Exact Algorithms and APX-Hardness Results for Geometric Packing and Covering Problems*

Timothy M. Chan† Elyot Grant‡

March 29, 2012

Abstract

We study several geometric set cover and set packing problems involving configurations of points and geometric objects in Euclidean space. We show that it is APX-hard to compute a minimum cover of a set of points in the plane by a family of axis-aligned fat rectangles, even when each rectangle is an ϵ-perturbed copy of a single unit square. We extend this result to several other classes of objects including almost-circular ellipses, axis-aligned slabs, downward shadows of line segments, downward shadows of graphs of cubic functions, fat semi-infinite wedges, 3-dimensional unit balls, and axis-aligned cubes, as well as some related hitting set problems. We also prove the APX-hardness of a related family of discrete set packing problems. Our hardness results are all proven by encoding a highly structured minimum vertex cover problem which we believe may be of independent interest.

In contrast, we give a polynomial-time dynamic programming algorithm for geometric set cover where the objects are pseudodisks containing the origin or are downward shadows of pairwise 2-intersecting x-monotone curves. Our algorithm extends to the weighted case where a minimum-cost cover is required. We give similar algorithms for several related hitting set and discrete packing problems.

1 Introduction

In a *geometric set cover problem*, we are given a range space $(X, \mathcal{S})$—a universe X of points in Euclidean space and a pre-specified configuration $\mathcal{S}$ of regions or geometric objects such as rectangles or half-planes. The goal is to select a minimum-cardinality subfamily $\mathcal{C} \subseteq \mathcal{S}$ such that each point in X lies inside at

*A preliminary version of this paper appeared in the 2011 Canadian Conference on Computational Geometry [10].

†David R. Cheriton School of Computer Science, University of Waterloo, tmchan@uwaterloo.ca

‡Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, elyot@mit.edu

least one region in $\mathcal{C}$. In the related *geometric hitting set problem*, the goal is instead to select a minimum cardinality subset $Y \subseteq X$ such that each set in $\mathcal{S}$ contains at least one point in Y. In the *weighted* generalizations of these problems, we are also given a vector of positive costs $\mathbf{w} \in \mathbb{R}^{\mathcal{S}}$ or $\mathbf{w} \in \mathbb{R}^{X}$ and we wish to minimize the total cost of all objects in $\mathcal{C}$ or Y respectively. Instances without costs (or with unit costs) are termed *unweighted*.

Geometric covering problems have found many applications to real-world engineering and optimization problems in areas such as wireless network design, image compression, and circuit-printing. Unfortunately, even for very simple classes of objects such as unit disks or unit squares in the plane, computing the exact minimum set cover is strongly NP-hard [20]. Consequently, much of the research surrounding geometric set cover has focused on approximation algorithms. A large number of constant and almost-constant approximation algorithms have been obtained for various hitting set and set cover problems of low VC-dimension via ϵ-net based methods [8] [16]. These methods have spawned a rich literature concerning techniques for obtaining small ϵ-nets for various weighted and unweighted geometric range spaces [14] [1] [24]. Results include constant-factor linear programming based approximation algorithms for set cover with objects like fat rectangles in the plane and unit cubes in $\mathbb{R}^3$.

However, these approaches have limitations. So far, ϵ-net based methods have been unable to produce anything better than constant-factor approximations, and typically the constants involved are quite large. Their application is also limited to problems involving objects with combinatorial restrictions such as low *union complexity* (see [14] for details). A recent construction due to Pach and Tardos has proven that small ϵ-nets need not always exist for instances of the *rectangle cover problem*—geometric set cover where the objects are axis-aligned rectangles in the plane [22]. In fact, their result implies that the integrality gap of the standard set cover LP for the rectangle cover problem can be as big as $\Theta(\log n)$. Despite this, a constant approximation using other techniques has not been ruled out.

The approximability of problems like rectangle cover also has connections to related capacitated covering problems [11]. Recently, Bansal and Pruhs used these connections, along with a weighted ϵ-net based algorithm of Varadarajan [24], to obtain a breakthrough in approximating a very general class of machine scheduling problems by reducing them to a weighted covering problem involving points and *4-sided boxes* in $\mathbb{R}^3$—axis-aligned cuboids abutting the xy and yz planes [9]. The 4-sided box cover problem generalizes the rectangle cover problem in $\mathbb{R}^2$ and thus inherits its difficulty.

In light of the drawbacks of ϵ-net based methods, Mustafa and Ray recently proposed a different approach. They gave a PTAS for a wide class of unweighted geometric hitting set problems (and consequently, related set cover problems) via a *local search* technique [21]. Cast in our framework, their algorithm works roughly as follows: take any feasible set cover $\mathcal{C}$, find a family of k sets in $\mathcal{C}$ that can be replaced by some family of $k-1$ sets while still covering every object in the universe, and make such replacements until no more are possible. For fixed k, this runs in polynomial time, and as Mustafa and Ray show, produces a

$1+O\left(\frac{1}{\sqrt{k}}\right)$-approximation for range spaces satisfying certain locality conditions. Their method yields PTASs for:

- Geometric hitting set problems involving half-spaces in $\mathbb{R}^3$ and pseudodisks (including disks, axis-aligned squares, and more generally homothetic copies of identical convex regions) in the plane.
- By implication, geometric set cover problems with lower half-spaces in $\mathbb{R}^3$ (by geometric duality, see [5]), disks in $\mathbb{R}^2$ (by a standard lifting transformation that maps disks to lower halfspaces in $\mathbb{R}^3$, see [5]), and translated copies of identical convex regions in the plane (again, by duality).

Their results currently do not seem applicable to set cover with general pseudodisks in the plane. On a related note, Erlebach and van Leeuwen have obtained a PTAS for the weighted version of geometric set cover for the special case of unit squares [17].

Of additional interest are the related "dual" problems encountered when attempting to solve the *maximum set packing* problem on geometric range spaces such as those described above. Given a geometric range space $(X, \mathcal{S})$, there are in fact two problems to discuss:

- The *pack-points* problem—that of finding a maximum cardinality subset $Y \subseteq X$ of points, no two of which are contained in a single region of $\mathcal{S}$.
- The *pack-regions* problem—that of finding a maximum-cardinality subfamily $\mathcal{C} \subseteq \mathcal{S}$ of regions, no two of which intersect at a point in X.

The pack-points and pack-regions problems can be thought of as the "dual" problems associated to the set cover and hitting set problems for geometric range spaces (where by dual, we mean in the sense of packing-covering duality or linear programming duality). The pack-regions problem sometimes goes by the name *discrete independent set* [12], but we avoid this terminology to avoid confusion between the two packing variations, instead emulating the terminology of [15].

As with the set cover and hitting set problems, we shall discuss both weighted and unweighted versions of the pack-points and pack-regions problems. In the weighted pack-points (pack-regions) problem, we are additionally given a vector of positive costs $\mathbf{w} \in \mathbb{R}^X$ ($\mathbf{w} \in \mathbb{R}^{\mathcal{S}}$) and we wish to maximize the total weight of the subset of points Y (subfamily of regions $\mathcal{C}$).

1.1 Hardness Results

In this article, we present two main classes of results. The first is a series of APX-hardness proofs for a number of unweighted geometric set cover, hitting set, pack-points, and pack-regions problems on various range spaces. Here, we outline these results.

For a set Y of points in the plane, we define the *downward shadow* of Y to be the set of all points (a, b) such that there is a point $(a, y) \in Y$ with $y \geq b$.

Theorem 1.1. *Unweighted geometric set cover is APX-hard with each of the following classes of objects:*

(C1) *Axis-aligned rectangles in* $\mathbb{R}^2$*, even when all rectangles have lower-left corner in* $[-1, -1+\epsilon]\times[-1, -1+\epsilon]$ *and upper-right corner in* $[1, 1+\epsilon]\times[1, 1+\epsilon]$ *for an arbitrarily small* $\epsilon > 0$*.*

(C2) *Axis-aligned ellipses in* $\mathbb{R}^2$*, even when all ellipses have centers in* $[0, \epsilon] \times [0, \epsilon]$ *and major and minor axes of length in* $[1, 1 + \epsilon]$*.*

(C3) *Axis-aligned slabs in* $\mathbb{R}^2$*, each of the form* $[a_i, b_i] \times [-\infty, \infty]$ *or* $[-\infty, \infty] \times [a_i, b_i]$*.*

(C4) *Axis-aligned rectangles in* $\mathbb{R}^2$*, even when the boundaries of each pair of rectangles intersect exactly zero times or four times.*

(C5) *Downward shadows of line segments in* $\mathbb{R}^2$*.*

(C6) *Downward shadows of (graphs of) univariate cubic functions in* $\mathbb{R}^2$*.*

(C7) *Unit balls in* $\mathbb{R}^3$*, even when all the balls contain a common point.*

(C8) *Axis-aligned cubes in* $\mathbb{R}^3$*, even when all the cubes contain a common point and are of similar size.*

(C9) *Half-spaces in* $\mathbb{R}^4$*.*

(C10) *Fat semi-infinite wedges in* $\mathbb{R}^2$*, each of which has an opening angle in* $[\pi - \epsilon, \pi)$ *and has its vertex within* ϵ *of a common point.*

Additionally, unweighted geometric hitting set is APX-hard with each of the following classes of objects:

(H1) *Axis-aligned slabs in* $\mathbb{R}^2$*.*

(H2) *Axis-aligned rectangles in* $\mathbb{R}^2$*, even when the boundaries of each pair of rectangles intersect exactly zero times or four times.*

(H3) *Unit balls in* $\mathbb{R}^3$*.*

(H4) *Half-spaces in* $\mathbb{R}^4$*.*

Mustafa and Ray ask if their local improvement approach outlined in [21] might yield a PTAS for a wider class of instances; Theorem 1.1 immediately rules this out for all of the covering and hitting set problems listed above by proving that no PTAS exists for them unless P = NP. Item (C1) demonstrates that even tiny perturbations in the input regions can destroy the behaviour of the local search method. (C2) rules out the possibility of a PTAS for arbitrarily fat ellipses (that is, ellipses that are within ϵ of being perfect circles). (C5) and (C6) stand in contrast to our dynamic programming algorithms given later in this paper, which prove that geometric set cover is polynomial-time solvable

when the objects are downward shadows of horizontal line segments or quadratic functions. In the case of (C4) and (H2), the intersection graph of the rectangles is a comparability graph (and hence a perfect graph); even then, neither set cover nor hitting set admits a PTAS. (C7), (C8), (C9), (H3), and (H4) complement the result of Mustafa and Ray by showing that their algorithm fails in higher dimensions. (C10) stands in contrast to the fact that covering with half-planes is exactly solvable in polynomial time (this result, alongside many others, is discussed in Section 1.3).

For unit cubes in $\mathbb{R}^3$, the existence of a PTAS remains an open question for both set cover and hitting set. Additionally, set cover with arbitrarily sized disks or squares in $\mathbb{R}^2$ remains open (Mustafa and Ray only provide a PTAS for the hitting set problem involving arbitrary pseudodisks, which only implies a PTAS for the covering version involving unit disks or unit squares).

All of our hardness results are proven via reduction from a restricted version of unweighted set cover, which we call *SPECIAL-3SC*:

Definition 1.2. In a *SPECIAL-3SC range space*, we are given a universe $U = A \cup W \cup X \cup Y \cup Z$ comprising disjoint sets $A = \{a_1, \ldots, a_n\}$, $W = \{w_1, \ldots, w_m\}$, $X = \{x_1, \ldots, x_m\}$, $Y = \{y_1, \ldots, y_m\}$, and $Z = \{z_1, \ldots, z_m\}$ where $2n = 3m$. We are also given a family $\mathcal{S}$ of $5m$ subsets of U satisfying the following two conditions:

- For each $1 \leq t \leq m$, there are integers $1 \leq i < j < k \leq n$ such that $\mathcal{S}$ contains the sets $\{a_i, w_t\}$, $\{w_t, x_t\}$, $\{a_j, x_t, y_t\}$, $\{y_t, z_t\}$, and $\{a_k, z_t\}$ (summing over all t gives the $5m$ sets contained in $\mathcal{S}$.)
- For all $1 \leq t \leq n$, the element a_t is in exactly two sets in S.

We denote by *SPECIAL-3SC* the minimum set cover problem on a SPECIAL-3SC range space.

We note that each element in a SPECIAL-3SC range space is contained in exactly two sets, so SPECIAL-3SC is a special case of the vertex cover problem in graphs of degree at most 3. Intuitively, we may think of SPECIAL-3SC as the problem obtained when one takes a 3-regular graph G, replaces each vertex v with a path P_v of length 4 (connecting the first, third, and fifth vertices of P_v to the neighbours of v in any order), and examines the minimum vertex cover problem in the resulting graph.

In section 2, we show:

Lemma 1.3. *SPECIAL-3SC is APX-hard, even in the unweighted case.*

We are also able to establish APX-hardness for a packing version of the SPECIAL-3SC problem:

Lemma 1.4. *The pack-regions problem is APX-hard for SPECIAL-3SC range spaces, even in the unweighted case.*

The pack-regions problem on SPECIAL-3SC range spaces is equivalent to the *maximum independent set* problem on the associated class of graphs of degree at most 3.

In section 3, we prove Theorem 1.1 by showing that all of the range spaces listed in the statement of Theorem 1.1 can directly encode SPECIAL-3SC. Via Lemma 1.4, we immediately obtain the following corollary:

Theorem 1.5. *We have the following APX-hardness results:*

(1) *The unweighted pack-regions problem is APX-hard for range spaces (C1) through (C10).*

(2) *The unweighted pack-points problem is APX-hard for range spaces (H1) through (H4).*

Item (2) follows by interchanging the roles of points and sets when switching from hitting set problems to covering problems.

1.2 Algorithmic Results

Our second main series of results is a collection of dynamic programming algorithms for the set cover, hitting set, pack-regions, and pack-points problems on a class of geometric range spaces exhibiting somewhat simpler structure. We show that, for all four problems, it is possible to obtain polynomial-time exact algorithms for the range space involving downward shadows of horizontal line segments in the plane (such regions might be called *3-sided rectangles*, *bottomless rectangles*, or *half-slabs*). In some cases, we are able to obtain more general results. In all cases, our methods work for both the weighted and unweighted versions of the problem (assuming weights are integers or other representations supporting fast arithmetic operations).

All proofs of nontrivial theorems in this subsection appear in section 4.

1.2.1 Set Cover

For set cover, our main algorithmic result is the following:

Theorem 1.6. *There exists a polynomial-time exact algorithm for the weighted geometric set cover problem involving downward shadows of pairwise 2-intersecting x-monotone curves in $\mathbb{R}^2$. Moreover, it runs in $O(mn^2(m+n))$ time on a range space consisting of n points and m regions.*

For the running time guarantees we provide in this and all other theorems, we assume a model in which $O(1)$ time is sufficient for arithmetic operations and primitive operations like computing the intersections of two curves. More details are provided in Section 4.

The algorithm we use for proving Theorem 1.6 is a generalization and simplification of a similar algorithm appearing in [11] for a combinatorial problem that turns out to be equivalent to geometric set cover with downward shadows

of horizontal line segments in $\mathbb{R}^2$. We believe that our current presentation is much shorter and cleaner; in particular, we do not require shortest path as a subroutine. We can also extend our algorithm to the set cover problem on some related geometric range spaces:

Corollary 1.7. *There exists a polynomial-time exact algorithm for the weighted geometric set cover problem involving a configuration of pseudodisks in $\mathbb{R}^2$ where the origin lies within each pseudodisk. Furthermore, it runs in $O(mn^2(m+n))$ time on a range space consisting of n points and m pseudodisks.*

Proof. By perturbing the pseudodisks if necessary, we may assume that the origin lies in the interior of each pseudodisk (i.e. not on the boundary of any of the regions). We refer the reader to Lemma 2.11 of [4], which shows us how to use a topological sweep curve method to transform the arrangement of pseudodisks into a topologically equivalent arrangement where all the pseudodisks are star-shaped about the origin. By examining the proof of this lemma, we note that this transformation can be accomplished in $O(m^2)$ time since, with m pseudodisks, the sweep curve must be advanced at most $2\binom{m}{2}$ times. We then apply a standard polar-to-cartesian projective transformation about the origin, sending each point (x, y) to $\left(\text{atan2}(y, x), \sqrt{x^2 + y^2}\right)$[1]. This transformation maps each star-shaped pseudodisk to the downward shadow of a positive valued x-monotone function on $[0, 2\pi)$, and the resulting family of curves is still pairwise 2-intersecting. Via the same transformation, we map each point from the original range space into the appropriate cell of the transformed arrangement to obtain a topologically identical range space involving downward shadows of pairwise 2-intersecting x-monotone curves in $\mathbb{R}^2$. Since the entire transformation preserves the element-set incidence relation, it suffices to apply Theorem 1.6. □

A further application is to set cover with lower half-planes in $\mathbb{R}^2$:

Corollary 1.8. *There exists a polynomial-time exact algorithm for the weighted geometric set cover problem involving lower half-planes in $\mathbb{R}^2$, and it runs in $O(mn^2(m+n))$ time on a range space consisting of n points and m half-planes.*

Proof. We first observe that, given a family $\mathcal{S}$ of lower half-planes in $\mathbb{R}^2$, there must exist a circle $\mathcal{C}$ that is disjoint from all of the half-planes (such a circle can be centered at some point P located at coordinates $(0, k)$ for sufficiently large k). We apply an inversive transformation about the circle C (see [13] for further information), mapping each half-plane to a disk whose boundary intersects P (and mapping each point to its corresponding location after inversion). Doing this, we obtain a new range space whose element-set incidence relation is unchanged, but is now a configuration of disks each containing point P. These disks are pseudodisks, and thus the previous corollary can be applied to achieve the desired result (substituting P for the origin). □

[1] By $\text{atan2}(y, x)$, we mean, for a nonzero vector (x, y), the unique angle θ in $[0, 2\pi)$ such that (x, y) is equal to the vector $(\sqrt{x^2 + y^2}, 0)$ rotated counterclockwise through an angle of θ about the origin.

1.2.2 Hitting Set

For the hitting set problem, we obtain a result for downward shadows of horizontal line segments:

Theorem 1.9. *There exists a polynomial-time exact algorithm for the weighted geometric hitting set problem involving downward shadows of horizontal line segments in $\mathbb{R}^2$. Moreover, it runs in $O(\min(m,n)^2(m+n)+n\log n+m\log m)$ time on a range space consisting of n points and m regions.*

Naive attempts to generalize this to downward shadows of 2-intersecting curves appear to fail. We leave it as an open problem to determine if the hitting set problem involving downward shadows of 2-intersecting x-monotone curves in the plane is APX-hard, has a PTAS, or is poly-time solvable. We remark that it appears unlikely that the problem can encode SPECIAL-3SC.

As an additional corollary of Theorem 1.6, we are able to deal with hitting set for lower half-planes:

Corollary 1.10. *There exists a polynomial-time exact algorithm for the weighted geometric hitting set problem involving lower half-planes in $\mathbb{R}^2$, running in $O(m^2n(m+n))$ time on a range space consisting of n points and m half-planes.*

Proof. This follows immediately from Corollary 1.8 and geometric duality [5]. Note that the role of m and n in the running time has been exchanged since points and regions have been interchanged. □

1.2.3 Pack-Points

For the pack-points problem, we are able to get polynomial algorithms for the same class of range spaces as we did for the set cover problem:

Theorem 1.11. *There exists a polynomial-time exact algorithm for the weighted pack-points problem involving downward shadows of pairwise 2-intersecting x-monotone curves in $\mathbb{R}^2$. Moreover, it runs in $O(mn^3)$ time on a range space consisting of n points and m regions.*

Via the same encoding techniques used previously, we can extend this result to some related range spaces:

Corollary 1.12. *There exist polynomial-time exact algorithms for the weighted pack-points problems involving the following types of regions, both running in time $O(mn^3+m^2)$ on range spaces consisting of n points and m regions:*

- *lower half-planes in $\mathbb{R}^2$.*
- *a configuration of pseudodisks in $\mathbb{R}^2$ where the origin lies within each pseudodisk.*

Proof. It suffices to apply precisely the same techniques as in the proofs of Corollary 1.7 and Corollary 1.8. The $O(m^2)$ term must be added to allow for the time required to perform the topology-preserving transformation described in the proof of Corollary 1.7. □

1.2.4 Pack-Regions

Finally, for the pack-regions problem, we obtain an algorithm for shadows of horizontal line segments:

Theorem 1.13. *There exists a polynomial-time exact algorithm for the weighted pack-regions problem involving downward shadows of horizontal line segments* $\mathbb{R}^2$*. Moreover, it runs in* $O(m \min(m,n)^2 + n \log n + m \log m)$ *time on a range space consisting of* n *points and* m *region*

As in the case of the hitting set problem, we leave open the question of whether this can be generalized to downward shadows of 2-intersecting x-monotone curves.

Again, we can also get a result for lower half-planes:

Corollary 1.14. *There exists a polynomial-time exact algorithm for the weighted pack-regions problem involving lower half-planes in* $\mathbb{R}^2$*, running in* $O(m^3 n + n^2)$ *time on a range space consisting of* n *points and* m *half-planes.*

Proof. This follows immediately from Corollary 1.12 and geometric duality [5]. □

1.3 Related Work

The problem of assembling a given rectilinear polygon from a minimum number of (possibly overlapping) axis-aligned rectangles was first proven to be MAX-SNP-complete by Berman and Dasgupta [6], which rules out the possibility of a PTAS unless P = NP. Since set cover with axis-aligned rectangles can encode these instances, it too is MAX-SNP-complete. However, the proof in [6] cannot be applied to produce an instance of geometric set cover using only fat rectangles.

In his recent Ph.D. thesis, van Leeuwen proves APX-hardness for geometric set cover and dominating set with axis-aligned rectangles and ellipses in the plane [25]. Har-Peled provides a simple proof that set cover with triangles is APX-hard, even when all triangles are fat and of similar size [18]. Har-Peled also notes that set cover with circles (that is, with boundaries of disks) is APX-hard for a similar reason. However, neither the results of van Leeuwen nor Har-Peled can be directly extended to fat axis-aligned rectangles or fat ellipses.

There are few non-trivial examples of geometric covering or hitting set problems that are known to be poly-time solvable. Har-Peled and Lee give a dynamic programming algorithm for weighted cover of points in the plane by half-planes [19]; their method runs in $O(n^5)$ time on an instance with n points and half-planes. Our algorithm reduces the running time by a factor of n in the case of lower half-planes. Ambühl et al. give a poly-time dynamic programming algorithm for weighted covering of points in a narrow strip using unit disks [3]; their methods appear to be unrelated to ours.

An interesting PTAS result is that of Har-Peled and Lee, who give a PTAS for minimum weight cover with any class of fat objects, provided that each

object is allowed to expand by a small amount [19]. Our results show that for many classes of objects, a PTAS cannot be obtained without allowing this.

The literature on geometric packing problems mostly deals with the situation in which one merely wishes to find a maximum cardinality disjoint family of geometric regions (one may think of this as the special case of pack-regions in which X consists of all points). The more general pack-points and pack-regions problems described herein are seen less frequently. Chan and Har-Peled have shown that the pack-regions problem admits a good approximation in cases of low union complexity [12]. Ene, Har-Peled, and Raichel have extended this to more general capacitated geometric packing problems, obtaining several constant and almost-constant approximations [15]. They also use standard encodings to rule out a PTAS for both the pack-points and pack-regions problems for the case of range spaces involving fat triangles in the plane. No PTAS results (via local search or other techniques) appear to be known.

2 APX-Hardness of Covering and Packing for SPECIAL-3SC Range Spaces

In this section, we prove the APX-hardness results given in Lemma 1.3 and Lemma 1.4. We recall that a pair of poly-time computable functions (f, g) is an L-reduction from an optimization problem A to an optimization problem B if there are positive constants α and β such that for each instance x of A, the following hold:

(L1) The function f maps instances of A to instances of B such that $\mathrm{OPT}(f(x)) \leq \alpha \cdot \mathrm{OPT}(x)$.

(L2) The function g maps feasible solutions y of $f(x)$ to feasible solutions $g(y)$ of x such that $|c_x(g(y)) - \mathrm{OPT}(x)| \leq \beta \cdot \left|c_{f(x)}(y) - \mathrm{OPT}(f(x))\right|$, where c_x and $c_{f(x)}$ are the cost functions of the instances x and $f(x)$ respectively.

To prove Lemma 1.3, we exhibit an L-reduction from minimum vertex cover on 3-regular graphs (hereafter known as 3VC) to SPECIAL-3SC. Since 3VC is APX-hard [2], this proves that SPECIAL-3SC is APX-hard (see [23] for details).

Proof of Lemma 1.3. Given an instance x of 3VC on a graph G having edges $\{e_1, \ldots, e_n\}$ and vertices $\{v_1, \ldots, v_m\}$ where $3m = 2n$, we define $f(x)$ be the SPECIAL-3SC instance containing the sets $\{a_i, w_t\}$, $\{w_t, x_t\}$, $\{a_j, x_t, y_t\}$, $\{y_t, z_t\}$, and $\{a_k, z_t\}$ for each 4-tuple (t, i, j, k) such that v_t is a vertex incident to edges e_i, e_j, and e_k with $i < j < k$. We may think of this as the vertex cover problem for the graph obtained when we take G and replace each vertex v in G with a path P_v of length 4, connecting the first, third, and fifth vertices of P_v to the neighbours of v in an arbitrary order.

To define g, we suppose we are given a solution y to the SPECIAL-3SC instance $f(x)$. We take vertex v_t in our solution $g(y)$ of the 3VC instance x if and only if at least one of $\{a_i, w_t\}$, $\{a_j, x_t, y_t\}$, or $\{a_k, z_t\}$ is taken in y. We

observe that g maps feasible solutions of $f(x)$ to feasible solutions of x since e_i is covered in $g(y)$ whenever a_i is covered in y.

Our key observation is the following:

Proposition 2.1. $\mathrm{OPT}(f(x)) = \mathrm{OPT}(x) + 2m$.

Proof. For $1 \leq t \leq m$, we define the sets $\mathcal{P}_t = \{\{w_t, x_t\}, \{y_t, z_t\}\}$ and $\mathcal{Q}_t = \{\{a_i, w_t\}, \{a_j, x_t, y_t\}, \{a_k, z_t\}\}$. Call a solution $\mathcal{C}$ of $f(x)$ *segregated* if for all $1 \leq t \leq m$, $\mathcal{C}$ either contains all sets in $\mathcal{P}_t$ and no sets in $\mathcal{Q}_t$, or contains all sets in $\mathcal{Q}_t$ and no sets in $\mathcal{P}_t$.

Via local interchanging, we observe that there exists an optimal solution to $f(x)$ that is segregated. Specifically, when given an arbitrary optimal solution $\mathcal{C}^*$ of $f(x)$, we can construct a new solution $\mathcal{C}'$ if, for each t, we simply take all sets in $\mathcal{Q}_t$ whenever $\mathcal{C}^*$ contains at least one set in $\mathcal{Q}_t$ and otherwise take all sets in $\mathcal{P}_t$. It follows immediately that $\mathcal{C}'$ is feasible if $\mathcal{C}^*$ is, and it is not hard to see that the cost of $\mathcal{C}'$ cannot exceed that of $\mathcal{C}^*$.

Additionally, our function g, when restricted to segregated solutions of $f(x)$, forms a bijection between them and feasible solutions of x. We check that g maps segregated solutions of size $2m + k$ to solutions of x having cost precisely k, and the result follows. □

Proposition 2.1 implies that f satisfies property (L1) with $\alpha = 5$, since $\mathrm{OPT}(x) \geq \frac{m}{2}$. Moreover, $c_x(g(y)) + 2m \leq c_{f(x)}(y)$ since both $\{w_t, x_t\}$ and $\{y_t, z_t\}$ must be taken in y whenever v_t is not taken in $g(y)$, and at least three of $\{\{a_i, w_t\}, \{w_t, x_t\}, \{a_j, x_t, y_t\}, \{y_t, z_t\}, \{a_k, z_t\}\}$ must be taken in y whenever v_t is taken in $g(y)$. Together with Proposition 2.1, this proves that g satisfies property (L2) with $\beta = 1$. Thus (f, g) is an L-reduction, completing the proof that SPECIAL-3SC is APX-hard. □

To prove APX-hardness of the pack-regions version of SPECIAL-3SC, we construct an L-reduction from SPECIAL-3SC itself. This is straightforward via standard methods:

Proof of Lemma 1.4. Let x be a SPECIAL-3SC instance on a range space $(U, \mathcal{S})$ encoding the vertex cover problem on a graph G having n vertices. We denote by $f(x)$ the pack-regions problem on $(U, \mathcal{S})$, equivalent to the independent set problem on G. We let g be a function mapping a pack-regions solution $\mathcal{I} \subseteq \mathcal{S}$ to the set $\mathcal{S} \setminus \mathcal{I}$. Since $\mathcal{I}$ corresponds to an independent set in G, $g(\mathcal{I})$ corresponds to a vertex cover in G. Consequently, if $\mathcal{I}$ is a feasible solution to $f(x)$, then $g(\mathcal{I})$ is a feasible solution to x.

To prove that (f, g) is an L-reduction, we simply verify the following:

(a) For any solution $\mathcal{I}$ to $f(x)$, we have $\mathrm{OPT}(f(x)) - |\mathcal{I}| = n - |\mathcal{I}| - \mathrm{OPT}(x)$, since a set of vertices is an independent set in G if and only if its complement is a vertex cover.

(b) By examining the structure of a SPECIAL-3SC range space, we observe that the value of $\mathrm{OPT}(x)$ is at least $\frac{2}{5}n$, and thus the value of OPT(f(x)) is at most $\frac{3}{5}n$.

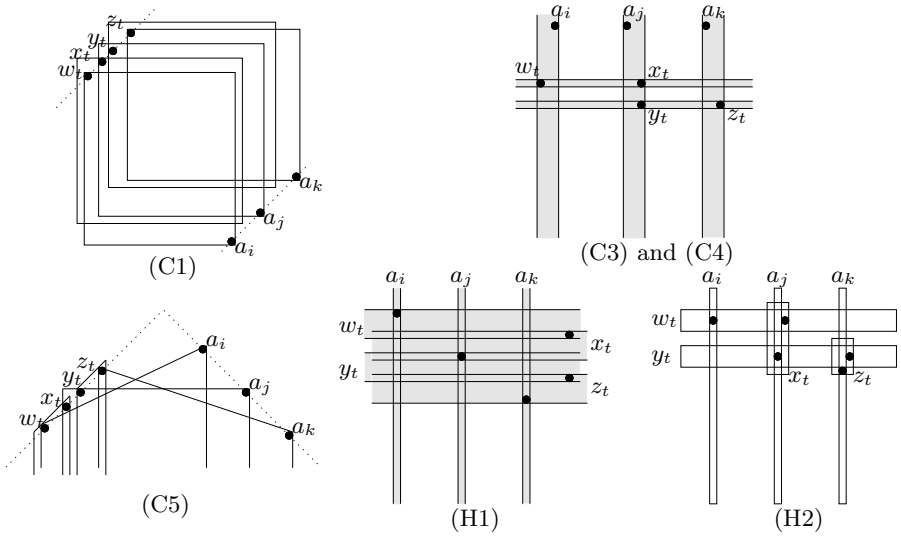

Figure 1: APX-hardness proofs of geometric set cover problems.

Facts (a) and (b) together imply that (f, g) is an L-reduction with $(\alpha, \beta) = (\frac{3}{2}, 1)$. The desired hardness result then follows. □

3 Encodings of SPECIAL-3SC via Geometric Set Cover

In this section, we prove Theorem 1.1 using Lemma 1.3, by encoding SPECIAL-3SC using various classes of geometric set cover and hitting set problems. The beauty of SPECIAL-3SC is that it allows many of our geometric APX-hardness results to follow immediately from simple "proofs by pictures" (see Figure 1). The key property of SPECIAL-3SC is that we can divide the elements into two sets A and $B = W \cup X \cup Y \cup Z$, and linearly order B in such a way that all sets in $\mathcal{S}$ are either two adjacent elements from B, one from B and one from A, or two adjacent elements from B and one from A. We need only make $[w_t, x_t, y_t, z_t]$ appear consecutively in the ordering of B.

For (C1), we simply place the elements of A on the line segment $\{(x, x-2) : x \in [1, 1+\epsilon]\}$ and place the elements of B, in order, on the line segment $\{(x, x+2) : x \in [-1, -1+\epsilon]\}$, for a sufficiently small $\epsilon > 0$. As we can see immediately from Figure 1, each set in $\mathcal{S}$ can be encoded as a fat rectangle in the class (C1).

(C2) is similar. It is not difficult to check that each set can be encoded as a fat ellipse in this class.

For (C3), we place the elements of A on a horizontal line (the top row). For each $1 \leq t \leq m$, we create a new row containing $\{w_t, x_t\}$ and another containing $\{y_t, z_t\}$ as shown in Figure 1. This time, we will need the second property in

Definition 1.2—that each a_i appears in two sets. If $\{a_i, w_t\}$ is the first set that a_i appears in, we place w_t slightly to the left of a_i; if it is the second set instead, we place w_t slightly to the right of a_i. Similarly, the placement of x_t, y_t (resp. w_t) depends on whether a set of the form $\{a_j, x_t, y_t\}$ (resp. $\{a_k, w_t\}$) is the first or second set that a_j (resp. a_k) appears in. As we can see from Figure 1, each set in $\mathcal{S}$ can be encoded as a thin vertical or horizontal slab.

(C4) is similar to (C3), with the slabs replaced by thin rectangles. For example, if $\{a_i, w_t\}$ and $\{a_i, w_{t'}\}$ are the two sets that a_i appears in, with w_t located higher than $w_{t'}$, we can make the rectangle for $\{a_i, w_t\}$ slightly wider than the rectangle for $\{a_i, w_{t'}\}$ to ensure that these two rectangles intersect 4 times.

For (C5), we can place the elements of A on the ray $\{(x, -x) : x > 0\}$ and the elements of B, in order, on the ray $\{(x, x) : x < 0\}$. The sets in $\mathcal{S}$ can be encoded as downward shadows of line segments as in Figure 1.

(C6) is similar to (C5). One way is to place the elements of A on the line segment $\ell_A = \{(x, x) : x \in [-1, -1 + \epsilon]\}$ and the elements of B (in order) on the line segment $\ell_B = \{(x, 0) : x \in [1.5, 1.5 + \epsilon]\}$. For any $a \in [-1, -1 + \epsilon]$ and $b \in [1.5, 1.5 + \epsilon]$, the cubic function $f(x) = (x - b)^2[(a + b)x - 2a^2]/(b - a)^3$ is tangent to ℓ_A and ℓ_B at $x = a$ and $x = b$. (The function intersects $y = 0$ also at $x = 2a^2/(a + b) \gg 1.5 + \epsilon$, far to the right of ℓ_B.) Thus, the size-2 sets in $\mathcal{S}$ can be encoded as cubics. A size-3 set $\{a_j, x_t, y_t\}$ can also be encoded if we take a cubic with tangents at a_j and x_t, shift it upward slightly, and make x_t and y_t sufficiently close.

For (C7), we place the elements in A on a circular arc $\gamma_A = \{(x, y, 0) : x^2 + y^2 = 1,\ x, y \geq 0\}$ and the elements in B (in order) on the vertical line segment $\ell_B = \{(0, 0, z) : z \in [1 - 2\epsilon, 1 - \epsilon]\}$. (This idea is inspired by a known construction [7], after much simplification.) We can ensure that every two points in A have distance $\Omega(\sqrt{\epsilon})$ if $\epsilon \ll 1/n^2$. The technical lemma below allows us to encode all size-2 sets (by setting $b = b'$) and size-3 sets by unit balls containing a common point.

Lemma 3.1. *Given any $a \in \gamma_A$ and $b, b' \in \ell_B$, there exists a unit ball that (i) intersects γ_A at an arc containing a of angle $O(\sqrt{\epsilon})$, (ii) intersects ℓ_B at precisely the segment from b to b', and (iii) contains $(1/\sqrt{2}, 1/\sqrt{2}, 1)$.*

Proof. Say $a = (x, y, 0)$, $b = (0, 0, z - h)$, $b' = (0, 0, z + h)$. Consider the unit ball K centered at $c = (\sqrt{1 - h^2}x, \sqrt{1 - h^2}y, z)$. Then (ii) is self-evident and (iii) is straightforward to check. For (i), note that a lies in K since $\|a - c\|^2 = (1 - \sqrt{1 - h^2})^2 + z^2 \leq \epsilon^2 + (1 - \epsilon)^2 < 1$. On the other hand, if a point $p \in \gamma_A$ lies in the unit ball, then letting $a' = (\sqrt{1 - h^2}x, \sqrt{1 - h^2}y, 0)$, we have $\|p - c\|^2 = \|p - a'\|^2 + z^2 \leq 1$, implying $\|p - a\| \leq \|p - a'\| + \|a' - a\| \leq \sqrt{1 - z^2} + (1 - \sqrt{1 - h^2}) = O(\sqrt{\epsilon})$. □

In order to implement this reduction in polynomial time using a Turing machine, we must ensure that all points are placed at locations having rational coordinates of polynomial size. To do this, we employ a standard trick based upon the *Weierstrass substitution*; a point $(\cos t, \sin t, 0)$ on the circular arc can

be replaced by a sufficiently close point $(\frac{1-s^2}{1+s^2}, \frac{2s}{1+s^2}, 0)$ for an appropriate value of s. It is straightforward to verify that the construction can still be carried out using polynomial-sized rational points of this form.

(C8) is similar to (C1); we place the elements in A on the line segment $\ell_A = \{(t, t, 0) : t \in (0, 1)\}$ and the elements in B on the line segment $\ell_B = \{(0, 3-t, t) : t \in (0, 1)\}$. For any $(a, a, 0) \in \ell_A$ and $(0, 3-b, b) \in \ell_B$, the cube $[-3+b+2a, a] \times [a, 3-b] \times [-3+a+2b, b]$ is tangent to ℓ_A at $(a, a, 0)$, is tangent to ℓ_B at $(0, 3-b, b)$, and contains $(0, 1, 0)$. Size-3 sets $\{a_j, x_t, y_t\}$ can be encoded by taking a cube with tangents at a_j and x_t, expanding it slightly, and making x_t and y_t sufficiently close.

(C9) follows from (C7) by the standard lifting transformation [5] which maps points $(x, y, z) \in \mathbb{R}^3$ to $(x, y, z, x^2+y^2+z^2) \in \mathbb{R}^4$ and balls $\{(x, y, z) : (x-a)^2+(y-b)^2+(z-c)^2 \leq r^2\}$ to halfspaces $\{(x, y, z, w) : w - 2ax - 2by + 2cz \leq r^2 - a^2 - b^2 - c^2\}$.

For (C10), we shall place the elements of A and B respectively along two circular arcs $\ell_A = \{(\cos t, \sin t) : t \in (0, \epsilon)\}$, and $\ell_B = \{(\cos t, 2 - \sin t) : t \in (0, \epsilon)\}$. Any point on either arc defines a unique tangent to that arc, and thus any pair of points, one on ℓ_A and one on ℓ_B, define a pair of tangents that meet at a unique intersection point (which will be close to $(1, 1)$). It follows that a wedge of opening angle close to π can be chosen to be tangent to any point on ℓ_A and any point on ℓ_B. Again, for the size-3 sets, it suffices to expand each wedge slightly and make x_t and y_t sufficiently close. Additionally, we can employ the Weierstrass substitution trick, as we did in (C7), to perform this construction using rational points.

For (H1), we map each element a_i to a thin vertical slab. For each $1 \leq t \leq m$, we map $\{w_t, x_t, y_t, z_t\}$ to a cluster of four thin horizontal slabs as in Figure 1. Each set in $\mathcal{S}$ can be encoded as a point in the arrangement.

(H2) is similar; see Figure 1.

(H3) follows from (C7) by duality.

(H4) follows from (C9) by duality.

4 Polynomial Time Dynamic Programming Algorithms for Weighted Packing and Covering Problems

Here, we give four dynamic programming algorithms yielding proofs of Theorems 1.6, 1.9, 1.11, and 1.13. Each proof proceeds by defining a class of subproblems, proving a recurrence relation among the optimal solutions to those subproblems, and constructing a dynamic programming algorithm from this recurrence relation. The recurrences are somewhat different for each of the four problems, and the running times of the resulting algorithms differ.

The running time bounds we give assume $O(1)$ time for arithmetic operations and primitive operations such as computing the intersections of boundaries of regions, determining whether a point lies inside a region, finding the projection

of a point vertically onto the boundary of a region, and so on. Additionally, in many cases, our subproblems are indexed by the x-coordinates of points. To reduce the number of subproblems that we must solve, we recognize that the number of distinct x-coordinates in the input can be reduced by moving points or regions, so long as the element-set incidence relation of the range space is unaffected. For range spaces involving downward shadows of horizontal line segments, it suffices to assume that the input contains only $O(\min(m, n))$ distinct x-coordinates, since the number of distinct x-coordinates can be reduced unless the points and region boundaries are interleaved. By simply iterating through the points and regions, such a reduction can be performed in $O(m + n)$ time if all necessary sorting of the input is done beforehand. If sorting is required, then $O(n \log n + m \log m)$ time is sufficient.

Throughout this section, we write w_x for the weight of a point $x \in X$, and w_R for the weight of a region $R \in \mathcal{S}$.

4.1 Set Cover

Here, we prove Theorem 1.6 by giving a polynomial-time dynamic programming algorithm for the weighted cover of a finite set of points $X \subseteq \mathbb{R}^2$ by a set $\mathcal{S}$ of downward shadows of 2-intersecting x-monotone curves $C_1, \ldots, C_m$. For $1 \leq i \leq m$, define the region $S_i \in \mathcal{S}$ to be the downward shadow of the curve C_i and let it have positive cost w_i. Define $n = |X|$.

We shall assume that each C_i is the graph of a smooth univariate function with domain $(-\infty, \infty)$, that all intersections are transverse (no pair of curves intersect tangentially), and that no points in X lie on any curve C_i. It is not difficult to get around these assumptions, but we retain them to simplify our explanation.

We shall abuse notation by writing $C_i(x)$ for the unique $y \in \mathbb{R}$ such that (x, y) lies on the curve C_i. We say curve C_i is *wider* than curve C_j (written $C_i \succ C_j$) whenever $C_i(x) > C_j(x)$ for all sufficiently large x. We may also write $S_i \succ S_j$ whenever $C_i \succ C_j$. We note that $\succ$ is a total ordering and thus we can order all curves by width, so we assume without loss of generality that $C_i \succ C_j$ whenever $i > j$. The width-based ordering of curves is useful because of the following key observation:

Proposition 4.1. *If $C_i \succ C_j$, then $S_j \setminus S_i$ is connected.*

Proof. This is clearly true if C_i and C_j intersect once or less. If C_i and C_j intersect transversely twice—say, at (x_1, y_1) and (x_2, y_2) with $x_2 > x_1$—then the area above C_i but below C_j can only be disconnected if $C_j(x) > C_i(x)$ for $x < x_1$ and $x > x_2$, implying $C_j \succ C_i$. □

For all $1 \leq i \leq m$ and all intervals $[a, b]$, define $X[a, b]$ to be all points in X with x-coordinate in $[a, b]$, and define $X[a, b, i]$ to be $X[a, b] \setminus S_i$. Define $\mathcal{S}_{<i}$ to be the set $\{S_1, \ldots, S_{i-1}\}$ of all regions of width less than S_i. Let $M[a, b, i]$ denote the minimum cost of a solution to the weighted set cover problem on the range space $(X[a, b, i], \mathcal{S}_{<i})$ (with weights inherited from the original problem).

If such a covering does not exist, $M[a, b, i] = \infty$. For simplicity, we assume that C_m, the widest curve, contains no points in its downward shadow (that is, $X \cap S_m$ is empty). Our goal is then to determine $M[-\infty, \infty, m]$ via dynamic programming; the key structural result we need is the following:

Claim 4.2. *If $X[a, b, i]$ is non-empty, then*

$$M[a, b, i] = \min\Big\{ \min_{c \in (a,b)} \{M[a, c, i] + M[c, b, i]\}, \min_{j<i}\{M[a, b, j] + w_j\}\Big\}.$$

Proof. Clearly $M[a, b, i] \leq M[a, c, i] + M[c, b, i]$ for all $c \in (a, b)$. Also, for $j < i$, $M[a, b, j] + w_j$ is the cost of purchasing S_j and then covering the remaining points in $X[a, b]$ using regions less wide than S_j (and hence less wide than S_i). Thus $M[a, b, j] + w_j$ is a cost of a feasible solution to $(X[a, b, i], \mathcal{S}_{<i})$ and hence is at least $M[a, b, i]$. It follows that $M[a, b, i]$ is bounded above by the right hand side.

To show that $M[a, b, i]$ is bounded below by the right hand side, we let $\mathcal{Z} \subseteq \mathcal{S}_{<i}$ be a feasible set cover for $(X[a, b, i], \mathcal{S}_{<i})$. We consider two cases:

Case 1: There is some $c \in (a, b)$ such that $(c, C_i(c))$ is not covered by $\mathcal{Z}$. Let $\mathcal{Z}_{<c}$ be the set of all regions in $\mathcal{Z}$ containing a point in $X[a, c, i]$, and let $\mathcal{Z}_{>c}$ be the set of all regions in $\mathcal{Z}$ containing a point in $X[c, b, i]$. Let $Z \in \mathcal{Z}$. Since $Z \prec S_i$, by Proposition 4.1, $Z \setminus S_i$ is connected and thus cannot contain points both in $X[a, c, i]$ and $X[c, b, i]$. Hence $\mathcal{Z}_{<c} \cap \mathcal{Z}_{>c} = \emptyset$ and thus the cost of $\mathcal{Z}$ is at least $M[a, c, i] + M[c, b, i]$.

Case 2: For all $c \in (a, b)$, the point $(c, C_i(c))$ is covered by $\mathcal{Z}$. Then $\mathcal{Z}$ covers $X[a, b, i] \cup S_i$ and hence covers all points in $X[a, b]$. Let C_j be the widest curve in $\mathcal{Z}$, noting that $j < i$. Then the cost of $\mathcal{Z}$ is at least $w_j + M[a, b, j]$ since $\mathcal{Z} \setminus S_j$ must cover all points in $X[a, b, j]$.

It follows that $\mathcal{Z}$ must cost as much as either $\min_{c \in (a,b)}\{M[a, c, i] + M[c, b, i]\}$ or $\min_{j<i}\{M[a, b, j] + w_j\}$, and the result follows. □

Claim 4.2 immediately implies the existence of a dynamic programming algorithm to compute $M[-\infty, \infty, m]$ and return a cover having that cost. There are at most $n + 1$ combinatorially relevant values of a and b when computing optimal costs $M[a, b, i]$ for subproblems, so there are $O(mn^2)$ distinct values of $M[a, b, i]$ to compute. Recursively computing $M[a, b, i]$ requires $O(m + n)$ table lookups, so the total running time of our algorithm is $O(mn^2(m+n))$, assuming a representation allowing primitive operations in $O(1)$ time.

4.2 Hitting Set

We next prove Theorem 1.9, giving a very simple and clean dynamic programming algorithm for weighted hitting set with shadows of horizontal line segments in the plane (bottomless rectangles, or simply *rectangles* for short). We suppose we are given a range space consisting of a set X of n points and a family $\mathcal{S}$ of m rectangles. We assume that the number of distinct x-coordinates in

the input is $O(\min(m,n))$, sorting the input if necessary to rewrite it (taking $O(n \log n + m \log m)$ time).

Define $\mathcal{S}[a,b]$ to be the family of all rectangles lying entirely inside $(a,b) \times (-\infty,\infty)$, and define $M[a,b]$ to be the minimum cost of hitting all rectangles in $\mathcal{S}[a,b]$. Define $y[a,b]$ to be the minimum y-coordinate of a top edge of a rectangle in $\mathcal{S}[a,b]$, or ∞ if $\mathcal{S}[a,b]$ is empty. Finally, define $X[a,b]$ to be the set of all $x \in X$ lying in $(a,b) \times (-\infty, y[a,b]]$.

Whenever $\mathcal{S}[a,b]$ is non-empty, hitting all rectangles in $\mathcal{S}[a,b]$ requires choosing at least one point in $X[a,b]$ (otherwise the lowest rectangle in $\mathcal{S}[a,b]$ will not be hit). However, any $x = (x_1, y_1) \in X[a,b]$ will hit all rectangles in $\mathcal{S}[a,b]$ whose horizontal range contains x_1, since x is lower than the top edge of all rectangles in $\mathcal{S}[a,b]$. After choosing such an x, it thus only remains to cover those rectangles in $\mathcal{S}[a,x_1]$ and $\mathcal{S}[x_1,b]$. Via this argument, the following recurrence is immediate:

Claim 4.3. *We have*

$$M[a,b] = \min_{x=(x_1,y_1) \in X[a,b]} \{M[a,x_1] + M[x_1,b] + w_x\}.$$

Via this claim, it is simple to implement a dynamic programming algorithm that computes $M[-\infty,\infty]$ and returns a hitting set having that cost. There are at most $O(\min(m,n))$ combinatorially relevant values of a and b for which subproblems must be computed, for a total of $O(\min(m,n)^2)$ subproblems. Moreover, each subproblem requires at most $O(m)$ time to find $y[a,b]$ and then $O(n)$ time to iterate through X and perform table lookups, so the total running time of the algorithm is $O((m+n)\min(m,n)^2 + n \log n + m \log m)$. This completes the proof of Theorem 1.9.

4.3 Pack-Points

Here, we prove Theorem 1.11 by giving an algorithm for the weighted pack-points problem involving downward shadows of 2-intersecting x-monotone curves. We borrow the notation from the proof of Theorem 1.6, letting X be the set of n points and $\{C_1, \ldots, C_m\}$ be the family of curves with downward shadows $\{S_1, \ldots, S_m\}$. Again, for simplicity, we assume that each C_i is the graph of a smooth univariate function with domain $(-\infty,\infty)$, that all intersections are transverse, and that no points in X lie on any curve C_i. We again assume without loss of generality that $C_i \succ C_j$ whenever $i > j$, where $\succ$ is a total ordering based on the *width* notion introduced in the proof of Theorem 1.6. We assume that each point is contained in at least one region (otherwise we simply take all points contained in no regions and consider the problem that remains).

We define a set of points to be *independent* whenever no region S_i contains two or more of them. For real numbers a and b, we denote by $X[a,b]$ the set of all points in X with x-coordinate in $[a,b]$. For $a, b \in \mathbb{R} \cup \{-\infty,\infty\}$, we define $X[a,b,i]$ to be the set of all points in $X[a,b]$ that are independent from the point $(x, C_i(x))$ for all $x \in (\{a,b\} \setminus \{-\infty,\infty\})$.

For a point $x \in X$, we let the *lowness* $\ell(x)$ of x be the largest value of i such that $x \in S_i$ (noting that this is well defined since each point is assumed to lie in at least one region). Observe that no independent set of points may ever contain two points having the same lowness. Immediately, we observe the following:

Proposition 4.4. *If $x \in X[a, b, i]$, then we have:*

(1) $\ell(x) \leq i$, and

(2) $X[a, b, \ell(x)] \subseteq X[a, b, i]$.

Proof. Write $x = (x_1, y_1)$. Suppose $\ell(x) = j$. Then S_j contains x but neither $C_i(a)$ nor $C_i(b)$ since $x \in X[a, b, i]$. From this it follows that $C_j(a) < C_i(a)$, $C_j(x_1) > C_i(x_1)$, and $C_j(b) < C_i(b)$, so $S_i \setminus S_j$ is disconnected and thus $j \leq i$ by Proposition 4.1, implying item (1). For item (2), since $C_j(a) < C_i(a)$ and $C_j(b) < C_i(b)$, any point independent from $(a, C_j(a))$ and $(b, C_j(b))$ is also independent from $(a, C_i(a))$ and $(b, C_i(b))$, so $X[a, b, j] \subseteq X[a, b, i]$. □

For $a, b \in \mathbb{R} \cup \{-\infty, \infty\}$, we define $M[a, b, i]$ to be the maximum total weight of an independent subset of $X[a, b, i]$. We denote by M the maximum weight of an independent subset of X. Our dynamic programming algorithm shall follow immediately from the following recurrence:

Claim 4.5. *We have the following:*

(1) If $X[a, b, i]$ is non-empty, then

$$M[a, b, i] = \max_{x=(x_1,y_1)\in X[a,b,i]} \{M[a, x_1, \ell(x)] + M[x_1, b, \ell(x)] + w_x\}.$$

(2) We have

$$M = \max_{x=(x_1,y_1)\in X} \{M[-\infty, x_1, \ell(x)] + M[x_1, \infty, \ell(x)] + w_x\}.$$

Proof. We give a full proof of (1); item (2) is similar. We first show that

$$M[a, b, i] \geq \max_{x=(x_1,y_1)\in X[a,b,i]} \{M[a, x_1, \ell(x)] + M[x_1, b, \ell(x)] + w_x\}.$$

We let $x = (x_1, y_1) \in X[a, b, i]$, and shall prove that $M[a, b, i] \geq M[a, x_1, \ell(x)] + M[x_1, b, \ell(x)] + w_x$. Suppose $Y_1 \subseteq X[a, x_1, \ell(x)]$ and $Y_2 \subseteq X[x_1, b, \ell(x)]$ with Y_1 having total weight $M[a, x_1, \ell(x)]$ and Y_2 having total weight $M[x_1, b, \ell(x)]$. Observe that $\{x\} \cup Y_1$ and $\{x\} \cup Y_2$ are independent sets since $X[a, x_1, \ell(x)]$ and $X[x_1, b, \ell(x)]$ contain only points independent from x. Moreover, $Y_1 \cup Y_2$ is necessarily independent, since any region S_j containing a point in Y_1 must necessarily have $j \leq i$ by Proposition 4.4 (and hence $S_j \setminus S_i$ connected), and thus $S_j \setminus S_i$ must lie entirely to the left of point x and S_j may then contain no point in Y_2. From this, it follows that $Y_1 \cup Y_2 \cup \{x\}$ is an independent set and so must lie in $X[a, b, i]$.

Finally, since $Y_1 \cup Y_2 \cup \{x\}$ has total weight $M[a, x_1, \ell(x)] + M[x_1, b, \ell(x)] + w_x$, we must necessarily have $M[a, b, i] \geq M[a, x_1, \ell(x)] + M[x_1, b, \ell(x)] + w_x$.

To prove that

$$M[a, b, i] \leq \max_{x=(x_1,y_1) \in X[a,b,i]} \{M[a, x_1, \ell(x)] + M[x_1, b, \ell(x)] + w_x\},$$

we suppose we are given an independent set $Y \subseteq X[a, b, i]$ having total weight $M[a, b, i]$. Let $x = (x_1, y_1)$ be any point in Y. Then all other points in $Y \backslash \{x\}$ are independent from (x_1, y_1) and are thus independent from $(x_1, C_{\ell(x)})$. Therefore all points in Y that are left of x lie in $X[a, x_1, \ell(x)]$, and all points in Y that are right of x lie in $X[x_1, b, \ell(x)]$. It follows that Y has total weight at most $\{M[a, x_1, \ell(x)] + M[x_1, b, \ell(x)] + w_x$. This completes the proof of the claim. □

Claim 4.5 implies the existence of a dynamic programming algorithm to compute M and return an independent set having that cost. As in the proof of Theorem 1.6, there are $O(mn^2)$ distinct values of $M[a, b, i]$ to compute. Each $M[a, b, i]$ value can be recursively computed using $O(n)$ table lookups, so the total running time of our algorithm is $O(mn^3)$ under the usual assumptions.

4.4 Pack-Regions

Finally, we give a proof of Theorem 1.13 via a dynamic programming algorithm for the pack-regions problem on downward shadows of horizontal line segments (again, we shall refer to these regions as *rectangles* in what follows). As usual, we assume we have a set X of n points and a family $\mathcal{S}$ of m rectangles and assume that the number of distinct x-coordinates in the input is $O(\min(m, n))$, sorting and rewriting the input to guarantee this. We shall say that a set of rectangles is *independent* if no pair of them intersect at a point in X. We again define $\mathcal{S}[a, b]$ to be the family of all rectangles lying entirely inside $(a, b) \times (-\infty, \infty)$, and this time define $M[a, b]$ to be the maximum weight of an independent set of rectangles in $\mathcal{S}[a, b]$ (that is, a set of rectangles, no two of which intersect at a point in X). Furthermore, for a rectangle $R \in \mathcal{S}$, we define L_R to be the set of all points (x, y) such that there is a point $(x, z) \in X \cap R$, and define $\mathcal{S}[a, b, R]$ to the the set of rectangles in $\mathcal{S}[a, b]$ that do not intersect L_R (that is, rectangles in $\mathcal{S}[a, b]$ that are not stabbed by any vertical lines passing through the points in $X \cap R$). Finally, we define $M[a, b, R]$ to be the maximum weight of an independent set in $\mathcal{S}[a, b, R]$.

To improve the running time of our dynamic program, we use two interdependent recurrences this time. Both are relatively straightforward.

Claim 4.6. *We have*

$$M[a, b] = \max_{R \in \mathcal{S}[a,b]} \{M[a, b, R] + w_R\}.$$

Proof. It is clear that $M[a, b] \geq M[a, b, R] + w_R$ for any $R \in \mathcal{S}[a, b]$, since any independent set in $\mathcal{S}[a, b, R]$ is, with the addition of R, still an independent set

in $\mathcal{S}[a,b]$. To prove that $M[a,b]$ is at most the right hand side, it suffices to let $\mathcal{Z} \subseteq \mathcal{S}[a,b]$ be an independent set of rectangles of total weight $M[a,b]$, and take R to be the rectangle in $\mathcal{Z}$ whose top edge is lowest. None of the rectangles in $\mathcal{Z} \setminus \{R\}$ can intersect L_R, because they all have a higher top edge than R and would thus intersect R at a point in X if ever they are stabbed by a line in L_R. Thus the remaining weight of the rectangles in $\mathcal{Z} \setminus \{R\}$ is at most $M[a,b,R]$, and hence the total weight of all the rectangles in $\mathcal{Z}$ is at most $M[a,b,R] + w_R$. The result follows. □

For the next result, we let $X[a,b]$ be the set of all points in X whose x-coordinate is in (a,b). Our second recurrence follows:

Claim 4.7. *Let $R \in \mathcal{S}[a,b]$. We have the following:*

(1) If $X[a,b] \cap R$ is empty, then $M[a,b,R] = M[a,b]$.

(2) If $X[a,b] \cap R$ is non-empty and (x_1,y_1) is the leftmost point in $X[a,b] \cap R$, then $M[a,b,R] = M[a,x_1] + M[x_1,b,R]$.

Proof. The first statement is immediate from the definitions. For the second statement, we simply observe that an independent set of rectangles in $\mathcal{S}[a,b,R]$ of weight $M[a,b,R]$ can always be partitioned into rectangles entirely left of (x_1,y_1) (having weight $M[a,x_1]$) and those entirely right of (x_1,y_1) (having weight $M[x_1,b,R]$). □

Via our two recurrence relations, it is simple to implement a dynamic programming algorithm that computes $M[-\infty,\infty]$ and returns a packing having that cost. As in the proof of Theorem 1.9, there are at most $\min(m,n)$ combinatorially relevant values of a and b for which subproblems must be computed. Computing each of the $O(\min(m,n)^2)$ subproblems of the form $M[a,b]$ requires $O(m)$ table lookups, for a total of $O(m\min(m,n)^2)$ time. For the $O(m\min(m,n)^2)$ subproblems of the form $M[a,b,R]$, we observe that each can be completed in $O(1)$ time with sufficient precomputation. To achieve the desired running time, it suffices to build a table containing, for each relevant a and b, the leftmost point in $X[a,b] \cap R$ for each R (or a note that $X[a,b] \cap R$ is empty). Such a table can easily be built in $O(m\min(m,n)^2)$ time. The total running time of our algorithm is then $O(m\min(m,n)^2 + n\log n + m\log m)$.

Acknowledgements

We thank Esther Ezra for discussions on set cover and hitting set for rectangles whose boundaries intersect pairwise exactly zero times or four times; and Sariel Har-Peled for discussions on the hardness of the pack-points and pack-regions problems.

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) (2010), 3248-3282.

[2] P. Alimonti and V. Kann. Some APX-completeness results for cubic graphs. *Theoretical Comp. Sci.* 237 (2000) 123-134.

[3] C. Ambühl, T. Erlebach, M. Mihalák, and M. Nunkesser. Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs. In *APPROX and RANDOM* (2006) 3-14.

[4] P. K. Agarwal, E. Nevo, J. Pach, R. Pinchasi, M. Sharir, and S. Smorodinsky. Lenses in arrangements of pseudo-circles and their applications. *J. ACM* 51(2) (2004), 139-186.

[5] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications (Third edition).* Springer-Verlag, Heidelberg, (2008).

[6] P. Berman and B. DasGupta. Complexities of efficient solutions of rectilinear polygon cover problems. *Algorithmica* 17(4) (1997), 331-356.

[7] H. Brönnimann and O. Devillers. The union of unit balls has quadratic complexity, even if they all contain the origin. arXiv:cs/9907025v1 [cs.CG] (1999).

[8] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete Comput. Geom.* 14 (1995), 263-279.

[9] N. Bansal and K. Pruhs. The geometry of scheduling. *IEEE 51st Annual Symposium on Foundations of Computer Science* (2010), 407-414.

[10] T. M. Chan and E. Grant. Exact Algorithms and APX-Hardness Results for Geometric Set Cover. In *Proc. 23rd Canadian Conference on Computational Geometry* (2011) 431-436.

[11] D. Chakrabarty, E. Grant, and J. Koenemann. On column-restricted and priority covering integer programs. In *Integer Programming and Combinatorial Optimization* (2010) 355-368.

[12] T. M. Chan and S. Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. In *Proc. 25th Annu. ACM Sympos. Comput. Geom.* (2009) 333-340.

[13] H. S. M. Coxeter. Introduction to geometry, second edition. John Wiley & Sons Inc., New York, 1969.

[14] K. Clarkson and K. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete Comput. Geom.* 37 (2007), 43-58.

[15] A. Ene, S. Har-Peled, and B. Raichel. Geometric packing under non-uniform constraints. arXiv:1107.2949v1 [cs.CG] (2011).

[16] G. Even, D. Rawitz, and S. Shahar. Hitting sets when the VC-dimension is small. *Information Processing Letters* 95(2) (2005), 358-362.

[17] T. Erlebach and E. J. van Leeuwen. PTAS for weighted set cover on unit squares. In *APPROX and RANDOM* (2010), 166-177.

[18] S. Har-Peled. Being Fat and Friendly is Not Enough. arXiv:0908.2369v1 [cs.CG] (2009).

[19] S. Har-Peled and M. Lee. Weighted geometric set cover problems revisited. Unpublished manuscript, (2008).

[20] D.S. Hochbaum and W. Maass. Fast approximation algorithms for a nonconvex covering problem. *J. Algorithms* 8(3) (1987), 305-323.

[21] N.H. Mustafa and S Ray. Improved results on geometric hitting set problems. *Discrete Comput. Geom.* 44(4) (2010), 883-895.

[22] J. Pach and G. Tardos. Tight lower bounds for the size of epsilon-nets. In *Proc. 27th ACM Sympos. Comput. Geom.* (2011) 458-463.

[23] C.H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Systems Sci.* 43 (1991), 425-440.

[24] K. Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *ACM Symposium on Theory of Computing* (2010) 641-648.

[25] E. J. van Leeuwen. *Optimization and Approximation on Systems of Geometric Objects.* PhD thesis, Universiteit van Amsterdam, (2009).