Weighted Capacitated, Priority, and Geometric Set Cover via Improved Quasi-Uniform Sampling

Timothy M. Chan[*] Elyot Grant[†] Jochen Könemann[†] Malcolm Sharpe[†]

October 6, 2011

Abstract

The *minimum-weight set cover problem* is widely known to be $O(\log n)$-approximable, with no improvement possible in the general case. We take the approach of exploiting problem structure to achieve better results, by providing a geometry-inspired algorithm whose approximation guarantee depends solely on an instance-specific combinatorial property known as *shallow cell complexity* (SCC). Roughly speaking, a set cover instance has low SCC if any column-induced submatrix of the corresponding element-set incidence matrix has few distinct rows. By adapting and improving Varadarajan's recent *quasi-uniform random sampling method* for weighted geometric covering problems, we obtain strong approximation algorithms for a structurally rich class of weighted covering problems with low SCC. We also show how to derandomize our algorithm.

Our main result has several immediate consequences. Among them, we settle an open question of Chakrabarty et al. [8] by showing that weighted instances of the *capacitated covering* problem with underlying network structure have $O(1)$-approximations. Additionally, our improvements to Varadarajan's sampling framework yield several new results for weighted geometric set cover, hitting set, and dominating set problems. In particular, for weighted covering problems exhibiting linear (or near-linear) *union complexity*, we obtain approximability results agreeing with those known for the unweighted case. For example, we obtain a constant approximation for the *weighted disk cover* problem, improving upon the $2^{O(\log^* n)}$-approximation known prior to our work and matching the $O(1)$-approximation known for the unweighted variant.

[*]Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, tmchan@uwaterloo.ca

[†]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, {egrant,jochen,masharpe}@uwaterloo.ca. Work supported by NSERC Discovery Grant 277224.

1 Introduction

In an instance of the classical *set cover* problem, we are given a ground set X of M elements, and a family $\mathcal{S}$ of N subsets of X. Each set $S \in \mathcal{S}$ has a non-negative weight w_S, and the goal is to find a collection $\mathcal{C} \subseteq \mathcal{S}$ of sets of minimum total weight such that each element $e \in X$ is contained in at least one of the sets in $\mathcal{C}$. Equivalently, if we let A be the element–set incidence matrix, then we are looking for a solution to the following 0, 1-integer program:

$$\text{(SC)} \qquad \min\{w^T x \,:\, Ax \geq \mathbb{1},\ x \geq 0,\ x \text{ integer}\}.$$

The set cover problem is rather well understood in terms of approximability; it admits efficient $O(\log M)$ approximation algorithms and this is best possible unless NP = P [15]. Nevertheless, there are many cases where one can get better approximations by exploiting the structure of matrix A. A classical example for this is given by the case where matrix A is *totally unimodular* [26], or where A exhibits other useful combinatorial structure (e.g., sparsity [18, 22, 27]).

Another relevant example for this paper is when (SC) encodes a *geometric* covering problem. Here X usually consists of points in some fixed-dimensional Euclidean space, $\mathcal{S}$ contains geometric objects (e.g., disks, triangles, or squares), and the goal is to pick a minimum cost collection of these objects to cover all points. The study of geometric covering problems is a vibrant area in itself, and much progress has been made in the last 20 years (e.g., see [1, 6, 9, 10, 13, 11, 29, 30]).

Until recently, much of the work in this area has focused on the *unweighted* setting, where $w_S = 1$ for all sets S. A key insight here links the integrality gap of the canonical linear programming relaxation (SC-LP) of (SC) to the existence of small ϵ-*nets*; $\mathcal{C} \subseteq \mathcal{S}$ is an L/N-net if it covers all points that are at least L-deep (i.e., that are contained in at least L sets in $\mathcal{S}$). Brönnimann and Goodrich [6] (see also [13]) showed that (SC-LP) has an integrality gap of $O(g(\texttt{opt}))$ if there are L/N-nets of size $O(N/L \cdot g(N/L))$, for some function g ($\texttt{opt}$

is the optimum value of (SC-LP)). Hence, improving bounds on the size of ϵ-nets is a central theme in the area of geometric covering.

Clarkson [9] and Haussler and Welzl [17] showed that L/N-nets of size $O(N/L \cdot \log N/L)$ exist for fairly general set families, including triangles, rectangles, and disks among others. A few years ago, Clarkson and Varadarajan [11] exhibited a connection between the existence of small ϵ-nets and the combinatorial complexity of the union of the corresponding geometric objects—the so-called *union complexity*. The authors showed that if the union complexity of any n objects is $O(n \cdot h(n))$ for some function h, then L/N-nets of size $O(N/L \cdot h(N/L))$ exist [11]; a recent improvement of this bound to $O(N/L \cdot \log h(N/L))$ can be found in [1, 29][1].

Recently, Varadarajan [30] gave an elegant algorithm that extends some of the unweighted results above to the weighted setting. The algorithm produces randomized L/N-nets that contain each set in $\mathcal{S}$ with (roughly) equal probability. This property, known as *quasi-uniformity*, yields a method for obtaining good approximation for weighted set cover instances. Specifically, when given a weighted covering problem in which any n objects have union complexity $O(n \cdot h(n))$, Varadarajan's algorithm produces a quasi-uniform L/N-net containing $O(2^{O(\log^* L)} \cdot N/L \cdot \log h(N/L))$ sets (where the leading exponential factor can be dropped if $h(n) \in \omega(\log^{(j)} n)$ for some constant j.)

The results in this paper are inspired by Varadarajan's [30]. We present an improvement of his result, and also extend the scope of his technique to more general, not necessarily geometric, set cover instances. In the following, we present a purely combinatorial property for set cover instances, under which improved approximations exist.

1.1 Cell complexity: geometric and combinatorial

One of the key technical concepts used by Varadarajan [30] is the *cell complexity* of a given configuration of geometric objects. Informally, in a configuration of objects in Euclidean space, a *cell* is a maximal connected region consisting of points that all lie within precisely the same set of objects. The *depth* of a cell is the number of objects that define the cell. In his algorithm, Varadarajan uses an earlier result by Clarkson and Shor [10] that shows that geometric set cover instances with low union complexity have a small number of cells of large depth.

[1]Technically, for the result in [29] to hold, we need $h(n) \in \omega(\log^{(j)} n)$ for some constant j, where $\log^{(j)}$ is the logarithm iterated j times .

For our purposes, we shall strip away the topological details underpinning the formal geometric definition of cells, leaving us with a purely combinatorial notion of "cell" for the matrix world. We call two rows A_i and A_j of a $0,1$-matrix A *equivalent* if they contain ones in precisely the same columns. The *cells* of A are then defined to be the resulting equivalence classes, and the *depth* of a cell is the number of ones in any one of its rows. For example, the first and third row of matrix A on the right are equivalent and have depth two. There are two more cells formed by rows two and four, of depth two and one, respectively. We define the key property used in our algorithm.

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Definition 1.1. (Cell Complexity) *Let $f(n,k)$ be a function that is non-decreasing in both n and k. A binary matrix A with N columns has* shallow cell complexity *(SCC) f if for all $1 \le k \le n \le N$ and for all sub-matrices A^* of A containing exactly n columns, the number of cells of A^* of depth k or fewer is at most $f(n,k)$. A set cover instance has SCC f if and only if its element–set incidence matrix does.*

For example, general binary matrices have SCC at most $\binom{n}{k}$. Binary matrices that do not contain the submatrix $[0,1]$ have SCC at most $k+1$. A set cover matrix has SCC at most $O(n^d)$ whenever its underlying set system has VC-codimension [28] at most d (see [17]). Finally, binary *network matrices* (a subclass of totally unimodular matrices [26]) have SCC $O(n)$ as we show in Lemma 4.1. Our main result shows that set cover instances with small SCC are well approximable:

Theorem 1.1. *Let $\phi(n)$ be a non-decreasing function of n, and let $c \ge 0$ be a constant. Suppose $\mathcal{I}$ is a class of set cover instances with SCC $f(n,k) = n\phi(n)k^c$. Then there is a randomized polynomial-time $O(\max\{1, \log\phi(n)\})$-approximation algorithm for the weighted set cover problem for $\mathcal{I}$.*

The performance guarantees stated in the above theorem are *LP-relative*, and hence provide upper bounds on the integrality gap of (SC-LP). As mentioned, our algorithm follows Varadarajan's general sampling framework [30], and it therefore reduces the problem of computing an approximate solution to a weighted set cover instance to that of computing small weighted ϵ-nets for an appropriately defined set system. Just like in [30], our algorithm ensures quasi-uniformity. Varadarajan's analysis gives an $O(\log \phi(n))$-approximation when $\phi(n) \in \omega(\log^{(j)} n)$ for some constant j, but obtains only a $2^{O(\log^* n)}$-approximation in the case of $\phi(n) = O(1)$. We provide a refined analysis that eliminates the $2^{O(\log^* n)}$ factor. At the same time, our algorithm also simplifies Varadarajan's in several ways. In Section 3 we also show how our algorithm can be derandomized.

1.2 Applications

Capacitated Covering. One of the main implications of Theorem 1.1 pertains to *capacitated* (or *column-restricted*) covering problems. This class of problems naturally generalizes the standard, $0,1$ set cover problem defined before. In a typical instance, we are given the usual parameters of a set cover instance, and in addition have a supply s_S and upper bound u_S for each set $S \in \mathcal{S}$, and a demand b_e for each element $e \in X$. We then define matrix $A[s]$ by letting $A[s]_{eS} = A_{eS} \cdot s_S$ for all e and S. The goal is to solve

$$\text{(CSC)} \quad \min\{c^T x \,:\, A[s]x \geq b,\, 0 \leq x \leq u,\, x \text{ integer}\}.$$

Chakrabarty et al. [8] recently considered this class of covering problems, and asked how their approximability relates to that of their underlying standard set cover problems. In particular, can one use the structure of A to find good approximations for CSC? The authors reduce the approximability of a given capacitated covering problem to the problem of bounding the integrality gaps of the standard LP relaxations of $0,1$-covering problems from two classes.

Underlying $0,1$***-Multicover Problem*** This is simply the family of weighted set-*multi*cover instances defined by matrix A, any positive right-hand side vector b, and upper-bounds u.

Priority Covering Problem A problem in this class is obtained from the given capacitated covering instance, and *priorities* π_S and π_e for all sets $S \in \mathcal{S}$, and elements $e \in X$. A set S now covers element e if $e \in S$, and if $\pi_S \geq \pi_e$. The goal is to find a min-weight solution to the resulting set cover instance.

The main result in [8] shows that if the integrality gaps of the two families above are bounded by α and β, then there is an $O(\alpha + \beta)$ approximation for the given capacitated covering instance.

Bansal et al. [3] very recently showed that if the integrality gap bound α of the underlying multicover family is *hereditary* in the sense that it also holds for row-induced sub-systems, then the integrality gap of the LP relaxations of the corresponding priority instances is $O(\alpha \log^2 k)$, where k is the number of distinct priorities. In addition, the authors show the hereditary multicover gap is α whenever a given instance has *hereditary discrepancy* at most α.

One of the main specific questions left open in [8] concerns the case where the set-cover matrix A is a network matrix. Does the addition of capacities make the problem harder in this case? The work in [3] implies an $O((\log \log s_{max})^2)$ approximation for the capacitated set-cover problem in this case, where s_{max} is the largest supply. We improve over [3] and settle the open question in [8].

Theorem 1.2. *There is a constant-factor approximation for the capacitated covering problem whenever the underlying set-cover matrix A is a network matrix.*

Whenever A is a network matrix, the LP for the underlying multicover problem is exact, and hence we only need to focus on the class of priority problems that arise. Here we first show that $0,1$ network matrices have `SCC` $O(n)$, and that adding priorities creates cover matrices whose `SCC` is larger by a factor k. The result then follows from Theorem 1.1. Unfortunately, the same argument does not work for all totally unimodular matrices, as shown by Example 1. Note also not all matrices with `SCC` $O(n)$ are network matrices. We obtain the following more general result.

Theorem 1.3. *There is an $O(\log \log N)$-approximation for the capacitated covering problem whenever the underlying set-cover matrix has `SCC` $O(n)$.*

The proof is similar to that of Theorem 1.2, and makes the additional observation that the induced multicover instances have integrality gap at most $O(\log \log N)$, using a trick from [4, §5].

Geometric Covering. Our results refine the sampling algorithm by Varadarajan [30], and thus there are naturally numerous consequences for geometric covering problems. A standard technique by Clarkson and Shor [10] links the shallow cell complexity to union complexity: for a family of objects in a constant dimension d with constant description complexity, if the union complexity is $O(n\phi(n))$, then the shallow cell complexity is bounded by $O((n/k)\phi(n/k) \cdot k^d) \leq O(n\phi(n)k^{d-1})$. We immediately obtain the following corollary of Theorem 1.1, which matches and generalizes previous results for the unweighted case [1, 29]:

Corollary 1.1. *Let $\mathcal{I}$ be a class of geometric set cover instances where the union of n objects has complexity $O(n\phi(n))$. Then there is a randomized polynomial-time LP-based $O(\max\{1, \log \phi(n)\})$-approximation algorithm for the weighted set cover problem for $\mathcal{I}$.*

This corollary enables us to resolve several of the open questions posed in [30] pertaining to weighted geometric set cover for objects with linear or near linear union complexity. It is, for example, well-known that a collection of n disks (or pseudodisks) has union complexity $O(n)$ [21]. Corollary 1.1 thus gives us an $O(1)$ approximation for weighted disk cover, and improving the $2^{O(\log^* n)}$-approximation obtained in [30]:

Corollary 1.2. *There is a randomized polynomial-time $O(1)$-approximation algorithm for weighted geometric disk (or pseudodisk) cover in $\mathbb{R}^2$.*

Similarly, *fat triangles* are known to have union complexity $O(n \cdot 2^{\alpha(n)} \log^* n)$ by a recent result of Ezra, Aronov, and Sharir [14] (with de Berg, these authors have apparently improved the bound to $O(n \log^* n)$). Applying Varadarajan's result readily gives a $2^{O(\log^* n)}$-approximation for the weighted fat triangle cover. Corollary 1.1 immediately implies the following strengthening:

Corollary 1.3. *There is a randomized polynomial-time $O(\log \log^* n)$-approximation algorithm for weighted fat triangle cover in $\mathbb{R}^2$.*

Axis-aligned octants and unit cubes in $\mathbb{R}^3$ have linear union complexity [5]. Halfspaces in $\mathbb{R}^3$ have linear union complexity, since the union is the complement of a convex polyhedron. We thus obtain $O(1)$-approximations for octants, unit cubes, and halfspaces in $\mathbb{R}^3$. Points and halfspaces can be mapped to halfspaces and points by duality [12], and so we also get an $O(1)$-approximation for the weighted *hitting set* problem for halfspaces in $\mathbb{R}^3$. Disks in $\mathbb{R}^2$ can be mapped to halfspaces in $\mathbb{R}^3$ by a well-known lifting transformation [12], and so we get an $O(1)$-approximation for weighted hitting set for disks in the plane.

Corollary 1.4. *There is a randomized polynomial-time $O(1)$-approximation algorithm for weighted set cover for axis-aligned octants and unit cubes in $\mathbb{R}^3$, and halfspaces in $\mathbb{R}^3$, as well as for weighted hitting set for disks in $\mathbb{R}^2$.*

Recently, Gibson and Pirwani [16] have applied Varadarajan's technique to the weighted *dominating set* problem in the intersection graph of a set of disks in $\mathbb{R}^2$. We can map a disk σ with center (a, b) and radius c to a point $p_\sigma = (a, b, c)$, and a disk σ' with center (a', b') and radius c' to a region $S_{\sigma'} = \{(x, y, z) : \sqrt{(x - a')^2 + (y - b')^2} \leq z + c'\}$, so that the two disks intersect if and only if p_σ is covered by $S_{\sigma'}$. The union of the $S_{\sigma'}$'s corresponds to a planar additively weighted Voronoi diagram, which is known to have linear complexity [2]. We immediately obtain the following improvement to Gibson and Pirwani's result:

Corollary 1.5. *There is a randomized polynomial-time $O(1)$-approximation algorithm for weighted dominating set for disks in $\mathbb{R}^2$.*

2 Proof of Theorem 1.1

We say that a set cover matrix A is k-deep if all rows of A contain at least k ones. Our goal is to approximate the minimum weight set cover problem on an $M \times N^0$ set cover matrix A^0. To this aim, we first reduce this to the problem of computing a small quasi-uniform cover of a related $M/2$-deep unweighted covering instance. This reduction is standard (e.g., see [30]), and we include it here only for completeness.

Suppose we are given the weights w and the matrix A^0 having $\texttt{SCC}$ at most $f(n, k) = n\phi(n)k^c$ for a fixed $c \geq 0$. We first solve the LP relaxation (SC-LP) of the problem, and let x^* be the optimal basic solution. Standard properties of basic solutions immediately imply that the support of x^* (the set of positive entries) has size at most M. We create a set family $\mathcal{S}^*$, by including $\lfloor 2M \cdot x^*_S \rfloor$ copies of each set $S \in \mathcal{S}$ with $x^*_S \geq 1/(2M)$; *small* sets S with $x^*_S < 1/(2M)$ are not included. For each element $e \in X$ we now have

(2.1)
$$\sum_{S \,:\, x^*_S \geq \frac{1}{2M}, e \in S} \lfloor 2M \cdot x^*_S \rfloor \geq M \sum_{S \,:\, x^*_S \geq \frac{1}{2M}, e \in S} x^*_S \geq \frac{M}{2},$$

where the second inequality uses the fact that small sets supply at most a 1/2 unit of coverage for each element e. Let A^* be the set cover matrix for set family $\mathcal{S}^*$ and elements X, and assume that it has N columns. It is not hard to see that A^* has the same $\texttt{SCC}$ $f(n, k)$ as A^0, and (2.1) shows that A^* is $M/2$-deep. We henceforth let $L := M/2$ so that A^* is L-deep.

The *key step* of our method is to employ an efficient randomized algorithm that produces a covering of X in which each set in $\mathcal{S}^*$ is included with probability $O\left(\frac{\ell(N)}{L}\right)$, where $\ell(N) = \max\{1, \log \phi(N)\}$. The expected weight of the covering is then

$$O\left(\frac{\ell(N)}{L}\right) \cdot w(\mathcal{S}^*) = O(\ell(N)) \cdot w^T x^*,$$

completing the proof of Theorem 1.1. We now fill in the details for the key step.

2.1 Improved quasi-uniform sampling Here, we provide a polynomial algorithm to produce a small, quasi-uniform L/N-net of an L-deep $M \times N$ set cover matrix A^* of $\texttt{SCC}$ $f(n, k) = n\phi(n)k^c$. Throughout this section, we let $\ell(N) = \max\{1, \log \phi(N)\}$. For simplicity, we shall assume that $f(n, k)$ is known a priori, although this is unnecessary if standard binary search techniques are employed to guess c and $\ell(N)$.

The algorithm proceeds in a series of sequential phases. At the start of a phase, we are given a k-deep $m \times n$ submatrix A of A^* with `SCC` $f(n,k)$ as inherited from A^*. The eventual goal is to produce a set cover for the subproblem induced by A, but much of the work will be put off until future phases, which operate on increasingly small nested sub-matrices of A. The purpose of a single phase is simply to partition the columns of A into three categories:

- *Forced* columns: those that will definitely be taken in our set cover;
- *Rejected* columns: those that will definitely not be taken in our set cover;
- *Retained* columns: those for which the decision to force or reject will be deferred until a future phase of the algorithm.

Given such a partition, we define B to be the submatrix of A obtained by deleting all forced columns, rejected columns, and rows with a one in any forced column (rows *covered* by a forced column). The subsequent phase of the algorithm operates on B.

The algorithm we present reduces the size and depth of A by a factor of approximately $\frac{1}{2}$ during each phase. Reducing by $\frac{1}{2}$ is a somewhat arbitrary choice that we make to optimize the simplicity of our presentation. We contrast our algorithm with that of Varadarajan [30], who reduces the depth of A from k to $\log k$ during each phase, picking up an unavoidable $2^{O(\log^* n)}$ in the approximation factor when the shallow cell complexity of A is very low. We avoid this by employing a forcing scheme that is somewhat simpler than Varadarajan's original "forced addition" scheme [30].

Roughly speaking, in each phase, we randomly and independently mark each column of A with probability $\frac{1}{2} + h(N,k)$ for a carefully-chosen function $h(N,k)$. We then force some columns of A in order to cover rows that contain ones in fewer than $\frac{k}{2}$ marked columns, and retain the marked columns that remain. The exact manner in which we force will be described later; the key trick is to use low `SCC` to obtain a forcing rule in which no column is forced with high probability. Formally, our marking and forcing rules will achieve the following during each phase:

1. The output B is $\frac{k}{2}$-deep (each row of B contains at least $\frac{k}{2}$ ones).
2. Each column of A is retained with probability at most $\frac{1}{2} + h(N,k)$.
3. Each column of A is forced with probability at most k^{-2}.

We will see later that it suffices to take $h(N,k) = O\left(\sqrt{\frac{\log k + \ell(N)}{k}}\right)$. Our analysis exploits the following key property of shallow cell complexity:

Lemma 2.1. *Suppose A is k-deep, has n columns, and has shallow cell complexity $f(n,k) = n\phi(n)k^c$. Then there is a column S of A such that the number of cells of depth exactly k that contain a one in S is at most $\phi(n)k^{c+1}$.*

Proof. Let A' be the submatrix of A obtained by deleting all rows that have more than k ones and eliminating duplicate copies of rows. Since A has shallow cell complexity f, so does A' and thus there are at most $n\phi(n)k^c$ rows in A'. Each row of A' contains exactly k ones, so A' contains at most $n\phi(n)k^{c+1}$ ones. Thus the average number of ones per column in A' is $\phi(n)k^{c+1}$ and it follows that some column of A' contains at most $\phi(n)k^{c+1}$ ones. The corresponding column of A then contains at most $\phi(n)k^{c+1}$ ones in cells of A having depth exactly k.

We now use this result to develop a procedure that takes a k-deep set system $(X, \mathcal{R})$ of low shallow cell complexity and returns a highly structured partition of X into *clusters* of points, each of which lies within the common intersection of some k sets in $\mathcal{R}$. We wish to assign each cluster to be the *responsibility* of some set containing it such that no set is responsible for too many clusters. In our full algorithm, we will force sets when the clusters they are responsible for are insufficiently covered, and the fact that no column is responsible for too many clusters will enable us to obtain an upper bound on the probability that a column is forced. We first give a definition:

Definition 2.1. *In a set cover matrix A, a group of rows R is called a k-*cluster *if there exists a set C of k columns of A that each contain a one in all rows of R. In such a case, C is said to* support *R.*

For example, rows 2 and 4 of matrix A to the right form a 2-cluster supported by columns 2 and 3. The next result follows immediately:

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Proposition 2.1. *Let A be a set cover matrix and let A' be a submatrix f A. Suppose R is a cell of depth k in A' (that is, a collection of identical rows in A', each containing k ones). Then the rows in R form a k-cluster when regarded as rows of A (note that they are not necessarily identical in A).*

We now state our key lemma:

Lemma 2.2. *Suppose a set cover matrix A is k-deep, has n columns, and has shallow cell complexity $f(n,k) = n\phi(n)k^c$. Denote by X and $\mathcal{S}$ the rows and columns of A respectively. Then there exists a function $\gamma : X \to \mathcal{S}$ such that:*

- *$\gamma(x)$ is a column containing a one in row x, and*
- *for each $S \in \mathcal{S}$, the pre-image $\gamma^{-1}(S) = \{x \in X : \gamma(x) = S\}$ can be partitioned into $\phi(n)k^{c+1}$ k-clusters of A.*

Moreover, such a function γ can be computed in polynomial time.

Proof. Figure 1 shows a polynomial-time iterative procedure to assign $\gamma(x)$ for each $x \in X$.

Figure 1 An iterative method to compute γ.

```
1: Initialize $A_1 \leftarrow A$ and $i \leftarrow 1$.
2: Find a column $S_i \in \mathcal{S}$ such that at most $\phi(n)k^{c+1}$ cells of $A_i$ having depth exactly $k$ contain a one in $S_i$ (one exists by Lemma 2.1). Let $Y_i \subseteq X$ be all rows of $A_i$ that are members of the $\phi(n)k^{c+1}$ cells.
3: Set $\gamma(x) = S_i$ for each row $x \in Y_i$.
4: $A_{i+1} \leftarrow$ submatrix of $A_i$ obtained by deleting the column $S_i$ and all rows in $Y_i$ from $A_i$.
5: If $A_{i+1}$ contains no rows, $\gamma(x)$ is defined for all rows $x \in X$ and we may terminate.
6: Otherwise, increment $i$ and go back to step 2.
```

We note that after the deletions in step 4, A_{i+1} is still k-deep because rows in $X \setminus Y_i$ either contain more than k ones or contain a zero in column S_i; in either case, their depth after the deletion of column S_i cannot be less than k. Additionally, the shallow cell complexity of A is inherited by sub-matrices, so A_{i+1} always has shallow cell complexity $f(n,k) = n\phi(n)k^c$ throughout the procedure. This permits the application of Lemma 2.1 throughout the iterations of the procedure, implying that our procedure terminates and thus assigns a value to $\gamma(x)$ for each $x \in X$. Additionally, $\gamma(x)$ is always a column containing a one in row x, as we require.

Additionally, the pre-image $Y_i = \gamma^{-1}(S_i)$ is a collection of at most $\phi(n)k^{c+1}$ cells of A_i. Proposition 2.1 then implies that $\gamma^{-1}(S_i)$ can be partitioned into at most $\phi(n)k^{c+1}$ k-clusters of A, completing the proof.

Formally, we say that S is *responsible* for x whenever $\gamma(x) = S$. With our key lemma in hand, we can finally provide a formal description of a phase in Figure 2. It is clear that, after a single phase, B will be $\frac{k}{2}$-deep, as any row x of A that does not lie in $\frac{k}{2}$ marked columns is deleted when $\gamma(x)$ is forced. It follows that each phase halves the depth of A, and thus the algorithm will terminate in at most $\lceil \log L \rceil$ phases when given a set cover matrix of depth L; hence, the algorithm runs in polynomial time.

Figure 2 A phase of the random sampling procedure.

```
1: **Input:** $m$ by $n$ set cover matrix $A$ of depth $k$ with SCC $n\phi(n)k^c$
2: **if** $\log k \geq \frac{k}{12(c+3)}$ **or** $\ell(N) \geq \frac{k}{12(c+3)}$ **then**
3:     Force every column of $A$ and terminate
4: **else**
5:     Mark each column of $A$ independently with probability $\frac{1}{2} + h(N,k)$
6:     Obtain a function $\gamma$ as described in Lemma 2.2
7:     **for all** rows $x$ of $A$ **do**
8:         **if** $x$ does not contain at least $\frac{k}{2}$ ones in marked columns **then**
9:             Force $\gamma(x)$
10:        **end if**
11:    **end for**
12:    Reject the remaining columns of $A$ that have been neither forced nor marked
13: **end if**
14: Obtain matrix $B$ from $A$ by deleting forced columns, rejected columns, and rows with a one in a forced column
15: **Output:** $B$
```

We also verify that the final output does indeed form a set cover of A. Rows of A are deleted during the algorithm if and only if they contain a one in a forced column, and thus are covered. Rows that are never deleted are covered in the final phase when all remaining columns of A are forced.

To ensure that our algorithm is feasible, we must verify that the marking probability $h(N,k) + \frac{1}{2}$ is at most one. This is easy from our choice of terminating condition. During non-terminating phases of the algorithm, we have both $\log k < \frac{k}{12(c+3)}$ and $\ell(N) < \frac{k}{12(c+3)}$, and thus:

$$\begin{aligned} & \log k + \ell(N) < \frac{k}{6(c+3)} \\ \implies \quad & \frac{3(c+3)}{2}\frac{\log k + \ell(N)}{k} < \frac{1}{4} \\ \implies \quad & \sqrt{\frac{3(c+3)}{2}\frac{\log k + \ell(N)}{k}} + \frac{1}{2} < 1. \end{aligned}$$

The final technical challenge remaining is to bound the probability that a column is forced at some point during the algorithm. We begin by bounding the probability that a single column is forced during a single phase:

Claim 1. *In a single non-terminating phase of the algorithm, each column S of A is forced with probability at most k^{-2}.*

Proof. We recall by Lemma 2.2 that the pre-image $\gamma^{-1}(S) = \{x \in X : \gamma(x) = S\}$ can be partitioned into $\phi(n)k^{c+1}$ k-clusters of A. Fix such a partition as obtained in Lemma 2.2. With this partition under consideration, suppose a row x of A has $\gamma(x) = S$ and lies in a k-cluster R supported by a set of columns C. A row in R can only cause S to be forced if fewer than $\frac{k}{2}$ columns in C are marked. We shall bound the probability of this happening in order to bound the probability that any row in R is insufficiently covered by the marked columns. We let Z be a random variable indicating the number of columns of C that are marked. Define

$$\mu = E[Z] = k\left[\frac{1}{2} + h(N,k)\right].$$

Then applying the Chernoff bound yields:

$$\begin{aligned}
&\Pr\left[\text{Fewer than } \frac{k}{2} \text{ columns in } C \text{ marked}\right]\\
\leq\ &\Pr\left[Z \leq \frac{k}{2}\right]\\
=\ &\Pr\left[Z \leq \left(1 - \frac{kh(N,k)}{\frac{k}{2} + kh(N,k)}\right)\mu\right]\\
\leq\ &\operatorname{Exp}\left(-\frac{1}{3}\left(\frac{kh(N,k)}{\frac{k}{2} + kh(N,k)}\right)^2\left(\frac{k}{2} + kh(N,k)\right)\right)\\
\leq\ &\operatorname{Exp}\left(-\frac{2}{3}kh(N,k)^2\right).
\end{aligned}$$

Since $\gamma^{-1}(S)$ can be partitioned into $\phi(n)k^{c+1}$ k-clusters of A, by the union bound, the probability that S is forced during an individual phase is at most

$$\phi(n)k^{c+1}\operatorname{Exp}\left(-\frac{2}{3}kh(N,k)^2\right).$$

Taking $h(N,k) = \sqrt{\frac{3}{2}\frac{(c+3)\log k + \ell(N)}{k}}$ and recalling that $n \leq N$ during all phases of the algorithm, this is at most

$$\phi(n)k^{c+1}\operatorname{Exp}\left(-(c+3)\log k - \log(\phi(n))\right) = k^{-2}.$$

The previous claim essentially proves that our additional sampling probability $h(N,k)$ is *big enough* to cause the forcing probability in each phase to be relatively low. Our next claim shows us that $h(N,k)$ is still *small enough* for the probability of a column surviving t phases of sampling to decay exponentially in t. The function $h(N,k)$ is indeed quite finely tuned in order to exhibit both of these features.

Claim 2. *After $t \geq 1$ phases of our algorithm, the probability of any given column of the original M by N matrix A^* still remaining is $\frac{O(1)}{2^t}$.*

Proof. This is clearly true after the terminating condition has occurred. Before that happens, A is still at least $\frac{L}{2^i}$-deep after i phases, so the probability that a row is retained during phase $i+1$ is at most $\frac{1}{2} + h(N, \frac{L}{2^i})$. Multiplying over all phases yields an upper bound of

$$\begin{aligned}
P_t &= \prod_{i=0}^{t-1}\left(\frac{1}{2} + h\left(N, \frac{L}{2^i}\right)\right)\\
&= \prod_{i=0}^{t-1}\left(\frac{1}{2} + O(1)\sqrt{\frac{(2^i)(\log L - i + \ell(N))}{L}}\right).
\end{aligned}$$

We recall that at a phase in which the initial depth is k, we must always have $\ell(N) < \frac{k}{12(c+3)}$, otherwise we terminate. If we have not yet terminated in phase t, we may take $k = \frac{L}{2^t}$, the depth after t phases, and obtain $\ell(N) < \frac{L}{12(c+3)2^t}$. Combining with the above yields:

$$\begin{aligned}
P_t &\leq \prod_{i=0}^{t-1}\left(\frac{1}{2} + O(1)\sqrt{\frac{(2^i)(\log L - i)}{L} + \frac{O(1)}{2^{t-i}}}\right)\\
&= \frac{1}{2^t}\prod_{j=\log L - t}^{\log L}\left(1 + O(1)\sqrt{\frac{j}{2^j} + 2^{\log L - t - j}}\right)\\
&\leq \frac{O(1)}{2^t}\operatorname{Exp}\left(\sum_{j=\log L - t}^{\log L}\left(\sqrt{\frac{j}{2^j} + 2^{\log L - t - j}}\right)\right)\\
&\leq \frac{O(1)}{2^t}\operatorname{Exp}\left(O(1)\right) = \frac{O(1)}{2^t}.
\end{aligned}$$

The final inequality follows from the fact that

$$\int_0^\infty \sqrt{\frac{x}{2^x}}dx = \frac{\sqrt{2\pi}}{(\log(2))^{3/2}} \approx 4.34362 \in O(1).$$

Claim 3. *A column is forced during the final phase of the algorithm with probability $O\left(\frac{\ell(N)}{L}\right)$, where $\ell(N) = \max\{1, \log\phi(N)\}$.*

Proof. The algorithm terminates when either $\log k \geq \frac{k}{12(c+3)}$ or $\ell(N) \geq \frac{k}{12(c+3)}$. In particular, during the terminating phase of the algorithm in which every column is forced, we must have $\log k + \ell(N) \geq \frac{k}{12(c+3)}$ from which it follows that $k \leq O(1)\ell(N)$. Thus at least $O(1)\log\left(\frac{L}{\ell(N)}\right)$ phases are required to reach the terminating condition. Consequently, taking $t = O(1)\log\left(\frac{L}{\ell(N)}\right)$ in the previous claim yields the desired result.

We finally prove that our algorithm returns a quasi-uniform cover:

CLAIM 4. *Throughout all phases of the algorithm, a given column is forced with probability at most $O\left(\frac{\ell(n)}{L}\right)$, where $\ell(n) = \max\{1, \log \phi(n)\}$.*

Proof. By the previous claim, it is sufficient to obtain the stated bound for non-terminating phases of the algorithm. After phase i, the depth is $\frac{L}{2^i}$ and thus the probability of a column both remaining after i phases and being forced during phase $i+1$ is

$$\frac{O(1)}{2^i}\left(\frac{L}{2^i}\right)^{-2} = \frac{O(1)}{L}\frac{2^i}{L}.$$

Summing over all phases yields a bound of at most

$$\frac{O(1)}{L}\sum_{i=0}^{\log L}\frac{2^i}{L} \leq \frac{O(1)}{L}(2) = \frac{O(1)}{L}.$$

This completes the proof that our algorithm computes a quasi-uniform cover.

3 Remark on derandomization

In this subsection, we note that our algorithm can be derandomized, and thus all the results in this paper hold deterministically. To our knowledge, derandomization of Varadarajan's technique [30] has not been observed before.

The idea is to replace the need for totally independent random choices with b-wise independent random choices for some constant b. Specifically, at the beginning of the algorithm, we generate a b-wise independent sequence of N integers $X_1, \ldots, X_N$ which are uniformly distributed in the range $[0, U)$ for a sufficiently large universe size U (for example, $U = \Theta(N)$ would suffice). The subset of columns that are retained during phase i is supposedly a uniform sample of the set of original columns, with a certain sampling probability P_i. To produce this sample, we take the subset of all columns S such that $X_S \in [0, P_i U)$. By a well known construction [20, 23], a b-wise independent sequence $X_1, \ldots, X_N \in [0, U)$ can be generated from b truly random integers in $[0, U)$ for a given prime $U \geq N$. Deterministically, we can try all $O(U^b)$ possible choices for these b integers and thus simulate the randomized algorithm by brute force in polynomial time.

It remains to show that this version of the algorithm using $X_1, \ldots, X_N$ still achieves the same expected bound on the weight of the computed set cover. For the analysis, we use the following alternative to the Chernoff bound (e.g., see [25, Theorem 4(III)]): if Z is a sum of b-wise independent 0-1 random variables with $E[Z] = \mu$ for an even b, then

$$\Pr[|Z - \mu| \geq t] \leq \left(\frac{b \cdot \max\{b, \mu\}}{e^{2/3}t^2}\right)^{b/2}.$$

In the proof of Claim 1, our application has $\mu = k \cdot (1/2 + h(N, k))$ and $t = kh(N, k)$. Thus,

$$\Pr[Z \leq k/2] \leq O\left(\frac{1}{kh(N, k)^2}\right)^{b/2}.$$

Taking $h(N, k) = 1/k^{1/3}$, for example, we can bound the right-hand side by $O(1/k^{b/6})$. Choosing b sufficiently large (as a function of c), we see that Claim 1 remains true. Claim 2 also remains true for our new choice of $h(N, k)$, by suitably replacing square roots with cube roots in the calculations. The final expected weight bound on the set cover follows by linearity of expectation, which does not require independence.

4 Proof of Theorem 1.2

In the *tree cover* problem, we are given an undirected graph $G = (V, E)$, and a spanning tree $T \subseteq E$; each of the edges $e \in E \setminus T$ has an associated cost $c_e \geq 0$. The goal is to select a minimum-cost set of edges $C \subseteq E \setminus T$ such that each $e \in T$ is on the fundamental cycle of at least one edge in C. The incidence-matrices of the resulting covering problem are the so-called *tree-fundamental-cycle* (TFC) matrices.

DEFINITION 4.1. *[7, 24, §6.4] Let $G = (V, E)$ be a connected graph and let $T \subseteq E$ be a tree spanning G. The TFC matrix for T and G is the $T \times (E \setminus T)$ matrix A where A_{ef} is 1 if e lies on the fundamental cycle of f in T and 0 otherwise.*

Note that 0,1-network matrices are TFC matrices, but not all TFC matrices are network matrices. The motivation for defining TFC matrices is that the notion more precisely captures the combinatorial structure leading to low SCC than does the notion of 0,1-network matrices. The importance of network matrices is that they are totally unimodular, but as we show later in Example 1, totally unimodular matrices do not generally share the favourable SCC properties of network matrices.

The following lemma is known to matroid theorists [19, 24, §14.10], but we prove it here for completeness without reference to matroids.

LEMMA 4.1. *Let $G = (V, E)$ be a graph, let $T \subseteq E$ be a tree spanning G, let A be the TFC matrix for T and G, and let n be the number of columns of A. Then A has at most $\max\{3n - 2, n + 1\}$ distinct rows. In particular, tree cover has $O(n)$ SCC.*

Proof. If every row of A has at most a single 1, then A has at most $n+1$ distinct rows, and we are done. Suppose A has some row with at least two 1's. i.e. Some two fundamental cycles intersect.

Suppose that G has a vertex v of degree at most 2. Then we are in one of the following three cases.

(i) v is incident to exactly one edge e of T and exactly one edge f of $E \setminus T$. In this case, v is a leaf, and f is the only edge whose fundamental cycle contains e. Thus e's row in the TFC matrix has a single 1.

(ii) v is incident to exactly one edge e of T and no other edge. In this case, v is a leaf, and e lies on no fundamental cycle. Thus e's row in the TFC matrix is zero.

(iii) v is incident to two edges e and f of T. This means that a non-tree edge contains e on its fundamental cycle iff it contains f, and hence the rows of e and f are identical in the TFC matrix.

We create a new graph $\tilde{G} = (\tilde{V}, \tilde{E})$ and a new tree $\tilde{T} \subseteq \tilde{E}$ spanning $\tilde{E}$ in the following way. While G has a vertex $v \in V$ of degree at most 2, contract one of its incident T edges. In matrix terms, contracting edge e of T corresponds to deleting e's row in the TFC matrix. Note that each contraction operation decreases the number of vertices in G by 1 and may delete only zero rows, rows with exactly one 1, and duplicate rows. Since we assumed A has a row with at least two 1's, we will never contract that tree edge, so $\tilde{G}$ therefore has at least two vertices.

The resulting graph $\tilde{G}$ has minimum degree at least 3, and we therefore have $3|\tilde{V}| \leq 2|\tilde{E}|$. Hence,

$$|\tilde{V}| - 1 \leq 2[|\tilde{E}| - (|\tilde{V}| - 1)] - 3 = 2n - 3,$$

which shows $\tilde{A}$ has at most $2n-3$ rows. Therefore, A has at most $3n-2$ distinct rows.

When a `SCC` bound holds independently of k, we may add priorities at loss of a factor of k in the `SCC`:

LEMMA 4.2. *Priority tree cover has $O(nk)$ SCC.*

Proof. Let $\tilde{A}$ be the incidence matrix of the priority tree cover problem with columns ordered by non-increasing priority, and let A be the incidence matrix of the underlying tree cover problem.

Consider a tree edge e. The effect of adding a priority to e is to set some suffix of its corresponding row to zero. Thus, each cell of A induces at most one cell of each depth in $\tilde{A}$—in particular, at most $k+1$ cells of depth k or fewer. Since A has $O(n)$ `SCC`, it follows that $\tilde{A}$ has $O(nk)$ `SCC`.

This completes the proof of Theorem 1.2. Unfortunately, not all totally-unimodular $0,1$ matrices have low `SCC` after adding priorities:

EXAMPLE 1. *Even transposes of $0,1$ network matrices, which are totally unimodular, can have SCC $\Omega(n^2)$ after adding priorities. We show this using a set cover problem where the elements are vertical paths in a rooted tree and each set consists of the paths meeting at a specified edge and not exceeding the priority of the edge.*

Fix $\ell \geq 1$. Let $v_0, v_1, \ldots, v_\ell$ be a path with ℓ edges, and let $w_1, \ldots, w_\ell$ be leaves, each adjacent to v_0. For each $1 \leq i \leq \ell$, assign priority i to edge $v_{i-1}v_i$, and assign priority ℓ to edge w_iv_0. Root the tree at v_ℓ.

The resulting set cover problem has $n = 2\ell$ sets, and we claim the number of cells of depth 2 is at least $\ell^2 = \frac{n^2}{4}$. Indeed, let $1 \leq i, j \leq \ell$, and consider the path P_{ij} from w_i to v_j having priority j. The only two edges whose corresponding sets contain P_{ij} are w_iv_0 and $v_{j-1}v_j$. Thus, each P_{ij} lies in a distinct depth-2 cell.

5 Acknowledgements

We thank Jim Geelen for pointing out the connection between Lemma 4.1 and matroid growth rates.

References

[1] B. Aronov, E. Ezra, and M. Sharir. Small-size ϵ-nets for axis-parallel rectangles and boxes. *SIAM Journal on Computing*, 39(7):3248–3282, 2010.

[2] F. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, September 1991.

[3] N. Bansal, R. Krishnaswamy, and B. Saha. On capacitated set cover problems. In *Proceedings of International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 2011.

[4] N. Bansal and K. Pruhs. The geometry of scheduling. In *Proceedings of IEEE Symposium on Foundations of Computer Science*, pages 407–414, 2010.

[5] J.-D. Boissonnat, M. Sharir, B. Tagansky, and M. Yvinec. Voronoi diagrams in higher dimensions under certain polyhedral distance functions. *Discrete Comput. Geom.*, 19(4):473–484, 1998.

[6] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete Comput. Geom.*, 14(4):463–479, 1995.

[7] T. H. Brylawski and D. Lucas. Uniquely representable combinatorial geometries. In *Proceedings, Colloquio Internazionale sulle Teorie Combinatorie*, pages 83–104, 1973.

[8] D. Chakrabarty, E. Grant, and J. Könemann. On column-restricted and priority covering integer programs. In *Proceedings of MPS Conference on Integer Programming and Combinatorial Optimization*, pages 355–368, 2010.

[9] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2(2):195–222, 1987.

[10] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4(1):387–421, 1989.

[11] K. L. Clarkson and K. R. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete Comput. Geom.*, 37(1):43–58, 2007.

[12] M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, Heidelberg, Germany, 3rd edition, 2008.

[13] G. Even, D. Rawitz, and S. Shahar. Hitting sets when the VC-dimension is small. *Inform. Process. Lett.*, 95(2):358–362, 2005.

[14] E. Ezra, B. Aronov, and M. Sharir. Improved bound for the union of fat triangles. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pages 1778–1785, 2011.

[15] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45, 1998.

[16] Matt Gibson and Imran A. Pirwani. Algorithms for dominating set in disk graphs: Breaking the $\log n$ barrier. In *Proceedings of European Symposium on Algorithms*, pages 243–254, 2010.

[17] D. Haussler and E. Welzl. ϵ-nets and simplex range queries. *Discrete Comput. Geom.*, 2:127–151, 1987.

[18] D. Hochbaum. Approximating covering and packing problems: Set cover, vertex cover, independent set, and related problems. In Dorit S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems.* PWS Publishing, 1997.

[19] F. Jaeger. Flows and generalized coloring theorems in graphs. *J. Combin. Th. Ser. B*, 26:205–216, 1979.

[20] A. Joffe. On a set of almost deterministic k-independent random variables. *Ann. Probab.*, 2(1):161–162, 1974.

[21] K. Kedem, R. Livne, J. Pach, and M. Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete Comput. Geom.*, 1:59–70, 1986.

[22] S. G. Kolliopoulos and N. E. Young. Approximation algorithms for covering/packing integer programs. *J. Comput. System Sci.*, 71(4):495–505, 2005.

[23] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge Univ. Press, New York, 1995.

[24] J. Oxley. *Matroid Theory.* Oxford University Press, New York, NY, 2011.

[25] J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. *SIAM J. Discrete Math.*, 8(2):223–250, 1995.

[26] A. Schrijver. *Combinatorial Optimization.* Springer, New York, 2003.

[27] A. Srinivasan. Improved approximation guarantees for packing and covering integer programs. *SIAM Journal on Computing*, 29(2):648–670, 1999.

[28] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probab. and its Applications*, 16(2):264–280, 1971.

[29] K. R. Varadarajan. Epsilon nets and union complexity. In *Proceedings of ACM-SIAM Symposium on Computational Geometry*, pages 11–16, 2009.

[30] K. R. Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *Proceedings of ACM Symposium on Theory of Computing*, pages 641–648, 2010.