Approximation Algorithms for Geometric Covering Problems for Disks and Squares

by

Nan Hu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Geometric covering is a well-studied topic in computational geometry. We study three covering problems: *Disjoint Unit-Disk Cover*, *Depth-*($\leq K$) *Packing* and *Red-Blue Unit-Square Cover*.

In the Disjoint Unit-Disk Cover problem, we are given a point set and want to cover the maximum number of points using disjoint unit disks. We prove that the problem is NP-complete and give a *polynomial-time approximation scheme* (PTAS) for it.

In Depth-($\leq K$) Packing for Arbitrary-Size Disks/Squares, we are given a set of arbitrary-size disks/squares, and want to find a subset with depth at most K and maximizing the total area. We prove a *depth reduction theorem* and present a PTAS.

In Red-Blue Unit-Square Cover, we are given a red point set, a blue point set and a set of unit squares, and want to find a subset of unit squares to cover all the blue points and the minimum number of red points. We prove that the problem is NP-hard, and give a PTAS for it. A *"mod-one"* trick we introduce can be applied to several other covering problems on unit squares.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Timothy M. Chan for guiding me in my research. Without his guidance, this thesis would not have been possible.

I would like to thank my family for their love and support.

Table of Contents

List of Figures

Chapter 1

Introduction

Geometric covering is an important topic in computational geometry. Geometric covering problems involve selecting geometric objects, such as disks and squares, to cover another set of objects (often points). These problems are motivated by applications in wireless networks and facility location, such as selecting transmitter locations with minimum inference, or building facilities with minimum transportation costs.

Geometric covering problems usually have a "continuous" version and a "discrete" version. In the continuous version, the object positions are not given, and the problems are to find the optimal positions for the objects. For example, in the *Unit-Disk Cover* problem, given a point set, one wants to find the minimum number of unit disks[1] to cover all the points. The unit disks can be placed at any position in the plane. In the discrete version, one is given a finite set of geometric objects, and wants to find the optimal subset. The discrete version of the Unit-Disk Cover problem will be called *Geometric Set Cover for Unit Disks*. Given a point set and a set of unit disks, the problem is to find the smallest subset of disks to cover all the points.

Geometric covering problems are natural to consider. For example, to find the minimum number of transmitters to cover all the clients in wireless networks, it is natural to model the clients as points, and the transmitters as disks. For some geometric problems, better approximation results have been found than for the general combinatorial versions of the problems.

[1] All the disks and squares in this dissertation are assumed to be in general position, and all the squares are axis-aligned.

1.1 The Set Cover Problem and Variations

Set Cover. One of the most important and best-known covering problem is the *Set Cover* problem. Given a set of elements and a collection of sets, the Set Cover problem is to find the smallest subcollection to cover all the elements. In the weighted version, a non-negative weight is associated with each set, and the objective is to find the subcollection of the minimum total weight. The general Set Cover problem cannot be approximated[2] to within the factor of $(1 - o(1)) \ln n$ in polynomial time, unless NP $\subset$ DTIME($n^{O(\log \log n)}$) [18]. A natural greedy approach can find a $(1 + \ln n)$-approximation algorithm for both the weighted and unweighted version [27, 30, 9].

In the Geometric Set Cover problem, the elements are always points, and the sets are geometric objects. Weights are associated with the geometric objects in the weighted version. Constant-factor approximation algorithms and even polynomial-time approximation schemes (PTASes) have been found for specific instances of the Geometric Set Cover problem. Brönnimann and Goodrich [4] proved that for a set system with finite VC-dimension, if one can compute so called ε-*nets* of size c/ε in polynomial time for some constant c, there is a c-approximation algorithm in polynomial time. Clarkson and Varadarajan [11] showed that small size ε-nets can be found in polynomial time for geometric objects with low *union complexity*, and approximation algorithms for various geometric objects have been found, such as an $O(\log \log OPT)$-approximation algorithm for fat triangles [2] and a constant-factor approximation algorithm for cubes in $\mathbb{R}^3$ [11]. The *linear programming* relaxation technique was also used as an alternative [17, 29] to Brönnimann and Goodrich's algorithm.

In 1985, Hochbaum and Maass [24] introduced a unified way, the *shifting technique*, to find PTASes for continuous geometric covering problems for fat objects of roughly the same size, such as the Unit-Disk Cover problem. Mustafa and Ray [31] gave the first PTAS for the discrete version of Set Cover for unit squares by the *local search* technique. However, the local search technique cannot be applied to the weighted versions. Erlebach and van Leeuwen [16] gave a PTAS for the Geometric Weighted Set Cover for Unit Squares, using the shifting technique together with a sophisticated dynamic programming algorithm.

[2] *Approximation algorithm* produces a solution "near" the optimal, often in polynomial time. For a maximization (or minimization) problem with optimal objective function value OPT, an algorithm is a factor-c approximation algorithm if it produces a solution with objective function value at least OPT/c (or at most $c \cdot OPT$). An algorithm is called a *polynomial-time approximation scheme* (PTAS) if it takes ε as an input, for any $0 < \varepsilon < 1$, and produces a solution with objective function value at least $(1 - \varepsilon)OPT$ (or at most $(1 + \varepsilon)OPT$) in polynomial time for constant ε.

Variations of Set Cover. Many variations of the Set Cover problem have also been studied. One natural variation is the *Maximum Coverage* problem. Given several sets and an integer k, while the sets may have some elements in common, the problem is to select at most k sets to cover the maximum number of elements. In the weighted version, each element has a profit, each set has a cost, and the problem is to find a collection of sets with total cost within a given budget, while maximizing the total profit of elements covered. For this problem, $(1 - \frac{1}{e})$-approximation algorithms have been found for both the unweighted and weighted versions [23, 28, 39]. Feige [18] proved that $1 - \frac{1}{e}$ is the best possible approximation ratio for polynomial-time algorithms, unless P = NP. For the geometric version, Erlebach and van Leeuwen [16] gave a PTAS for the Weighted Maximum Coverage problem where sets are unit squares and elements are points.

The *k-Partial Cover* problem is another natural variation of the Set Cover problem. Given a set of elements, a collection of sets and an integer k, the problem is to find the smallest subcollection of sets to cover at least k elements. The k-Partial Cover problem can be viewed as a general version of the Set Cover problem. The classical bounds and performance of the greedy algorithm for approximating the Set Cover problem has been proved valid for k-Partial Cover [38]. Gandhi et al. [19] studied the continuous geometric version, where the elements are points, and one wants to find the smallest number of unit disks to cover at least k points. The positions of the unit disks are not given. They obtained a PTAS using Hochbaum and Maass's shifting technique and additional ideas.

Demaine and Feige [12] introduced the *Unique Coverage* problem. Given a set of elements and a collection of sets, the Unique Coverage problem is to find sets to maximize the number of elements that are covered exactly once. One application is in wireless networks: when building the transmitters, we want to maximize the number of clients that receive signals without interference. Demaine and Feige [12] proved that for any $\varepsilon > 0$, the problem is inapproximable to within $O(1/\log^{\sigma(\varepsilon)} n)$ in polynomial time, assuming that NP $\not\subseteq$ BPTIME($2^{n^{\varepsilon}}$), where $\sigma(\varepsilon)$ is some constant dependent on ε. They also gave an $\Omega(1/\log n)$-approximation algorithm. Erlebach and van Leeuwen [15] introduced the geometric version where the elements are points and the sets are geometric objects. They proved that the problem remains NP-hard when the objects are unit disks and gave a constant-factor approximation algorithm. Ito et al. [26] gave the first PTAS for Geometric Unique Coverage for Unit Squares using the shifting technique together with a sophisticated dynamic programming algorithm.

The *Red-Blue Set Cover* problem was introduced by Carr et al. [5]. Given a set of blue elements, a set of red elements and a collection of sets, the problem is to find a subcollection to cover all the blue elements, and minimize the red elements covered. The problem arises in data mining. For example, given a large set of data records including valid records (blue

elements) and fraudulent records (red elements), and different classifications (sets) of these records where each classification may contain both valid and fraudulent records, we want to find a new classifier which covers all the valid records but covers a minimum number of fraudulent records. Carr et al. [5] proved the inapproximability to within $2^{O(\log^{1-\delta} n)}$ in polynomial time, where $\delta = 1/(\log\log n)^c$, for any constant $c < 1/2$, even when every set contains only one blue element and two red elements. They also gave a $2\sqrt{n}$-approximation algorithm where every set contains only one blue element.

1.2 Geometric Packing

Geometric Packing, which is a geometric version of *Maximum Independent Set*, is a related class of geometric optimization problems. The Maximum Independent Set problem is a well-studied problem in graph theory. Given an undirected graph, the problem is to find the largest subset of vertices (or the subset of the maximum total weight in the weighted version), in which no two vertices are adjacent. The Maximum Independent Set problem is known to be NP-complete. In the Geometric Packing problem, given a set of geometric objects, one wants to find the largest subset in which no two objects intersect. In the weighted Geometric Packing problem, each object has a non-negative weight. A common choice for the weight is the area (or volume in high dimensions) of each geometric object. The Geometric Packing problem for fat objects of roughly the same size can be solved using Hochbaum and Maass's shifted grid technique. Erlebach et al. [14] gave a PTAS for disks of arbitrary radii with running time $n^{O(1/\varepsilon^2)}$ with a modified shifted grid algorithm. Chan [6] improved the time complexity to $n^{O(1/\varepsilon)}$.

The Geometric Packing problem is related to a mathematical problem raised by T. Rado [35] in 1928. The problem is: what is the largest value ξ such that, for any set S of squares, there is always an independent set $I \subseteq S$ with $\mathit{area}(I)/\mathit{area}(S) \geq \xi$, where $\mathit{area}(\bullet)$ is defined as the area of $\bullet$. The conjecture of Rado is $\xi = 1/4$. A simple greedy approach can prove that $\xi > 1/9$. R. Rado [32, 33, 34] improved the bound to $\frac{\pi}{8\sqrt{3}} \approx 1/4.4107$. Many researchers have studied the problem, but a tight bound has not been found yet.

1.3 Our Results and Motivation

We study three problems in this dissertation:

1. *Disjoint Unit-Disk Cover*,

2. *Depth-$\leq K$ Packing for Arbitrary-Size Disks/Squares*, and

3. *Red-Blue Unit-Square Cover*.

1.3.1 Disjoint Unit-Disk Cover

A disk configuration puzzle has been studied by several researchers recently [1, 25, 40]: what is the smallest set of points so that no collection of disjoint unit disks can cover the entire point set. We consider an algorithmic version of the problem.

Definition 1 (Disjoint Unit-Disk Cover) *Give a point set P, the Disjoint Unit-Disk Cover problem is to find a set of disjoint unit disks to cover the maximum number of points.*[3]

Similar to the Unique Coverage Problem, the disjointness constraint may be useful for the avoidance of inference in wireless networks. We prove that the combinatorial problem is NP-complete by reducing from the *Planar 3SAT* problem, and give a PTAS using the shifting technique. The above problem is a continuous problem. The hardness result directly implies that the discrete version of the problem is NP-complete as well.

We notice that Hearn [22] has proved the same result by reducing the problem from *Constraint Graph Satisfiability*. Our work is independent to Hearn's work.

1.3.2 Depth-($\leq K$) Packing

Given a set of objects $\mathcal{O}$ and a point p, we define the depth of p to be the number of objects in $\mathcal{O}$ that contain p. When saying the depth of $\mathcal{O}$, we refer to the maximal depth of all the points covered by $\bigcup \mathcal{O}$.

Definition 2 (Depth-($\leq K$) Packing for Arbitrary-Size Disks/Squares) *Given a set T of disks/squares of arbitrary sizes, and a non-constant integer K, the problem is to find a subset of disks/squares with depth at most K and the maximum total area of the union.*

The Depth-($\leq K$) Packing problem is a general version of the Geometric Packing problem. When $K = 1$, any point is allowed to be covered at most once, and the problem is to find the independent set of maximal area. We give a PTAS for arbitrary-size disks/squares.

[3] The disks are closed disks, but two disks touching each other are viewed as disjoint.

The key to our PTAS is a *depth reduction theorem*: given a set T of arbitrary-size disks/squares, the theorem finds a subset of depth $O(1/\delta^4)$ and covers a fraction of at least $1-\delta$ of the area, for any $0 < \delta < 1$. The theorem is related to T. Rado's problem, which is about finding the best fraction for subsets of depth 1, whereas our theorem gives asymptotic bounds on the fraction as a function of the depth. The theorem is also a generalization of Brass et al.'s theorem [3] for *k-Coloured Disk Packing*. Given a set A of unit disks and k colours where $k \in \{i^2 + ij + j^2 | i, j \in \mathbb{N}\}$, they proved that one can select and colour a subset S of disks such that the same-coloured disks are disjoint and $area(A)/area(T) \geq \frac{1}{(1+\delta_k)^2}$, where $\delta_k = \frac{2}{\sqrt{3}}\left(\frac{2}{\sqrt{k}-\frac{2}{3}}\right)$. Since it is known that a set of disks of depth k can be coloured using $O(k)$ colours so that the same-coloured disks are disjoint [7, 37], our theorem implies that given a set T of arbitrary-size disks, one can select and colour a subset of disks using $O(k)$ colours which covers a fraction of at least $(1 - \frac{1}{\sqrt[4]{k}})$ of the area. This is similar to Brass et al.'s result, but for arbitrary-size disks.

The theorem may have other applications where one needs to reduce the depth of a set without losing much area of the union.

1.3.3 Red-Blue Unit-Square Cover

As mentioned, the general Red-Blue Set Cover is hard to approximate. We study a geometric version of the problem where the elements are points and the sets are unit squares.

Definition 3 (Red-Blue Unit-Square Cover) *Given a blue point set B, a red point set R, and a set T of unit squares, the problem is to find a subset of T such that all the blue points are covered, minimizing the number of red points covered.*

We prove that the geometric version remains NP-hard. We present the first PTAS for Geometric Red-Blue Set Cover.

There have already been a number of PTASes for problems related to geometric covering problems in the literature. We note that most of the known techniques can be classified into a few categories: 1. Hochbaum and Maass's original shifted grid technique; 2. Mustafa and Ray's local search technique; 3. sophisticated *dynamic programming* combined with shifting technique, such as Erlebach and van Leeuwen's [16] and Ito et al.'s [26] PTASes that we have mentioned before. Our PTAS belongs to the third category.

Arguably the most interesting aspect of our work lies not so much in the specific result about Red-Blue Set Cover, but in our technique, which can be applied to several other

geometric covering problems for unit squares, such as Geometric (Weighted) Set Cover, Unique Coverage, Budget Maximum Coverage and Geometric k-Partial Cover and other potential problems.

We feel our technique is conceptually simpler than Erlebach and van Leeuwen's [16] PTAS for Geometric Weighted Set Cover and Ito et al.'s PTAS for Unique Coverage [26]. The descriptions of the dynamic programming algorithm in both papers are lengthy. For example, the algorithm by Erlebach and van Leeuwen is obtained by simulating a plane sweep that involves multiple sweep lines moving at different speeds. We get around most of the complications by one very simple idea: a "*mod-one* trick". Our work will be presented at the Canadian Conference on Computational Geometry 2013 [8].

1.4 Organization

In chapter 2, we review the known shifted grid and shifted quadtree techniques, which are the main techniques we use in our work. In chapter 3, we give an NP-hardness proof of Disjoint Unit-Disk Cover, and a simple PTAS for it. In chapter 4, we prove the depth reduction theorem, and give a PTAS for Depth-($\leq K$) Packing. In chapter 5, we introduce a "mod-one" trick, and present a PTAS for Red-Blue Unit-Square Cover. We also show how "mod-one" trick could be applied to other problems to find PTASes. Chapter 6 is the conclusion, where we discuss some open problems related to our work.

Chapter 2

Background

In this chapter, we discuss two main techniques we use in our work: Hochbaum and Maass's shifted grid technique [24] and Chan's shifted quadtree technique [6].

2.1 Hochbaum and Maass's Shifted Grid Technique

Hochbaum and Maass introduced the shifted grid technique which is applicable to a broad range of geometric covering and packing problems [24]. Generally, the shifting technique has two parts. The first part is to find a local algorithm which solves the problem within a bounded area in polynomial time. The local algorithm can either be an exact algorithm or an approximation algorithm. In the second part, the instance of the general problem is partitioned into bounded areas. Each area is solved using the local algorithm, and the solutions are combined. The final solution is the "best" solutions among all the possible partitions.

We use the same problem as [24], the *Unit Disk Cover* problem, as an example of the shifting technique. Given a point set in the plane, the problem is to find the smallest number of unit disks to cover all the points.

For a $k \times k$ square, while k is a constant, one can cover the whole square using $O(k^2)$ unit disks. For a point set P' with all points inside the $k \times k$ square, there are at most $O(k^2)$ disks in the optimal solution. Furthermore, we can always move a disk in the optimal solution, so that there are two points on its boundary. Then there are $O(n^2)$ positions for the disks, where n is the size of the point set. The complexity analysis still holds for the

special case where some disk covers only one point. A simple brute-force algorithm can find the optimal solution in $O(n^{O(k^2)})$ time. This is the local algorithm for a bounded area.

We draw a uniform grid of unit side length. For a constant integer k, we partition the plane into $k \times k$ squares by grid lines $x = ik + a$ and $y = jk + b$, where $0 \le a, b < k$ and $a, b, i, j \in \mathbb{Z}$. We call each partition a "shift", and there are k^2 different shifts in total. For a shift with shift indices (a, b), we use the local algorithm to find the optimal solution in each $k \times k$ square. Let OPT denote the optimal disk set, and let $OPT^{(a,b)}$ be the union of all the local solutions. Let $\widetilde{OPT}^{(a,b)}$ denote the set of disks in OPT intersecting lines $X = ik + a$ or $Y = jk + b$, with $i, j \in \mathbb{Z}$. We have

$$\left|OPT^{(a,b)}\right| \le |OPT| + 3\left|\widetilde{OPT}^{(a,b)}\right|,$$

and when we consider all the shifts,

$$\sum_{0\le i,j<k}\left|OPT^{(i,j)}\right| \le \sum_{0\le i,j<k}(|OPT| + 3\left|\widetilde{OPT}^{(i,j)}\right|) = k^2|OPT| + 3\sum_{0\le i,j<k}\left|\widetilde{OPT}^{(i,j)}\right|),$$

since

$$\sum_{0\le i,j<k}\left|\widetilde{OPT}^{(i,j)}\right| \le 2k|OPT|,$$

implying that

$$\sum_{0\le i,j<k}\left|OPT^{(i,j)}\right| \le (k^2 + 6k)|OPT|$$

and

$$\min_{0\le i,j<k}\left|OPT^{(i,j)}\right| \le \left(1 + \frac{6}{k}\right)|OPT|.$$

Setting $k = \lceil 6/\varepsilon \rceil$ gives a $(1 + \epsilon)$-approximation algorithm (PTAS).

The shifted grid technique can also be modified to solve other discrete covering problems. The hard part of applying shifted grid technique to other covering problems is often in designing the local algorithm.

2.2 Chan's Shifted Quadtree Technique

Based on the shifted grid technique, Chan [6] presented a shifted quadtree technique, a generalization of the shifted grid technique, for the Geometric Packing problems for *fat*

objects[1] in d dimensions, with any constant $d \geq 2$ [6]. In this section, we will give a sketch of the shifted quadtree technique for the example of *Geometric Independent Set for Arbitrary-Size Squares*. Given a set of squares of arbitrary sizes, the problem is to find the largest subset in which no two squares intersect.

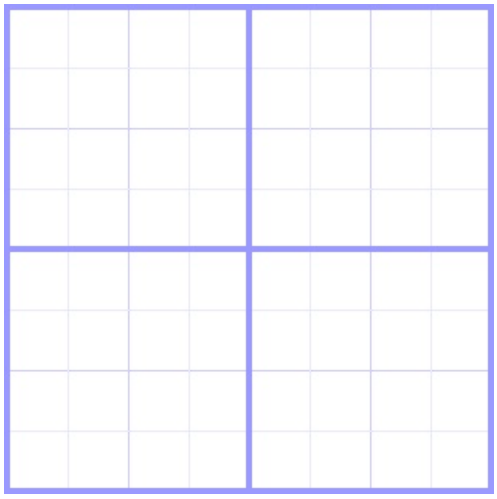

Figure 2.1: A region quadtree.

The quadtree in 2D represents a partition of space by decomposing the region into four equal quadrants, each of which is decomposed into four equal subquadrants, and so on. See Figure 2.1. We call the regions from this decomposition the quadtree cells.

To solve Geometric Independent Set for Arbitrary-Size Squares, we draw a quadtree over $[0, 1) \times [0, 1)$ and rescale so that all the squares are inside a $[0, 1) \times [0, 1)$ square. For a constant integer $k > 2$ and some integer $0 \leq j \leq k$, a shift of the quadtree with respect to j involves moving the whole quadtree by a vector of $(j/k, j/k)$. There are k different shifts.

A square of size r is called *k-aligned*, if it is inside a quadtree cell of size at most kr. Given a constant number k and a shift, we consider all the squares that are k-aligned. For a quadtree cell R of size r, any square s intersecting R must have size at least r/k, otherwise s should be inside a quadtree cell of size smaller than r by the definition of k-aligned. Given an independent set, the number of squares intersecting R is at most $4r/(r/k) = 4k$, which is a constant.

[1]Fat objects: "A collection $\mathcal{C}$ of objects is fat if for any r and hypercube of diameter R, we can choose a constant number c of points such that every object that intersects R and has diameter at least r contains one of the chosen points." (Definition from [6].)

For a set S of squares and a quadtree cell R, let S_R and $S_{|\partial R}$ denote the subset of squares in S that are inside R and intersect R respectively. For an independent subset $B \subseteq S_{|\partial R}$, define $pack[R, B]$ to be the maximum cardinality of a subset $A \subseteq S_R$ such that $A \cup B$ is an independent set. We then have

$$pack[R, B] = \max_{B'} \left(\sum_{i=1}^{4} pack[R_i, (B' \cup B)_{|\partial R_i}] + |B'| \right),$$

where R_i is the i^{th} sub-quadrant of R, and the maximum is over all subset $B' \subseteq \bigcup_i S_{|\partial R_i} \backslash S_{|\partial R}$ such that $B' \cup B$ is independent. Since $|B| \leq 4k$, the problem can be solved by dynamic programming in $n^{O(k)}$.

Given a constant number k and a shift $(i/k, i/k)$ of the quadtree, where $0 \leq i < k$, any square s is not $2k$-aligned at most twice, by the proof in [6] (omitted).

Let OPT be the optimal solution, and OPT^i be the maximum independent among all the squares that is $2k$-aligned for the shift $(i/k, i/k)$. Then

$$\sum_{i=0}^{k-1} |OPT^i| \geq \sum_{i=0}^{k-1} |\{s \in OPT : s \text{ is } 2k\text{-aligned for the shift } (i/k, i/k)\}|,$$

since each s is not $2k$-aligned at most twice, then

$$\sum_{i=0}^{k-1} |OPT^i| \geq (k-2)|OPT|,$$

and

$$\max_{i=0}^{k-1} |OPT^i| \geq (1 - 2/k)|OPT|.$$

Setting $k = \lceil 2/\varepsilon \rceil$ gives a PTAS.

Chapter 3

Disjoint Unit-Disk Cover

We discuss the Disjoint Unit-Disk Cover problem in this chapter. The problem has motivation from wireless networks, where we would like to avoid interference of signals. Our main result is an NP-completeness proof of its decision problem. A PTAS could be easily obtained using the shifted grid technique.

3.1 NP-Hardness

In this section, we prove that the decision problem of Disjoint Unit-Disk Cover is NP-complete: *given a point set, decide if all the points can be covered by disjoint unit disks.*

There are some important properties we should observe. For three unit disks (of unit diameter) touching each other, there exists a disk of radius $r = \sqrt{3}/3 - 1/2$ touching them all. See Figure 3.1. Given a square lattice (other shapes may work as well) in which the side length is smaller than r, it is impossible to cover all the lattice points using disjoint unit disks, since we can always place a disk of radius r disjoint to all the existing unit disks in the plane, and at least one lattice point falls inside the disk and is not covered by any unit disk.

If we place a sufficient number of lattice points inside a unit disk $\mathcal{D}$ and points along the boundary of $\mathcal{D}$, where the distance of two adjacent points is small enough, there is only one way to cover all the points by placing a unit disk on $\mathcal{D}$. Let $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$ be three unit disks, $\mathcal{D}_1, \mathcal{D}_3$ touching each other, and $\mathcal{D}_2$ intersecting them both. See Figure 3.2. Similarly, we place a sufficient number of lattice points inside and points along the boundary of the

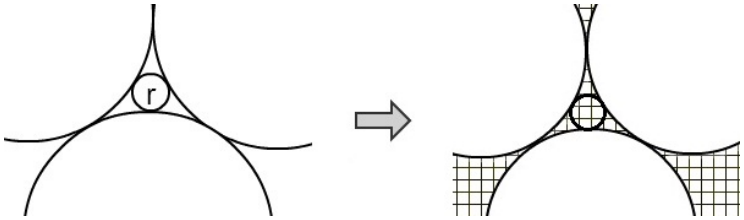

Figure 3.1: The small disk has size $r = \sqrt{3}/3 - 1/2$

.

common regions $\mathcal{D}_1 \cap \mathcal{D}_2$ and $\mathcal{D}_2 \cap \mathcal{D}_3$. To cover all the points by disjoint unit disks, one can either put two disks on $\mathcal{D}_1, \mathcal{D}_3$, or put one disk on $\mathcal{D}_2$.

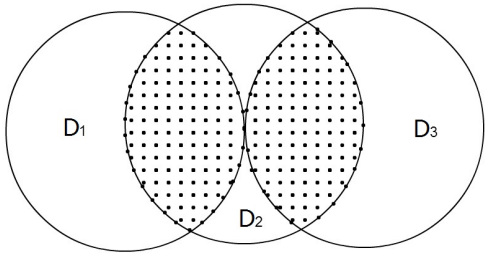

Figure 3.2: We place a sufficient number of points in $\mathcal{D}_1 \cap \mathcal{D}_2$ and $\mathcal{D}_2 \cap \mathcal{D}_3$. There are only two ways to cover all the points by disjoint disks.

Furthermore, we can expand three disks to a disk chain. See Figure 3.3. We place points as in Figure 3.2. There are only two ways to cover all the points, either using the bold boundary disks, or the light boundary disks. It is not difficult to see that $\Theta(1)$ points in each common region are sufficient to ensure the desired property if the area of the common region is $\Theta(1)$. This is the main trick to construct our gadget.

We next prove that the decision problem of Disjoint Unit-Disk Cover is NP-complete.

Theorem 4 *Given a point set P, deciding if all the points in P can be covered by disjoint unit disks is NP-complete.*

Proof. We reduce the problem from the *Planar 3-SAT* problem, which is known to be NP-complete.

Lemma 5 [36] *Every planar graph $G = (V, E)$ has an rectilinear planar layout on an $O(|V|) \times O(|V|)$ grid (i.e., each vertex is represented by a horizontal line segment and*

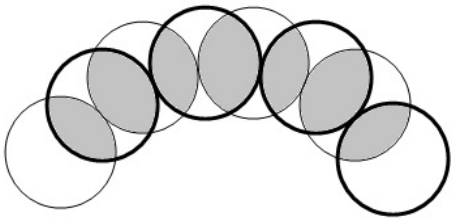

Figure 3.3: A disk chain. The shaded area represents that the points are filled inside and along the boundary.

each edge is represented by a vertical line segment, and all endpoints of the segments have integer coordinates).

Given a planar graph corresponding to a Planar 3SAT instance, we create a rectilinear drawing by Lemma 5. See Figure 3.4. We rescale the drawing by a factor of 4 so that there is enough space for disks. The construction is basically to put virtual unit disks along the line segments, and then place points in the common regions of disks, as in Figure 3.2, so that there are only two ways to cover all the points along each line segment using disjoint unit disks.

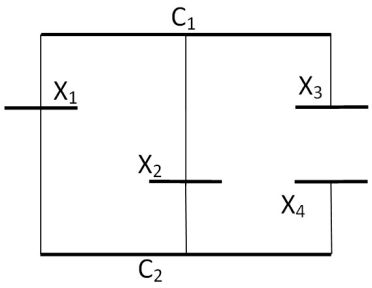

Figure 3.4: An example of the rectilinear layout of a Planar 3SAT Instance I. $I = C_1 \wedge C_2$, $C_1 = x_1 \vee \bar{x}_2 \vee x_3$, $C_2 = x_1 \vee x_2 \vee \bar{x}_4$.

For each horizontal line segment representing a variable, we put virtual unit disks along the line segment. See Figure 3.5. We place points in the common region of any two intersecting virtual disks as in Figure 3.2. We colour the leftmost virtual disk black, and the second disk white, and the third disk black, and so on. To cover all the points in a variable gadget by disjoint disks, we can either use all the black disks, or use all the white disks.

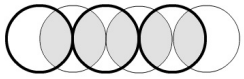

Figure 3.5: The variable gadget. The bold boundary disks represent the black disks and the light boundary disks represent the white disks.

Note that each clause contains exactly three variables. For each horizontal line segment representing a clause, it intersects three vertical line segments (edges). We first put three virtual unit disks around the middle intersection point. See Figure 3.6. The three disks intersect each other. We call them the "*central disks*" of the clause, and we call the common region of all the three disks the "*central region*" of the clause. We place a sufficient number of points inside and along the boundary of the central area. We then put another three virtual disks, called "*outer disks*", with each outer disk intersecting a central disk, and tangent to the other two central disks. Each outer disk will be joined to a variable contained in the clause using a disk chain. Note that we do not place points in the common regions of the outer disks and the central disks.

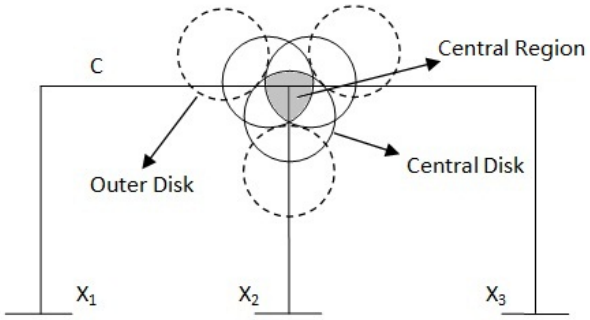

Figure 3.6: The clause gadget.

Finally, we connect the variable gadgets and the corresponding clause gadgets. See Figure 3.7. For each variable x, if a clause C contains its positive form x, we connect the corresponding outer disk of C with a white disk in the gadget of x, by putting disks roughly along the line segments. Similarly, if C contains the negated form $\bar{x}$, we connect the corresponding outer disk of C with a black disk in the gadget of x. The disk chains are not strictly straight, since we may need to "bend" them to meet certain requirements. Each bending can be done using a constant number of disks. We then place points in the common regions of the newly added disks as in Figure 3.2. We colour the uncoloured disks from the variable gadgets along the way to the corresponding outer disks, in a way so that

if an uncoloured disk intersects a black disk, we colour it white, and if an uncoloured disk intersects a white disk, we colour it black. All the virtual disks except the central disks of each clause are then coloured either white or black.

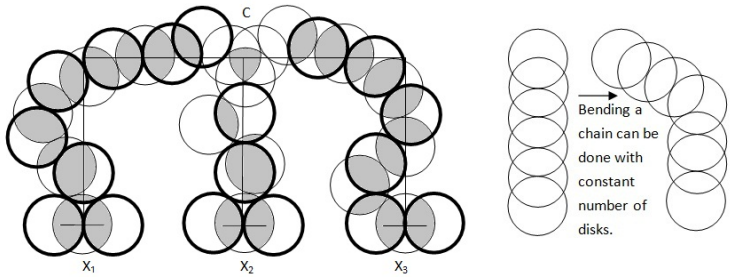

Figure 3.7: $C = x_1 \vee x_2 \vee \bar{x}_3$.

The number of black and white disks is bounded by $O(|V|^2)$. Since bending the disk chains can be achieved using a constant number of disks, the common region of any two disks has $\Theta(1)$ area, and the number of points we place is constant. The total number of points in the gadgets is $O(|V|^2)$. The reduction takes polynomial time.

For an instance of Planar 3SAT, if there is a setting that satisfies all the clauses, for the variables setting to true, we use white disks to cover points in the variable gadgets, and black disks for the variables setting to false. The disks used from a variable gadget to an outer disk have the same colour as the disks used in the variable gadget. Finally, to cover the central area of each clause, there shall be at least one central disk that does not intersect any existing disks. For a central area, if there are two or three central disks that can be used, we arbitrarily choose one. Note that in the case where two or three central disks that can be used, we may be able to place a disk in the position other than the central disks to cover the central area, but this does not affect the result.

Similarly, if we can cover all the points in the constructed instance of Disjoint Unit-Disk Cover, we set a variable to true if the points in the corresponding gadget are covered by white disks, otherwise we set it to false. Then all the clauses can be satisfied. Thus, the original Planar 3SAT instance has a solution if and only if the constructed Disjoint Unit-Disk Cover instance has a solution. □

3.2 PTAS

A PTAS for Disjoint Unit-Disk Cover can be found by applying the shifted grid technique.

Lemma 6 *Given a point set P with all the points inside a $k \times k$ square, there is a polynomial time algorithm to cover the maximum number of points using disjoint disks.*

Proof. As mentioned in section 2.1, one can cover the whole $k \times k$ square using $2k^2$ disks. Let $n = |P|$, and let t be an integer with $0 < t \leq 2k^2$. For a point (a_i, b_i) in P and a disk in the optimal solution of centre (x_j, y_j) that covers the point, where $0 < i \leq n$ and $0 < j \leq t$, we have

$$(a_i - x_j)^2 + (b_i - y_j)^2 \leq 1.$$

Any two disks in the optimal solution of centres (x_p, y_p), (x_q, y_q) with $0 < p, q \leq t$ $(p \neq q)$, should be disjoint and thus

$$(x_p - x_q)^2 + (y_p - y_q)^2 \geq 1.$$

For a point $\xi = (x_1, y_1, \ldots, x_t, y_t) \in \mathbb{R}^{2t}$, let

$$f_{(i,j)}(\xi) = (x_i - x_j)^2 + (y_i - y_j)^2 - 1$$

and

$$g_{(i,l)}(\xi) = (x_i - a_l)^2 + (y_i - b_l)^2 - 1$$

with $0 < i, j \leq t$, $0 < l \leq n$ and $i, j, l \in \mathbb{Z}$.

Each $f_{(i,j)}(\xi)$ and each $g_{(i,l)}(\xi)$ represent surfaces in $\mathbb{R}^{2t}$. The number of all surfaces is $O(n)$. And the number of all the cells is $n^{O(t)} = n^{O(k^2)}$. We can compute the arrangement of the surfaces with respect to all $f_{(i,j)}(\xi)$ and $g_{(i,l)}(\xi)$ in polynomial time by known algorithms [21] for each $0 < t \leq 2k^2$, and select the best assignment to maximize the number of points covered. □

Lemma 7 *There is a PTAS for Disjoint Unit-Disk Cover.*

Proof. We draw a uniform grid of unit side length. Given a constant $k \geq 4$ and a shift (a, b) with $0 \leq a, b < k$ and $a, b \in \mathbb{Z}$, we apply the local algorithm in lemma 6 for the points inside every square $[ik + a + 1, (i+1)k + a - 1)] \times [jk + b + 1, (j+1)k + b - 1)]$ with $i, j \in \mathbb{Z}$. Let OPT denote the optimal point set, let $OPT^{(a,b)}$ denote the union of the point sets found by the local algorithm, and let $\widetilde{OPT}^{(a,b)}$ denote the set of points covered by the disks that are in OPT and intersect the lines $x = ik + a$ and $y = jk + b$. We have

$$\left|OPT^{(a,b)}\right| + \left|\widetilde{OPT}^{(a,b)}\right| \geq |OPT|,$$

and

$$\sum_{0\le i,j<k}\left(\left|OPT^{(i,j)}\right|+\left|\widetilde{OPT}^{(i,j)}\right|\right)\ge k^2|OPT|,$$

since

$$\sum_{0\le i,j<k}\left|\widetilde{OPT}^{(i,j)}\right|\le 2k|OPT|,$$

implying that

$$\sum_{0\le i,j<k}\left|OPT^{(i,j)}\right|\ge (k^2-2k)|OPT|,$$

and

$$\max_{0\le i,j<k}\left|OPT^{(i,j)}\right|\ge\left(1-\frac{2}{k}\right)|OPT|.$$

Setting $k=\lceil 2/\varepsilon\rceil$ gives a PTAS. □

Chapter 4

Depth-($\leq K$) Packing for Arbitrary-Size Disks/Squares

We study the Depth-($\leq K$) Packing for Arbitrary-Size Disks/Squares in this chapter. Depth-($\leq K$) Packing is a generalization of the Geometric Packing problem. When $K = 1$, the problem reduces to the Geometric Independent Set problem, which admits a PTAS by Chan's shifted quadtree technique, as we have shown in section 2.2. When K is a constant, the problem can be directly solved by the same algorithm with slight modification to the dynamic programming part: since the number of objects that intersect a quadtree cell is bounded by $O(Kk)$ where k is the number of shifts, we only need to remember $O(Kk)$ disks for each cell. So the following theorem holds:

Theorem 8 *When K is a constant, the Depth-($\leq K$) Packing problem for fat objects has a PTAS in $n^{O(K/\varepsilon)}$ time, for any $0 < \varepsilon < 1$.*

However, when K is not a constant, the number of objects that intersect a quadtree cell is no longer a constant, and Theorem 8 does not guarantee polynomial time. In this chapter, we show how the case of non-constant K can actually be reduced to the constant K case. Given a set of disks/squares, we provide an algorithm which finds a subset with bounded depth, and the area missing is bounded by $\Theta(\delta)$ times the total area, for any given $0 < \delta < 1$.

4.1 Depth Reduction Theorem

In this section, we present a *depth reduction theorem*, which is the main technique we use to solve Depth-($\leq K$) Packing. We prove the theorem for disks, and the proof for squares is basically same. Given a set A of geometric objects, we define $area(A)$ as the area of the union of all objects in A.

Theorem 9 (Depth Reduction Theorem) *Given a set T of disks, there is a subset S of T with depth at most $O(1/\delta^4)$, and $area(S) \geq (1 - \Theta(\delta))area(T)$, for any given δ with $0 < \delta < 1$.*

Before proving the theorem, we first give a brief introduction of the *Minkowski sum*.

Definition 10 (Minkowski sum) *Given two sets A, B of position vectors in Euclidean space, the Minkowski sum of A and B is the set*

$$A \oplus B = \{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in A, \mathbf{b} \in B\}.$$

See Figure 4.1 for an example of the Minkowski sum of a shape and a disk.

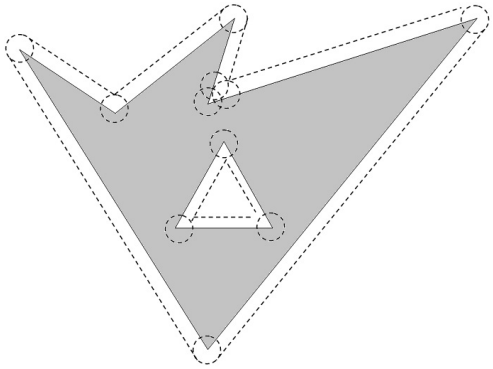

Figure 4.1: The shaded part is the original shape. The region enclosed by the dotted lines is the Minkowski sum.

Depth Reduction Algorithm For a given set T of disks, the following algorithm finds the subset S in Theorem 9.

Given a disk set $T = \{d_1, d_2, \ldots, d_n\}$, where the sizes of $d_1, d_2, \ldots, d_n$ are in descending order. Let S be an empty set initially. For an index i from 1 to n, if $d_i \not\subset \bigcup_{d\in S} d \oplus B_d$, where B_d is a disk of radius of $r_d\delta$, we add d_i to S. Otherwise, we do nothing. Return S at the end of the loop.

We prove the correctness of the depth reduction theorem/algorithm in the following sections.

4.1.1 S Has Area At Least $(1 - \Theta(\delta))area(T)$

Given a set T of disks, and the subset S of T obtained by the depth reduction algorithm, we prove that $area(S) \geq (1 - \Theta(\delta))area(T)$.

Lemma 11 *Given a set S of disks, $area\left(\bigcup_{d\in S} d \oplus B_d\right) - area(S) \leq \Theta(\delta)area(S)$, where B_d is a disk of radius $r_d\delta$ for a given $0 < \delta < 1$.*

Proof. We use the *additively weighted Voronoi Diagram* (AWVD) in Euclidean space to help prove the lemma. Each point site in the AWVD is given a positive weight, and the distance between an arbitrary point and a site is defined to be their Euclidean distance minus the weight of that site. The cell of the AWVD is star-shaped [13], i.e., a line segment between the site and any point in the cell always lies inside the cell.

We expand the radius of each disk $d \in S$ by a factor of $1+\delta$, and let S' be the resulting set. We consider the AWVD of S', where the sites are the centres of the disks in S', and the weights are the disk radius. Abusing notation, for a disk in S' of centre O, we also use O to refer to that disk. We use $cell(O)$ to denote the AWVD cell of O. For an arbitrary point p, we use $d(p, O)$ and $dist(p, O)$ to denote the weighted distance and the Euclidean distance between p and O respectively. By definition, $d(p, O) = dist(p, O) - r$, where r is the radius of disk O.

If a point p is inside a disk O_1 but outside a disk O_2, where $O_1, O_2 \in S'$, then p is not in $cell(O_2)$. This is because

$$d(p, O_1) = dist(p, O_1) - r_1 < 0 < dist(p, O_2) - r_2 = d(p, O_2),$$

where r_1, r_2 are the radii of O_1, O_2 respectively. Thus if a point p is covered by several disks, it is in the disk that corresponds to the AWVD cell containing p.

Let $\Delta = \bigcup S' \setminus \bigcup S$ be the area that is in $\bigcup S'$ but outside $\bigcup S$. Consider a fixed site O and a connected component R of $\Delta \cap cell(O)$. See Figure 4.2. The grey portion represents Δ, and the black portion represents the area R. Since the black portion is known to be covered by one or several disks in S', it is in one of the corresponding AWVD cells. We assume it is in $cell(O)$, where O is some site. Let d and d' be the disks of centre O in the sets S and S' respectively. The area R is in d' and outside d. Let s' denote the sector of d' that contains R with the smallest central angle. Let s be the sector of d with the same angle. We have $area(R) \leq area(s') - area(s) = \Theta(\delta) area(s)$.

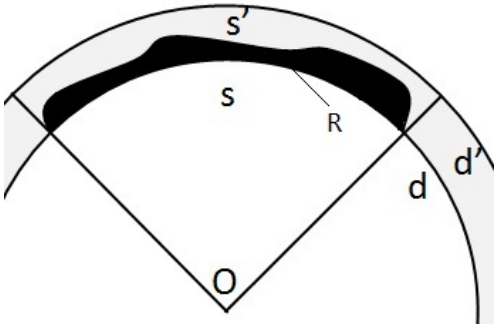

Figure 4.2: A connected component R of $\Delta \cap cell(O)$

For all the small areas R and the corresponding sectors s and s', we have

$$area(\Delta) \leq \sum \Theta(\delta) area(s).$$

Furthermore, all the sectors s are disjoint since each AWVD cell is star-shaped and any two AWVD cells are disjoint. So

$$\sum \Theta(\delta) area(s) \leq \Theta(\delta) area(S),$$

and

$$area\left(\bigcup_{d \in S} d \oplus B_d\right) = area(\Delta) + area(S) \leq (1 + \Theta(\delta)) area(S).$$

□

We apply Lemma 11 to the subset S found by the depth reduction algorithm. We have

$$area\left(\bigcup_{d \in S} d \oplus B_d\right) \leq (1 + \Theta(\delta)) area(S),$$

and

$$area\left(\bigcup_{d\in S} d \oplus B_d\right) - area(S) \leq \Theta(\delta)(S).$$

Observe that $area(S) \leq area(T) \leq area\left(\bigcup_{d\in S} d \oplus B_d\right)$. The right-hand side holds since any disk in T and not in $\bigcup_{d\in S} d \oplus B_d$ would be added to S in the algorithm. So we have

$$area(T) - area(S) \leq \Theta(\delta) area(T)$$

and

$$area(S) \geq (1 - \Theta(\delta)) area(T).$$

4.1.2 Bounded Depth

Next we prove that the subset S obtained by the depth reduction algorithm has bounded depth. Assume that the point p has the maximal depth t among all the points covered by $\bigcup S$, and let $D = \{d_1, \ldots, d_t\}$ be the subset of S that contains p. We prove that t is bounded by $O(1/\delta^4)$.

Assume that the sizes of $d_1, \ldots, d_t$ are in descending order. The area $\bigcup D$ is connected since all the disks in D contain p. Let r_i denote the radius of d_i, for $0 < i \leq t$. Then $r_1 > r_2 > \cdots > r_t$. Let S_i be the subset S at the beginning of iteration i, before d_i is added. The disk d_i should cover some point p' outside $\bigcup_{d\in S_i} d \oplus B_d$, otherwise d_i will not be added. Observe that the distance between p' and any point in $\bigcup S_i$ is at least $\delta r_i \geq \delta r_t$. Thus, there exists a disk of diameter δr_t inside d_i and outside $\bigcup S_i$. See Figure 4.3.

All the disks in D are inside a square of size $4r_1$. Each disk d_i covers an area outside $d_1 \cup \cdots \cup d_{i-1}$ of size at least $\pi r_t^2 \delta^2/4$. There are at most $16r_1^2/(\pi r_t^2\delta^2/4) = O(\frac{r_1^2}{r_t^2\delta^2})$ disks. Observe that $d_t \not\subset d_1 \oplus B_{d_1}$, otherwise d_t would not be added to S. Since d_1 and d_t contain a common point p, this implies that $2r_t \geq \delta r_1$, and $O\left(\frac{r_1^2}{r_t^2\delta^2}\right) = O(1/\delta^4)$. Thus the maximal depth t is bounded by $O(1/\delta^4)$.

4.2 PTAS

We have proved the depth reduction theorem for disks in section 4.1.1 and 4.1.2. Given an instance of the Depth-($\leq K$) Packing problem on a set T of disks, for any $0 < \varepsilon < 1$, we use

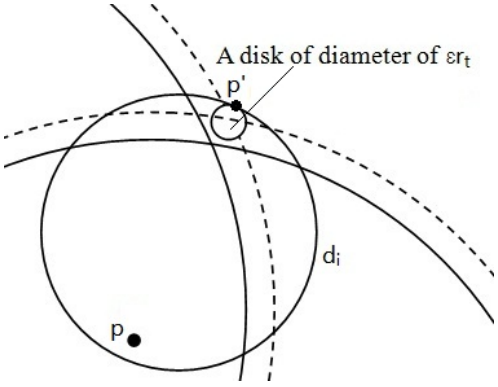

Figure 4.3: The dotted curves are the boundaries of $d_j \oplus B_{d_j}$, for $j = 0, \ldots, i-1$. Point p' is outside $\bigcup_{d \in S_i} d \oplus B_d$.

$\widetilde{OPT}$ to denote the subset of T that has depth at most $1/\varepsilon^4$, and that has the maximum area of the union. Applying Theorem 8 for the Depth-($\leq 1/\varepsilon^4$) Packing problem on T, we can find a subset $\widetilde{S}$ of T in time $n^{O(1/\varepsilon^5)}$, such that the depth of $\widetilde{S}$ is at most $1/\varepsilon^4$ and $area(\widetilde{S}) \geq (1-\varepsilon) area(\widetilde{OPT})$.

Let OPT be the optimal solution of the Depth-($\leq K$) Packing problem on T, where OPT has depth at most K. By the depth reduction theorem, there is a subset S of OPT of depth $1/\varepsilon^4$ and $area(S) \geq (1 - c \cdot \varepsilon) area(OPT)$, where c is a constant. Since $\widetilde{OPT}$ has the largest area among all the subsets of T of depth at most $1/\varepsilon^4$, we have

$$area(\widetilde{OPT}) \geq area(S),$$

and

$$area(\widetilde{S}) \geq (1-\varepsilon)(1 - c \cdot \varepsilon) area(OPT) = (1 - \Theta(\varepsilon)) area(OPT).$$

It implies that $\widetilde{S}$ is a $(1 - \Theta(\varepsilon))$-approximation of the Depth-($\leq K$) Packing problem on T. We then have the following theorem:

Theorem 12 *There is a PTAS for Depth-($\leq K$) Packing for Disks of Arbitrary Radii.*

The proof can also be applied to squares with a similar analysis using the additively weighted L_∞-Voronoi Diagrams.

Chapter 5

Red-Blue Unit-Square Cover

We study Red-Blue Unit-Square Cover in this chapter. We prove that the problem is NP-hard, and introduce a "mod-one" operation which can be applied to Red-Blue Unit-Square Cover and several other problems to find a PTAS.

5.1 NP-Hardness

Theorem 13 *Red-Blue Unit-Square Cover is NP-hard.*

Proof. We reduce from the vertex cover problem on degree-3 planar graphs, which is well known to be NP-hard [20].

Lemma 14 [10] *Every planar graph $G = (V, E)$ of maximum degree at most 4 has an orthogonal planar drawing on an $O(|V|) \times O(|V|)$ grid (i.e., vertices are placed at grid points and edges are drawn as a rectilinear polygonal chain with corners at grid points, with no crossings).*

Lemma 15 (Folklore) *Given a graph G and an edge e in G, define a new graph G' obtained from G by subdividing e through the addition of two new "dummy" vertices. Then the size of a minimum vertex cover of G' is precisely the size of a minimum vertex cover of G plus 1.*

Given a degree-3 planar graph G with n vertices, we create an orthogonal drawing by Lemma 14. We define a new graph G' by subdividing each edge e through the addition of new dummy vertices at each grid point along e. Each edge in G' is now a horizontal or vertical line segment of length 1 in the drawing. If e contains an odd number of dummy vertices, we insert an extra new dummy vertex at the midpoint of a line segment. Then all edge lengths in G' are $1/2$ or 1. By rescaling by a factor slightly less than 2, we can ensure that all edge lengths in G' are strictly between $2/3$ and 2. Now, each edge in the original graph G has an even number of dummy vertices, and by repeated applications of Lemma 15, finding the size of the minimum vertex cover of G is equivalent to finding the size of the minimum vertex cover of G'.

To construct an instance of Red-Blue Unit-Square Cover from G', we replace each vertex in G' by a red point r_i. For each edge $r_i r_j$ in G', we create a blue point b_{ij} in the middle of the edge and add a unit square containing precisely b_{ij} and r_i and a unit square containing precisely b_{ij} and r_j. See Figure 5.1. Such squares exist since the distance between two adjacent blue and red points is strictly between $1/3$ and 1.

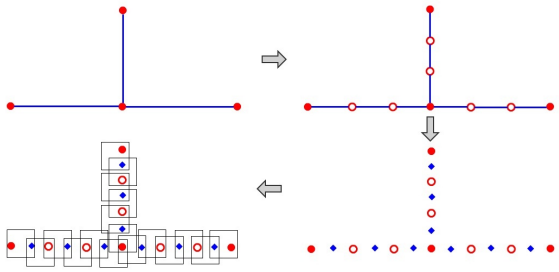

Figure 5.1: The reduction from vertex cover. (Red points are drawn as dots, and blue points are drawn as diamonds.)

Correctness of the reduction is easy to see: Given a vertex cover of G' of size k, we can select all the squares that cover the corresponding k red points; these squares would cover all blue points. Conversely, given a subset of squares covering all blue points, the red points covered by these squares form a vertex cover of G'.

□

5.2 PTAS

We now present a PTAS for Red-Blue Unit-Square Cover. We begin with a definition:

Definition 16 *Let $S = \{s_1, \ldots, s_t\}$ be a set of unit squares, where $s_1, \ldots, s_t$ are arranged in increasing x-order of their centers. We say that S forms a monotone set, if the centers of $s_1, \ldots, s_t$ are in increasing or decreasing y-order.*

Note that the boundary of the union of the squares in a monotone set S consists of two monotone chains ("staircases"), as as shown in figure 5.2. We say that these two chains are *complementary*.

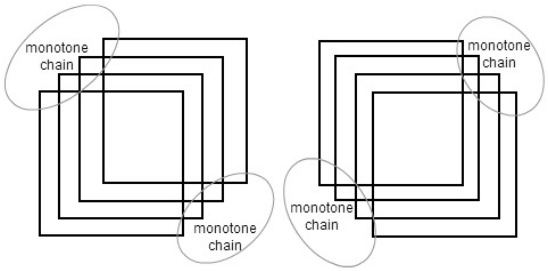

Figure 5.2: A monotone set may be "increasing" (left) or "decreasing" (right).

Lemma 17 *Let OPT be an optimal solution for an instance where the blue point set B is inside a $k \times k$ square. Then OPT can be decomposed into $O(k^2)$ monotone sets.*

Proof. We may assume that all squares in OPT intersect the $k \times k$ square. Draw a grid with unit side length over the $k \times k$ square. Consider a grid point p. Let $S(p)$ be the set of squares in OPT containing p; every square in OPT belongs to one of the $S(p)$'s. Let $U(p)$ denote the boundary of the union of the squares of $S(p)$. We may assume that each square in $S(p)$ appears on $U(p)$, for otherwise we could remove the square from OPT and the resulting solution is no worse than OPT.

Divide the plane into 4 quadrants at p. For each $i \in \{1, 2, 3, 4\}$, let $S_i(p)$ be the subset of squares in OPT containing p that contributes to the portion of $U(p)$ inside the i-th quadrant. Then each $S_i(p)$ is a monotone set. Thus, we have decomposed OPT into $4(k+1)^2$ monotone sets. (These sets may not be disjoint, but can be made disjoint by deleting elements from sets, since a subset of a monotone set is still monotone.) □

The heart of our PTAS is an exact dynamic programming solution for the special case of the problem where all points are inside a $k \times k$ grid for a constant k. The idea is to use a sweep-line algorithm to guess the $O(k^2)$ monotone sets at the same time. We can

"remember" a constant number ($O(k^2)$) of intersections of the monotone chains with a vertical sweep line as we sweep from left to right. However, each monotone set defines two complementary monotone chains, and the guess of one chain should be consistent with the guess of its complementary chain; but by the time the sweep line gets to the second chain, we would have forgotten information about the first chain. This is why Erlebach and van Leeuwen [16] needed a more complicated approach involving multiple sweep lines moving at different speeds.

To avoid this difficulty, we overlay all the monotone sets into one grid cell by introducing a "mod-one" transformation:

Definition 18 *We define the* mod-one *mapping* $(x, y) \mapsto (x \bmod 1, y \bmod 1)$*, where* $z \bmod 1$ *denotes the fractional part of a real number* z*.*

With this transformation, a unit square is rearranged into four pieces covering the unit grid cell, as shown in figure 5.3.

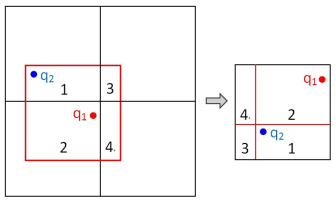

Figure 5.3: Applying the mod-one transformation to a unit square.

Furthermore, the union of the squares in a monotone set is rearranged as shown in figure 5.4. Notice that the two complementary monotone chains are mapped to two monotone chains that are connected at the corner points. This is the key property we need about the mod-one transformation. By redesigning the sweep-line algorithm to sweep over the unit grid cell in the transformed space, we can guess the two complementary monotone chains of each monotone set at the same time. The remaining pieces of the union consists of two rectangles defined by the start and end square of the monotone set; we can guess these two squares in advance.

Theorem 19 *For any instance of Red-Blue Unit-Square Cover where* B *is inside a* $k \times k$ *square for a constant* k*, we can find the optimal solution in* $O(mn^{O(k^2)})$ *time.*

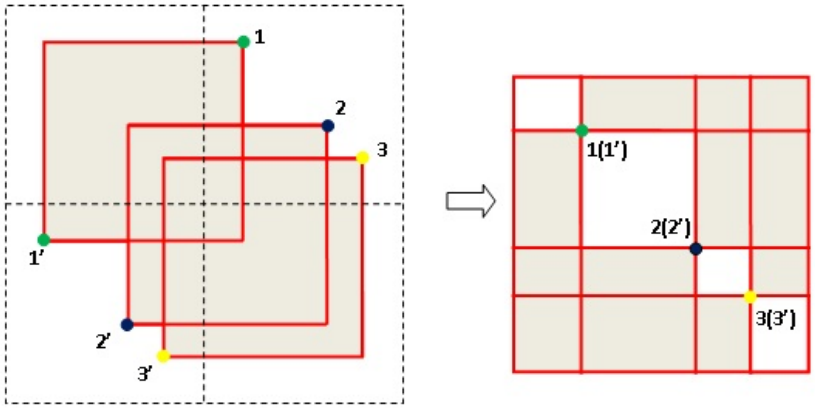

Figure 5.4: Applying the mod-one transformation to a monotone set.

Proof. We find it best to describe our dynamic programming algorithm in terms of a state-transition diagram. We define a *state* to consist of

- a vertical sweep line ℓ that passes through a corner of an input square, after taking mod 1;
- $O(k^2)$ 4-tuples of the form $(s_{\text{start}}, s_{\text{prev}}, s_{\text{curr}}, s_{\text{end}})$, subject to the conditions that $s_{\text{start}}, s_{\text{prev}}, s_{\text{curr}}, s_{\text{end}}$ are in increasing x-order and form a monotone set, and that ℓ lies between the corners of s_{prev} and s_{curr}, mod 1.

Intuitively, a state represents current information about a decomposition of a solution into monotone sets at the sweep line (the monotone sets are not required to be disjoint). Specifically, each 4-tuple corresponds to a monotone set S; s_{start} and s_{end} represent the start and end square of S; and s_{prev} and s_{curr} represent the squares that define intersections of the sweep line with the two complementary monotone chains of S, after taking mod 1. These two squares s_{prev} and s_{curr} are adjacent in the monotone set S.

Given this state, we create a *transition* into a new state as follows: We pick the 4-tuple $(s_{\text{start}}, s_{\text{prev}}, s_{\text{curr}}, s_{\text{end}})$ such that the corner point of s_{curr} has the smallest x-coordinate, mod 1. The new sweep line ℓ' will be at the corner of s_{curr}. This 4-tuple is replaced by a new 4-tuple $(s_{\text{start}}, s_{\text{curr}}, s', s_{\text{end}})$ satisfying the stated conditions for some square s'. All other 4-tuples are unchanged. Let j_r (resp. j_b) be the number of red (resp. blue) points

that lie between ℓ and ℓ', after taking mod 1, and are covered (resp. not covered) by the squares in the $O(k^2)$ 4-tuples (before taking mod 1). If $j_b > 0$, we remove this transition. Otherwise, we set the cost of this transition to j_r.

The problem is thus reduced to finding the shortest path in this state-transition diagram (a directed acyclic graph), after adding suitable transitions involving start and end states. There are at most $O(mn^{O(k^2)})$ states, and each state has at most $O(n)$ outgoing transitions (since there are $O(n)$ choices for s'). Thus, we can construct the graph and find the shortest path by dynamic programming in $O(mn^{O(k^2)})$ time. □

We now apply the grid shifting technique to obtain our final result:

Theorem 20 *There is a PTAS for Red-Blue Unit-Square Cover.*

Proof. For each shift $a, b \in \{0, \ldots, k-1\}$, let $S^{a,b}$ be the union of the solutions found by Theorem 19 for the blue points inside every $k \times k$ square $[ik+a, (i+1)k+a] \times [jk+b, (j+1)k+b]$, with $i, j \in \mathbb{Z}$. We return the $S^{(a,b)}$ with the smallest $c(S^{(a,b)})$, where $c(S)$ denotes the number of red points covered by S.

To analyze the approximation factor, let OPT be the optimal solution. Let $OPT^{a,*}$ (resp. $OPT^{*,b}$) be the subset of squares in OPT intersecting the lines $x = ik+a$ with $i \in \mathbb{Z}$ (resp. the lines $y = jk+b$ with $j \in \mathbb{Z}$). Since the algorithm in Theorem 2 covers the minimum number of red points for the subproblem for each $k \times k$ square, we have

$$c(S^{a,b}) \;\le\; c(OPT) + 2\,c(OPT^{a,*}) + 2\,c(OPT^{*,b}).$$

Since $\sum_{0 \le a < k} c(OPT^{a,*})$ and $\sum_{0 \le b < k} c(OPT^{*,b})$ are both at most $2\,c(OPT)$,

$$\sum_{0 \le a,b < k} c(S^{a,b}) \;\le\; (k^2 + 8k)\,c(OPT),$$

implying that

$$\min_{0 \le a,b < k} c(S^{a,b}) \;\le\; (1 + 8/k)\,c(OPT).$$

Setting $k = \lceil 8/\varepsilon \rceil$ gives a $(1+\varepsilon)$-approximation algorithm. □

5.3 Related Problems

Weighted Unit-Square Cover. Erlebach and van Leeuwen [16] studied the following related problem: Given a set P of points and a set $\mathcal{S}$ of unit squares in 2D where each square has a positive weight, we want to find a smallest-weight subset of $\mathcal{S}$ to cover all the points in P.

Our algorithm can easily be modified to solve this problem. Specifically, in the proof of Theorem 19, if there are any points that lie between ℓ and ℓ', after taking mod 1, and are not covered by any of the squares in the $O(k^2)$ 4-tuples, then we remove the transition. Otherwise, we set the cost of the transition to the weight of the square s'.

Budgeted Maximum Coverage for Unit Squares. Erlebach and van Leeuwen [16] also considered the following problem: Given a set P of points where each point has a positive profit value, and given a set $\mathcal{S}$ of unit squares where each square has a positive cost, and given a budget B, we want to find a subset of $\mathcal{S}$ with total cost at most B, maximizing the total profit of all points in P that are covered by the subset.

Erlebach and van Leeuwen [16] described how a modification of their dynamic programming algorithm combined with additional ideas can yield a PTAS for this problem. Our approach can be used to simplify the dynamic programming part of their PTAS.

Partial Unit-Square Cover. Gandhi et al. [19] studied the Partial Set Cover problem. A geometric version can be stated as follows: Given a set P of points and a set $\mathcal{S}$ of unit squares in 2D, and given an integer K, we want to find a smallest subset of squares in $\mathcal{S}$ to cover at least K points in P.

Gandhi et al. gave a PTAS for a continuous version of the problem based on Hochbaum and Maass' shifted grid technique [24]. For the discrete version, we can obtain a PTAS by using an appropriate modification of our dynamic programming algorithm, in conjunction with shifted grids as in Gandhi et al.'s paper.

Unique Unit-Square Coverage. Ito et al. [26] studied the following problem: Given a set P of points and a set $\mathcal{S}$ of unit squares in 2D, find a subset of $\mathcal{S}$ to maximize the number of points in P that are covered exactly once by the subset.

Again our algorithm can be modified to solve this problem. In the proof of Theorem 19, we use 6-tuples $(s_{\text{start}}, s_{\text{prev2}}, s_{\text{prev}}, s_{\text{curr}}, s_{\text{curr2}}, s_{\text{end}})$ instead of 4-tuples, where intuitively

s_{prev2} represents the predecessor of s_{prev} and s_{curr2} represents the successor of s_{curr} in a monotone set. We set the cost of the transition to be the number of points that lie between ℓ and ℓ', after taking mod 1, and are uniquely covered by the squares in the $O(k^2)$ 6-tuples. This works because squares that are not part of these 6-tuples are irrelevant as to whether a point is uniquely covered.

Chapter 6

Conclusions

In this thesis, we have studied three problems: Disjoint Unit-Disk Cover, Depth-($\leq K$) Packing, and Red-Blue Unit-Square Cover.

For Disjoint Unit-Disk Cover, we prove that it is NP-complete and give a PTAS.

For the Depth-($\leq K$) Packing, we give a PTAS by proving the depth reduction theorem. However, the theorem only works in the continuous version, and it cannot be applied to the discrete version or the weighted version where the weights do not equal to the areas. We prove the theorem for disks and squares for the continuous version. We do not know if the theorem can be applied to other fat objects. The theorem may have other potential applications where one wants to bound the depth and does not want to lose much area of the union.

The "mod-one" trick we introduce can be applied to several geometric covering problems for unit squares: the Red-Blue Set Cover problem, the Set Cover problem, the Weighted Set Cover problem, the Partial Set Cover problem, the Unique Coverage problem and the discrete Depth-($\leq K$) Coverage where K is a constant. The technique may also be applicable to other problems where the objects are unit squares. However, the time complexity of our algorithm is $n^{O(1/\varepsilon^2)}$, which is slower than some known PTASes of running time $n^{O(1/\varepsilon)}$. We do not know if there is a faster dynamic programming based PTAS. Furthermore, we do not know if the technique can be applied to other objects beside unit squares, and to 3D or higher dimensions.

References

[1] G. Aloupis, R. A. Hearn, H. Iwasawa, and R. Uehara. Covering points with disjoint unit disks. In *Proceedings of the Twenty-Fourth Canadian Conference on Computational Geometry*, 2012.

[2] 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.

[3] P. Brass, F. Hurtado, B. Lafreniere, and A. Lubiw. A lower bound on the area of a 3-coloured disk packing. *International Journal of Computational Geometry & Applications*, 20(03):341–360, 2010.

[4] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14(1):463–479, 1995.

[5] R. D. Carr, S. Doddi, G. Konjevod, and M. Marathe. On the red-blue set cover problem. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 345–353, 2000.

[6] T. M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *Journal of Algorithms*, 46(2):178–189, 2003.

[7] T. M. Chan. Low-dimensional linear programming with violations. *SIAM Journal on Computing*, 34(4):879–893, 2005.

[8] T. M. Chan and N. Hu. Geometric red-blue set cover for unit squares and related problems. In *Proceedings of the Twenty-Fifth Canadian Conference on Computational Geometry*, 2013. (to appear).

[9] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[10] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1):165–177, 1990.

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

[12] E. D. Demaine, U. Feige, M. Hajiaghayi, and M. R. Salavatipour. Combination can be hard: Approximability of the unique coverage problem. *SIAM Journal on Computing*, 38(4):1464–1483, 2008.

[13] A. Dobrin. A review of properties and variations of Voronoi diagrams. *Whitman College*, 2005.

[14] T. Erlebach, K. Jansen, and E. Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM Journal on Computing*, 34(6):1302–1323, 2005.

[15] T. Erlebach and E. J. van Leeuwen. Approximating geometric coverage problems. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1267–1276, 2008.

[16] T. Erlebach and E. J. van Leeuwen. PTAS for weighted set cover on unit squares. In *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, pages 166–177. 2010.

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

[18] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[19] R. Gandhi, S. Khuller, and A. Srinivasan. Approximation algorithms for partial covering problems. *Journal of Algorithms*, 53(1):55–84, 2004.

[20] M. R. Garey and D. S. Johnson. Computers and Intractability. A Guide to the Theory of NP-completeness, 1979.

[21] D. Halperin. Arrangements. Citeseer, 1997.

[22] Robert A. Hearn. Complexity of Inaba's coin-covering problem. In *the Tenth Gathering for Gardner Conference*, 2012.

[23] D. S. Hochbaum. Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. In *Approximation algorithms for NP-hard problems*, pages 94–143. PWS Publishing Co., 1996.

[24] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32(1):130–136, 1985.

[25] N. Inaba. http://inabapuzzle.com/hirameki/suuri_4.html, 2008. In Japanese.

[26] T. Ito, S. Nakano, Y. Okamoto, Y. Otachi, R. Uehara, T. Uno, and Y. Uno. A polynomial-time approximation scheme for the geometric unique coverage problem on unit squares. In *the Thirteenth Scandinavian Symposium and Workshops on Algorithm Theory*, pages 24–35. 2012.

[27] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

[28] S. Khullera, A. Mossb, and J. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70:39–45, 1999.

[29] P. M. Long. Using the pseudo-dimension to analyze approximation algorithms for integer programming. In *Workshop on Algorithms and Data Structures*, pages 26–37. 2001.

[30] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13(4):383–390, 1975.

[31] N. H. Mustafa and S. Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010.

[32] R. Rado. Some covering theorems (I). *Proceedings of the London Mathematical Society*, 2(1):232–264, 1949.

[33] R. Rado. Some covering theorems (II). *Proceedings of the London Mathematical Society*, 2(1):243–267, 1951.

[34] R. Rado. Some covering theorems (III). *Proceedings of the London Mathematical Society*, 1(1):127–130, 1968.

[35] T. Radó. Sun un problème relatif à un théorème de Vitali. *Fundamenta Mathematicae*, 11(1):228–229, 1928.

[36] P. Rosenstiehl and R. E. Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete & Computational Geometry*, 1(1):343–353, 1986.

[37] M. Sharir. On k-sets in arrangements of curves and surfaces. *Discrete & Computational Geometry*, 6(1):593–613, 1991.

[38] P. Slavík. Improved performance of the greedy algorithm for partial cover. *Information Processing Letters*, 64(5):251–254, 1997.

[39] R. V. Vohra and N. G. Hall. A probabilistic analysis of the maximal covering location problem. *Discrete Applied Mathematics*, 43(2):175–183, 1993.

[40] P. Winkler. Puzzled figures on a plane. *Communications of the ACM*, 53(8), 2010.