_2}, \ldots, X_{i_k}$ get defined as follows. Initially, each leaf of T sends its label (of the form (i_j, q_j)) to its parent. In general, suppose an internal node u gets the pairs (i_a, α) and (i_b, β) from its children. Node u then runs simplify$(X_{i_a}, X_{i_b}, \alpha, \beta)$. If this call to simplify fixes X_{i_a} at some $\psi \in \{0, 1\}$ and sets $\Pr[X_{i_b} = 1] = \kappa$ for some κ (possibly $\kappa \in \{0, 1\}$), then X_{i_a} gets fixed at ψ, and the pair (i_b, κ) gets sent up to u's parent. Otherwise if this call to simplify sets $\Pr[X_{i_a} = 1] = \kappa$ for some $\kappa \in (0, 1)$ and fixes X_{i_b} at ψ, then X_{i_b} gets fixed at ψ, and the pair (i_a, κ) gets sent up to u's parent. The random choices made by the simplify operations at the different nodes of T are all independent. It is easy to see that after any sequence of simplify operations, the sum over $j = 1, 2, \ldots, k$ of the current probability of X_{i_j} being 1, equals r. Thus, when a call simplify$(X_{i_a}, X_{i_b}, \alpha, \beta)$ is finally made at the root of T, we are guaranteed that $\alpha + \beta \in \{0, 1, 2\}$, and hence that both X_{i_a} and X_{i_b} get fixed at the root.

To use the above for our problem of sampling $(X_1, X_2, \ldots, X_t)$ from $D(t; P)$, we use the above with the label-set $L = \{(1, p_1), (2, p_2), \ldots, (t, p_t)\}$, and use any *short* pairing tree T of L. It is easy to see that the above is a linear-time randomized algorithm. It is also easily implemented as a $t/\log t$ processor, $O(\log t)$ time RNC algorithm on an EREW PRAM, since T has height $\lceil \log_2 t \rceil$ and since simplify operations on nodes at the same level of T can be done in parallel.

Given a set of labels $L = \{(i_1, q_1), (i_2, q_2), \ldots, (i_k, q_k)\}$ and a pairing tree T of L as above, let $\mathcal{D}(L, T)$ denote the distribution on $(X_{i_1}, X_{i_2}, \ldots, X_{i_k})$ induced by our above algorithm.

Theorem 2.1 *Let k be a positive integer. Suppose we are given any set of labels $L = \{(i_1, q_1), (i_2, q_2), \ldots, (i_k, q_k)\}$, where $Q = (q_1, q_2, \ldots, q_k)$ is an r-valid vector for some integer r. Let T be any pairing tree of L. Then, $\mathcal{D}(L, T)$ is a valid choice for the distribution $D(k; Q)$.*

Proof. We start with some definitions and a useful observation. Suppose $k \geq 2$ and that (i_a, q_a) and (i_b, q_b) are two leaves of T that have a common parent, say u. (Since T is a pairing tree, there must exist two leaves with a common parent.) For $x \in \{a, b\}$, define $T[x, y]$ to be the tree obtained by deleting the nodes (i_a, q_a) and (i_b, q_b) from T, and giving the node u (which is now a leaf) the label (i_x, y). Note that $T[x, y]$ is a pairing tree for the leaf-set $L[x, y] \doteq (L - \{(i_a, q_a), (i_b, q_b)\}) \cup \{(i_x, y)\}$.

For $x \in \{a, b\}$, letting z denote the element of $\{a, b\}$ that is *not* equal to x, define $X[x]$ to be the sequence $(X_{i_1}, X_{i_2}, \ldots, X_{i_k})$ *without* the element X_{i_z}. A simple and useful observation is that our sampling from $\mathcal{D}(L, T)$ can be viewed as follows. Run $\texttt{simplify}(X_{i_a}, X_{i_b}, q_a, q_b)$. Suppose this call to `simplify` fixes X_{i_z} at $\psi \in \{0, 1\}$ and sets $\Pr[X_{i_x} = 1] = y$ for some y, where $x \in \{a, b\}$ and z denotes the element of $\{a, b\}$ that is *not* equal to x. (If both X_{i_a} and X_{i_b} get fixed, choose x arbitrarily.) Then, fix $X_{i_z} = \psi$, and sample $X[x]$ recursively from $\mathcal{D}(L[x, y], T[x, y])$. It is easy to see that this is a valid way of viewing $\mathcal{D}(L, T)$. Whatever x and y are, $L[x, y]$ has length one less than L, and this will lead us to a proof by induction on $|L|$, i.e., on k, as follows.

The base case $k = 1$ is straightforward. Assuming the theorem to hold for $k - 1$, let us establish it now for $k \geq 2$: i.e., we will show that $\mathcal{D}(L, T)$ is a valid choice for $D(k; Q)$. For (A3), we will only prove (1); the proof of (2) is identical. The proofs of (A1) and (A2) are even simpler: they follow the same inductive process as below, and follow easily using (B1) and (B2). These simpler proofs are omitted from this extended abstract.

To prove (1) for $D(k; Q)$, we need to show the following. Let $S \subseteq [k]$ be arbitrary. Then, we want to show that if $(X_{i_1}, X_{i_2}, \ldots, X_{i_k})$ is sampled from $\mathcal{D}(L, T)$, then

$$\Pr[\bigwedge_{j \in S} (X_{i_j} = 0)] \leq \prod_{j \in S} (1 - q_j). \qquad (3)$$

Recall the definitions relating to (i_a, q_a), (i_b, q_b), $T[x, y]$ etc. from the first paragraph of this proof. We have three cases: $|\{a, b\} \cap S|$ being 0, 1, or 2. Suppose both a and b lie in S. Consider the `simplify` process run at the parent node u of (i_a, q_a) and (i_b, q_b). If $q_a + q_b \geq 1$, then at least one of X_{i_a} and X_{i_b} gets fixed at 1 by this call to `simplify`, and so the l.h.s. of (3) will be zero. So suppose $q_a + q_b < 1$; we start with the interesting subcase where $q_a + q_b > 0$. Then the above-seen randomized reduction from $\mathcal{D}(L, T)$ to $\mathcal{D}(L[x, y], T[x, y])$ and the induction hypothesis about $\mathcal{D}(L[x, y], T[x, y])$ show that $\Pr[\bigwedge_{j \in S} (X_{i_j} = 0)]$ is at most

$$\frac{q_a}{q_a + q_b} \cdot (1 - (q_a + q_b)) \cdot [\prod_{j \in S:\ j \notin \{a, b\}} (1 - q_j)] +$$
$$\frac{q_b}{q_a + q_b} \cdot (1 - (q_a + q_b)) \cdot [\prod_{j \in S:\ j \notin \{a, b\}} (1 - q_j)];$$

the two terms in this sum relate to which of X_{i_a} and X_{i_b} gets fixed at 0 by the call to `simplify`. Thus,

$$\begin{aligned} \Pr[\bigwedge_{j \in S} (X_{i_j} = 0)] &\leq \frac{1 - (q_a + q_b)}{(1 - q_a)(1 - q_b)} \cdot \prod_{j \in S} (1 - q_j) \\ &\leq \prod_{j \in S} (1 - q_j), \end{aligned}$$

proving (3). Finally, the case where $q_a = q_b = 0$ yields a similar, even easier proof of (3).

Next, suppose $|\{a, b\} \cap S| = 1$: say $a \in S$ but $b \notin S$. Here we observe that with some probability γ, the call to `simplify` at u fixes X_{i_a} at 0, and with probability $1 - \gamma$, the call re-sets the probability of $X_{i_a} = 1$ to some value κ. The relevant property here which follows from (B1), is that

$$\gamma + (1 - \gamma) \cdot (1 - \kappa) = 1 - q_a. \qquad (4)$$

Now, our reduction from $\mathcal{D}(L, T)$ to $\mathcal{D}(L[x, y], T[x, y])$ and the induction hypothesis about $\mathcal{D}(L[x, y], T[x, y])$ show that $\Pr[\bigwedge_{j \in S} (X_{i_j} = 0)]$ is at most

$$\gamma \cdot [\prod_{j \in S:\ j \neq a} (1 - q_j)] +$$
$$(1 - \gamma) \cdot (1 - \kappa) \cdot [\prod_{j \in S:\ j \neq a} (1 - q_j)].$$

We now use (4) to see that this sum equals $\prod_{j \in S} (1 - q_j)$.

The case where neither a nor b lies in S is even simpler: whatever happens in the `simplify` process at u, we get (3) from the induction hypothesis. □

3. Applications to approximation algorithms

Before presenting our applications, we first invoke the results of [21] which show that the negative correlation property (A3) has the following interesting large-deviations consequence. Suppose $P = (p_1, p_2, \ldots, p_t)$ is ℓ-valid for some integer ℓ, and that we sample a vector $(X_1, X_2, \ldots, X_t)$ from $D(t; P)$. Then, the following theorem states that for any given sequence $a_1, a_2, \ldots, a_t \in [0, 1]$, the random variable $\sum_i a_i X_i$ is sharply concentrated around its mean; in fact, the tail bounds on $\sum_i a_i X_i$ are at least as good as any bound obtainable by a Chernoff-type approach [21].

Theorem 3.1 ([21]) *Let $a_1, a_2, \ldots, a_t$ be reals in $[0, 1]$, and $X_1, X_2, \ldots, X_t$ be random variables taking values in $\{0, 1\}$.*
(i) Suppose (2) holds for all $S \subseteq [t]$; also suppose $\mathbf{E}[\sum_i a_i X_i] \leq \mu_1$. Then, for any $\delta \geq 0$,

$$\Pr[\sum_i a_i X_i \geq \mu_1 (1 + \delta)] \leq \left(\frac{e^\delta}{(1 + \delta)^{1 + \delta}} \right)^{\mu_1}.$$

(ii) Suppose (1) holds for all $S \subseteq [t]$; also suppose $\mathbf{E}[\sum_i a_i X_i] \geq \mu_2$. Then, for any $\delta \in [0, 1]$,

$$\Pr[\sum_i a_i X_i \leq \mu_2 (1 - \delta)] \leq e^{-\mu_2 \delta^2 / 2}.$$

We present the applications (a), (b), (c), and (d) of §1 in the next four subsections. The reader is referred to §1 for the definitions of the applications; we do not re-define them. We denote input graphs throughout by $G = (V, E)$, with $|V| = n$ and $|E| = m$.

3.1. Low-congestion Multi-path Routing

There is a natural LP relaxation for the problem, as described a few sentences below. Letting C^* be the optimum of this relaxation, we get the same bounds on the integral relative congestion C as are obtained in [23, 17] for the case where $\ell_i = 1$ for all i:

$$C \leq O\left(\frac{\log m}{\log(2\log m/C^*)}\right) \text{ if } C^* \leq \log m; \quad (5)$$

$$C \leq C^* + O(\sqrt{C^* \log m}) \text{ if } C^* > \log m. \quad (6)$$

Suppose $\mathcal{P}_i = \{P_{i,1}, P_{i,2}, \ldots\}$. There is a natural integer linear programming formulation for the problem, with a variable $z_{i,j}$ for each (i,j): $z_{i,j} = 1$ if $P_{i,j}$ is chosen, and is 0 otherwise. The LP relaxation lets each $z_{i,j}$ lie in $[0,1]$. Let the variable C denote the optimal fractional relative congestion. We have the constraints:

$$\forall i,\ \sum_j z_{i,j} = \ell_i; \quad \forall f \in E,\ \sum_{(i,j):\ f \in P_{i,j}} \rho_i z_{i,j} \leq c_f C. \quad (7)$$

Let $|\mathcal{P}_i| = t_i$. We solve the LP, and suppose $r_i = (z^*_{i,j} : j \in [t_i])$ is the vector of values for the $z_{i,j}$ in this optimal solution, with C^* being the optimal fractional relative congestion. Independently for each $i \in [k]$, we sample from $D(t_i; r_i)$, and select the paths that are chosen (i.e., rounded to 1) by this process. By the first family of constraints in (7) and (A2), we have with probability 1 that ℓ_i paths are chosen for each i. Next, by (A1) and the second family of constraints in (7), the expected relative congestion on any given edge f is at most C^*. Suppose the constants implicit in the $O(\cdot)$ notation of (5) and (6) are large enough. Then, (A3) and Theorem 3.1(i) can be used to show that for any given edge f, the probability that it gets relative congestion more than C is at most $1/(2m)$. Adding over all edges, we get a relative congestion of at most C, with probability at least $1/2$.

We are not aware of any other approach that will yield (5) and (6). In particular, if we try to bin-pack the $z^*_{i,j}$ for each given i and round independently from each bin, the approximation ratio can be about twice our approximation ratio (e.g., if most of the $z^*_{i,j}$ are just above $1/2$, the optimal bin-packing will need roughly $2\ell_i$ bins). Note from (6) that we get $(1 + o(1))$–approximations for families of instances where C^* grows faster than $\log m$; it appears difficult to get such results from any other approach (e.g., the above bin-packing) that we are aware of.

3.2. Maximum Coverage Versions of Set Cover

We will work with a natural LP relaxation of the problem. This relaxation has variables $x_i \in [0,1]$ for $i \in [n]$, and $z_j \in [0,1]$ for $j \in [m]$; it seeks to maximize $\sum_i w_i x_i$ subject to: (i) $\sum_j c_j z_j \leq L$, and (ii) $\forall i,\ x_i \leq \sum_{j:i \in S_j} z_j$. For all of §3.2, let s denote the maximum number of sets in which an element i lies. We assume that $s \geq 2$, since the case $s = 1$ is just a knapsack problem.

Definition of D'. As in the setting of §1, suppose we have a vector of reals $P = (p_1, p_2, \ldots, p_t)$ where $p_i \in [0,1]$, but where $\ell = \sum_i p_i$ may not be an integer. Let $P' = (p_1, p_2, \ldots, p_t, \lceil \ell \rceil - \ell)$, and define $D'(t; P)$ to be the joint distribution of the first t co-ordinates of the distribution $D(t+1; P')$. Note that $D'(t; P)$ satisfies (A1) and (A3); instead of (A2), we have with probability 1 that $|\{i \in [t] : X_i = 1\}| \in \{\lfloor \ell \rfloor, \lceil \ell \rceil\}$.

We first focus on the case where all the set-costs c_j are 1. Here we can assume that L is an integer (the problem remains unchanged if we replace L by $\lfloor L \rfloor$). Let $\{x^*_i, z^*_j\}$ be the set of values in an optimal LP solution, and let y^* denote the optimal LP value. Our randomized rounding scheme here is to sample $(z_1, z_2, \ldots, z_m)$ from $D'(m; (z^*_1, z^*_2, \ldots, z^*_m))$ and then to define x_i naturally to be $\max\{z_j : i \in S_j\}$. The properties of D' ensure that at most L sets are chosen. What is the expected objective function value? Recall that $\alpha_s = 1 - (1 - 1/s)^s$. We next prove that the following inequality holds:

$$\begin{aligned}
\Pr[x_i = 1] &= 1 - \Pr[\bigwedge_{j:i \in S_j} (z_j = 0)] \\
&\geq 1 - \prod_{j:i \in S_j} \Pr[z_j = 0] \qquad (8) \\
&= 1 - \prod_{j:i \in S_j} (1 - z^*_j) \\
&\geq \alpha_s \cdot x^*_i. \qquad (9)
\end{aligned}$$

Inequality (8) follows from (A3). Since $x^*_i = \min\{1, \sum_{j:i \in S_j} z_j\}$ and since $|\{j : i \in S_j\}| \leq s$, it can be verified that $\prod_{j:i \in S_j} (1 - z^*_j) \leq (1 - x^*_i/s)^s$; since $x^*_i \in [0,1]$, elementary calculus shows that $1 - (1 - x^*_i/s)^s \geq \alpha_s \cdot x^*_i$. Thus we get (9); so, the expected value of the objective function is at least $\alpha_s \cdot y^*$. This improved approximation for the unit-cost case can be generalized further. Suppose we are given several pairwise-disjoint families $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_u$, where each $\mathcal{S}_j$ is a collection of subsets of X and with its own integer budget L_j; we aim to choose a collection of sets from $\mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \mathcal{S}_u$ (at unit cost for each set) to maximize the total weight of the elements covered, while respecting the budget of each $\mathcal{S}_j$. Essentially the same approach carries over here: we just need to independently sample from a distribution of the form $D'(|\mathcal{S}_j|, \cdot)$ for each j. Here again, we get an α_s-approximation: in particular, a constant-factor approximation.

We now move on to the general-cost case of the problem, which needs more work; we start with a useful lemma.

Lemma 3.1 *Suppose we are given: (i) a sequence $P = (p_1, p_2, \ldots, p_t)$ with $p_i \in [0,1]$, (ii) a sequence $\vec{a} = (a_1, a_2, \ldots, a_t)$ with $a_i \geq 0$, and (iii) a real $\psi > 0$. Then, there is an efficient random process $g(\vec{a}, P, \psi)$ to generate a vector of binary random variables $(X_1, X_2, \ldots, X_t)$ for which (A1) and (A3) holds, and such that with probability one, $\sum_i a_i X_i \leq (1 + 1/\psi) \cdot (\max_i a_i) + (1+\psi) \cdot (\sum_i a_i p_i)$.*

Proof. Let $A = \max_i a_i$. For $s \geq 0$, define an index set $I_s \subseteq [t]$ to be $\{i : (1+\psi)^{-s-1} A < a_i \leq (1+\psi)^{-s} A\}$. Let P_s be the restriction of P to I_s. Independently run $D'(|I_s|; P_s)$ for each s with a non-empty I_s, and re-arrange the outputs in the "correct" order $1, 2, \ldots, t$, to get a vector $(X_1, X_2, \ldots, X_t)$. It is simple to check that (A1) and (A3) hold. Also, for each s, we have with probability 1 that

$$|\{i \in [I_s] : X_i = 1\}| \leq 1 + \sum_{i \in I_s} p_i.$$

Thus,

$$\sum_{i \in I_s} a_i X_i \leq (1+\psi)^{-s} A + (1+\psi) \cdot \sum_{i \in I_s} a_i p_i.$$

Adding over all s completes the proof. □

Returning to the general-cost case, recall that we are given a constant $0 < \epsilon < \alpha_s$, and want an $(\alpha_s - \epsilon)$–approximation for the given instance. Define $\delta = \epsilon^3/[(1+\epsilon) \cdot (1 + \epsilon + \epsilon^2)]$, and $\beta = (\alpha_s - \epsilon)/(\alpha_s - \alpha_s \epsilon)$. Since ϵ is a constant in $(0, \alpha_s)$ and $1 - 1/e < \alpha_s \leq 3/4$, $\beta = \beta(\epsilon)$ is a constant in $(0,1)$. Define λ to be the smallest positive integer such that $\beta^\lambda \leq 1 - \alpha_s$; we can verify that $\lambda = O(1/\epsilon)$. We can assume that $0 < c_j \leq L$ for all j. Given our collection of sets $\mathcal{S}$, let $\mathcal{S}(a,b]$ denote the subcollection of $\mathcal{S}$ that has sets with cost in the range $(a,b]$. Fix some optimal solution $\mathcal{S}' \subseteq \mathcal{S}$. Let the actual budget used by $\mathcal{S}'$ be $L_0 \leq L$, and the total weight of the elements covered by $\mathcal{S}'$—i.e., the optimal solution value for the given instance—be W_0. (We do not know these values, but will shortly "guess" these and some related values.) We now define some real values $L_{-1}, L_1, L_2, \ldots, L_\lambda$ and $W_1, W_2, \ldots, W_\lambda$, as well as pairwise-disjoint families $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_\lambda$ (with $\mathcal{S}_i \subseteq \mathcal{S}$ for each i); all of these quantities are completely determined by $\mathcal{S}'$. Define $L_{-1} = L_0/\delta$, and $\mathcal{S}_0 = \emptyset$. For $0 \leq i \leq \lambda - 1$, L_{i+1}, $\mathcal{S}_{i+1}$, and W_{i+1} are defined inductively as follows. $\mathcal{S}_{i+1} = \mathcal{S}' \cap \mathcal{S}(L_i \delta, L_{i-1}\delta]$; L_{i+1} is L_0 minus the total cost of the sets in $\mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_{i+1}$, and W_{i+1} is W_0 minus the total weight of the elements covered by $\mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_{i+1}$.

We guess the values of W_0 and L_0. We also guess the families $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_\lambda$; this way, all the L_i, W_i and $\mathcal{S}_i$ above are seen to be determined. We now argue that this guessing can be implemented efficiently. First of all, W_0 and L_0 can be guessed to an arbitrary desired accuracy by a two-dimensional bisection search. Second, we claim that $|\mathcal{S}_{i+1}| < 1/\delta$ for $0 \leq i \leq \lambda - 1$; if true, this implies that there are most $1/\delta^\lambda = O(1)$ choices for the tuple $(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_\lambda)$, which can all be enumerated in $m^{O(1)}$ time. To see why $|\mathcal{S}_{i+1}| < 1/\delta$, let $\mathcal{F} = \mathcal{S}' - \bigcup_{i \leq \lambda} \mathcal{S}_i$. We can see from the above definition of L_i that the total cost of the sets in $\mathcal{S}_{i+1}, \mathcal{S}_{i+2}, \ldots, \mathcal{S}_\lambda, \mathcal{F}$ is L_i. However, each set in $\mathcal{S}_{i+1}$ has cost more than $L_i \delta$; so $|\mathcal{S}_{i+1}| < 1/\delta$.

Theorem 3.2 *Suppose $W_\ell \geq \beta W_{\ell-1}$ for some ℓ, $1 \leq \ell \leq \lambda$. Then, we can efficiently compute a solution of value at least $(\alpha_s - \epsilon) W_0$.*

Proof Sketch. Construct a modified instance of our problem in which: (i) the collection of sets is $\mathcal{S}(0, L_{\ell-1}\delta]$, and (ii) each element $i \in [n]$ gets a *modified* weight: all elements covered by $\bigcup_{i \leq \ell} \mathcal{S}_i$ have a modified weight of 0, and all other elements have a modified weight that is the same as their original weight w_i. We will construct a collection $\mathcal{T} \subseteq \mathcal{S}(0, L_{\ell-1}\delta]$ such that:
(R1): the total cost of $\mathcal{T}$ is at most $L_{\ell-1}$, and
(R2): the elements covered by $\mathcal{T}$ have total *modified* weight at least $(\alpha_s - \epsilon) W_{\ell-1}$.

Then, the (disjoint) union of $\mathcal{T}$ and $\bigcup_{i \leq \ell-1} \mathcal{S}_i$ has: (a) a total cost of at most $L_{\ell-1} + L_0 - L_{\ell-1} = L_0$, and (b) covers a set of elements of total weight at least $(\alpha_s - \epsilon) W_{\ell-1} + (W_0 - W_{\ell-1}) \geq (\alpha_s - \epsilon) W_0$. In this calculation (b), "weight" refers to the original weight w_i of each element. The sum in this calculation is justified because the elements covered by $\bigcup_{i \leq \ell-1} \mathcal{S}_i$ are not double-counted: their weights were taken to be zero in the requirement (R2). We now show how to construct such a $\mathcal{T}$. We solve the LP relaxation (sketched earlier in §3.2) for our modified instance with budget L_ℓ, and solve it to optimality. Let $\{x_i^*, z_j^*\}$ be the set of values in an optimal LP solution, and let W_ℓ^* denote the optimal LP value. We claim that $W_\ell^* \geq W_\ell$; this is true because the collection $\mathcal{S}' \cap \mathcal{S}(0, L_{\ell-1}\delta]$ has a total cost of L_ℓ, and has an objective function value of W_ℓ for our modified instance. For notational simplicity, suppose $\mathcal{S}(0, L_{\ell-1}\delta] = \{S_1, S_2, \ldots, S_r\}$ for some r. Recall the process $g(\cdot,\cdot,\cdot)$ guaranteed by Lemma 3.1. For a parameter $\gamma \in (0,1]$ to be chosen later, we run $g((c_1, c_2, \ldots, c_r), (\gamma z_1^*, \gamma z_2^*, \ldots, \gamma z_r^*), \epsilon)$. By essentially the same reasoning as for (9), we now get

$$\begin{aligned} \Pr[x_i = 1] &\geq 1 - \prod_{j \in [r]:\ i \in S_j} (1 - \gamma z_j^*) \\ &\geq 1 - (1 - \gamma x_i^*/s)^s \\ &\geq \gamma \cdot \alpha_s \cdot x_i^*. \end{aligned} \tag{10}$$

Thus, the expected modified weight of the elements chosen is at least

$$\begin{aligned}\gamma \cdot \alpha_s \cdot W_\ell^* &\geq \gamma \cdot \alpha_s \cdot W_\ell \\ &\geq \gamma \cdot \alpha_s \cdot \beta \cdot W_{\ell-1} \\ &= \gamma \cdot (\alpha_s - \epsilon) \cdot W_{\ell-1}/(1-\epsilon), \quad (11)\end{aligned}$$

where the second inequality follows from the hypothesis "$W_\ell \geq \beta W_{\ell-1}$" of the theorem. By Lemma 3.1, the total cost of the chosen family of sets $\mathcal{T} \subseteq \mathcal{S}(0, L_{\ell-1}\delta]$ is at most

$$(1 + 1/\epsilon) \cdot L_{\ell-1} \cdot \delta + (1+\epsilon) \cdot \gamma \cdot L_\ell. \quad (12)$$

If $L_\ell < L_{\ell-1}/(1+\epsilon+\epsilon^2)$ we use $\gamma = 1$; else we use $\gamma = 1 - \epsilon$. In either case, the bounds (12) and (11), along with the bound $L_\ell \leq L_{\ell-1}$, can be used to show that (R1) and (R2) are fulfilled. □

The remaining case is where $W_\ell < \beta W_{\ell-1}$ for all ℓ, $1 \leq \ell \leq \lambda$: i.e., $W_\lambda < \beta^\lambda W_0 \leq (1-\alpha_s)W_0$. This means that the total weight of the elements covered by $\mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_\lambda$ is at least $\alpha_s W_0$. In this case, we just choose $\mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_\lambda$ as our solution. This completes the description of our algorithm.

We are currently studying distributions with some negative correlation properties that are stronger than (A3), in order to see if we can develop certain *multi-criteria* approximations for such maximum coverage versions of set cover.

3.3. Partial Vertex Cover

We first present the relevant approximation algorithm of [11], and show how to modify it. An LP relaxation of the problem has variables $x_j, z_{i,j} \in [0,1]$, for each vertex $j \in V$ and each edge $z_{i,j} \in E$. It is simple to verify that the following is a valid relaxation: minimize $\sum_{j\in[n]} x_j$ subject to: (i) $x_i + x_j \geq z_{i,j}$ for all $(i,j) \in E$; and (ii) $\sum_{(i,j)\in E} z_{i,j} \geq k$. The rounding approach of [11] is as follows. Let $\{x_i^*\}$, $\{z_{i,j}^*\}$ denote an optimal LP solution, and let $\lambda = 2(1-\epsilon)$, where $\epsilon \in [0,1)$. Let $S_1 = \{v_j : x_j^* \geq 1/\lambda\}$, and $S_2 = V - S_1$. Choose all vertices in S_1. Next, independently for each $j \in S_2$, round x_j to 0 or 1: the probability of rounding to 1 is λx_j^*. Thus, we have chosen the vertices of $S_1 \cup S_2'$ for some $S_2' \subseteq S_2$. Let W denote the number of covered edges now. If $W < k$, we augment our solution: choose any $k - W$ uncovered edges and cover them by arbitrarily choosing one end-point for each. A case analysis and suitable choice of ϵ is then shown to yield a $(2 - \Omega(1/d))$–approximation in $n^{O(d^5)}$ time for graphs of *bounded* maximum degree d, in [11]. Recall the definition of the distribution D' from §3.2. Now, as in [11], we choose all vertices in S_1. Then, instead of independently choosing the vertices in S_2, we set $P = (\lambda x_j^* : j \in S_2)$, and sample from $D'(|S_2|; P)$ to choose a subset of S_2 (i.e., the elements of S_2 that get rounded to 1 in the sampling). Let y^* be the LP optimum. With probability one, the number of vertices chosen so far is at most $\lceil y^*\lambda \rceil \leq 2(1-\epsilon)y^* + 1$. Finally, in the augment step, we add at most $\max\{k - W, 0\}$ vertices. Let X_i be the binary random variable indicating whether vertex i was chosen or not *before* our augmentation. ($\Pr[X_i = 1] = 1$ if $i \in S_1$.) Note that $\mathbf{E}[W]$ equals

$$\sum_{(i,j)\in E} (\Pr[X_i = 1] + \Pr[X_j = 1] - \Pr[X_i = X_j = 1]). \quad (13)$$

Now, it is shown in [11] that for the rounding scheme there, $\mathbf{E}[W] \geq k(1-\epsilon^2)$. Usage of (13), (A1) and (A3) shows that $\mathbf{E}[W] \geq k(1-\epsilon^2)$ for our scheme also! Furthermore, we have deterministically that: the number of edges covered is k, and the number of vertices chosen by our sampling from $D'(|S_2|; P)$ is at most $2(1-\epsilon)y^* + 1$. Finally, since W takes values in $\{0, 1, \ldots, m\}$, the bound $\mathbf{E}[W] \geq k(1-\epsilon^2)$ can be used to show that $\Pr[W \leq k(1-\epsilon^2) - 1] \leq 1 - \Omega(1/m)$. Thus, if we repeat the above scheme Cm times for a large enough constant C, with high probability at least one of the iterations will give us a solution of value at most $2(1-\epsilon)y^* + 2 + k\epsilon^2$. Choosing $\epsilon = y^*/k$, we get the following theorem. (We had earlier restricted ϵ to be smaller than 1, but it is possible that $y^* = k$. However, if $y^* = k$, then choosing one end-point each of any k edges is an optimal solution; so we can assume that $y^* < k$.)

Theorem 3.3 *Our scheme constructs a feasible solution of value at most $y^*(2 - y^*/k) + 2$ with high probability. In particular, we get a $(2 - \Omega(1))$–approximation for families of instances where $y^* \geq \Omega(k)$: e.g., for bounded-degree graphs.*

3.4. Group Steiner Trees

Recall the problem definition from §1. The work of [12] shows that by invoking the approach of [1], we can reduce the problem to the case where the given graph is a rooted tree, and where we need to pick the root; this reduction involves a multiplicative $O(\log n \log\log n)$ loss in the approximation ratio. By presenting a novel randomized rounding scheme for the problem on trees, the work of [12] developed the first polylogarithmic approximation algorithm for the problem. Letting $N = \max_i |S_i|$, the approximation ratio of [12] for trees is $O(\log N \log k)$. The $\log k$ factor is apparently necessary since the problem generalizes set cover: it has been an interesting open question whether the $\log N$ factor is necessary. We make progress on this by presenting a new rounding scheme with worst-case approximation ratio at most $O(\log N \log k)$; however, for families of instances such as one shown in [12] for which the approximation ratio of [12] is $\Theta(\log N \log k)$, our approximation ratio is $O(\log k)$.

We first sketch the rounding scheme of [12] for a tree T with root r. We may assume w.l.o.g. that all nodes in $\bigcup_i S_i$ are leaves in T. An integer linear programming formulation for the problem is as follows. We have a variable $x_f \in \{0,1\}$ for each edge f: this is 1 if f is chosen and is zero otherwise. The spanning constraints are that if we interpret x_f as the "capacity" of f, then for each $i \in [k]$, these capacities can support a unit flow from r to S_i. (The capacities *do not* need to support a unit flow simultaneously for all the S_i.) The objective function is to minimize $\sum_f c_f x_f$, where c_f is the given cost of edge f. Relaxing each x_f to lie in $[0,1]$ gives an LP relaxation.

Let $\{x^*(f) : f \in E\}$ be an optimal solution to the LP relaxation, with objective function value y^*. The rounding approach of [12] is as follows; we describe it in a level-by-level manner. Each edge f incident with the root r is independently chosen with probability $p(f) \doteq x^*(f)$; this is the rounding scheme for the edges in the first level of the tree. Suppose we have rounded the edges in the first i levels, and need to round the $(i+1)$st level. For each vertex u at a distance of i from the root that is reachable from the root in the subtree obtained by the rounding so far, and for each edge f connecting u to some child of u, edge f is chosen with probability $p(f) \doteq x^*(f)/x^*(g)$, where g is the "parent edge" of f. All these choices are made independently. It is not hard to see that the expected cost of the subtree chosen is y^*. More interestingly, it is shown in [12] that any given S_i is hit with probability at least $\Omega(1/\log N)$. Thus, if we repeat this process $a \log N \log k$ times for a sufficiently large constant a and take the union of all the subtrees chosen, we will hit all the S_i with high probability. Also, by Markov's inequality, the total cost of the final tree will be at most $(2a \log N \log k) \cdot y^*$ with probability at least $1/2$. Thus we have an $O(\log N \log k)$–approximation. However, it is also shown in [12] that there are families of examples where the basic scheme does need to be run $\Omega(\log N \log k)$ times as above, to have a reasonable probability of hitting all the S_i. We now sketch a new rounding scheme which has an approximation ratio of at most $O(\log N \log k)$, but which is an $O(\log k)$–approximation for families of potentially "difficult" examples such as the one mentioned above.

The new rounding scheme is best described as a level-by-level rounding scheme. Recall the definition of D' from §3.2. Let the probabilities $p(f)$ be the same as above. Suppose there are t edges incident on the root. Then, instead of rounding them independently as above, we round them using $D'(t; (p_1, p_2, \ldots, p_t))$, where the p_i are the values $p(f)$ for the t edges. In general, suppose we now need to round the edges in the $(i+1)$st level. For each vertex u at a distance of i from the root that is reachable from the root in the subtree rounded so far, we proceed as follows, independently of all other vertices. Suppose there are t_u edges f connecting u to some child of u; denote their probabilities (the same "$p(f) \doteq x^*(f)/x^*(g)$" as in [12]) by $p_1, p_2, \ldots, p_{t_u}$. Instead of rounding these t_u edges independently as in [12], we round them using $D'(t_u; (p_1, p_2, \ldots, p_{t_u}))$.

This is our new rounding scheme. It is easy to show using (A1) that the expected cost of the subtree chosen is y^*. More interestingly, we can show that the correlations implicit in our usage of D' only help, once again via (A3): we are able to show again that any given S_i is hit with probability at least $\Omega(1/\log N)$. Thus, our approximation ratio is again at most $O(\log N \log k)$. However, for the hard examples shown in [12], we can show that our approximation ratio is at most $O(\log k)$. (In this context, the following example is provided in [12]. Consider a binary tree, where the values x^*_f are $1/2$ in the first level, $1/4$ in the second level, and so on; we take k such disjoint trees, each reaching down separately to some S_i, and merge their roots [12].) Further details are deferred to the full version.

Acknowledgments. This work evolved over a period of time, and benefited much from discussions with several people. We thank Moses Charikar, Chandra Chekuri, Rajiv Gandhi, Sanjeev Khanna, Samir Khuller, Seffi Naor, Iraj Saniee, Peter Winkler, and Leonid Zosin for helpful discussions.

References

[1] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proc. ACM Symposium on Theory of Computing*, pages 161–168, 1998.

[2] R. Bar-Yehuda. Using homogeneous weights for approximating the partial cover problem. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 71–75, 1999.

[3] S. Belardo, J. Harrald, W. Wallace, and J. Ward. A partial covering approach to siting response resources for major maritime oil spills. *Management Science*, 30:1184–1196, 1984.

[4] O. Berman, D. Bertsimas, and R. C. Larson. Locating discretionary service facilities, II: maximizing market size, minimizing inconvenience. *Operations Research*, 43:623–632, 1995.

[5] O. Berman, R. C. Larson, and N. Fouska. Optimal location of discretionary service facilities. *Transportation Science*, 26:201–211, 1992.

[6] N. Bshouty and L. Burroughs. Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem. In *Proc. Symposium on the Theoretical Aspects of Computer Science*, pages 298–308, 1998.

[7] M. Charikar, C. Chekuri, A. Goel, and S. Guha. Rounding via trees: deterministic approximation algorithms for Group Steiner trees and k-median. In *Proc. ACM Symposium on Theory of Computing*, pages 114–123, 1998.

[8] M. Conforti and G. Cornuéjols. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete Applied Math.*, 7:257–274, 1984.

[9] R. D. Davis, K. Kumaran, G. Liu, and I. Saniee. SPIDER: a simple and flexible tool for design and provisioning of protected lightpaths in optical networks. *Bell Labs Technical Journal*, Vol. 6, January–June 2001.

[10] B. T. Doshi, S. Dravida, P. Harshavardhana, O. Hauser, and Y. Wang. Optical network design and restoration. *Bell Labs Technical Journal*, Issue on Optical Networking, Vol. 4, January-March 1999.

[11] R. Gandhi, S. Khuller, and A. Srinivasan. Approximation algorithms for partial covering problems. In *Proc. International Colloquium on Automata, Languages, and Programming*, pages 225–236, 2001.

[12] N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the Group Steiner Tree problem. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 253–259, 1998.

[13] D. S. Hochbaum. Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. In *Approximation Algorithms for NP-Hard Problems*, D. S. Hochbaum, ed., PWS Publishing Company, Boston, pages 94–143, 1997.

[14] D. S. Hochbaum. The t-vertex cover problem: Extending the half integrality framework with budget constraints. In *Proc. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 111–122, 1998.

[15] D. S. Hochbaum and A. Pathria. Analysis of the greedy approach in covering problems. Unpublished manuscript, 1994.

[16] S. Khuller, A. Moss, and J. (Seffi) Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[17] J. Kleinberg. *Approximation algorithms for disjoint paths problems*. Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, 1996.

[18] D. Logothetis and K. Trivedi. The effect of detection and restoration times for error recovery in communication networks. *Journal of Network and Systems Management*, 5:173–195, 1997.

[19] N. Megiddo, E. Zemel, and S. L. Hakimi. The maximum covering location problem. *SIAM Journal on Algebraic and Discrete Methods*, 4:253–261, 1983.

[20] G. L. Nemhauser and L. Wolsey. Maximizing submodular set functions: formulations and studies of algorithms. In *Studies on Graphs and Discrete Programming*, North-Holland, Amsterdam, pages 279–301, 1981.

[21] A. Panconesi and A. Srinivasan. Randomized distributed edge coloring via an extension of the Chernoff-Hoeffding bounds. *SIAM J. Comput.*, 26:350–368, 1997.

[22] G. Reich and P. Widmayer. Beyond Steiner's problem: a VLSI oriented generalization. In *Proc. Graph Theoretic Concepts of Computer Science*, Lecture Notes in Computer Science 411, Springer-Verlag, pages 196–210, 1990.

[23] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.

[24] SDH Frequently Asked Questions. See *http://www1.biz.biglobe.ne.jp/∼worldnet/faq/sdh.html*

[25] T. E. Stern and K. Bala. *Multiwavelength optical networks: a layered approach*. Prentice Hall, 1999.

[26] C. Toregas, R. Swain, C. Revelle, and L. Bergman. The location of emergency facilities. *Operations Research*, 19:1363–1373, 1971.

[27] W. T. Trotter and P. Winkler. Ramsey theory and sequences of random variables. *Combinatorics, Probability and Computing*, 7:221–238, 1998.

[28] R. V. Vohra and N. G. Hall. A probabilistic analysis of the maximal covering location problem. *Discrete Applied Math.*, 43:175–183, 1993.