Spring 2010

Clusters and covers: geometric set cover algorithms

Matthew Richard Gibson
University of Iowa

CLUSTERS AND COVERS:

GEOMETRIC SET COVER ALGORITHMS

by

Matthew Richard Gibson

An Abstract

Of a thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Computer Science
in the Graduate College of
The University of Iowa

May 2010

Thesis Supervisor: Associate Professor Kasturi Varadarajan

ABSTRACT

We consider several geometric special cases of the *set cover problem*. The first problem we consider is the *decomposing coverings* problem. Here, we consider a combinatorial problem: given a collection of points in the plane and a collection of objects in the plane such that each point is contained in at least k objects, partition the objects into as many sets as possible so that each set covers all of the points. We show that if the objects are translates of a convex polygon, then it is possible to partition the translates into $\Omega(k)$ covers.

The second problem we consider is the *planar sensor cover* problem. This problem is a generalization of the decomposing coverings problem. We are given a collection of points in the plane and a collection of objects in the plane. Each of the objects can be thought of as a sensor. The sensors have a *duration* which can be thought of as the battery life of the sensor. The planar sensor cover problem is to schedule a start time to each of the sensors so that the points are covered by a sensor for as long as possible. We give a constant factor approximation for this problem. The key contribution to this result is a constant factor approximation to a one-dimensional version of the problem called the *restricted strip cover* (RSC) problem. Our result for RSC improves upon the previous best $O(\log \log \log n)$-approximation, and our result for the planar sensor cover problem improves upon the previous best $O(\log n)$-approximation.

The next problem we consider is the *metric clustering to minimize the sum*

of radii problem. Here, we are given an n-point metric (P, d), and an integer $k > 0$. We are interested in covering the points in P with at most k balls so that the sum of the radii of the balls is minimized. We give a randomized algorithm which solves the problem exactly in $n^{O(\log n \log \Delta)}$ time, where Δ is the ratio of the maximum interpoint distance to the minimum interpoint distance. We also show that the problem is NP-hard, even in metrics induced by weighted planar graphs and when the metric has constant doubling dimension.

The last problem we consider is the *minimum dominating set* problem for *disk graphs*. In this problem, we are given a set of disks in the plane, and we want to choose a minimum-cardinality subset of disks such that every disk is either in the set or intersects a disk in the set. For any $\epsilon > 0$, we show that a simple local search algorithm is a $(1+\epsilon)$-approximation for the problem which improves upon the previous best $O(\log n)$-approximation algorithm.

Abstract Approved: _____
Thesis Supervisor

_____
Title and Department

_____
Date

CLUSTERS AND COVERS:

GEOMETRIC SET COVER ALGORITHMS

by

Matthew Richard Gibson

A thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Computer Science
in the Graduate College of
The University of Iowa

May 2010

Thesis Supervisor: Associate Professor Kasturi Varadarajan

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Matthew Richard Gibson

has been approved by the Examining Committee for the thesis requirement for the Doctor of Philosophy degree in Computer Science at the May 2010 graduation.

Thesis Committee: _____
Kasturi Varadarajan, Thesis Supervisor

_____
Sriram Pemmaraju

_____
Sukumar Ghosh

_____
Alberto Segre

_____
Jeffrey Ohlmann

ACKNOWLEDGEMENTS

There have been many people who have made a major influence in my experience doing algorithms research. Firstly, I must thank Kasturi Varadarajan. His guidance has helped me to read papers well and learn how to break a problem down in order to understand a problem inside and out. He is a coauthor on all of the work presented in this document, as well as several other papers that I coauthored. I am very fortunate for the opportunity to work with him.

I must also thank Gaurav Kanade, Erik Krohn, and Imran Pirwani. Between the four of us, we have spent many hours reading papers and discussing problems. The time spent with them was invaluable as I felt like I learned something new every time we met. Our hard work was rewarded with several publications, with much of the credit going to those three. They are coauthors of the work in Chapter 6. Also, Imran Pirwani and Mohammad Salavatipour are coauthors of the work in Chapter 7.

I would also like to thank Sriram Pemmaraju. I took three algorithms-related classes from Sriram and learned an incredible amount from him. Two of the three classes were very early in my time here, and Sriram's ability to clearly communicate sophisticated algorithmic techniques helped me to "grow up" as an algorithms researcher very quickly.

I would also like to thank Jeffrey Ohlmann. I enjoyed learning about heuristic techniques from him and had a great time working on our paper. While most of my time as a graduate student was spent doing algorithms research, I very much enjoy

working on heuristics, and because of my time working with Jeff, I feel confident that I could develop interesting heuristics in the future.

I finally would like to thank all members of the Algorithms Reading Group, both past and present. There have been many bright minds researching algorithms at the University of Iowa within the past five years, and the opportunity to learn from them has benefited me greatly. Thank you to all who have made the University of Iowa such a great place to do algorithms research.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

CHAPTER 1
INTRODUCTION

In this document are several pieces of work on different variants of the *geometric set cover problem*. The general set cover problem is well studied, see for example [27, 17, 39, 46]. Here, we are given a set of elements X and a set $\mathcal{F}$ of subsets of X. The goal is to pick some subset of $\mathcal{F}$ that satisfies some constraints and optimizes some objective value. For example, one might want to find the smallest subset of $\mathcal{F}$ that covers (contains) all of the elements in X. Or perhaps there is a weight assigned to each element in $\mathcal{F}$, and the goal is to find a subset of $\mathcal{F}$ that covers everything in X such that the sum of the weights of the elements in the subset is minimized.

In 1974, Johnson showed that a greedy algorithm for set cover gives an $O(\log n)$ approximation [39], where $n = |X|$. That is, the algorithm will return a solution which is at most an $O(\log n)$ factor worse than an optimal solution. In 1993, Lund and Yannakakis [47] and Bellare, Goldwasser, Lund, and Russell [9] showed that there is a positive constant c such that the general set cover problem cannot be approximated in polynomial time within a $c \log n$ factor unless $NP \subseteq DTIME(n^{O(\log \log n)})$. In 1997, Raz and Safra [58] showed that the problem cannot be approximated within a $c \log n$ factor for some constant c unless $NP = P$. In 1998, Feige [27] showed that the problem cannot be approximated to within a $(1 - o(1)) \log n$ factor unless $NP \subseteq DTIME(n^{O(\log \log n)})$. Note that the lower bounds and upper bounds are the same up to a constant factor, and thus the computational complexity for this problem is settled.

Suppose now that instead of dealing with the general problem, the set X is a collection of points in some geometric space and the sets $\mathcal{F}$ are subsets of X induced by the intersection of X with some geometric object. For example, X might be a collection of points in the plane, and each set in $\mathcal{F}$ might be the subset of points in X contained within some disk or triangle in the plane. For simplicity, we will call such a problem the set cover problem with disks, the set cover problem with triangles, and so on. Since we cannot hope to do better than an $O(\log n)$ approximation for general set cover, it would be interesting if the additional structure of the geometry can lead to better approximation algorithms, e.g. a constant factor approximation algorithm.

How well a geometric set cover problem can be approximated depends largely upon the problem at hand. For many problems, the best known approximation algorithm is the $O(\log n)$ approximation algorithm for general set cover, e.g. the set cover problem with axis-aligned rectangles. There are problems in which there are known approximation algorithms with approximation ratios better than $O(\log n)$ yet are still superconstant. For example, Varadarajan [61] and Aronov, Ezra, and Sharir [6] give an $O(\log \log \log n)$ approximation for set cover with "fat" triangles. As we will see in Section 1.2, Aloupis et al. [2] give an $O(1)$ approximation for a sensor cover problem when each sensor is a translate of a special type of convex polygon. Gibson, Kanade, Krohn, and Varadarajan showed that the problem of guarding an x-monotone polygonal chain has a polynomial time approximation scheme (PTAS) [34]. That is, for any $\epsilon > 0$, their algorithm returns a solution in polynomial time whose size is at most a factor of $1+\epsilon$ worse than the size of an optimal solution. Some

geometric set cover problems can be solved exactly. For example, Gibson, Kanade, Krohn, Pirwani, and Varadarajan showed that the *geometric clustering to minimize the sum of radii* problem is polynomial time solvable [33]. Very few hardness of approximation results are known. One recent hardness result due to Har-Peled [35] is that there is no PTAS for set cover with fat triangles. Compare this to the best known $O(\log\log\log n)$ algorithm, and it is easy to see that there is much left to understand about this problem specifically, and many variants of set cover in general.

In this document, we will describe some algorithms for some variants of the set cover problem. The first being the *decomposing multiple coverings* problem. Here, we are given a collection of points X in the plane and a collection of translates of some object P such that each point in X is contained in at least k of the translates. We would like to partition the translates into as many sets as possible so that each set covers all of the points in X. We show that if P is any convex polygon, then we can partition the translates of P into $\Omega(k)$ covers. This can be contrasted with the case when P is a concave polygon. We will see in Section 1.1.2 that there are examples for any $k > 0$ that it is not possible to partition the translates into 2 sets so that both sets cover all of the points.

Next, we will look at two variants of the *sensor cover* problem. Here, we are given a collection of points (in the plane, on a line, etc.) and a collection of objects, each of which has a duration and covers some geometric subset of the points. We think of the duration as being the battery life of some sensor, i.e. the maximum amount of time that the sensor can cover the points that it contains. We would like

for the points to be monitored by the sensors for as long as possible. We could turn all of the sensors on initially, but in doing so we may be redundantly covering the points. It may be advantageous to turn on only a subset of the sensors initially, and then activate more later after some of the first sensors durations have been depleted. We will give constant factor approximation algorithms for both of the variants that we consider.

Then, we will look at the *metric clustering problem to minimize the sum of radii*. This is a generalization of the geometric version mentioned earlier. In this problem, we are given a positive integer k and a set of points with interpoint distances that satisfy the condition of being a "metric." We want to cover all of the points using at most k balls, each of which is centered at one of the input points, so that the sum of the radii of the balls is minimized. When the input points lie in a geometric space and interpoint distances are the corresponding geometric distances, there is an exact polynomial time algorithm for the problem. We will show that in the general metric case, we can get a $n^{O(\log n \log \Delta)}$ time exact algorithm where Δ is the aspect ratio of the problem (the ratio of the longest interpoint distance to the shortest interpoint distance). This is a quasipolynomial time algorithm when Δ is bounded by a polynomial in n. However, when Δ is exponential in n, we will show that the problem is NP-hard, even when the metric is induced by a planar graph and when the metric has the so called property "constant doubling dimension."

Finally, we will consider the *minimum dominating set problem for disk graphs*. Here, we are given a collection of disks $\mathcal{D}$ in the Euclidean plane. We say that two

disks are neighbors if and only if they intersect. A subset of the disks $\mathcal{D}' \subseteq \mathcal{D}$ is a *dominating set* if for every disk $d \in \mathcal{D}$, either $d \in \mathcal{D}'$ or d has a neighbor in $\mathcal{D}'$. The minimum dominating set problem (MDS) is to compute the minimum cardinality dominating set. We show that a local search algorithm is a PTAS for this problem for disk graphs, improving upon the previous best $O(\log n)$ approximation.

1.1 Decomposing Multiple Coverings

Given a collection of objects (sets) in the plane that cover every point in the plane, we say that the objects are *cover-decomposable* if they can be partitioned into two covers. Let us call an object (set) P in the plane *cover-decomposable* if there there exists a constant $c > 0$ (which may depend on P) such that any collection of translates of P, with the property that every point in the plane has c or more translates covering it, can be partitioned into two covers. Pach conjectured in the 1980s that every convex object is cover decomposable [51, 52], and this remains open. Let us focus on a finite version of this definition. Given a set of points and a collection of objects in the plane, we say that the objects are cover-decomposable if they can be partitioned into two sets so that both sets cover all of the points. Given a set of points, we call a covering of the points with a collection of objects a *c-fold covering* if every point is contained within at least c of the objects. We say that P is cover-decomposable if there exists a constant $c > 0$ such that any c-fold covering of a point set X by a finite collection of translates of P can be partitioned into two sub-collections, so that each sub-collection covers every point in X. That is, if whenever a collection of translates

of P form a c-fold covering of a point set X the translates can be partitioned into two 1-fold covers of X, then we say that P is cover-decomposable.

1.1.1 An Example of Cover-Decomposability

To see an example of cover-decomposability, consider the following one dimensional problem. Suppose we are given a collection of points that all lie on a horizontal line ℓ, and suppose we are also given a collection of intervals I of ℓ. We will show that if every point is contained in at least 3 intervals, then we can partition the set of intervals into two sub-collections such that each sub-collection covers all of the points.

For a collection of intervals I', we say an interval $v \in I'$ is redundant if the intervals in $I' \setminus v$ covers the exact same subset of ℓ as do the intervals in I'. Consider Algorithm 1.1.

Algorithm 1.1

1: $I' \leftarrow I$

2: **while** there exists a redundant interval $v \in I'$ **do**

3: $\quad I' \leftarrow I' \setminus \{v\}$

4: **end while**

5: $C_1 \leftarrow I'$

6: $C_2 \leftarrow I \setminus I'$

Consider the intervals in C_1. Clearly, C_1 is a cover of the input points. Note

that no interval in C_1 is redundant, i.e. every interval in C_1 uniquely covers some point of ℓ. We will show that for any point $l \in \ell$, l is covered by at most two intervals in C_1. Note that if we are able to show this, then we will be done because every point is covered by at least 3 intervals in I, and thus there must be an interval in C_2 that also covers l.

Suppose there is some point $l \in \ell$ that is covered by 3 intervals in C_1. Since we are thinking of ℓ as being a horizontal line, we will use the natural notion of a left (resp. right) endpoint of an interval. Let v_1, v_2, and v_3 be any three distinct intervals that cover l. Assume without loss of generality that the left endpoint of v_1 is at least as far to the left as the left endpoints of v_2 and v_3. If the right endpoint of v_1 is at least as far to the right as the right endpoints of v_2 and v_3, then clearly both v_2 and v_3 are redundant. Otherwise, assume that the right endpoint of v_2 extends at least as far to the right as the right endpoints of v_1 and v_3. Clearly v_3 must be a subset of $v_1 \cup v_2$, and thus v_3 is redundant, a contradiction.

1.1.2 An Example of a Polygon that is Not Cover-Decomposable

To contrast the notion of an object being cover-decomposable, we will now give the high level ideas of a recent result of Pálvölgyi [53] that shows that there exist concave polygons which are not cover-decomposable. That is, for any $c > 0$, we can construct a point set and a c-fold covering using translates of a concave polygon P such that for every partition of the translates into two sets C_1 and C_2, one of the two

sets does not cover some point that needs to be covered.

Using standard techniques (which will be formally stated in Chapter 2), we reduce the problem to coloring points inside of wedges which correspond to the wedges of the polygon P. In this setting, we have a point set and a set of translates of wedges such that each wedge contains at least c points. To show cover decomposability, we would need to show that we can color the points either red or blue so that every wedge contains at least one point of each color (we think of the points as covering the wedges, and the points colored red correspond to C_1 and the points colored blue correspond to C_2). However Pálvölgyi shows that it is possible to construct an example using translates of a wedge V and a wedge W (which correspond with the wedges of a concave polygon, see Figure 1.1) so that no matter how you color the points, there either exists a translate of V which contains all red points or there is a translate of W which contains all blue points.

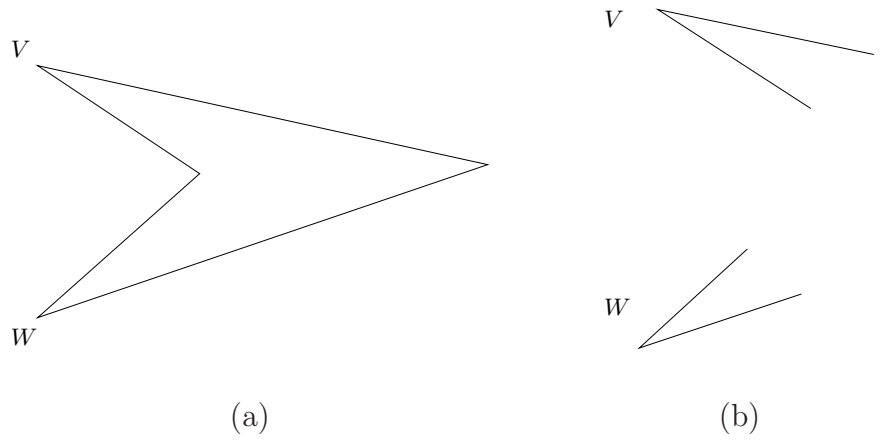


Figure 1.1: An illustration of the wedges of a polygon. (a) Suppose this is our polygon with vertices V and W labeled accordingly. (b) The construction will use translates of a wedge V and of a wedge W.

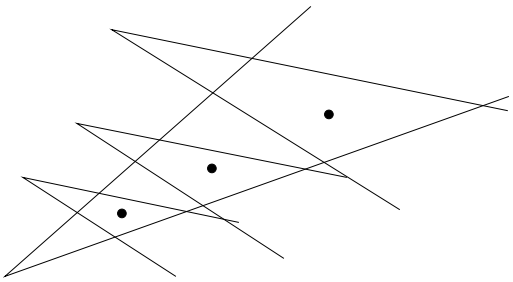


Figure 1.2: Illustration for $P(k,l)$ with $k=1$ and $l=3$.

We will construct a gadget $P(k,l)$ which consists of points plus translates of V and W. The gadget will have the property that no matter which red-blue coloring of the points we choose, there is a translate of V containing k red points and no blue points or there is a translate of W containing l blue points and no red points. When $k=1$, place l points and l translates of V so that each translate contains exactly one of the points and each of the points lies within exactly one translate of V. We then add one translate of W that contains all of the l points. See Figure 1.2 for an example. Clearly either the translate of W contains l blue points and no red points, or one of the translates of V contains $k=1$ red point and no blue points. A similar construction holds for the case when $l=1$.

Now suppose that we want to construct $P(k,l)$ for $k>1$ and $l>1$. Assume that we already have $P(k',l')$ for all $k'+l'<k+l$. We will show how to construct $P(k,l)$ using $P(k-1,l)$ and $P(k,l-1)$. Place a point p, then place a copy of $P(k-1,l)$ so that all translates of V from $P(k-1,l)$ contain p and none of the translates of W from $P(k-1,l)$ contain p. Similarly, place a copy of $P(k,l-1)$ so that all translates of W from $P(k,l-1)$ contain p and none of the translates of V from $P(k,l-1)$ contain

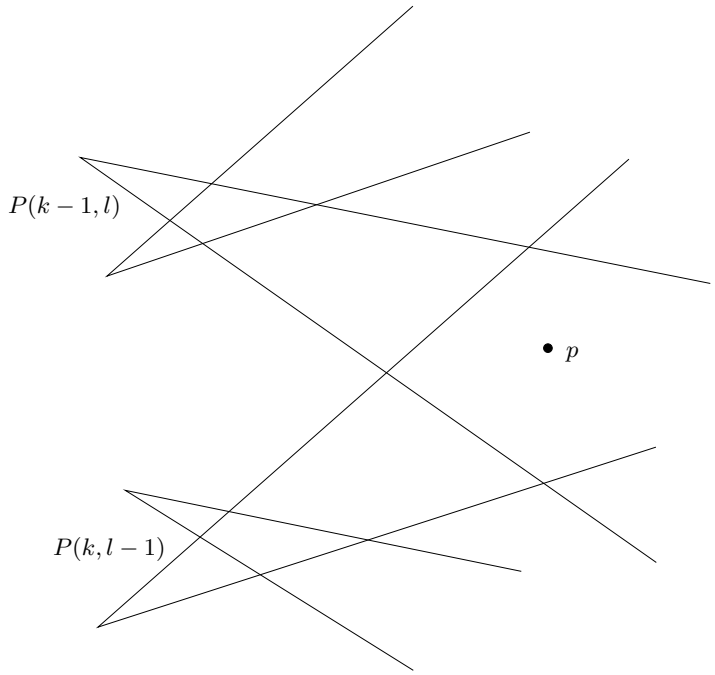


Figure 1.3: Illustration for $P(k,l)$.

p. See Figure 1.3. We will now show that this construction satisfies the requirements for $P(k,l)$.

Suppose p is colored red. One of the following two cases must occur:

1. There is a translate of V from the copy of $P(k-1,l)$ which contains $k-1$ red points and no blue points. This wedge now contains p, which gives it k red points and no blue points.

2. There is a translate of W from the copy of $P(k-1,l)$ which contains l blue points and no red points. Since this translate does not contain p, then it still holds that it contains l blue points and no red points.

We can argue in a similar fashion for the wedges of $P(k,l-1)$ in the case when p is

colored blue. Thus we have that there is either a translate of V which contains k red points and no blue points or there is a translate of W which contains l blue points and no red points.

Now for any $c > 0$, we can construct $P(c, c)$. Thus for any coloring of the points, there will either be a translate of V which contains c red points and no blue points, or there will be a translate of W which contains c blue points and no red points. Note that all wedges necessarily must contain at least c points. This then implies that there is a c-fold covering of a point set by translates of a concave polygon that cannot be partitioned into two covers.

1.1.3 Previous Work

In the 1980's, Mani and Pach [48] showed in an unpublished manuscript that the unit disk is cover decomposable (with the constant c being 33.) Also in the 1980's, Pach [51] showed that any centrally symmetric convex polygon is cover-decomposable. Tardos and Tóth [60] showed somewhat more recently that any triangle is cover-decomposable. Finally, a very recent result due to Pálvölgyi and Tóth [54] shows that any convex polygon is cover-decomposable. The constant c in the results of [51] and [54] depends on the convex polygon, in particular the number of its sides, and that is why these results say nothing about the original conjecture of Pach. Examples of non-convex polygons that are not cover-decomposable are known [53] (as was described in Section 1.1.2).

Motivated partly by questions in scheduling sensors [11], an extension of the

cover-decomposability question has recently attracted a lot of attention: Given a collection of translates of P and any integer k, partition the collection into as many sub-collections as possible so that each sub-collection covers every point covered by k or more of the original translates. That is, we would like to be able to decompose a k-fold cover into as many disjoint covers as possible. While the original results on cover-decomposability do yield non-trivial bounds for this question, these are usually far from optimal. For instance, Tardos and Tóth [60] implies that a k-fold cover with translates of a triangle can be partitioned into $\Omega(\log k)$ covers.

In this line of work, Pach and Tóth [52] showed that a k-fold cover with a centrally symmetric convex polygon P can be decomposed into $\Omega(\sqrt{k})$ covers, where the constant as before depends on P. Aloupis et al. [2] improved this result and obtained an optimal bound, showing that one can obtain $\Omega(k)$ covers. Both of these results have corresponding efficient algorithms that compute the desired decompositions. Aloupis et al. [3] consider other related problems.

The problem of decomposing multiple coverings seems to be harder if instead of a convex polygon we have a unit disk. Pandit, Pemmaraju and Varadarajan [55] consider a special case where the universe that needs to be covered is the same as the centers of the covering disks. For this version of the problem, better known as the domatic partition problem for *unit disk graphs* [56], they show that it is possible to compute $\Omega(k)$ disjoint covers in polynomial time.

1.1.4 Our Contribution

We obtain an optimal result for translates of an arbitrary convex polygon:

Theorem 1. *For any convex polygon P in the plane, there exists a constant $\alpha \geq 1$ so that for any $k \geq 1$ and any finite collection of translates of P, we can partition the collection into k/α sub-collections, each of which covers any point in the plane that is covered by k or more translates in the original collection. Such a partition can be computed by an efficient algorithm.*

Our techniques build upon the recent work of Aloupis et al. [2] for centrally symmetric convex polygons. A polygon is centrally symmetric if there is a point $o \in \Re^2$ such that whenever the polygon contains point $o + p$ it also contains $o - p$. A key idea of theirs is to focus on the level curves corresponding to the wedges at the vertices of P. The interaction of these level curves can be complex, but they show that is sufficient to work within a region where the interaction is much more controlled. It is only for centrally symmetric convex polygons that they establish such nice properties of the interaction. The notion of level curves is also central to our work, but the main point of departure is the simplicity of the new way in which we handle the level curve interactions.

In Chapter 2, we give some preliminaries and an algorithm for centrally symmetric convex polygons which will serve as a foundation for proving Theorem 1 in Chapter 3.

1.2 The Planar Sensor Cover Problem

Suppose we have a universe, which is simply some collection of points, and a set of sensors such that each sensor covers some subset of the universe. Further suppose that each sensor is powered by a battery and thus can only be turned on for some amount of time. We refer to this amount of time as the sensor's *duration*. We are interested in scheduling a start time to each of the sensors such that the entire universe is covered for as long as possible. This problem was introduced by Buchsbaum et al. [11] as the *sensor cover problem*. We only consider the non-preemptive case in which once a sensor has been turned on, it will remain on until its duration has been depleted. In the preemptive case, research has shown that partitioning the sensors into covers and iterating through the covers helps to increase the lifetime of a sensor network [1, 19, 57, 59, 21].

We now formally define the general sensor cover problem, followed by the geometric instances that are the subject of Chapters 4 and 5. We are given a finite universe U that we wish to cover, and a set $\mathbf{S}$ of n sensors. For each sensor $s \in \mathbf{S}$, we let $R(s) \subseteq U$ denote the region that s covers. We call this region the *range* of s. For each $x \in R(s)$, we say that s is *live* at x. Each sensor s also has a duration $d(s)$ which is some positive integer.

A schedule of the set $\mathbf{S}$ of sensors is an assignment of a positive integer, called the start time, to each sensor in some subset $S' \subseteq \mathbf{S}$. We will denote by $t(s)$ the start time of sensor s. The sensors in $\mathbf{S} \setminus S'$ are said to be unassigned. A sensor s that is assigned a start time $t(s)$ is said to be *active* at times $\{t(s), t(s)+1, ..., t(s)+d(s)-1\}$.

Let S be some schedule of $\mathbf{S}$. A point $x \in U$ is said to be covered at time $t > 0$ if there is a sensor s such that $x \in R(s)$ and s is active at time t. For each $x \in U$, define the duration of x in the schedule to be $M(S, x) = \max\{j : \forall j' \leq j, \exists s \in S, s \text{ covers } x \text{ at time } j'\}$. (If no sensor covers x at time 1, then define $M(S, x) = 0$.) The duration of the schedule S is defined to be $M(S) = \min_x M(S, x)$. The goal of the problem is to compute a schedule of maximum duration.

The *load* at a point $x \in U$ is $L(x) = \sum_{s \in \mathbf{S}: x \in R(s)} d(s)$. The *load* of the problem instance is $L = \min_x L(x)$. Let OPT denote the duration of an optimal schedule. Clearly, $OPT \leq L$, and thus any approximation ratio that is with respect to L is also with respect to OPT.

We are particularly interested in two geometric instances of the sensor cover problem:

- **Restricted Strip Cover.** Here, the universe U is a set of points on the real line, and the range $R(s)$ of each sensor s is equivalent to the intersection of U with an interval on the real line.

- **Planar Sensor Cover for Convex Polygons.** Here, the universe U is a set of points in $\Re^2$, and the range $R(s)$ of each sensor is equivalent to the intersection of U with a translate of a fixed convex polygon. In the remainder of this paper, we will refer to this problem as simply the planar sensor cover problem.

1.2.1 Previous Work.

The General Sensor Cover. A closely related problem to the sensor cover problem is the *domatic partition problem*. In this problem, we are given a graph with the goal of finding the maximum number of disjoint dominating sets. A *dominating set* is a subset of the vertices such that for each vertex in the graph, either it is in the set or it has a neighbor in the set. Domatic partition can be viewed as a special case of the sensor cover problem where the universe is the vertex set, each vertex of the graph is a sensor, the range of each sensor is its corresponding vertex's closed neighborhood, and each sensor has unit duration. Feige et al. [28] show that it is NP-hard to approximate this problem to within a $\log n$-factor and give a simple randomized algorithm that achieves an $O(\log n)$-approximation, where n is the number of vertices in the graph.

As pointed by Buchsbaum et al. [11], the lower bound above given by Feige et al. implies that general sensor cover cannot be approximated to better than a $\log n$ factor. On the positive side, Buchsbaum et al. [11] present a poly-time algorithm for the sensor cover problem that returns an $O(\log |U|)$ approximation. This algorithm extends an algorithm for the set cover packing problem [28], which is the special case of the sensor cover problem with the duration of all sensors being 1. In many applications, the sensors do not cover arbitrary subsets of the universe, but rather the points in the universe lie in some geometric space and the sensors cover some geometric subset of the universe. In such cases, we will see that it is possible to do better than the $\log n$ lower bound for general sensor cover.

Restricted Strip Cover. The Restricted Strip Cover problem (RSC) was introduced and studied by Buchsbaum et al. [11]. Here, we think of the universe as being points on a horizontal line, and each sensor covers some sub-interval of the line. If there are m points to be monitored, then we number them from left-to-right, i.e. the leftmost point is 1 and the rightmost point is m. Buchsbaum et al. showed that if each of the sensors have unit duration, then a greedy algorithm will achieve a schedule of duration L. The algorithm proceeds left-to-right computing a schedule S and maintains two invariants: (a) no sensors overlap at any point $\geq i$, and (b) $M(S, i) = L$.

The algorithm starts at point $i = 1$. Select any L sensors which are live at 1 and schedule them so that they do not overlap, i.e. the point 1 will be covered at all times $\leq L$. Note that after this initial step, the two invariants are satisfied. Now, for point $i+1$, if $M(S, i+1) = L$, then we are done, and can go on to $i+2$. If not, then there are $k > 0$ unit-duration gaps at $i+1$. The invariants imply that there must be at least k sensors which are live at $i+1$ which have not yet been scheduled, and so we will be able to fill these gaps in without violating the invariants.

The problem becomes much more difficult if the sensors have arbitrary durations. As Buchsbaum et al. pointed out [12], there are instances of RSC where $OPT < L$. See Figure 1.4 for an illustration of this which appeared in [12].

Buchsbaum et al. [11] showed that RSC is NP-hard and give a polynomial-time $O(\log \log \log n)$-approximation algorithm, where n is the number of sensors. They show that their algorithm does better for special cases of RSC. In particular, they

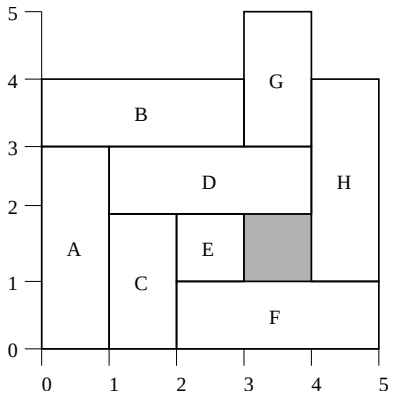


Figure 1.4: Figure from [12]. The shaded region is a gap. In this example, $L = 4$ and $OPT = 3$ which is realized by shifting sensor G down to cover the shaded region.

show that their algorithm is a $(2 + \epsilon)$-approximation for any $\epsilon > 0$ when the sensors are non-nested; this includes the case where the ranges of all the sensors have the same size.

The RSC bears some resemblance to the well studied *dynamic storage allocation* [32, 12]. The RSC problem can be viewed in the following way. We are given a set of rectangles, and we are allowed to slide each rectangle vertically; the goal is to find a placement of the rectangles so that we cover a horizontal strip that is as tall as possible. In the dynamic storage allocation problem, we are also given a set of rectangles, each of which we are allowed to slide vertically; the goal is to find a placement of the rectangles such that no two of them overlap and the rectangles are contained in a horizontal strip that is as short as possible. The dynamic storage allocation problem admits constant factor approximation algorithms [32, 12], and these are with respect to the load, where now the load is the *maximum* of the pointwise loads. We refer the reader to [11] for a review of the similarities of the RSC to other

problems [1, 19, 21, 30, 41, 57, 59] studied in the literature.

Planar Sensor Cover. The special case of the planar sensor cover problem where each sensor has unit duration has received a lot of attention recently. Consider an instance with universe U, sensors $\mathbf{S}$, and load L. We know that each $x \in U$ is covered by at least L sensors (convex polygons). One may ask a combinatorial question: Is it possible to partition $\mathbf{S}$ into $\Omega(L)$ sets each of which is a cover for U? As seen in Subsection 1.1.3, Aloupis et al. [2] showed that the answer is "yes", and in fact they give a polynomial time algorithm to construct $\Omega(L)$ covers. Thus they obtain a poly-time algorithm for computing a constant factor approximation to the planar sensor problem when each sensor has unit duration.

For the planar sensor cover problem with the durations of the sensors not being the same, the best known result is the logarithmic approximation inherited from the combinatorial sensor cover problem.

1.2.2 Our Contribution.

We have two main results: firstly, we give a constant factor approximation algorithm for RSC, and secondly, we use it to give a constant factor approximation for the planar sensor cover problem.

Restricted Strip Cover. We improve upon the $O(\log \log \log n)$ approximation of [11] and give the first constant factor approximation (a ratio of 5) for RSC. The work of [11] starts off with the observation that if all the sensors have unit duration then it

is possible to compute a schedule whose duration is equal to the load of the instance. The case of non-uniform duration is handled by reduction to several instances of the uniform duration case. The tool used for this is a technique called *grouping* where several sensors of small duration are combined to form one sensor of large duration. The question of how the groups are to be formed is addressed in a clever way, but the reduction entails a non-constant loss in the load and hence the $O(\log \log \log n)$ approximation factor.

We take a different and conceptually simpler approach here. Our algorithm is greedy and schedules sensors one by one. The scheduling rule manages to ensure that we do not have more than 5 sensors overlapping any particular point at one time. Hence we obtain a schedule whose duration is at least a fifth of the load. One idea that the scheduling rule uses is that if there are two sensors s and s' such that $R(s)$ is strictly contained in $R(s')$, we schedule s' before we schedule s. Another idea is to consider the duration of the sensors in an indirect way – for the next sensor to be scheduled, the durations of the unscheduled sensors is irrelevant but only their ranges; however the durations of the already scheduled sensors does play a crucial rule. Since our algorithm is greedy it has a simple implementation with a reasonable running time. We have not attempted to optimize the factor of 5 that our analysis guarantees.

The algorithm and its analysis can be found in Chapter 4.

Planar Sensor Cover. We give a constant factor approximation for the planar sensor cover problem, where the range of each sensor is a translate of a convex polygon, improving upon the previous best logarithmic factor. Our work on this problem is inspired by the approach taken by Aloupis et al. [2] and generalizes their results to the case where each of the sensors has varying duration. Essentially, we show that we can obtain a constant factor approximation for a convex polygon with μ vertices by invoking μ instances of the RSC which we solve using our 5-approximation. The simplicity of our greedy algorithm for RSC and some of its properties play a crucial role here.

This work can be found in Chapter 5.

1.3 Metric Clustering to Minimize the Sum of Radii

A *metric* d is a function on a set of points P, such that each of the following holds:

1. $d(u, u) = 0$ for each $u \in P$.
2. $d(u, v) = d(v, u)$ for each $u, v \in P$.
3. $d(u, v) \leq d(u, z) + d(z, v)$ for each $u, v, z \in P$.

Given a metric d defined on a set P of n points, we define the ball $\mathrm{B}(v, r)$ centered at $v \in P$ and having radius $r \geq 0$ to be the set $\{q \in P | d(v, q) \leq r\}$. In Chapter 6, we consider the problem of computing a minimum cost k-cover for the given point set P, where $k > 0$ is some given integer which is also part of the input.

For $\kappa > 0$, a κ-cover for subset $Q \subseteq P$ is a set of at most κ balls, each centered at a point in P, whose union covers (contains) Q. The cost of a set $\mathcal{D}$ of balls, denoted cost($\mathcal{D}$), is the sum of the radii of those balls.

1.3.1 Previous Work

This problem and its variants have been well examined, motivated by applications in clustering and base-station coverage [23, 15, 44, 10, 4].

Doddi et al. [23] consider the metric min-cost k-cover problem and the closely related problem of partitioning P into a set of k clusters so as to minimize the sum of the cluster diameters. Following their terminology, we will call the latter problem *clustering to minimize the sum of diameters*. They present a bicriteria poly-time algorithm that returns $O(k)$ clusters whose cost is within a multiplicative factor $O(\log(n/k))$ of the optimal. For clustering to minimize the sum of diameters, they also show that the existence of a polynomial time algorithm that returns k clusters whose cost is strictly within 2 of the optimal would imply that $P = NP$. Notice that this hardness result does not imply the NP-hardness of the k-cover problem. Charikar and Panigrahy [15] give a poly-time algorithm based on the primal-dual method that gives a constant factor approximation – around 3.504 – for the k-cover problem, and thus also a constant factor approximation for clustering to minimize the sum of diameters.

The well known k-center problem is a variant of the k-cover problem where the cost of a set of balls is defined to be the maximum radius of any ball in the

set. The problem is NP-hard and admits a polynomial time algorithm that yields a 2-approximation [37]. Several other formulations of clustering such as k-median and min-sum k-clustering are NP-hard as well [40, 20].

Gibson et al. [33] consider the geometric version of the k-cover problem where $P \subset \Re^l$ for some constant l. When the L_1 or L_∞ norm is used to define the metric, they obtain a polynomial time algorithm for the k-cover problem. With the L_2 norm, they give an algorithm that runs in time polynomial in n, the number of points, and in $\log(1/\epsilon)$ and returns a k-cover whose cost is within $(1+\epsilon)$ of the optimal, for any $0 < \epsilon < 1$.

1.3.2 Our Contribution

Our first result generalizes the algorithmic approach of Gibson et al. [33] to the metric case. For the k-cover problem in the general metric setting, we obtain an exact algorithm whose running time is $n^{O(\log n \cdot \log \Delta)}$, where Δ is the *aspect ratio* of the metric space, the ratio between the maximum interpoint distance and the minimum interpoint distance. The algorithm is randomized and succeeds with high probability. Thus when Δ is bounded by a polynomial in n, the running time of the algorithm is quasi-polynomial. This result for the k-cover problem should be contrasted with the NP-hardness results for problems such as k-center, k-median, and min-sum k-clustering, which hold when the aspect ratio is bounded by a polynomial in n.

The main idea that underlies this result is that if we probabilistically partition the metric into sets with at most half the original diameter [8, 26], then with high

probability only $O(\log n)$ balls in the optimal k-cover of P are "cut" by the partition. A recursive approach is then used to compute the optimal k-cover.

This algorithmic result raises the question of whether an algorithm whose running time is quasi-polynomial in n is possible even when the aspect ratio is not polynomially bounded. Our second result shows that this is unlikely by establishing the NP-hardness of the k-cover problem. The aspect ratio in the NP-hardness construction is about 2^n. The metrics obtained are induced by weighted planar graphs, thus establishing the NP-hardness of the k-cover problem for this special case.

Our final result is that the k-cover problem is NP-hard in metrics of constant doubling dimension for a large enough constant. This result is somewhat surprising given the positive results of [33] for fixed dimensional geometric spaces.

This work can be found in Chapter 6.

1.4 Minimum Dominating Set for Disk Graphs

Given a graph $G = (V, E)$, a *dominating set* is a subset of the vertices $V' \subseteq V$ such that for every vertex $v \in V$, either $v \in V'$ or v has a neighbor in V'. The *minimum dominating set* (MDS) problem is to find a minimum cardinality dominating set. See Figure 1.5 for an illustration.

MDS is motivated by applications in sensor networks in which the vertices of the graph represent sensors and an edge connecting vertices u and v represents the fact that u and v are able to communicate with each other. If a dominating set is

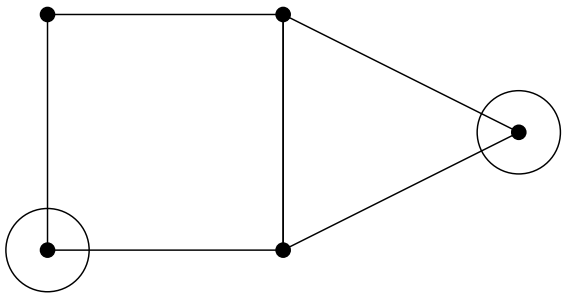


Figure 1.5: An example of a dominating set. The circled vertices form a dominating set for this graph.

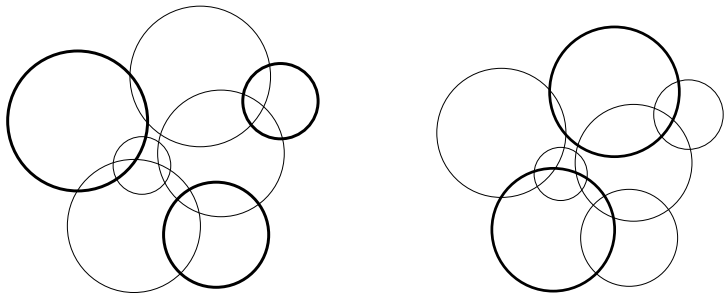
computed, each sensor in the dominating set can monitor all of its neighbors, and by the definition of dominating set, every sensor in the network will be monitored. One might be interested in monitoring the network as cheaply as possible, and finding a MDS might be a good solution for doing so.

In Chapter 7, we will focus on the MDS problem on disk graphs. A *disk graph* of a given set $\mathcal{D}$ of n disks in the Euclidean plane is simply the *intersection graph* of the disks. That is, the vertices of the graph are the disks, and two disks are neighbors if and only if they intersect. We are interested in the MDS problem for disk graphs. That is, we want to find a smallest cardinality set of disks $\mathcal{D}' \subseteq \mathcal{D}$ such that for every $d \in \mathcal{D}$, either $d \in \mathcal{D}'$ or d intersects a disk in $\mathcal{D}'$. See Figure 1.6 for an illustration.

1.4.1 Previous Work on Dominating Set

On general graphs, the problem is $(1 - \epsilon) \ln n$ hard to approximate for any $\epsilon > 0$ [27, 16], while a greedy algorithm yields an $O(\log n)$ approximation [62] due to its connection to the *set cover problem*.

Nevertheless, better approximations are possible for restricted domains. For



(a) (b)

Figure 1.6: An illustration of a dominating set for a disk graph. (a) A dominating set of size 3 for a disk graph. (b) A dominating set of size 2 for the same disk graph. In the MDS problem, we would prefer the solution with 2 disks.

example, the problem admits a *polynomial-time approximation scheme* (PTAS) for unit disk graphs and *growth-bounded graphs* [38, 50]. The problem is NP-hard on these domains [18]. However, for the disk graph case, $o(\log n)$ approximations have remained elusive – perhaps, in part, because known techniques for unit disk graphs and solutions to other problems on disk graphs have either relied on packing properties [38, 50, 24, 13], or when packing properties do not hold, as in the *minimum weighted dominating set* on unit disk graphs, the fact that disk radii are uniform [5, 55]. Erlebach and van Leeuwen recently studied the dominating set problem on *fat objects*, e.g., disk graphs, [25]. They note that existing techniques for disk graphs do not seem sufficient to solve MDS [25]; they also give an $O(1)$-approximation for fat objects of *bounded ply*.

1.4.2 Our Result

We give a PTAS for MDS on disk graphs, improving upon the algorithm for general graphs. Our result is inspired by two recent results given by Chan and Har-Peled [14] and Mustafa and Ray [49]. Both papers (independently) show how a simple *local search* algorithm on certain geometric graphs yields a PTAS for some problems; Chan and Har-Peled [14] show local search yields a PTAS for maximum independent set problem on admissible objects, while Mustafa and Ray [49] show local search yields a PTAS for the minimum hitting set problem given a collection of points and half-spaces in $\mathbb{R}^3$, and also for points and admissible regions in $\mathbb{R}^2$. They both use the *planar separator theorem* to relate the cost of the local search solution with the optimum solution. In the framework, at the crux lies an argument proving the existence of an appropriate planar bi-partite graph whose vertices are objects found by local search and ones that belong to an optimum solution, and the edges are only between the two kinds of vertices. Mustafa and Ray [49] refer to this as the *locality condition*.

Our main contribution is to show that the appropriate planar graph does indeed exist for the MDS problem on disk graphs. We then can use the planar separator theorem to show that a local search algorithm is a PTAS for this problem. This work can be found in Chapter 7.

CHAPTER 2
DECOMPOSING COVERINGS: CENTRALLY-SYMMETRIC POLYGONS

In this Chapter, we will consider the decomposing coverings problem for centrally symmetric convex polygons. Here, we are given a set of points Q and a set of translates of a centrally-symmetric convex polygon P. We will show that it is possible to partition the translates into $\Omega(k)$ sets such that each point in Q that is contained within at least k of the translates is contained within a translate from each of the sets.

In Section 2.1, we recall crucial tools from previous work on the problem of decomposing multiple coverings. In Section 2.2, we describe a simple algorithm to decompose a k-fold covering with translates of a centrally-symmetric polygon into $\Omega(k)$ covers.

2.1 Preliminaries

It is convenient to prove Theorem 1 in its dual form as done in [60, 2]. Suppose we are given a polygon P. Fix O, the centroid of P, as the origin in the plane. For a planar set T and a point x in the plane, let $T(x)$ denote the translate of T with centroid x. Let $\bar{P}$ be the reflection through O of the polygon P. For points p and x in the plane, $p \in P(x)$ if and only if $x \in \bar{P}(p)$.

Because of this transformation, it is sufficient for us to show that there exists a constant $\alpha \geq 1$ so that for any $k \geq 1$ and any collection Q of points in the plane, it

is possible to assign each point in Q a color from $\{1, 2, \ldots, \frac{k}{\alpha}\}$, so that any translate of $\bar{P}$ with $|\bar{P} \cap Q| \geq k$ contains a point colored i, for each $1 \leq i \leq \frac{k}{\alpha}$.

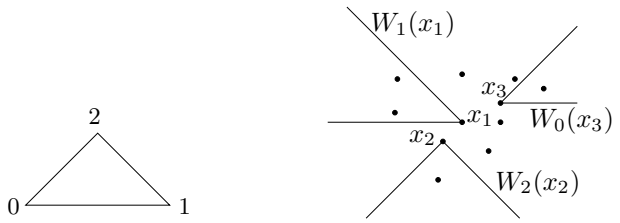
Polygons to Wedges. Denote the vertices of $\bar{P}$ to be $p_0, p_1, p_2, \ldots, p_{\mu-1}$ in counterclockwise order. Addition and subtraction of indices of these vertices will be taken modulo μ throughout the paper. The set of indices between index i and index j in counterclockwise order are denoted $[i, j]$. We now transform the problem further, so that instead of dealing with translates of $\bar{P}$, we can deal with translates of the μ *wedges* corresponding to the vertices of $\bar{P}$ [52, 60, 2].

Let c be equal to half the minimum distance between two points on non-consecutive edges of $\bar{P}$. We lay a square grid of side c on the plane; any translate of $\bar{P}$ intersects $\beta \in O(1)$ grid cells, and each grid cell intersects at most two sides of a translate; moreover, if a grid cell does intersect two sides of a translate, then these sides must be adjacent in $\bar{P}$.

For a subset (region) R of the plane and for a finite subset X of points, denote $\text{load}_X(R)$ to be the number of points in X that lie in R. We call this value the load of region R with respect to X. Since each translate $\bar{P}(u)$ intersects at most β grid cells, $\bar{P}(u)$ must contain load at least k/β within some grid cell if its load with respect to Q is at least k. We can therefore make the points of Q within such a grid cell "responsible" for $\bar{P}(u)$.

Since each grid cell intersects at most two edges of $\bar{P}(u)$, it must be that the intersection of a grid cell with $\bar{P}(u)$ is the same as the intersection of the grid cell

with a wedge whose boundaries are parallel to two adjacent edges of $\bar{P}(u)$. If one boundary of the wedge is parallel to the edge $p_{i-1}p_i$ of $\bar{P}$ and the other is parallel with p_ip_{i+1} of $\bar{P}$, then we call the wedge an i-wedge. For a point q in the plane, we denote $W_i(q)$ to be the i-wedge with apex q. See Figure 2.1 for an illustration.



(a) (b)

Figure 2.1: An illustration for the wedges of a polygon. (a) Suppose this triangle is our polygon with vertices indexed accordingly. (b) A 0-wedge, 1-wedge, and 2-wedge with respect to the polygon.

Because of these observations, Theorem 1 is established by applying the following theorem to the points Y within each grid cell G.

Theorem 2. *There exists a constant $\alpha' \geq 1$ so that for any $k \geq 1$ and any collection Y of points in the plane, it is possible to assign each point in Y a color from $\{1, 2, \ldots, \frac{k}{\alpha'}\}$, so that any i-wedge that contains k or more points from Y contains a point colored j, for each $1 \leq j \leq \frac{k}{\alpha}$.*

We prove Theorem 2 for wedges of a centrally-symmetric convex polygon in this chapter, and we prove the same theorem for wedges of a general convex polygon in Chapter 3. We assume that the point set Y is in general position – a line parallel

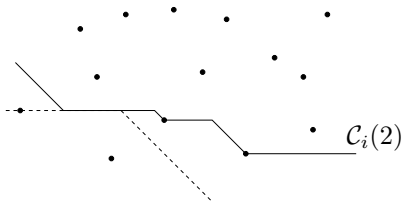


Figure 2.2: An example of a level curve $\mathcal{C}_i(r)$ for $r = 2$. Note that any i-wedge with apex on $\mathcal{C}_i(2)$ (e.g. the dotted wedge) contains load at least 2.

to a side of $\bar{P}$ contains at most one point in Y. It is straightforward to perturb the input to the original problem so that this assumption holds for Y.

Level Curves. We will now define a boundary for an $i \in \{0, 1, \ldots \mu-1\}$ and positive integer r. This boundary has the property that any i-wedge placed on or "inside" the boundary has load at least r with respect to Y, and any i-wedge placed "outside" the boundary has load less than r. That is, the number of points in $W_i(x) \cap Y$ for any x inside the boundary or on the boundary is at least r and is less than r for any x outside the boundary. This boundary is called a *level curve* [2] and extends the definition of *boundary points* [51, 52]. Let $\mathcal{W}_i^j$ be the set of apices of all i-wedges W such that $\text{load}_Y(W) = j$. For each $i = 0, 1, \ldots \mu - 1$, let the level curve $\mathcal{C}_i(r)$ be the boundary of the region $\mathcal{W}_i^{\geq r} = \bigcup_{j \geq r} \mathcal{W}_i^j$ for each $i = 0, 1, \ldots \mu - 1$.

Note that $\mathcal{C}_i(r)$ is a monotone staircase polygonal path with edges that are parallel to the edges of an i-wedge. See Figure 2.2. We have the following observations:

Observation 3. *For any $x \in \mathcal{C}_i(r)$, $r \leq load_Y(W_i(x)) \leq r + 1$.*

Observation 4. *Any i-wedge W such that $load_Y(W) \geq r$ contains an i-wedge whose*

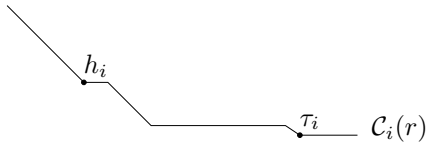


Figure 2.3: Level curve $\mathcal{C}_i(r)$ with h_i and τ_i denoted.

apex belongs to $\mathcal{C}_i(r)$.

Observe that one of the two extreme edges of the level curve $\mathcal{C}_i(r)$ is a semi-infinite ray parallel to edge $p_{i-1}p_i$. Let h_i denote the origin of this ray. We call h_i the *head* of $\mathcal{C}_i(r)$. Note that for all points y on the ray, $W_i(y) \cap Y = W_i(h_i) \cap Y$. The other extreme edge of $\mathcal{C}_i(r)$ is parallel to edge p_ip_{i+1}. Let τ_i denote the origin of this ray. We call τ_i the *tail* of $\mathcal{C}_i(r)$. Note that for all points y on the ray, $W_i(y) \cap Y = W_i(\tau_i) \cap Y$. See Figure 2.3.

2.1.1 Simple Algorithm for One Level Curve

Observation 4 implies that is sufficient to prove Theorem 2 for the i-wedges with apex on $\mathcal{C}_i(k)$, for each $0 \leq i \leq \mu - 1$. In order to do this, we will need a procedure that takes as input one level curve $\mathcal{C}_i(k)$, a positive integer t, and a subset $Q \subseteq Y$. The input to the procedure has the guarantee that for any i-wedge W with apex on $\mathcal{C}_i(k)$, we have $|W \cap Q| \geq 2t$. The goal is to output a partial coloring of the points of Q with colors $\{1, 2, \ldots, t\}$ so that any i-wedge W with apex on $\mathcal{C}_i(k)$ (a) contains a point colored j, for $1 \leq j \leq t$, and (b) contains at most $2t$ colored points.

It is known [2] that such a procedure exists. The reason is that for any $q \in Q$, the set $I(q) = \{u \in \mathcal{C}_i(k) | q \in W_i(u)\}$ of apexes of i-wedges containing q is an

"interval" of $\mathcal{C}_i(k)$. See Figure 2.4 for an illustration. We consider these intervals in an order such that if interval I properly contains interval I', then we consider I before I'. Considering intervals in such an order, we add an interval into our working set if it covers a point of $\mathcal{C}_i(k)$ that is not covered by previous intervals in the working set. Notice that after all intervals have been considered, the working set forms a cover of $\mathcal{C}_i(k)$. Now, we repeatedly throw out intervals from the working set that are redundant – an interval is redundant if throwing it out of the current working set does not affect coverage of $\mathcal{C}_i(k)$.

The final non-redundant working set covers $\mathcal{C}_i(k)$, but also has no more than two intervals covering any point of $\mathcal{C}_i(k)$. We give the color 1 to the points in Q that give rise to the intervals in our working set. We repeat this process $t-1$ more times after removing the colored points from Q. It is easy to verify that the overall procedure, which we call computeCover(i, Q, t), successfully achieves properties (a) and (b). We have the following observation whose second claim easily follows from the manner in which we pick our non-redundant working set.

Observation 5. *The partial cover computed by computeCover(i, Q, t) has the property that any i-wedge with apex on $\mathcal{C}_i(k)$ has at most $2t$ colored points. Furthermore, if q and q' are points in Q such that $q \in W_i(q')$ (that is, $I(q)$ properly contains $I(q')$), then q' is colored only if q is colored.*

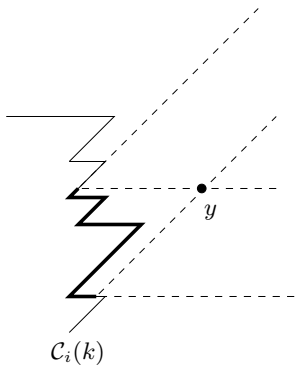


Figure 2.4: An example of an interval $I(y)$ (in bold). Note that the i-wedges with apex on $\mathcal{C}_i(k)$ that contain y are the dotted wedges and all wedges with apex "in between" the apices of the dotted wedges.

2.2 Centrally-Symmetric Polygons

In this section, we give an algorithm which will construct $\Omega(k)$ covers when P is a centrally-symmetric polygon. Such an algorithm was given by Aloupis et al. [2]. The algorithm in this section is slightly different, and some key ideas from the analysis of this algorithm will play a crucial part in the analysis of the algorithm for general convex polygons.

Algorithm 2.1 calls computeCover(i, X_i, t) for each $0 \leq i \leq \mu - 1$. The set X_i in the i-th iteration consists of all the points in $Y \cap \mathcal{W}_i^{\leq k}$ not colored in iterations $0, 1, \ldots, i-1$. At the beginning of the i-th iteration, let L denote, as in the algorithm, the smallest number of uncolored points in a j-wedge with apex on $\mathcal{C}_j(k)$, for $i \leq j \leq \mu - 1$. The parameter t is chosen to be $\frac{L}{64\mu}$. After the call to computeCover$(i, X_i, \frac{L}{64\mu})$, any i-wedge with apex on $\mathcal{C}_i(k)$ contains points colored $1, 2, \ldots, L/64\mu$. This is $\Omega(k)$ colors provided $L \in \Omega(k)$. This is established in the remainder of the section.

Algorithm 2.1

1: $Y' \leftarrow Y$

2: **for** $i \leftarrow 0$ to $\mu - 1$ **do**

3: $L \leftarrow \min\{\text{load}_{Y'}(W_z(x)) : x \in \mathcal{C}_z(k) \text{ and } z = i, i+1, \ldots, \mu - 1\}$

4: $X_i \leftarrow Y' \cap \mathcal{W}_i^{\leq k}$

5: Run computeCover$(i, X_i, \frac{L}{64\mu})$. Let $Y_i \subseteq X_i$ be the points assigned a color during this call.

6: Let Y' denote the uncolored points (i.e. $Y' \leftarrow Y' \setminus Y_i$).

7: **end for**

We will show that L, which equals k before the 0-th iteration, drops by at most a constant factor (i.e. $O(\mu)$) with each iteration. More specifically, we will show that L drops by at most a constant factor during iteration i for all $j > i$.

We use the following terminology for iteration i: for two distinct points q and q', if $W_i(q) \subseteq W_i(q')$, we say that q *dominates* q'. Notice that if q and q' are both uncolored before iteration i, then q' is colored in iteration i only if q is already colored. (This is Observation 5.) For the rest of this chapter, let Y' denote the points that are not colored just before iteration i, let X_i denote set of candidate points that are eligible to be colored in iteration i (as constructed in the algorithm), and let Y_i denote the points that are actually colored in iteration i.

The analysis will rely heavily upon the following observation, which follows directly from Observation 5 and the fact that in Step 5 of the algorithm, we invoke

computeCover(i, X_i, t) with $t = \frac{L}{64\mu}$.

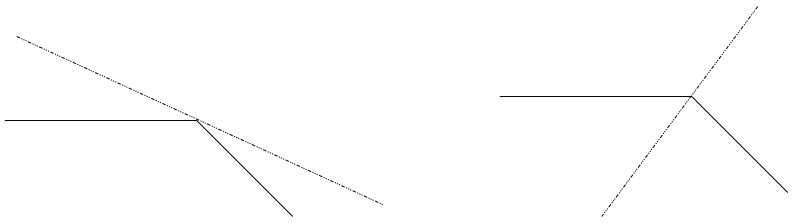
Observation 6. *For any $z \in \mathcal{C}_i(k)$, we have that $load_{Y_i}(W_i(z)) \leq \frac{L}{32\mu}$.*

A line through a vertex of $\bar{P}$ is *tangent* to $\bar{P}$ if it intersects $\bar{P}$ only at the vertex. Similarly, a line through the apex of a wedge is *tangent* to the wedge if it intersects the wedge only at the apex. See Figure 2.5. We will consider the following two cases:

- p_i and p_j are antipodal vertices of $\bar{P}$ – that is, there are parallel lines through p_i and p_j such that both of the lines are tangent to $\bar{P}$.
- p_i and p_j are not antipodal vertices of $\bar{P}$.

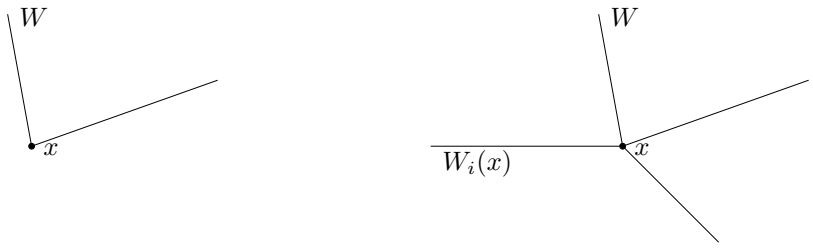
Suppose we have a j-wedge W . If p_i and p_j are not antipodal vertices of $\bar{P}$, then we say that W is *nonantipodal with respect to an i-wedge*. Now suppose that W is a wedge of any type, i.e. not necessarily a type corresponding to a vertex of $\bar{P}$. Suppose the apex of W is the point x, and consider the i-wedge $W_i(x)$. See Figure 2.6. We say that W is *subantipodal with respect to an i-wedge* if (1) every line that is tangent to $W_i(x)$ is also tangent to W, and (2) W and $W_i(x)$ are on "opposite sides" of each of these tangent lines. See Figure 2.7.

We will argue that each wedge W (of type other than i) will have at most a constant factor of its uncolored points assigned a color during iteration i of the algorithm. Lemma 8 handles the case when W is nonantipodal with respect to an i-wedge, and Lemma 9 handles the case when W is subantipodal with respect to an i-wedge. The lemmas are written in this general form as they will be referred to multiple



(a) (b)

Figure 2.5: An illustration for the definition of tangent. (a) This line is tangent to the wedge. (b) This line is not tangent to the wedge.



(a) (b)

Figure 2.6: An illustration of W and $W_i(x)$. (a) The wedge W with apex at x. (b) $W_i(x)$ and W both have their apex at x.

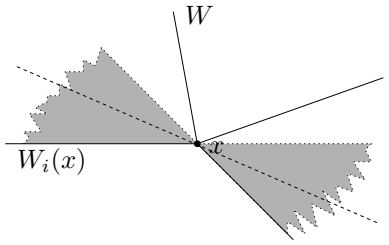


Figure 2.7: All of the lines tangent to $W_i(x)$ must lie in the shaded region (e.g. the dotted line). Every such line is also tangent to W, and the wedges are on opposite sides of every such line. By definition, W is subantipodal with respect to an i-wedge.

times in the paper; however, note that every wedge of a centrally-symmetric convex polygon of type other than i is either nonantipodal or subantipodal with respect to i. Assuming Lemmas 8 and 9 are true, the following theorem easily follows.

Theorem 7. *Let $\bar{P}$ be any centrally-symmetric convex polygon with μ vertices. Let $k \geq 1000\mu \cdot (5\mu)^{\mu}$ be a parameter. For any input $Y \subset \Re^2$, Algorithm 2.1 colors the points in Y with $\Omega(k)$ colors in a way such that for any j-wedge W corresponding to $\bar{P}$ such that $|W \cap Y| \geq k$, W contains a point of each color.*

Proof. In iteration i, we color points in X_i so that each i-wedge contains points of $t = \frac{L}{64\mu}$ different colors, where L is as defined in the algorithm. This is $\Omega(k)$ different colors as long as L (which is k prior to iteration 0 of the algorithm) is $\Omega(k)$ prior to iteration i of the algorithm. Lemma 8 and Lemma 9 ensure this by showing that L falls by at most a constant factor in each iteration of the algorithm for all j-wedges such that $j > i$.

Let W be any wedge of type j such that $j > i$. Note that if $\text{load}_{X_i}(W) < \frac{L}{6}$, then clearly W will contain at least $L - \frac{L}{6}$ uncolored points after iteration i (only points from X_i will be assigned a color). Therefore we only need to handle the case when $\text{load}_{X_i}(W) \geq \frac{L}{6}$. Also note that in the case of centrally-symmetric convex polygons, every j-wedge with $j \neq i$ is either nonantipodal with respect to an i-wedge or is subantipodal with respect to an i-wedge. If W is nonantipodal with respect to an i-wedge then we can invoke Lemma 8. If W is subantipodal with respect to an i-wedge then we can invoke Lemma 9. In both cases, the number of uncolored points in W falls by a factor of at most 5μ in iteration i. Therefore, when we reach iteration

j of the algorithm, W will still contain $\Omega(k)$ points which have not yet received a color. □

The following lemma handles the case for wedges which are nonantipodal with respect to an i-wedge.

Lemma 8. *Suppose that W is a wedge that is nonantipodal with respect to an i-wedge and $load_{X_i}(W) \geq \frac{L}{6}$. Further suppose that at the beginning of iteration i, all i-wedges with apex on $\mathcal{C}_i(k)$ have load at least $\frac{L}{2}$ from points in X_i. After the i-th iteration of the algorithm, W has load at least $\frac{L}{5\mu}$ from points in Y'. (Note that Y' always denotes the uncolored points in the algorithm.)*

Proof. Let x denote the apex of W. The argument is trivial if $W \cap \mathcal{W}_i^{\leq k} = \emptyset$. Suppose that $W \cap \mathcal{W}_i^{\leq k} \neq \emptyset$ and $W \cap \mathcal{C}_i(k) = \emptyset$. For this to be the case, one of the boundaries of W must be parallel with a boundary of an i-wedge. We will focus on the case when W has a boundary parallel to the side $p_i p_{i+1}$ as the other case is symmetric. See Figure 2.8. In this case, we have that $W \cap X_i \subseteq W_i(\tau_i)$ by the definition of the tail τ_i. We thus have $\text{load}_{Y_i}(W) \leq \text{load}_{Y_i}(W_i(\tau_i)) \leq \frac{L}{32\mu}$ where the last inequality comes from Observation 6. Therefore, the load of uncolored points in W after iteration i is at least $\frac{L}{6} - \frac{L}{32\mu} > \frac{L}{5\mu}$.

So let us assume that $W \cap \mathcal{C}_i(k) \neq \emptyset$. Recall that since W is nonantipodal with respect to an i-wedge, it is of type j where j corresponds to a vertex p_j of $\bar{P}$. There are two cases to cases to consider – in the first, we encounter p_j after p_i and before the vertices antipodal to p_i when walking counter-clockwise around $\bar{P}$, and in

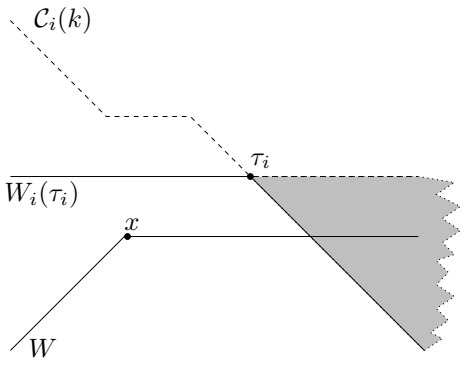


Figure 2.8: Illustration for Lemma 8. Note that there cannot be a point in the shaded region by definition of the tail τ_i.

the second, we encounter p_j after the vertices antipodal to p_i and before p_i. We will focus on the first case, since the other is symmetric. Let z be the intersection point of the boundary of W and $\mathcal{C}_i(k)$ (to be precise, let z be the last point on $W \cap \mathcal{C}_i(k)$ as one walks "clockwise" around the boundary of W). If W does not contain in its interior the tail τ_i of the level curve $\mathcal{C}_i(k)$, then $W \cap Y_i \subseteq W_i(z) \cap Y_i$, and so $\text{load}_{Y_i}(W) \leq \text{load}_{Y_i}(W_i(z)) \leq \frac{L}{32\mu}$. It follows that the load of uncolored points in W after iteration i is at least

$$\frac{L}{6} - \frac{L}{32\mu} > \frac{L}{5\mu}.$$

Let us therefore assume that W does contain in its interior the tail τ_i of $\mathcal{C}_i(k)$. See Figure 2.9. Let a denote the point where the boundaries of the wedges $W_i(z)$ and $W_i(\tau_i)$ intersect. If $\text{load}_{X_i}(W_i(a)) \geq \frac{L}{8\mu}$, then since $\text{load}_{Y_i}(W_i(a)) \leq \text{load}_{Y_i}(W_i(\tau_i)) \leq \frac{L}{32\mu}$, there are uncolored points in $W_i(a)$ after iteration i. Since any point in $W_i(a)$ dominates points in $W \cap Y_i$ that are not contained in $W_i(z) \cup W_i(\tau_i)$, we conclude

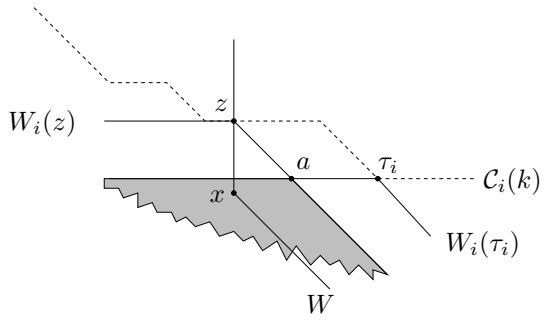


Figure 2.9: Illustration for the nonantipodal case.

that $W \cap Y_i \subseteq (W_i(z) \cup W_i(\tau_i)) \cap Y_i$. Thus,

$$\text{load}_{Y_i}(W) \leq \text{load}_{Y_i}(W_i(z)) + \text{load}_{Y_i}(W_i(\tau_i)) \leq \frac{L}{16\mu}.$$

Therefore there must be at least $\frac{L}{6} - \frac{L}{16\mu} > \frac{L}{5\mu}$ uncolored points left in W.

Let us therefore consider the case where $\text{load}_{X_i}(W_i(a)) < \frac{L}{8\mu}$. This means that $\text{load}_{X_i}(W_i(\tau_i) \backslash W_i(a)) > \frac{L}{2} - \frac{L}{8\mu} > \frac{L}{3}$. Again, $\text{load}_{Y_i}(W_i(\tau_i) \backslash W_i(a)) \leq \text{load}_{Y_i}(W_i(\tau_i)) \leq \frac{L}{32\mu}$, and this means the load of the points in $W_i(\tau_i) \setminus W_i(a)$ that are uncolored after iteration i is at least $\frac{L}{3} - \frac{L}{32\mu} > \frac{L}{5\mu}$. But $W_i(\tau_i) \setminus W_i(a) \subseteq W$, and this means that the load of the uncolored points in W after iteration i is at least $\frac{L}{5\mu}$. □

The following lemma handles wedges which are subantipodal with respect to an i-wedge.

Lemma 9. *Suppose that W is a wedge that is subantipodal with respect to an i-wedge and $load_{X_i}(W) \geq \frac{L}{6}$. Further suppose that at the beginning of iteration i, all i-wedges with apex on $\mathcal{C}_i(k)$ have load at least $\frac{L}{2}$ from points in X_i. After the i-th iteration of the algorithm, W has load at least $\frac{L}{5\mu}$ from points in Y'. (Note that Y' always denotes the uncolored points in the algorithm.)*

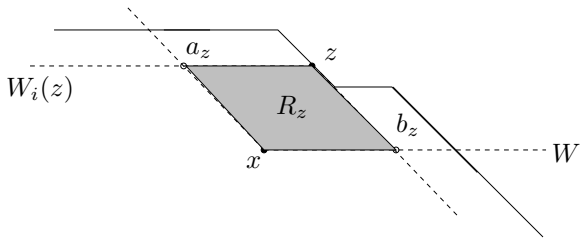


Figure 2.10: Illustration for Case 2(a): the region R_z.

Proof. Let x denote the apex of W. Again, the argument is trivial if $W \cap \mathcal{W}_i^{\leq k} = \emptyset$. So let us assume that $W \cap \mathcal{W}_i^{\leq k} \neq \emptyset$. We first consider the case when both boundaries of W intersect $\mathcal{C}_i(k)$. Consider any point $z \in W \cap \mathcal{C}_i(k)$. Let a_z denote the "leftmost" point where the boundaries of W and $W_i(z)$ intersect, and let b_z denote the "rightmost" point where the boundaries of W and $W_i(z)$ intersect. Let R_z be the quadrilateral with vertices a_z, x, b_z, and z. That is, $R_z = W \cap W_i(z)$ [1]. Suppose that $\text{load}_{X_i}(R_z) \geq \frac{L}{5\mu} + \frac{L}{32\mu}$. By Observation 6, $\text{load}_{Y_i}(W_i(z)) \leq \frac{L}{32\mu}$, and since all points in R_z are in $W_i(z)$, R_z contains (uncolored) load at least $\frac{L}{5\mu}$ after iteration i. Since $R_z \subseteq W$, W contains (uncolored) load at least $\frac{L}{5\mu}$ after iteration i, and we are done. See Figure 2.10 for an illustration.

So we now assume that $\text{load}_{X_i}(R_z) \leq \frac{L}{5\mu} + \frac{L}{32\mu}$ for each $z \in W \cap \mathcal{C}_i(k)$. Since $\text{load}_{X_i}(W_i(z)) \geq \frac{L}{2}$, we must have $\text{load}_{X_i}(W_i(a_z) \cup W_i(b_z)) \geq \frac{L}{2} - (\frac{L}{5\mu} + \frac{L}{32\mu}) > \frac{L}{8}$. Let z_1 be the "leftmost" point on $\mathcal{C}_i(k) \cap W$, and let z_2 be the "rightmost" point on $\mathcal{C}_i(k) \cap W$. Notice that a_{z_1} is just z_1 itself, and so $\text{load}_{X_i}(W_i(a_{z_1})) \geq \frac{L}{2}$. Similarly, $\text{load}_{X_i}(W_i(b_{z_2})) \geq \frac{L}{2}$. Let z' be the last point on $\mathcal{C}_i(k)$, while walking from z_1 to z_2,

[1] To be more precise, a_z is encountered after x and before z while traversing the boundary of R_z clockwise. Then b_z is encountered after z and before x.

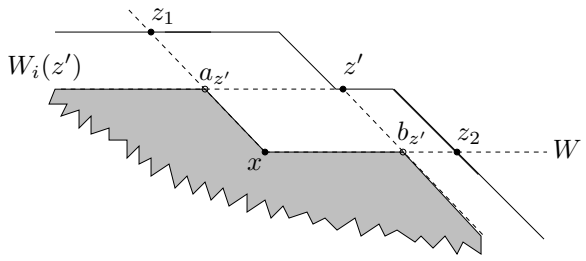


Figure 2.11: Illustration for Case 2(a): the constructed point z'.

such that $\text{load}_{X_i}(W_i(a_{z'})) \geq \frac{L}{16}$. Thus

$$\text{load}_{X_i}(W_i(b_{z'})) \geq \text{load}_{X_i}(W_i(a_{z'}) \cup W_i(b_{z'})) - \frac{L}{16} \geq \frac{L}{8} - \frac{L}{16} = \frac{L}{16}.$$

See Figure 2.11 for an illustration.

Now consider any point $z'' \in W \backslash W_i(z')$. It must be that $W_i(z'')$ either contains $W_i(a_{z'})$ or contains $W_i(b_{z'})$ which both have load in X_i of at least $\frac{L}{16}$. Suppose that $W_i(z'')$ contains $W_i(a_{z'})$; the other case is similar. The points in $W_i(a_{z'})$ dominate z'' and we will not color z'' in iteration i until we have colored all points in $W_i(a_{z'}) \cap X_i$. But since $\text{load}_{Y_i}(W_i(a_{z'})) \leq \frac{L}{32\mu} < \frac{L}{16} \leq \text{load}_{X_i}(W_i(a_{z'}))$, this means we will not color z''.

It follows that $W \cap Y_i \subseteq W_i(z') \cap Y_i$, and thus $\text{load}_{Y_i}(W) \leq \text{load}_{Y_i}(W_i(z')) \leq \frac{L}{16\mu}$. And so the load of uncolored points in W after iteration i is at least $\frac{L}{2} - \frac{L}{32\mu} \geq \frac{L}{5\mu}$.

We now consider the case when one or both boundaries of W do not intersect with $\mathcal{C}_i(k)$. See Figure 2.12. We will show that we can find a wedge W' such that both boundaries of W' intersect $\mathcal{C}_i(k)$, $W' \cap Y = W \cap Y$, and W' is subantipodal with respect to an i-wedge. Given that W' exists, we can use the previous arguments to show that W' contains $\Omega(k)$ uncolored points after iteration i. It then follows that

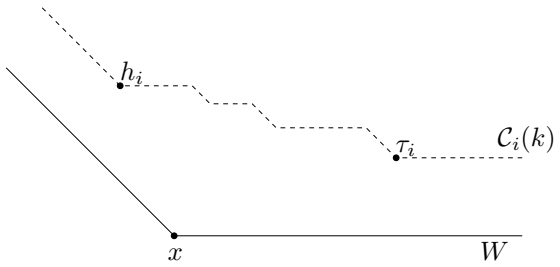


Figure 2.12: Illustration of the case when $W \cap \mathcal{W}_i^{\leq k} \neq \emptyset$ and the boundaries of W do not intersect $\mathcal{C}_i(k)$.

W contains $\Omega(k)$ uncolored points after iteration i because $W' \cap Y = W \cap Y$.

We will now describe how to find the wedge W'. We begin by placing a wedge identical to W "just behind" W so that the new wedge contains exactly the same points of Y that W contains. See Figure 2.13. We then "bend in" the boundaries of the wedge just enough so that the wedge still contains the same points as W and the boundaries are no longer parallel with the edges of $\mathcal{C}_i(k)$. This is our wedge W'. See Figure 2.14. Both boundaries now intersect $\mathcal{C}_i(k)$ and W' is subantipodal with respect to an i-wedge. By our previous analysis, W' contains at least $\frac{L}{5\mu}$ uncolored points after iteration i. Since W and W' contain the same points in Y, it follows that W contains $\frac{L}{5\mu}$ uncolored points after iteration i.

□

The previous two lemmas complete the proof of Theorem 7, which then in turn completes the proof of Theorem 2 for the wedges of a centrally-symmetric convex polygon.

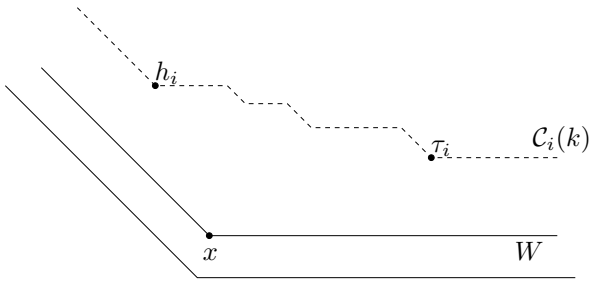


Figure 2.13: The first step in constructing W'.

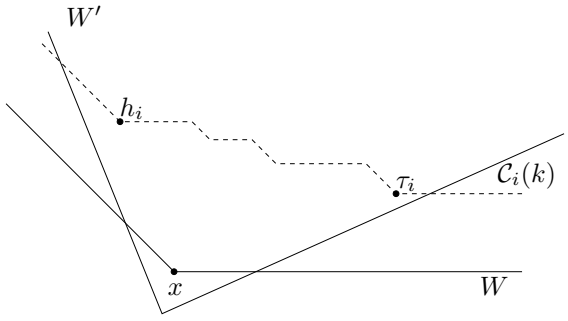


Figure 2.14: The second step in constructing W'.

CHAPTER 3
DECOMPOSING COVERINGS: CONVEX POLYGONS

In Section 3.1, we show that the algorithm for centrally-symmetric polygons does not work in the case of triangles, a polygon which is not centrally-symmetric. We then show how the algorithm can be modified to handle all convex polygons.

3.1 General Convex Polygons

In this section, we will show that Algorithm 2.1 does not work when $\bar{P}$ is not a centrally-symmetric polygon. More specifically, we will show that the algorithm does not work when $\bar{P}$ is a triangle. We will then give an algorithm that does indeed work for general convex polygons, thus proving Theorem 2.

A bad example for Algorithm 2.1. The reason that Algorithm 2.1 works for centrally-symmetric polygons but does not work for triangles is because of the ways that wedges from a triangle can intersect. Using the definitions from the previous chapter, a triangle could have a wedge that is not nonantipodal or subantipodal with respect to an i-wedge. Because of this, it could be possible that when running computeCover() for $\mathcal{C}_i(k)$ that all of the points inside of some j-wedge could be eligible to be colored in iteration i, and it could be possible that they are all assigned the same color.

For an example of this, see Figures 3.1 and 3.2. Suppose we are working with the triangle in Figure 3.1. In Figure 3.2, each of the small y_i dots correspond

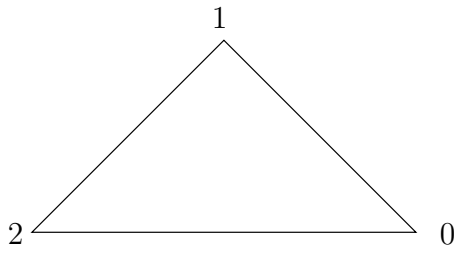


Figure 3.1: The triangle used in Figure 3.2.

to a single point in Y, and the larger $\mathcal{Y}_i$ dots correspond with $k-1$ points in Y. The level curve $\mathcal{C}_0(k)$ is drawn with dotted lines, and a 2-wedge $W_2(x)$ is drawn with solid lines. $W_2(x)$ contains only the points y_i for $i = 1, 2, \ldots, k$. We want to run computeCover() for $\mathcal{C}_0(k)$ up through time $\Omega(k)$ while leaving $\Omega(k)$ points from $W_2(x)$ uncolored. However computeCover() could pick all of the y_i points in its first iteration and assign all of these points the same color. $W_2(x)$ contains at least k points from Y, but all of the points it contains have the same color. Thus, the algorithm fails.

The Algorithm for General Convex Polygons. Recall that computeCover() takes as input a set of points $X_i \subseteq Y$. Only points from X_i are eligible to be assigned a color during iteration i. The way we modify the algorithm to work for all convex polygons is by more carefully choosing our set X_i. For centrally-symmetric polygons, it suffices to let X_i consist of all uncolored points in $\mathcal{W}_i^{\leq k}$. In the case of triangles, we want to choose a subset of the uncolored points in $\mathcal{W}_i^{\leq k}$. Intuitively, we want to only color the points which are the "furthest away" from $\mathcal{C}_i(k)$. By doing this, we can show that if we assign a color to a point within some j-wedge, then there must

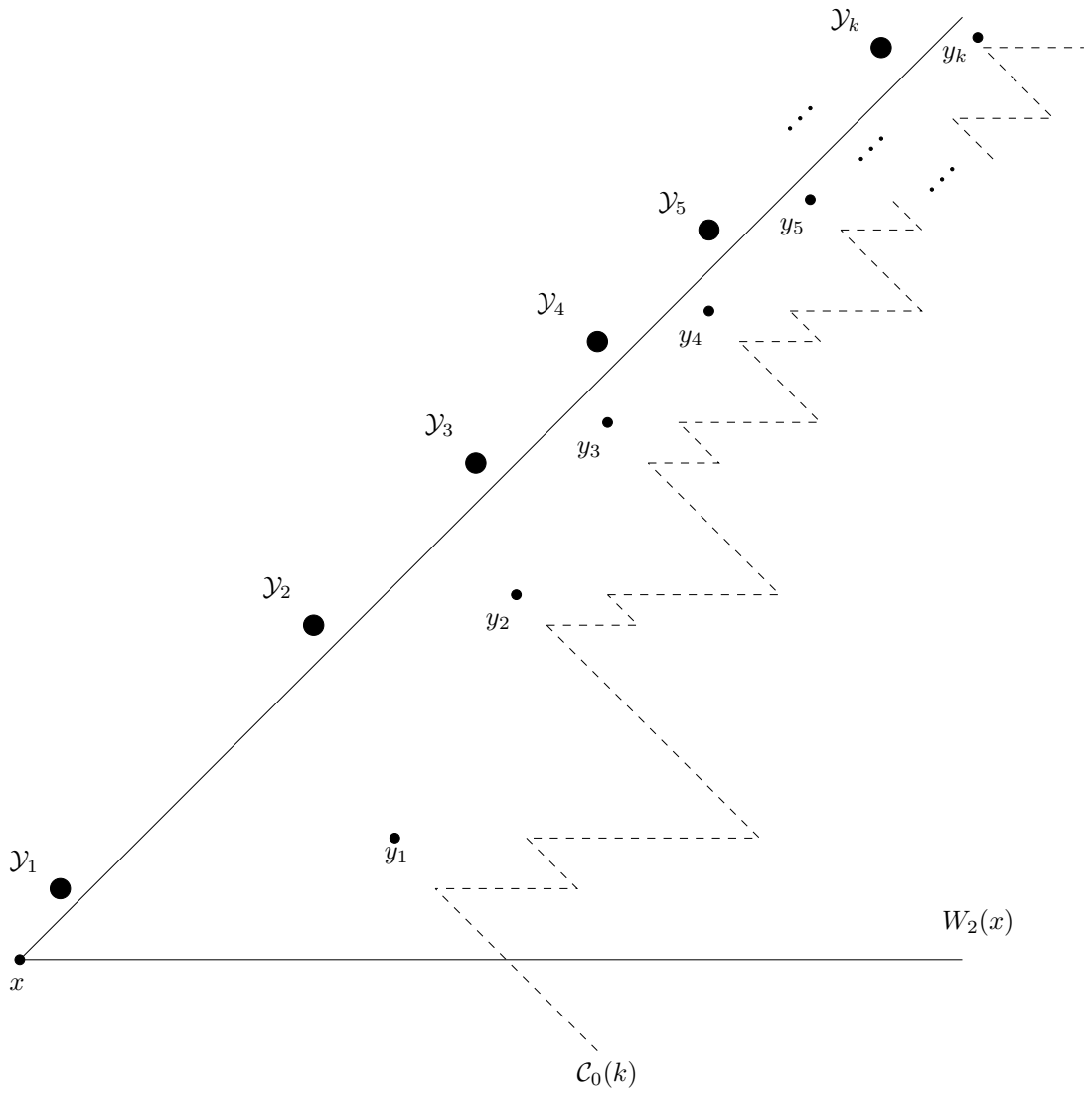


Figure 3.2: Algorithm 2.1 would assign all of the points in $W_2(x)$ the same color when coloring points for $\mathcal{C}_0(k)$.

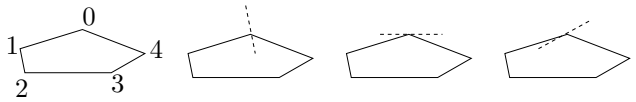


Figure 3.3: An example of a set A_i. In this example, $A_0 = \{2\}$ since only the side of $\bar{P}$ parallel with p_2p_3 has the qualifying property.

be "a lot" of points remaining within the j-wedge.

Now we will introduce a notion which will be crucial in determining which points are "far away" from $\mathcal{C}_i(k)$. Consider the natural linear ordering of the lines parallel to side p_jp_{j+1} of $\bar{P}$ with the line through vertices p_j and p_{j+1} being smaller than the line through any of the other vertices of $\bar{P}$. For $x, y \in \Re^2$, we define the partial order $<_j$ such that $x <_j y$ if the p_jp_{j+1} parallel line through x is less than the p_jp_{j+1} parallel line through y. A similar notion was used in [60].

For a vertex p_i, let A_i denote the set of all indices j such that the intersection of $\bar{P}$ with the line parallel to p_jp_{j+1} and through p_i is only the point p_i. See Figure 3.3 for an example.

The algorithm is Algorithm 3.1. Note that the only difference between this algorithm and Algorithm 2.1 is in how X_i is chosen.

We will first analyze the algorithm in the case when $\bar{P}$ is a triangle. We will then show that we can use the ideas from the triangle case as well as the centerally-symmetric case to analyze the algorithm when $\bar{P}$ is an arbitrary convex polygon.

Analysis for Triangles. First, we will elaborate on how Algorithm 3.1 chooses the set X_i for a triangle. Recall the definition of the set A_i. For triangles, $A_i = \{i + 1\}$

Algorithm 3.1

1: $Y' \leftarrow Y$

2: **for** each $i \in \{0, 1, 2, \ldots, \mu - 1\}$ **do**

3: $L \leftarrow \min\{\text{load}_{Y'}(W_z(x)) : x \in \mathcal{C}_z(k) \text{ and } z = i, i+1, \ldots, \mu - 1\}$

4: **for** each $c \in \mathcal{C}_i(k)$ **do**

5: **for** each $j \in A_i$ **do**

6: Let $X^j(c)$ be the first $\frac{L}{2\mu}$ points in $W_i(c) \cap Y'$ in decreasing order with respect to the ordering $<_j$.

7: **end for**

8: $X(c) \leftarrow \{W_i(c) \cap Y'\} \setminus \bigcup_{j \in A_i} X^j(c)$

9: **end for**

10: $X_i \leftarrow \bigcup_{c \in \mathcal{C}_i(k)} X(c)$

11: Run computeCover(i, X_i, $\frac{L}{64\mu}$). Let $Y_i \subseteq X_i$ be the points assigned a color during this call.

12: Let Y' denote the uncolored points (i.e. $Y' \leftarrow Y' \setminus Y_i$).

13: **end for**

since only the line parallel with $p_{i+1}p_{i+2}$ has the qualifying property. Note that this line is parallel with the side of the triangle opposite of p_i.

So when we get to the loop in Step 5 of the algorithm in iteration i, we will only compute the set $X^{i+1}(c)$ in Step 6. This set will consist of the first $\frac{L}{6}$ points in $W_i(c) \cap Y'$ in decreasing order with respect to the ordering $<_{i+1}$. The sorting is

thus done with respect to the side of the triangle opposite of p_i. Thus in Step 7, the set $X(c)$ is the last $\frac{5L}{6}$ points in $W_i(c) \cap Y'$ in decreasing order with respect to the ordering $<_{i+1}$. Then finally in Step 8, X_i is simply the union of $X(c)$ for all $c \in \mathcal{C}_i(k)$.

The analysis for this algorithm has the same flavor as the analysis of Algorithm 2.1. In iteration i of the algorithm, we will assign colors to some of the points in X_i so that any i-wedge with apex on $\mathcal{C}_i(k)$ contains points colored with $\Omega(L)$ colors where L is as defined in the algorithm. This is $\Omega(k)$ different colors as long as L, which is k before the 0th iteration, is $\Omega(k)$ before the ith iteration. This is ensured in the remainder of this section.

The analysis will again rely upon Observation 6 and the following observation.

Observation 10. *For any $z \in \mathcal{C}_i(k)$, $load_{X_i}(W_i(z)) \geq \frac{L}{2}$.*

To see Observation 10, note that for each $c \in \mathcal{C}_i(k)$, we have:

$$\text{load}_{X_i}(W_i(c)) \geq \text{load}_{X(c)}(W_i(c)) \geq \frac{L}{2}$$

Suppose we have a wedge W of any type other than i. We say that W is *triangular with respect to an i-wedge* if there exists a triangle T with vertices t_1 and t_2 such that the boundaries of T adjacent to t_1 are parallel with the boundaries of W and the boundaries of T adjacent to t_2 are parallel with the boundaries of an i-wedge. Note that for this to be possible, W must have a boundary that is parallel with a boundary of an i-wedge. Also note that by the definition, a wedge of a triangle is triangular with respect to every other type of wedge of the same triangle. See Figures 3.4 and 3.5.

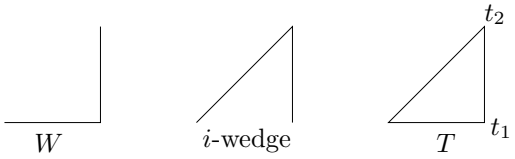


Figure 3.4: The wedge W is triangular with respect to an i-wedge because there is a triangle T with the required property.

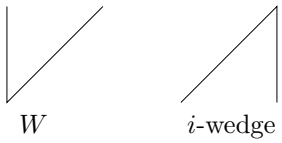


Figure 3.5: The wedge W is not triangular with respect to an i-wedge because the required triangle does not exist.

We will argue that each wedge W (of type other than i) will have at most a constant factor of its uncolored points assigned a color during iteration i of the algorithm. Again, we write the lemma in a general form as we will use it again in a later section.

Lemma 11. *Suppose that W is a wedge that is triangular with respect to an i-wedge and $load_{X_i}(W) \geq \frac{L}{6}$. Further suppose that at the beginning of iteration i, all i-wedges with apex on $\mathcal{C}_i(k)$ have load at least $L \geq \frac{L}{2}$ from points in X_i. After the i-th iteration of the algorithm, W has load at least $\frac{L}{5\mu}$ from points in Y'. (Note that Y' always denotes the uncolored points in the algorithm.)*

Proof. Let x denote the apex of W. If $W \cap \mathcal{W}_i^{\leq k} = \emptyset$, then the lemma trivially holds. So for now on, we will assume $W \cap \mathcal{W}_i^{\leq k} \neq \emptyset$. First suppose that $W \cap \mathcal{C}_i(k) = \emptyset$. Note

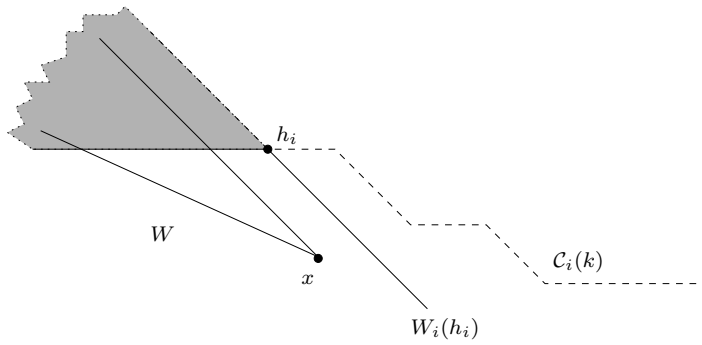


Figure 3.6: Illustration for Lemma 11. Note that there cannot be any points in the shaded region due to the definition of the head h_i.

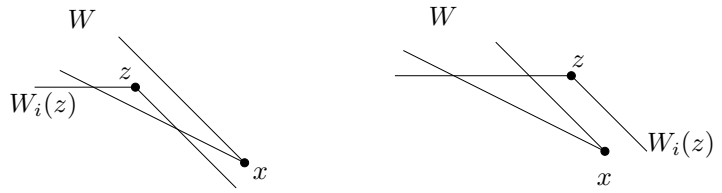
that one of the boundaries of W is parallel with one of the boundaries of an i-wedge. We will assume that W has a boundary parallel with the side $p_{i-1}p_i$. The other case is symmetric. See Figure 3.6. In this case, we have that $W \cap X_i \subseteq W_i(h_i) \cap X_i$ by the definition of the head h_i. Since $W \cap X_i \subseteq W_i(h_i) \cap X_i$, we have that $\text{load}_{Y_i}(W) \leq \frac{L}{32\mu}$. Therefore there is (uncolored) load at least $\frac{L}{6} - \frac{L}{32\mu} > \frac{L}{5\mu}$ after iteration i.

Now we will assume that $W \cap \mathcal{C}_i(k) \neq \emptyset$. Let $z \in \mathcal{C}_i(k)$ be a point such that $W_i(z) \cap W \neq \emptyset$. Since we know that W has a boundary that is parallel with one of the boundaries of $W_i(z)$, we know that there are only two types of intersections between these two wedges:

1. $z \in W$, one boundary of W intersects both boundaries of $W_i(z)$, and the other boundary of W does not intersect with $W_i(z)$.

2. $x \in W_i(z)$, one boundary of $W_i(z)$ intersects both boundaries of W, and the other boundary of $W_i(z)$ does not intersect with W.

See Figure 3.7 for an illustration.



(a) (b)

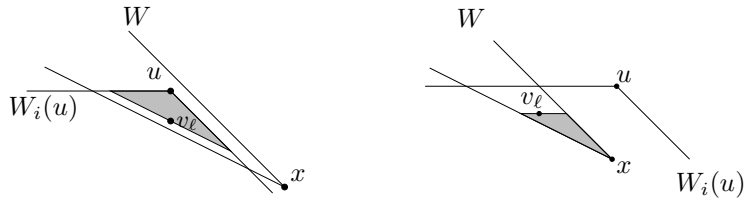
Figure 3.7: An illustration for the triangular case. (a) A type 1 intersection. (b) A type 2 intersection.

Let $\{v_1, v_2, v_3, \ldots\}$ denote the points in $X_i \cap W$ in decreasing order according to the partial order $<_i$. Let $\ell = \max\{t | v_t \in Y_i\}$. If $\ell \leq \frac{L}{10}$ then $\text{load}_{Y'}(W) \geq \frac{L}{6} - \frac{L}{10} \geq \frac{L}{5\mu}$ after iteration i, so assume that $\ell > \frac{L}{10}$. For $t \in \{0, 1, \ldots, \mu - 1\}$ and for a point $s \in \Re^2$, let $H_t(s)$ denote the halfplane consisting of all points y such that $y \leq_t s$.

Since $v_\ell \in X_i$ there is a $u \in \mathcal{C}_i(k)$ so that $v_\ell \in X(u)$ in iteration i of the algorithm. Suppose that the intersection between $W_i(u)$ and W is a type 1 intersection. Let $T_{v_\ell} = W_i(u) \setminus H_{i+1}(v_\ell)$. Since $v_\ell \in X(u)$, we know that v_ℓ is one of the last L points in $W_i(u) \cap Y'$ with respect to the ordering $<_{i+1}$. (See Algorithm 3.1 for the notation.) Because we chose the points for $X(u)$ with respect to this ordering, we know that the first $\frac{L}{2\mu}$ points with respect to the ordering $<_{i+1}$ from $W_i(u) \cap Y'$ must be in T_{v_ℓ}. Since, $\text{load}_{Y_i}(W_i(u)) \leq \frac{L}{32\mu}$, there must be at least $\frac{L}{2\mu} - \frac{L}{32\mu} > \frac{L}{5\mu}$ uncolored points left in T_{v_ℓ} after iteration i. Since we are dealing with a type 1 intersection, $T_{v_\ell} \subset W$, and thus W will contain at least $\frac{L}{5\mu}$ uncolored points after iteration i and the lemma holds. See Figure 3.8(a) for an illustration.

Now suppose that the intersection between $W_i(u)$ and W is a type 2 intersection. Consider the region $T'_{v_\ell} = W \setminus H_i(v_\ell)$. Since we are assuming $\ell > \frac{L}{10}$ and since

$v_\ell \in W_i(u)$, it must be that $\text{load}_{Y'}(T'_{v_\ell}) \geq \frac{L}{10}$. Since we are dealing with a type 2 intersection, it must be that $T'_{v_\ell} \subset W_i(u)$. Since $\text{load}_{Y_i}(W_i(u)) \leq \frac{L}{16\mu}$, we have that $\text{load}_{Y_i}(T'_{v_\ell}) \leq \frac{L}{16\mu}$ and thus there will be at least $\frac{L}{10} - \frac{L}{16\mu} > \frac{L}{5\mu}$ uncolored points left in T'_{v_ℓ} after iteration i. Since $T'_{v_\ell} \subseteq W$, there must be (uncolored) load at least $\frac{L}{5\mu}$ in W after iteration i. See Figure 3.8(b) for an illustration. □



(a) (b)

Figure 3.8: An example of the key triangular regions. (a) An illustration of T_{v_ℓ}. (b) An illustration of T'_{v_ℓ}.

At this point, we could prove a theorem similar to Theorem 7 to show that Theorem 2 holds for the wedges of a triangle.

Analysis for Convex Polygons. In this section, we will analyze Algorithm 3.1 in the case when $\bar{P}$ is an arbitrary convex polygon, thus proving Theorem 2. We will show that any j-wedge that contains $\Omega(k)$ uncolored points prior to coloring points for $\mathcal{C}_i(k)$ will still contain $\Omega(k)$ uncolored points after doing the coloring. We will use Lemma 8, Lemma 9, and Lemma 11 in the analysis. If p_i and p_j are not antipodal vertices of $\bar{P}$, then we can use the analysis of Lemma 8. If p_i and p_j are antipodal

vertices of $\bar{P}$, then we show that we can partition the j-wedge into at most 3 wedges. One of these wedges can be analyzed with Lemma 9 (antipodal wedges of centrally symmetric polygons), and the other two wedges (if they exist) can be analyzed with the same arguments from Lemma 11 (the triangle case).

Lemma 12. *Suppose at the beginning of iteration i, all j-wedges with apex on $\mathcal{C}_j(k)$ have load at least L from points in Y' for $j \geq i$, where L is larger than some absolute constant. (Note that Y' always denotes the uncolored points in the algorithm.) After the i-th iteration of the algorithm, any j-wedge $W_j(x)$, for $j > i$, and with apex x on $\mathcal{C}_j(k)$ has load at least $\frac{L}{5\mu}$ from points in Y'.*

Proof. There are two main cases to consider:

- p_i and p_j are antipodal vertices of $\bar{P}$ – that is, there are parallel lines through p_i and p_j such that both of the lines are tangent to $\bar{P}$.
- p_i and p_j are not antipodal vertices of $\bar{P}$.

Case 1: p_i and p_j are not antipodal vertices of $\bar{P}$.

In this case, we can invoke Lemma 8 for each j-wedge $W_j(x)$ with apex $x \in \mathcal{C}_j(k)$ such that $\text{load}_{X_i}(W_j(x)) \geq \frac{L}{6}$ (Lemma 12 trivially holds for $W_j(x)$ if $\text{load}_{X_i}(W_j(x)) < \frac{L}{6}$). We simply use the arguments from Lemma 8 with $W := W_j(x)$.

Case 2: p_i and p_j are antipodal vertices of $\bar{P}$.

Since $\text{load}_{Y'}(W_j(x)) \geq L$, if $\text{load}_{X_i}(W_j(x)) \leq \frac{L}{2}$ then $W_j(x)$ will clearly have load at least $\frac{L}{5\mu}$ after iteration i. So assume that $\text{load}_{X_i}(W_j(x)) > \frac{L}{2}$.

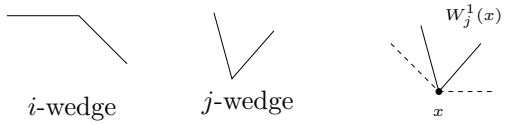
Consider the line parallel with $p_{i-1}p_i$ through x and the line parallel with p_ip_{i+1} through x. Note these lines are parallel with the boundaries of an i-wedge. Let $H_t(x)$ denote the halfplane consisting of all points y such that $y \leq_t x$. Let $W_j^1(x) = H_{i-1}(x) \cap H_i(x) \cap W_j(x)$. Let $W_j^2(x) = (H_i(x) \cap W_j(x)) \setminus W_j^1(x)$. Let $W_j^3(x) = (H_{i-1}(x) \cap W_j(x)) \setminus W_j^1(x)$. See Figure 3.9 and Figure 3.10 for an illustration. Note that $W_j^1(x), W_j^2(x)$, and $W_j^3(x)$ form a partition of $W_j(x)$. Also note that $W_j^1(x)$ cannot be empty but $W_j^2(x)$ or $W_j^3(x)$ could be empty. Since these three sets form a partition of $W_j(x)$ and $\text{load}_{X_i}(W_j(x)) > \frac{L}{2}$, it must be that one of the three sets has load at least $\frac{L}{6}$ from X_i.

(a) (b)

Figure 3.9: An illustration for the antipodal case. If we are working with the corresponding i-wedge and j-wedge (part (a)), then we obtain the corresponding $W_j^1(x)$, $W_j^2(x)$, and $W_j^3(x)$ (part (b)).

Note that $W_j^1(x)$ is subantipodal with respect to an i-wedge. We thus handle the case when $\text{load}_{X_i}(W_j^1(x)) \geq \frac{L}{6}$ by invoking Lemma 9 with $W := W_j^1(x)$.

Note that $W_j^2(x)$ and $W_j^3(x)$ (assuming they are nonempty) are triangular with respect to an i-wedge. Thus we can handle the cases when $\text{load}_{X_i}(W_j^2(x)) \geq \frac{L}{6}$ and $\text{load}_{X_i}(W_j^3(x)) \geq \frac{L}{6}$ with Lemma 11 with $W := W_j^2(x)$ or $W := W_j^3(x)$, completing the proof. □



(a) (b)

Figure 3.10: An illustration for the antipodal case. If we are working with the corresponding i-wedge and j-wedge (part (a)), then $W_j^1(x) = W_j(x)$ and $W_j^2(x) = W_j^3(x) = \emptyset$ (part (b)).

We will now show that Theorem 2 holds for the wedges of any convex polygon.

Theorem 13. *Let $\bar{P}$ be any convex polygon with μ vertices. Let $k \geq 1000\mu \cdot (5\mu)^\mu$ be a parameter. For any input $Y \subset \Re^2$, Algorithm 3.1 colors the points in Y with $\Omega(k)$ colors in a way such that for any j-wedge W corresponding to $\bar{P}$ such that $|W \cap Y| \geq k$, W contains a point of each color.*

Proof. Algorithm 3.1 calls computeCover(i, X_i, t) for each $0 \leq i \leq \mu - 1$. The set X_i in the i-th iteration is an appropriately chosen subset of the points in Y not colored in iterations $0, 1, \ldots, i-1$. At the beginning of the i-th iteration, let L denote, as in the algorithm, the smallest number of uncolored points in a j-wedge with apex on $\mathcal{C}_j(k)$, for $i \leq j \leq \mu - 1$. The parameter t is chosen to be $\frac{L}{64\mu}$, and we have $\min_{c \in \mathcal{C}_i(k)} |W_i(c) \cap X_i| \geq \frac{L}{2}$ (due to the manner in which X_i is chosen in the algorithm, see Observation 10). After the call to computeCover$(i, X_i, \frac{L}{64\mu})$, any i-wedge with apex on $\mathcal{C}_i(k)$ contains points colored $1, 2, \ldots, L/64\mu$. Thus, the algorithm produces a coloring as required in Theorem 2, provided $L \in \Omega(k)$. This is established by Lemma 12. It states that L, which equals k before the 0-th iteration, drops by a factor of at most 5μ with each iteration. □

This completes the proof of Theorem 1.

CHAPTER 4
RESTRICTED STRIP COVERING

In this chapter, we consider the Restricted Strip Cover problem (RSC). Here, we are given a universe U of m points which can be thought of as points on a horizontal line. We are also given a set $\mathbf{S}$ of n sensors, each of which covers some interval of the line and has a duration. The RSC problem is the problem of assigning a start time to each of the sensors so that all of the points in U are "covered" by an active sensor for as long as possible. We give a simple, greedy algorithm for RSC and show that the algorithm is a 5-approximation.

4.1 Restricted Strip Covering

In the Restricted Strip Covering problem, the universe U is an ordered set of m *points* or *coordinates*, $\{1, 2, \ldots, m\}$. We may assume without loss of generality that $m \leq 2n$. For each $s \in \mathbf{S}$, $R(s)$ is a range $\{\ell(s), \ell(s)+1, \ldots, r(s)\}$ where $1 \leq \ell(s) \leq r(s) \leq m$ and $\ell(s), r(s) \in U$. We can think of the coordinates as lying on a horizontal line, and we use the natural notion of some coordinate being to the left or right of another coordinate.

4.1.1 The Algorithm

We now describe our algorithm which takes an instance of RSC consisting of sensors $\mathbf{S}$ and universe U and returns a schedule S of the sensors. We will later show that the schedule produced by the algorithm has duration at least $L/5$. The

algorithm starts with the empty schedule where no sensor is assigned, and assigns a start time to one sensor in each iteration. We will also denote the current schedule of the algorithm at any stage of its execution by S.

With respect to a schedule S, we say the sensor s *dominates coordinate* x *to the right* if s extends as far to the right as possible (maximizes $r(s)$) among all sensors that have not been assigned and are live at x. In the event of a tie, we take the sensor that extends as far to the left as possible. Further ties are broken arbitrarily. The sensor that *dominates coordinate* x *to the left* is defined similarly. For ease of description, define $\mathrm{M}(S, 0) = \mathrm{M}(S, m+1) = \infty$ (recall that $\mathrm{M}(S, i)$ is the duration of schedule S at coordinate i).

Consider Algorithm 4.1. Let us denote by t_f the duration of the final schedule produced by the algorithm. At termination, there is a point $x \in U$ so that $\mathrm{M}(S, x) = t_f$ and there is no unassigned sensor that is live at x.

Running Time. We will iterate through the while loop at most n times because we schedule a sensor in each iteration of the while loop. Each iteration of the while is readily implemented in $O(m+n) = O(n)$ time. Thus the algorithm runs in $O(n^2)$ time. It may be possible to improve the running time by using data structures such as segment trees.

4.1.2 Approximation Ratio

For an instance **S** of RSC, let OPT denote the duration of the optimal solution for **S**. In this section we will prove the following theorem.

Algorithm 4.1

1: $t \leftarrow 0$

2: $S \leftarrow \emptyset$

3: **while** TRUE **do**

4: $t \leftarrow \mathrm{M}(S) + 1$

5: Let i be the first uncovered coordinate at time t and let j be max $\{x \in U \mid [i, x]$ is completely uncovered at time $t\}$.

6: Let s' be the sensor that dominates i to the right. If s' does not exist, go to step 9.

7: If s' is not live at j, $t(s') \leftarrow t$ and $S \leftarrow S \cup \{s'\}$.

8: If s' is live at j, let s'' be the sensor that dominates j to the left. If $\mathrm{M}(S, i-1) \geq \mathrm{M}(S, j+1)$, $t(s') \leftarrow t$ and $S \leftarrow S \cup \{s'\}$. Otherwise, $t(s'') \leftarrow t$ and $S \leftarrow S \cup \{s''\}$.

9: **end while**

10: Return S.

Theorem 14. *Given any instance $\boldsymbol{S}$ of Restricted Strip Cover, our algorithm returns a schedule S such that $M(S) \geq OPT/5$.*

Lemma 15. *Given some instance of Restricted Strip Cover $\boldsymbol{S}$, let S be the schedule returned by our algorithm. Let $u, v \in \boldsymbol{S}$ be any two, distinct sensors. If $R(u)$ is strictly contained in $R(v)$, then u is scheduled after v and in fact $t(u) \geq t(v) + d(v)$.*

Proof. Suppose that u and v are two unscheduled sensors such that $R(u)$ is strictly contained in $R(v)$. Sensors are only scheduled when they dominate some coordinate

to the left or to the right. Suppose we want to find the sensor that dominates some coordinate $i \in [\ell(u), r(u)]$. We will consider both u and v, but will always prefer v to u from the definition of domination. Therefore, we will schedule v before u and will not consider another sensor to dominate a coordinate in $[\ell(u), r(u)]$ until after time $t(v) + d(v)$. □

For $x \in U$ and $t > 0$, we define coverage(x, t) to be the number of sensors that cover x at time t in the schedule output by our algorithm.

We need the following observation.

Lemma 16. *If coverage$(x, t) \leq c$ for each $x \in U$ and $t > 0$, then the duration t_f of the schedule we output is at least L/c.*

Proof. At termination, there is a point $x \in U$ so that $\mathrm{M}(S, x) = t_f$ and there is no unassigned sensor that is live at x. Thus, $ct_f \geq L(x) \geq L$, and so $t_f \geq L/c$. □

We will now show that coverage$(x, t) \leq 5$ for each $x \in U$ and $t > 0$. Theorem 14 then follows immediately from this and Lemma 16.

In each iteration the algorithm schedules (assigns) a sensor s which is either s' or s''. Let us call the corresponding interval $[i, j]$ the *interval for which s is scheduled.* (Please refer to the algorithm for what i and j stand for.) If $s = s'$, we call s a *right going* sensor to remember that it was chosen to dominate i to the right. In this case, i was not covered at time $t(s)$ before s was scheduled, but i was covered at time $t(s)$ after s was scheduled. We say that $s = s'$ *closes* i at time $t(s)$. Similarly, if $s = s''$ we call s a *left going* sensor and say that it closes j at time $t(s)$.

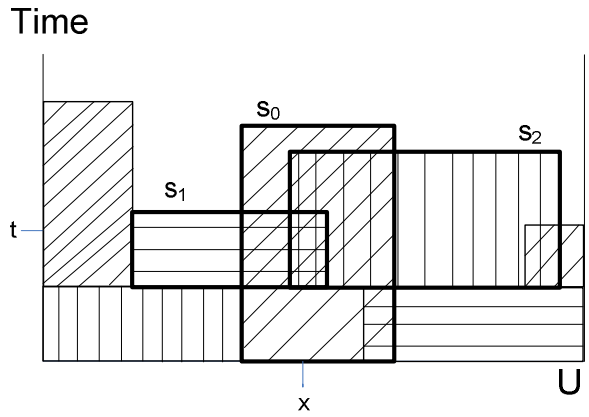


Figure 4.1: There are 3 sensors covering (x, t). The first scheduled sensor to cover (x, t) is s_0. The next sensor to cover (x, t) is a type 2 sensor s_1. Finally, (x, t) is covered by a type 4 sensor s_2.

Lemma 17. *For any $x \in U$ and $t > 0$, coverage$(x, t) \leq 5$.*

Proof. Fix an $x \in U$ and $t > 0$. If no sensor in the output schedule covers (x, t), then coverage(x, t) is 0. Let us therefore suppose that some sensor covers (x, t), and let s_0 denote the first scheduled sensor that covers (x, t). Let us classify any other sensor s that covers (x, t) into exactly one of the following four types: (1) s closes some $i < x$ and is left going; (2) s closes some $i < x$ and is right going; (3) s closes some $i > x$ and is left going; (4) s closes some $i > x$ and is right going. We show that there are at most two sensors of types 1 and 2 put together. A symmetric argument shows that there are at most two sensors of types 3 and 4 put together.

Suppose that at some point the algorithm adds a sensor l of type 1. We claim that after l is added no more sensors of types 1 or 2 are added. Suppose l closed $i < x$ at time $t(l)$. Consider some sensor l' that is live at x and that closes some $i' < x$ when it is later added by the algorithm. Since l was chosen because it dominated

i to the left, we can conclude that $\ell(l') \geq \ell(l)$. (If this were not the case, then l' would necessarily be live at i and would have been preferred to l.) Observe that the interval $[\ell(l), x]$ is covered by l for all times between $t(l)$ and $t(l) + d(l) - 1$. Since $i' \in [\ell(l'), x] \subseteq [\ell(l), x]$, we must have $t(l') > t(l) + d(l) - 1 \geq t$. Thus l' will not cover x at time t. We conclude that once we schedule a sensor of type 1, we do not schedule any more sensors of types 1 or 2.

We will therefore consider the case where the first sensor of type 1 or 2 is a type 2 sensor which we denote r_1. We need the following claim.

Claim 18. *Let v be a sensor scheduled after r_1 such that (a) v is live at x, (b) $t(v) \leq t$, and (c) v closes some $z < x$ at time $t(v)$. Let $[x', y']$ be the interval for which v is scheduled. Then $y' + 1 = \ell(r_1)$.*

We prove the claim after completing the rest of the proof of the lemma. If the next sensor of type 1 or 2 that we schedule after r_1 is a type 1 sensor, we do not schedule any more sensors of types 1 and 2. So let us assume that the next sensor of types 1 or 2 that we schedule after r_1 is a type 2 sensor r_2. Since $t \geq t(r_2) \geq t(r_1)$, and r_2 closes some $i' < x$ at time $t(r_2)$, we must have $i' < \ell(r_1)$, and thus $\ell(r_2) < \ell(r_1)$. Now assume for the sake of a contradiction that there is some sensor r_3 of type 1 or 2 that is scheduled after l_2. Reasoning as above, the interval $[x'', y'']$ that it is scheduled for has $y'' < \ell(r_2)$. Thus $y'' + 1 \leq \ell(r_2) < \ell(r_1)$, a contradiction to Claim 18.

Thus we have completed the proof of the lemma and we now prove Claim 18.

Proof of Claim 18. First, we clarify to the reader that in the statement of the claim condition (a) does not require v to cover (x, t), but only requires v to be live

at x. Let $v_1, \ldots, v_k$ be the sensors satisfying the hypothesis of the claim, ordered in the sequence in which they were chosen by the algorithm. Let $[x_j, y_j]$ be the interval for which v_j is scheduled. Since $t(r_1) \leq t(v_j) \leq t$ and v_j closes some point strictly less than x, we must have $y_j < \ell(r_1)$. We have argued that $y_j \leq \ell(r_1) - 1$, and we want to show $y_j = \ell(r_1) - 1$. Suppose for the sake of contradiction that $y_j < \ell(r_1) - 1$ for some j, and consider the first j for which this happens.

When v_j was being scheduled, $y_j + 1$ is covered at time $t(v_j)$ by a sensor w. Clearly, it must be that $y_j + 1 = \ell(w)$. So we have $\ell(v_j) < \ell(w) < \ell(r_1)$.

If w was scheduled before r_1, then $r(w) < i' < x$ where i' is the point that r_1 closes. Thus $R(v_j)$ properly contains $R(w)$ but v_j is scheduled after w, a contradiction to Lemma 15.

If w was scheduled after r_1, then w must be live at x, for otherwise $R(v_j)$ properly contains $R(w)$ and we derive a contradiction as above. Also, we must have $r(w) < r(r_1)$ for otherwise $R(w)$ properly contains $R(r_1)$ and we reach a contradiction. Since $r(w) < r(r_1)$, w closes some $i < \ell(r_1)$. Thus w is scheduled after r_1, $t(w) \leq t(v_j) \leq t$, w is live at x, and w closes some point strictly to the left of x. Thus $w = v_{j'}$ for some $j' < j$. If $j = 1$, we have reached a contradiction. We therefore assume $j > 1$. We observe that w cannot be a left going sensor, because v_j is also live at the point i that w closes, and would have been preferred to w otherwise since $\ell(v_j) < \ell(w)$. Since $j' < j$, we have $y_{j'} + 1 = \ell(r_1)$. Thus $\mathrm{M}(S, y_{j'} + 1) \geq t$ at the time $w = v_{j'}$ is being scheduled. Since w is right going, we must have $\mathrm{M}(S, x_{j'} - 1) \geq \mathrm{M}(S, y_{j'} + 1) \geq t$ at the time w is being scheduled. Let s denote the sensor that covers $x_{j'} - 1$ at times

$t(w), t(w)+1, \ldots, t$ when w is being scheduled. We have $r(s) = x_{j'} - 1 < x_{j'} < x \leq r(v_j)$. Now v_j closes some point $i'' \in [x_j, y_j]$ at time $t(v_j)$, and since $t \geq t(v_j) \geq t(w)$, i'' cannot be in $R(s)$. Since $i'' \leq y_j = \ell(w) - 1 \leq x_{j'} - 1 = r(s)$, it must be that $i'' < \ell(s)$. So $\ell(v_j) \leq i'' < \ell(s)$ and we have already argued that $r(v_j) > r(s)$. So $R(v_j)$ properly contains $R(s)$ but v_j is scheduled after s, a contradiction to Lemma 15. □

The following observation about our algorithm for RSC is evident from the analysis.

Observation 19. *Suppose that we stop our algorithm once the duration of the schedule becomes greater than equal to t. Then the total load of all the scheduled sensors that are live at some point x of the universe is at most $5(t + d_{max})$, where d_{max} is the maximum duration of any input sensor.*

CHAPTER 5
THE PLANAR SENSOR COVER PROBLEM

In this chapter, we consider the planar sensor cover problem. Here, we are given a universe U which is a set of m points in the plane. We are also given a set of n sensors $\mathbf{S}$, each of which is a translate of a convex polygon and has a duration. We want to assign a start time to each of the sensors so that all of the points in U are monitored for as long as possible. We use the RSC algorithm to give a constant factor approximation for the planar sensor cover problem.

5.1 Planar Sensor Cover via RSC

We now describe a polynomial time algorithm for the planar sensor cover problem. Recall that the input here is a universe U which is a set of m points in the plane and a set of n sensors $\mathbf{S}$; each sensor $s \in \mathbf{S}$ has a integer duration $d(s)$, and a range $R(s)$ which is a translate of a convex polygon P. If the load of the problem instance is L, our algorithm computes a schedule of $\mathbf{S}$ of duration at least L/α for some constant $\alpha \geq 1$, where the constant α depends on P.

It is convenient to think of the problem in its dual form as done in [60, 2]. Fix O, the centroid of P, as the origin in the plane. For a planar set T and a point x in the plane, let $T(x)$ denote the translate of T with centroid x. Let $\bar{P}$ be the reflection through O of the polygon P. For points p and x in the plane, $p \in P(x)$ if and only if $x \in \bar{P}(p)$.

We now view this problem in a slightly different sense. We can map each

sensor s to the point which is the centroid of the polygon $R(s)$; abusing notation, we denote by $\mathbf{S}$ the point set thus obtained. We can map each $x \in U$ to a translate $\bar{P}(x)$ of $\bar{P}$ centered at x. From this point of view, we are given a set of n points in the plane, each of which has a duration, and we are given a set of convex polygons such that the sum of the durations of the points that lie within each polygon is at least L. We would like to schedule a start time to each of the points so that each polygon contains an active point for all times $t \leq L/\alpha$.

5.1.1 From Polygons to Wedges

Denote the vertices of $\bar{P}$ to be $p_0, p_1, p_2, \ldots, p_{\mu-1}$ in counterclockwise order. Addition and subtraction of indices of these vertices will be taken modulo μ throughout the paper. The set of indices between index i and index j in counterclockwise order are denoted $[i, j]$. We now transform the problem further, so that instead of dealing with translates of $\bar{P}$, we can deal with translates of the μ *wedges* corresponding to the vertices of $\bar{P}$ [52, 60, 2].

Let c be equal to half the minimum distance between two points on non-consecutive edges of $\bar{P}$. We lay a square grid of side c on the plane; any translate of $\bar{P}$ intersects $\beta \in O(1)$ grid cells, and each grid cell intersects at most two sides of a translate; moreover, if a grid cell does intersect two sides of a translate, then these sides must be adjacent in $\bar{P}$.

Since each grid cell intersects at most two edges of $\bar{P}(u)$, it must be that the intersection of a grid cell with $\bar{P}(u)$ is the same as the intersection of the grid cell

with a wedge whose boundaries are parallel to two adjacent edges of $\bar{P}(u)$. If one boundary of the wedge is parallel to the edge $p_{i-1}p_i$ of $\bar{P}$ and the other is parallel with p_ip_{i+1} of $\bar{P}$, then we call the wedge an i-wedge. For a point q in the plane, we denote $W_i(q)$ to be the i-wedge with apex q. See Figure 5.1 for an illustration.

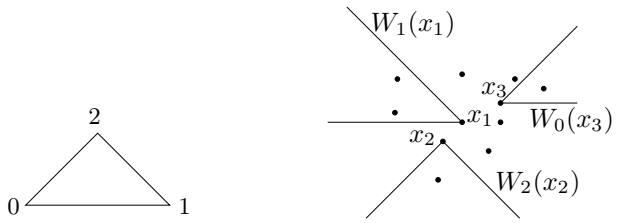


Figure 5.1: An illustration for the wedges of a polygon. (a) Suppose this triangle is our polygon with vertices indexed accordingly. (b) A 0-wedge, 1-wedge, and 2-wedge with respect to the polygon.

Given a subset R of the plane and a subset X of the point set $\mathbf{S}$, let us denote by $\text{load}_X(R)$ the load $\sum_{s\in X\cap R} d(s)$ of region R with respect to X. The number of squares that some $\bar{P}(u)$ intersects is bounded by a constant β. Since for each $u \in U$, we have $\text{load}_{\mathbf{S}}(\bar{P}(u)) \geq L$, some square that $\bar{P}(u)$ intersects must have load at least L/β. We can therefore make the points within such a grid cell "responsible" for $\bar{P}(u)$. It therefore suffices to solve the following problem separately for each square in the tiling:

> Let $Y \subseteq \mathbf{S}$ be the subset of the sensors (viewed as points) that fall inside this square. Assign a start time to each $s \in Y$ such that for any $0 \leq i \leq$

$\mu - 1$ and for any i-wedge W with $\text{load}_Y(W) \geq k \equiv L/\beta$ and any time t with $1 \leq t \leq k/\alpha'$, there is a point in $Y \cap W$ that is active at time t. Here $\alpha' \geq 1$ is a constant that depends only on μ.

In the rest of this section, we describe an efficient algorithm for this problem. Some of the ideas are adopted from [2]. We assume that the point set Y is in general position – a line parallel to a side of $\bar{P}$ contains at most one point in Y. It is straightforward to perturb the input to the original problem so that this assumption holds for Y.

5.1.2 The Structure of Heavy Wedges

We will now define a boundary for an $i \in \{0, 1, \ldots \mu - 1\}$ and a positive integer r. This boundary has the property that any i-wedge placed on or "inside" the boundary has load at least r, and any i-wedge placed "outside" the boundary as load less than r. That is, the sum of the durations of $W_i(x) \cap Y$ for any x inside the boundary or on the boundary is at least r and is less than r for any x outside the boundary. This boundary is called a *level curve* and is a generalization of level curves used in [2] which extends the definition of *boundary points* [51, 52]. Let $\mathcal{W}_i^j$ be the set of apices of all i-wedges W such that $\text{load}_Y(W) = j$. For each $i = 0, 1, \ldots \mu - 1$, let the level curve $\mathcal{C}_i(r)$ be the boundary of the region $\mathcal{W}_i^{\geq r} = \bigcup_{j \geq r} \mathcal{W}_i^j$ for each $i = 0, 1, \ldots \mu - 1$.

Note that $\mathcal{C}_i(r)$ is a monotone staircase polygonal path with edges that are parallel to the edges of an i-wedge. See Figure 5.2. Let d_{max} denote the maximum

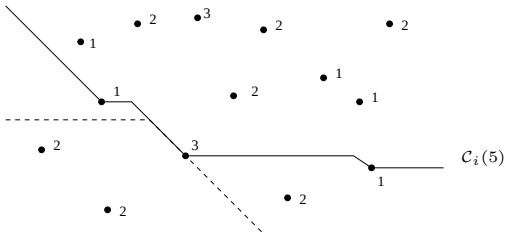


Figure 5.2: An example of a level curve $\mathcal{C}_i(r)$ for $r = 5$. Note that any i-wedge with apex on $\mathcal{C}_i(5)$ (e.g. the dotted wedge) contains load at least 5.

duration of any sensor in Y. We have the following observations:

Observation 20. *For any $x \in \mathcal{C}_i(r)$, $r \leq load_Y(W_i(x)) \leq r + d_{max}$.*

Observation 21. *Any i-wedge W such that $load_Y(W) \geq r$ contains an i-wedge whose apex belongs to $\mathcal{C}_i(r)$.*

Observe that one of the two extreme edges of the level curve $\mathcal{C}_i(r)$ is a semi-infinite ray parallel to edge $q_{i-1}q_i$. Let h_i denote the first point along this ray such that for all points y on the ray that lie after h_i, $W_i(y) \cap Y = W_i(h_i) \cap Y$. We call h_i the *head* of $\mathcal{C}_i(r)$. The other extreme edge of $\mathcal{C}_i(r)$ is parallel to edge q_iq_{i+1}. Let τ_i denote the first point along this ray such that for all points y on the ray that lie after τ_i, $W_i(y) \cap Y = W_i(h_i) \cap Y$. We call τ_i the *tail* of $\mathcal{C}_i(r)$. See Figure 5.3.

5.1.3 Multiple RSC instances

Suppose that we are given a subset $Y' \subseteq Y$, and we want to assign start times to some of the points in Y' so that for any $x \in C_i(k)$, the i-wedge $W_i(x)$ has an active point for a sufficiently long duration. Note that because of Observation 21, such a schedule would satisfy all i-wedges which have load at least k with respect to Y.

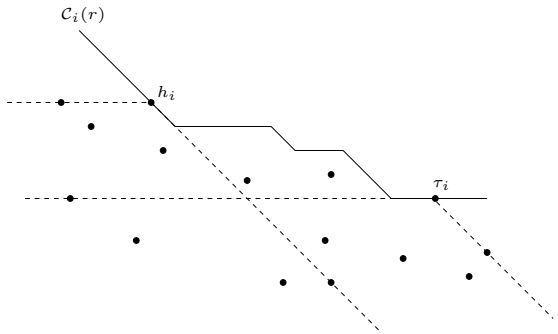


Figure 5.3: Level curve $\mathcal{C}_i(r)$ with h_i and τ_i denoted.

This can be mapped to an instance of RSC where the universe is the one-dimensional level curve $\mathcal{C}_i(k)$; each point $p \in Y'$ maps to the set $I(p) = \{x \in \mathcal{C}_i(r) \mid p \in W_i(x)\}$. Each $I(p)$ is an "interval" in this one-dimensional universe, see Figure 5.4, and we associate with it the duration $d(p)$. Finding a partial schedule with duration L' for this RSC problem is equivalent to assigning start times for some of the sensors in Y' so that for each $W_i(x)$ for $x \in \mathcal{C}_i(k)$ and each time $t \leq L'$, some point in $W_i(x) \cap Y'$ is active at time t. This equivalence is used by the algorithm for the planar sensor cover problem.

Before we describe the algorithm, we need to define two notions. Consider the natural linear ordering of the lines parallel to side p_ip_{i+1} of $\bar{P}$ with the line through vertices p_i and p_{i+1} being smaller than the line through any of the other vertices of $\bar{P}$. For $x, y \in \Re^2$, we define the partial order $<_i$ such that $x <_i y$ if the p_ip_{i+1} parallel line through x is less than the p_ip_{i+1} parallel line through y.

For a vertex p_i, let A_i denote the set of all indices j such that the intersection of $\bar{P}$ with the line parallel to p_jp_{j+1} and through p_i is only the point p_i. See Figure

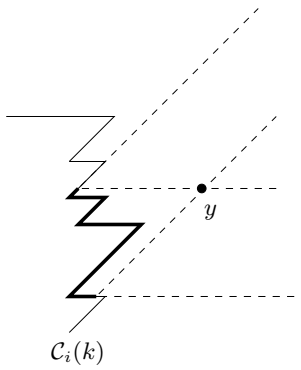


Figure 5.4: An example of an interval $I(y)$ (in bold). Note that the i-wedges with apex on $\mathcal{C}_i(k)$ that contain y are the dotted wedges and all wedges with apex "in between" the apices of the dotted wedges.

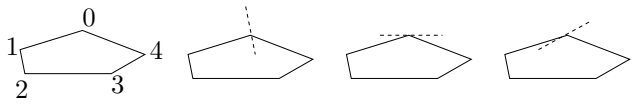


Figure 5.5: An example of a set A_i. In this example, $A_0 = \{2\}$ since only the side of $\bar{P}$ parallel with p_2p_3 has the qualifying property.

5.5 for an example.

Algorithm 5.1 calls the RSC algorithm for each $0 \leq i \leq \mu - 1$. The set X_i in the i-th iteration is an appropriately chosen subset of the points in Y not scheduled in iterations $0, 1, \ldots, i-1$. At the beginning of the i-th iteration, let L denote, as in the algorithm, the smallest load of a j-wedge with apex on $\mathcal{C}_j(k)$ with respect to Y', for $i \leq j \leq \mu - 1$. The RSC algorithm will schedule sensors up until time $\frac{L}{100\mu}$, and we have $\min_{c \in \mathcal{C}_i(k)} \text{load}_{X_i}(W_i(c)) \geq \frac{L}{2}$ (due to the manner in which X_i is chosen in the algorithm, see Observation 24 below). After the execution of the RSC algorithm, any i-wedge with apex on $\mathcal{C}_i(k)$ contains an active point for all times $1, 2, \ldots, \frac{L}{100\mu}$.

Thus, the algorithm produces a schedule as required, provided $L \in \Omega(k)$. This is established by the following key Lemma. It states that L, which equals k before the 0-th iteration, drops by a factor of at most 5μ with each iteration. In the proof of the lemma, we will assume $d_{max} \leq \frac{L}{100\mu}$. If some sensor has duration greater than $\frac{L}{100\mu}$, then we can set its start time to 1, and all wedges that contain this sensor will be actively covered up through time $\frac{L}{100\mu}$.

Lemma 22. *Suppose at the beginning of iteration i, all j-wedges with apex on $\mathcal{C}_j(k)$ have load at least L from points in Y' for $j \geq i$, where L is larger than some absolute constant. (Note that Y' always denotes the unscheduled points in the algorithm.) After the i-th iteration of the algorithm, any j-wedge $W_j(x)$, for $j > i$, and with apex x on $\mathcal{C}_j(k)$ has load at least $\frac{L}{5\mu}$ from points in Y'.*

Proof. There are two main cases to consider:

- p_i and p_j are antipodal vertices of $\bar{P}$ – that is, there are parallel lines through p_i and p_j with the convex polygon sandwiched between them.
- p_i and p_j are not antipodal vertices of $\bar{P}$.

We use the following terminology for iteration i: for two distinct points q and q', if $W_i(q) \subseteq W_i(q')$, we say that q *dominates* q'. Notice that in this case, we have $I(q') \subseteq I(q)$. Thus by Lemma 15, if q and q' are both unscheduled before iteration i then q' is scheduled in iteration i only if q is already scheduled. For the rest of this proof, let Y' denote the points that are not scheduled just before iteration i, let X_i denote the set of candidate points that are eligible to be scheduled in iteration

i (as constructed in the algorithm), and let Y_i denote the points that are actually scheduled in iteration i.

The analysis will rely heavily upon the following two observations.

Observation 23. *For any $z \in \mathcal{C}_i(k)$, we have that $load_{Y_i}(W_i(z)) \leq 5(\frac{L}{100\mu} + \frac{L}{100\mu}) \leq \frac{L}{10\mu}$.*

Observation 24. *For any $z \in \mathcal{C}_i(k)$, $load_{X_i}(W_i(z)) \geq \frac{L}{2}$.*

Observation 23 is a consequence of Observation 19. To see Observation 24, note that for each $c \in \mathcal{C}_i(k)$, $\text{load}_{X_i}(W_i(c)) \geq \text{load}_{X(c)}(W_i(c)) \geq L - \mu \cdot \frac{L}{2\mu} = \frac{L}{2}$.

Case 1: p_i and p_j are not antipodal vertices of P.

The argument is trivial if $W_j(x) \cap \mathcal{W}_i^{\leq k} = \emptyset$. Suppose that $W_j(x) \cap \mathcal{W}_i^{\leq k} \neq \emptyset$ and $W_j(x) \cap \mathcal{C}_i(k) = \emptyset$. For this to be the case, one of the boundaries of a j-wedge must be parallel with a boundary of an i-wedge. We will focus on the case when the j-wedge has a boundary parallel to the side p_ip_{i+1} as the other case is symmetric. See Figure 5.6. In this case, we have that $W_j(x) \cap X_i' \subseteq W_i(\tau_i)$ by the definition of the tail τ_i. We thus have $\text{load}_{Y_i}(W_j(x)) \leq \text{load}_{Y_i}(W_i(\tau_i)) \leq \frac{L}{10\mu}$ where the last inequality comes from Observation 23. Therefore, the load of unscheduled points in $W_j(x)$ after iteration i is at least $L - \frac{L}{10\mu} > \frac{L}{5\mu}$.

So let us assume that $W_j(x) \cap \mathcal{C}_i(k) \neq \emptyset$. There are two cases – in the first, we encounter p_j after p_i and before the vertices antipodal to p_i when walking counter-clockwise around $\bar{P}$, and in the second, we encounter p_j after the vertices antipodal to p_i and before p_i. We will focus on the first case, since the other is symmetric. Let

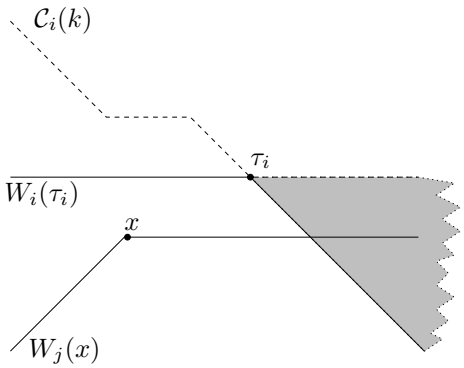


Figure 5.6: Illustration for Case 1. Note that there cannot be a point in the shaded region by definition of the tail τ_i.

z be the intersection point of the boundary of $W_j(x)$ and $\mathcal{C}_i(k)$. If $W_j(x)$ does not contain in its interior the tail τ_i of the level curve $\mathcal{C}_i(k)$, then $W_j(x) \cap Y_i \subseteq W_i(z) \cap Y_i$, and so $\text{load}_{Y_i}(W_j(x)) \leq \text{load}_{Y_i}(W_i(z)) \leq \frac{L}{10\mu}$. It follows that the load of unscheduled points in $W_j(x)$ after iteration i is at least

$$L - \frac{L}{10\mu} > \frac{L}{5\mu}.$$

Let us therefore assume that $W_j(x)$ does contain in its interior the tail τ_i of $\mathcal{C}_i(k)$. See Figure 5.7. Let a denote the point where the boundaries of the wedges $W_i(z)$ and $W_i(\tau_i)$ intersect. If $\text{load}_{X_i}(W_i(a)) \geq \frac{L}{5\mu}$, then since $\text{load}_{Y_i}(W_i(a)) \leq \text{load}_{Y_i}(W_i(\tau_i)) \leq \frac{L}{10\mu}$, there are unscheduled points in $W_i(a)$ after iteration i. Since any point in $W_i(a)$ dominates points in $W_j(x) \cap Y_i$ that are not contained in $W_i(z) \cup W_i(\tau_i)$, we conclude that $W_j(x) \cap Y_i \subseteq (W_i(z) \cup W_i(\tau_i)) \cap Y_i$. Thus,

$$\text{load}_{Y_i}(W_j(x)) \leq \text{load}_{Y_i}(W_i(z)) + \text{load}_{Y_i}(W_i(\tau_i)) \leq \frac{L}{5\mu}.$$

Therefore there must be at least $L - \frac{L}{5\mu} > \frac{L}{5\mu}$ unscheduled points left in $W_j(x)$.

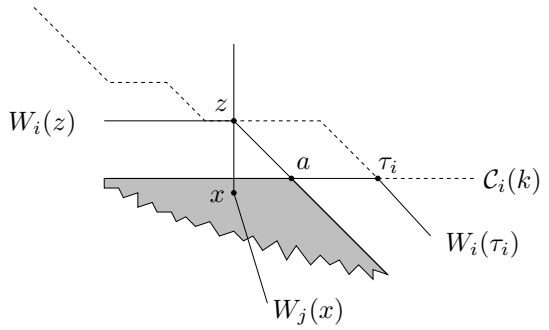


Figure 5.7: Illustration for the nonantipodal case.

Let us therefore consider the case where $\text{load}_{X_i}(W_i(a)) < \frac{L}{5\mu}$. This means that $\text{load}_{X_i}(W_i(\tau_i) \setminus W_i(a)) > \frac{L}{2} - \frac{L}{5\mu} > \frac{L}{3}$. Again, $\text{load}_{Y_i}(W_i(\tau_i) \setminus W_i(a)) \leq \text{load}_{Y_i}(W_i(\tau_i)) \leq \frac{L}{10\mu}$, and this means the load of the points in $W_i(\tau_i) \setminus W_i(a)$ that are unscheduled after iteration i is at least $\frac{L}{3} - \frac{L}{10\mu} > \frac{L}{5\mu}$. But $W_i(\tau_i) \setminus W_i(a) \subseteq W_j(x)$, and this means that the load of the unscheduled points in $W_j(x)$ after iteration i is at least $\frac{L}{5\mu}$.

Case 2: p_i and p_j are antipodal vertices of $\bar{P}$.

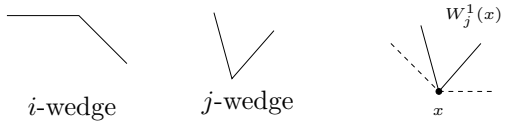
Again, the argument is trivial if $W_j(x) \cap \mathcal{W}_i^{\leq k} = \emptyset$. So let us assume that $W_j(x) \cap \mathcal{W}_i^{\leq k} \neq \emptyset$. Since $\text{load}_{Y'}(W_j(x)) \geq L$, if $\text{load}_{X_i}(W_j(x)) \leq \frac{L}{2}$ then $W_j(x)$ will clearly have load at least $\frac{L}{5\mu}$ after iteration i. So assume that $\text{load}_{X_i}(W_j(x)) > \frac{L}{2}$.

Consider the line parallel with $p_{i-1}p_i$ through x and the line parallel with p_ip_{i+1} through x. Note these lines are parallel with the boundaries of an i-wedge. Let $H_t(x)$ denote the halfplane consisting of all points y such that $y \leq_t x$. Let $W_j^1(x) = H_{i-1}(x) \cap H_i(x) \cap W_j(x)$. Let $W_j^2(x) = (H_i(x) \cap W_j(x)) \setminus W_j^1(x)$. Let $W_j^3(x) = (H_{i-1}(x) \cap W_j(x)) \setminus W_j^1(x)$. See Figure 5.8 and Figure 5.9 for an illustration. Note that $W_j^1(x), W_j^2(x)$, and $W_j^3(x)$ form a partition of $W_j(x)$. Also note that $W_j^1(x)$

cannot be empty but $W_j^2(x)$ or $W_j^3(x)$ could be empty. Since these three sets form a partition of $W_j(x)$ and $\text{load}_{X_i}(W_j(x)) > \frac{L}{2}$, it must be that one of the three sets has load at least $\frac{L}{6}$ from X_i. We first handle the case when $\text{load}_{X_i}(W_j^1(x)) \geq \frac{L}{6}$ and then conclude the proof with the case when $\text{load}_{X_i}(W_j^2(x)) \geq \frac{L}{6}$. The case when $\text{load}_{X_i}(W_j^3(x)) \geq \frac{L}{6}$ has a symmetric proof with the $W_j^2(x)$ case.

(a) (b)

Figure 5.8: An illustration for the antipodal case. If we are working with the corresponding i-wedge and j-wedge (part (a)), then we obtain the corresponding $W_j^1(x)$, $W_j^2(x)$, and $W_j^3(x)$ (part (b)).



(a) (b)

Figure 5.9: An illustration for the antipodal case. If we are working with the corresponding i-wedge and j-wedge (part (a)), then $W_j^1(x) = W_j(x)$ and $W_j^2(x) = W_j^3(x) = \emptyset$ (part (b)).

Case 2(a): $\text{load}_{X_i}(W_j^1(x)) \geq \frac{L}{6}$.

We first consider the case when both boundaries of $W_j^1(x)$ intersect with $\mathcal{C}_i(k)$.

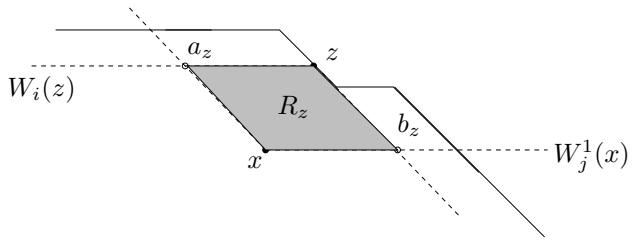


Figure 5.10: Illustration for Case 2(a): the region R_z. Note that although this figure is drawn with respect to a scenario as in Figure 5.8, the analysis still holds for the scenario as in Figure 5.9 (i.e. when the boundaries of an i-wedge are not parallel with the boundaries of $W_j^1(x)$).

Consider any point $z \in W_j^1(x) \cap \mathcal{C}_i(k)$. Let a_z denote the "leftmost" point where the boundaries of $W_j^1(x)$ and $W_i(z)$ intersect, and let b_z denote the "rightmost" point where the boundaries of $W_j^1(x)$ and $W_i(z)$ intersect. Let R_z be the quadrilateral with vertices a_z, x, b_z, and z. That is, $R_z = W_j^1(x) \cap W_i(z)$. Suppose that $\text{load}_{X_i}(R_z) \geq \frac{L}{5\mu} + \frac{L}{10\mu}$. Again, since $\text{load}_{Y_i}(W_i(z)) \leq \frac{L}{10\mu}$, and all points in R_z are in $W_i(z)$, R_z contains (unscheduled) load at least $\frac{L}{5\mu}$ after iteration i. Since $R_z \subseteq W_j^1(x)$, $W_j^1(x)$ contains (unscheduled) load at least $\frac{L}{5\mu}$ after iteration i, and we are done. See Figure 5.10 for an illustration.

So we now assume that $\text{load}_{X_i}(R_z) \leq \frac{L}{5\mu} + \frac{L}{10\mu}$ for each $z \in W_j^1(x) \cap \mathcal{C}_i(k)$. Since $\text{load}_{X_i}(W_i(z)) \geq \frac{L}{2}$, we must have $\text{load}_{X_i}(W_i(a_z) \cup W_i(b_z)) \geq \frac{L}{2} - (\frac{L}{5\mu} + \frac{L}{10\mu}) > \frac{L}{8}$. Let z_1 be the "leftmost" point on $\mathcal{C}_i(k) \cap W_j^1(x)$, and let z_2 be the "rightmost" point on $\mathcal{C}_i(k) \cap W_j^1(x)$. Notice that a_{z_1} is just z_1 itself, and so $\text{load}_{X_i}(W_i(a_{z_1})) \geq \frac{L}{2}$. Similarly, $\text{load}_{X_i}(W_i(b_{z_2})) \geq \frac{L}{2}$. Let z' be the last point on $\mathcal{C}_i(k)$, while walking from z_1 to z_2,

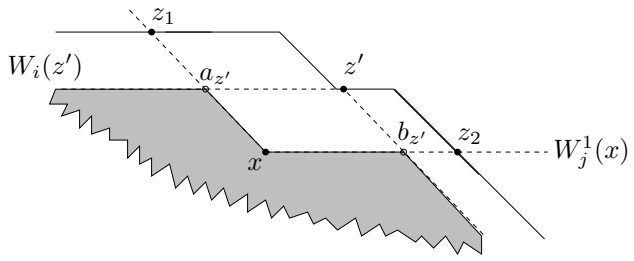


Figure 5.11: Illustration for Case 2(a): the constructed point z'.

such that $\text{load}_{X_i}(W_i(a_{z'})) \geq \frac{L}{30}$ (if it exists). Thus

$$\text{load}_{X_i}(W_i(b_{z'})) \geq \text{load}_{X_i}(W_i(a_{z'}) \cup W_i(b_{z'})) - (\frac{L}{30} + d_{max}) \geq \frac{L}{8} - (\frac{L}{30} + \frac{L}{100\mu}) \geq \frac{L}{30}.$$

See Figure 5.11 for an illustration.

Now consider any point $z'' \in W_j^1(x) \setminus W_i(z')$. It must be that $W_i(z'')$ either contains $W_i(a_{z'})$ or contains $W_i(b_{z'})$ which both have load in X_i of at least $\frac{L}{30}$. Suppose that $W_i(z'')$ contains $W_i(a_{z'})$; the other case is similar. The points in $W_i(a_{z'})$ dominate z'' and we will not schedule z'' in iteration i until we have scheduled all points in $W_i(a_{z'}) \cap X_i$. But since $\text{load}_{Y_i}(W_i(a_{z'})) \leq \frac{L}{10\mu} \leq \frac{L}{30} \leq \text{load}_{X_i}(W_i(a_{z'}))$, this means we will not schedule z''.

It follows that $W_j^1(x) \cap Y_i \subseteq W_i(z') \cap Y_i$, and thus

$$\text{load}_{Y_i}(W_j^1(x)) \leq \text{load}_{Y_i}(W_i(z')) \leq \frac{L}{10\mu}$$

. And so the load of unscheduled points in $W_j^1(x)$ after iteration i is at least $\frac{L}{6} - \frac{L}{10\mu} > \frac{L}{5\mu}$.

We now consider the case when one or both boundaries of $W_j^1(x)$ do not intersect with $\mathcal{C}_i(k)$. See Figure 5.12. We will show that we can find a wedge W' such

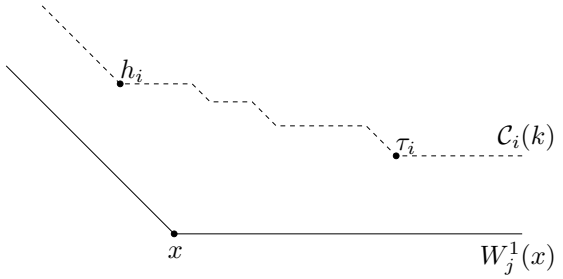


Figure 5.12: Illustration of the case when $W_j^1(x) \cap \mathcal{W}_i^{\leq k} \neq \emptyset$ and the boundaries of $W_j^1(x)$ do not intersect $\mathcal{C}_i(k)$.

that both boundaries of W' intersect $\mathcal{C}_i(k)$, $W' \cap Y = W_j^1(x) \cap Y$, and W' is antipodal with respect to an i-wedge. Given that W' exists, we can use the previous arguments to show that W' contains $\Omega(k)$ uncolored points after iteration i. It then follows that $W_j(x)$ contains $\Omega(k)$ uncolored points after iteration i because $W' \cap Y = W_j^1(x) \cap Y$.

We will now describe how to find the wedge W'. We begin by placing a wedge identical to $W_j^1(x)$ "just behind" $W_j^1(x)$ so that the new wedge contains exactly the same points of Y that $W_j^1(x)$ contains. See Figure 5.13. We then "bend in" the boundaries of the wedge just enough so that the wedge still contains the same points as $W_j^1(x)$ and the boundaries are no longer parallel with the edges of $\mathcal{C}_i(k)$. This is our wedge W'. See Figure 5.14. Both boundaries now intersect $\mathcal{C}_i(k)$ and W' is antipodal with respect to an i-wedge. By our previous analysis, W' contains at least $\frac{L}{5\mu}$ uncolored points after iteration i. Since $W_j^1(x)$ and W' contain the same points in Y, it follows that $W_j^1(x)$ contains $\frac{L}{5\mu}$ uncolored points after iteration i.

Case 2(b): $\text{load}_{X_i}(W_j^2(x)) \geq \frac{L}{6}$.

We first consider the case when $W_j^2(x) \cap \mathcal{C}_i(k) = \emptyset$. Note that one of the

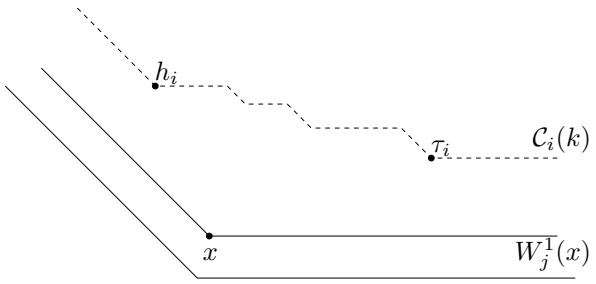


Figure 5.13: The first step in constructing W'.

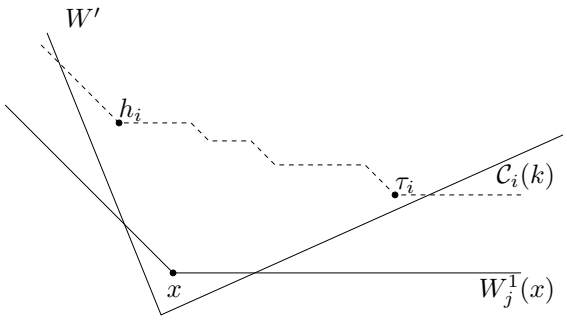


Figure 5.14: The second step in constructing W'.

boundaries of $W_j^2(x)$ is parallel with one of the boundaries of an i-wedge. We will assume that the j-wedge has a boundary parallel with the side $p_{i-1}p_i$. The other case is symmetric. See Figure 5.15. In this case, we have that $W_j^2(x) \cap X_i \subseteq W_i(h_i) \cap X_i$ by the definition of the head h_i. We know that $\text{load}_{Y_i}(W_i(h_i)) \leq \frac{L}{10\mu}$. Since $W_j^2(x) \cap X_i \subseteq W_i(h_i) \cap X_i$, we have that $\text{load}_{Y_i}(W_j^2(x)) \leq \frac{L}{10\mu}$. Therefore there is (unscheduled) load at least $\frac{L}{6} - \frac{L}{10\mu} > \frac{L}{5\mu}$ after iteration i.

Now we will assume $W_i^2(x) \cap \mathcal{C}_i(k) \neq \emptyset$. Let $z \in \mathcal{C}_i(k)$ be a point such that $W_i(z) \cap W_j^2(x) \neq \emptyset$. Note that both $W_j^2(x)$ and $W_i(z)$ have a boundary parallel with the side $p_{i-1}p_i$. There are only two types of intersections between these two wedges:

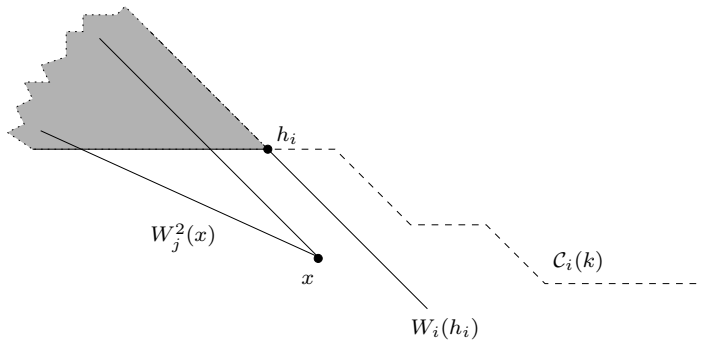
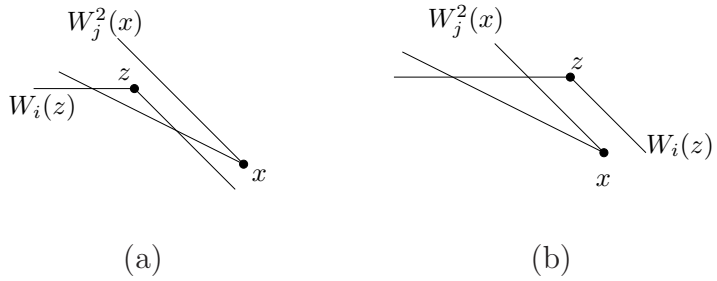


Figure 5.15: Illustration for Case 2(b). Note that there cannot be any points in the shaded region due to the definition of the head h_i.

1. $z \in W_j^2(x)$, the boundary of $W_j^2(x)$ parallel with the side $p_{j-1}p_j$ intersects both boundaries of $W_i(z)$, and the boundary of $W_j^2(x)$ parallel with the side $p_{i-1}p_i$ does not intersect with $W_i(z)$.

2. $x \in W_i(z)$, the boundary of $W_i(z)$ parallel with the side p_ip_{i+1} intersects both boundaries of $W_j^2(x)$, and the boundary of $W_i(z)$ parallel with the side $p_{i-1}p_i$ does not intersect with $W_j^2(x)$.

See Figure 5.16 for an illustration.



(a) (b)

Figure 5.16: Illustration for case 2(b): (a) A type 1 intersection. (b) A type 2 intersection.

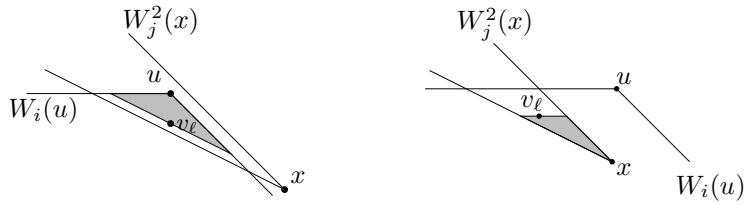
Let $\{v_1, v_2, v_3, \ldots\}$ denote the points in $X_i \cap W_j^2(x)$ in decreasing order according to the partial order $<_i$. Let $\ell = \max\{t | v_t \in Y_i\}$. Consider the value $b_\ell = \sum_{l=1}^{\ell} d(v_l)$. If $b_\ell \leq \frac{L}{10}$ then $\text{load}_{Y'}(W_j^2(x)) \geq \frac{L}{6} - \frac{L}{10} \geq \frac{L}{5\mu}$ after iteration i, so assume that $b_\ell > \frac{L}{10}$.

Since $v_\ell \in X_i$ there is a $u \in \mathcal{C}_i(k)$ so that $v_\ell \in X(u)$ in iteration i of the algorithm. Suppose that the intersection between $W_i(u)$ and $W_j^2(x)$ is a type 1 intersection. Let $T_{v_\ell} = W_i(u) \setminus H_{j-1}(v_\ell)$. Note that since $W_j^2(x) \neq \emptyset$, it must be that $j - 1 \in A_i$. Combining this with the fact that $v_\ell \in X(u)$, we know that $X^{j-1}(u) \subset T_{v_\ell} \subset W_i(u)$. (See Algorithm 5.1 for the notation.) Thus $\text{load}_{Y'}(T_{v_\ell}) \geq \text{load}_{Y'}(X^{j-1}(u)) \geq \frac{L}{2\mu}$. Since, $\text{load}_{Y_i}(W_i(u)) \leq \frac{L}{10\mu}$, there must be at least $\frac{L}{2\mu} - \frac{L}{10\mu} > \frac{L}{5\mu}$ unscheduled points left in T_{v_ℓ} after iteration i. Since we are dealing with a type 1 intersection, $T_{v_\ell} \subset W_j^2(x)$, and thus $W_j^2(x)$ will contain at least $\frac{L}{5\mu}$ unscheduled points after iteration i and the lemma holds. See Figure 5.17 for an illustration.

Now suppose that the intersection between $W_i(u)$ and $W_j^2(x)$ is a type 2 intersection. Consider the region $T'_{v_\ell} = W_j^2(x) \setminus H_i(v_\ell)$. Since we are assuming $b_\ell > \frac{L}{10}$, it must be that $\text{load}_{Y'}(T'_{v_\ell}) \geq \frac{L}{10}$. Since we are dealing with a type 2 intersection, it must be that $T'_{v_\ell} \subset W_i(u)$. Since $\text{load}_{Y_i}(W_i(u)) \leq \frac{L}{10\mu}$, we have that $\text{load}_{Y_i}(T'_{v_\ell}) \leq \frac{L}{10\mu}$ and thus there will be at least $\frac{L}{10} - \frac{L}{10\mu} \geq \frac{L}{5\mu}$ unscheduled points left in T'_{v_ℓ} after iteration i. Since $T'_{v_\ell} \subseteq W_j^2(x)$, there must be (unscheduled) load at least $\frac{L}{5\mu}$ in $W_j^2(x)$ after iteration i. See Figure 5.17 for an illustration.

□

Theorem 25. *There is a polynomial time algorithm that, given an instance of the*



(a) (b)

Figure 5.17: Illustration for case 2(b): (a) An illustration of T_{v_ℓ}. (b) An illustration of T'_{v_ℓ}.

planar sensor problem where the range of each sensor is a translate of a convex, centrally symmetric $2k$-gon Q, returns a schedule of duration at least L/α; here L is the load of the input instance and $\alpha \geq 1$ is a constant that depends only on Q.

Remark 26. *Notice that the constant α in the theorem above is proportional to $\beta_k \cdot k$. We can improve the constant α to be proportional to just β_k by using ideas in the proof of Lemma 22 together with a generalization of a more involved construction of Aloupis et al. [2].*

Algorithm 5.1

1: $Y' \leftarrow Y$

2: **for** each $i \in \{0, 1, 2, \ldots, \mu - 1\}$ **do**

3: $L \leftarrow \min\{\text{load}_{Y'}(W_z(x)) : x \in \mathcal{C}_z(k) \text{ and } z = i, i+1, \ldots, \mu - 1\}$

4: **for** each $c \in \mathcal{C}_i(k)$ **do**

5: **for** each $j \in A_i$ **do**

6: Order the points in $W_i(c) \cap Y'$ in decreasing order with respect to the ordering $<_j$. Let $X^j(c)$ be the smallest subset of these points taken in order such that $\text{load}_{X^j(c)}(W_i(c)) \geq \frac{L}{2\mu}$.

7: **end for**

8: $X(c) \leftarrow \{W_i(c) \cap Y'\} \setminus \bigcup_{j \in A_i} X^j(c)$

9: **end for**

10: $X_i \leftarrow \bigcup_{c \in \mathcal{C}_i(k)} X(c)$

11: Run the RSC algorithm with universe $\mathcal{C}_i(k)$ and sensors with intervals and durations corresponding to X_i until the duration of the schedule becomes $\frac{L}{100\mu}$. Let Y_i denote the sensors that are assigned a start time during this call to the algorithm.

12: Let Y' denote the unscheduled points (that is, $Y' \leftarrow Y' \setminus Y_i$).

13: **end for**

CHAPTER 6
METRIC CLUSTERING TO MINIMIZE THE SUM OF RADII

In this chapter, we consider the metric clustering to minimize the sum of radii problem. Here, we are given a metric (P, d) and a positive integer k, and we want to find a set of at most k balls, each of which must be centered at a point in P, that cover all of the points in P such that the sum of the radii of the balls is minimized. In Section 6.1, we present our algorithm for this problem. In Section 6.2, we establish the NP-hardness of the k-cover problem for metrics induced by weighted planar graphs. In Section 6.3, we establish NP-hardness for metrics of constant doubling dimension.

6.1 Algorithm for General Metrics

We consider the k-cover problem whose input is a metric (P, d), where P is a set of n points and d is a function giving the interpoint distances, and an integer $k > 0$. We assume without loss of generality that the minimum interpoint distance is 1. Let Δ denote $\text{diam}(P)$, the maximum interpoint distance. We present a randomized algorithm that runs in $n^{O(\log n \log \Delta)}$ time and with high probability returns the best k-cover for P. We will assume below that $k \leq n$.

The main idea for handling the metric case is that probabilistic partitions [8, 26] can play a role analogous to the line separators were used in the geometric case [33]. To formalize this, let Q denote some subset of P such that $\text{diam}(Q) \geq 50$, and consider the following randomized algorithm (taken from [26]) that partitions Q into sets of diameter at most $\text{diam}(Q)/2$:

Algorithm 6.1 Partition(Q)

1: Let π denote a random permutation of the points in Q.

2: Let β denote a random number in the range $[\text{diam}(Q)/8, \text{diam}(Q)/4]$.

3: Let $R \leftarrow Q$.

4: **for all** $i \leftarrow 1$ to $|Q|$ **do**

5: Let $Q_i \leftarrow \{p \in R | d(p, \pi(i)) \leq \beta\}$.

6: Let $R \leftarrow R \setminus Q_i$.

7: **end for**

Since each Q_i is contained in a ball of radius at most $\text{diam}(Q)/4$, we have that $\text{diam}(Q_i) \leq \text{diam}(Q)/2$. Clearly, the Q_i also partition Q. Let us say that a ball $B \subseteq P$ is *cut* by this partition of Q if there are two distinct indices i and j such that $(B \cap Q) \cap Q_i \neq \emptyset$ and $(B \cap Q) \cap Q_j \neq \emptyset$. The main property that the probabilistic partition enjoys is encapsulated by the following lemma, whose proof follows via the methods of Fakcharoenphol et al. [26].

Lemma 27. *Let $B \subseteq P$ be some ball of radius r. The probability that B is cut by the partition of Q output by Partition(Q) is at most $\frac{r}{diam(Q)} O(\log |Q|)$.*

Proof. Let $q_1, \ldots q_{|Q|}$ denote the ordering of the points in Q according to increasing order of distance from $B' = B \cap Q$, with ties broken arbitrarily. We may assume that $B' \neq \emptyset$ for otherwise the lemma trivially holds. For each q_j let a_j (resp. b_j) denote the distance to the closest (resp. furthest) point in B'. By the triangle inequality it follows that $b_j - a_j \leq 2r$. We say that $\pi(i)$ *settles* B if i is the first index for which

some point in B' belongs to Q_i. Note that exactly one point in Q settles B. We say that $\pi(i)$ *cuts* B if $\pi(i)$ settles B and at least one point in B' is not assigned to Q_i. The probability that B is cut by the partition equals

$$\sum_i \Pr[\pi(i) \text{ cuts } B] = \sum_j \Pr[q_j \text{ cuts } B].$$

The event that q_j cuts B requires the occurrence of two events: E_1, the event that β lands in the interval $[a_j, b_j)$, and E_2, the event that q_j appears before $q_1, \ldots, q_{j-1}$ in the ordering π. Using independence,

$$\begin{aligned} \Pr[q_j \text{ cuts } B] &\leq \Pr[E_1] * \Pr[E_2|E_1] = \Pr[E_1] * \Pr[E_2] \\ &\leq \frac{2r}{\text{diam}(Q)/8} \cdot \frac{1}{j} = \frac{16r}{\text{diam}(Q)} \cdot \frac{1}{j}. \end{aligned}$$

So the probability that B is cut by the partition is bounded above by

$$\frac{16r}{\text{diam}(Q)} \sum_j \frac{1}{j} = \frac{r}{\text{diam}(Q)} O(\log |Q|).$$

□

Let S denote the optimal κ-cover for Q some $\kappa > 0$. The following states the main structural property that S enjoys.

Lemma 28. *The expected number of balls in S that are cut by Partition(Q) is $O(\log |Q|)$. Consequently, the probability is at least $1/2$ that the number of balls in S that are cut by Partition(Q) is at most $c \log n$, where $c > 0$ is some constant.*

Proof. The expected number of balls in S cut is equal to

$$\sum_{B \in S} \Pr[\text{B is cut}] = O(\log |Q|) \sum_{B \in S} \frac{\text{radius}(B)}{\text{diam}(Q)} = O(\log |Q|) \frac{\text{cost}(S)}{\text{diam}(Q)}.$$

The Lemma follows by observing that $\text{cost}(S) \leq \text{diam}(Q)$ since Q can be covered by a single ball of radius $\text{diam}(Q)$. □

The Randomized Algorithm

We describe a recursive algorithm `BC-Compute` that takes as arguments a set $Q \subseteq P$ and an integer $0 \leq \kappa \leq n$ and returns with high probability an optimal κ-cover for Q. We begin by noting that we may restrict our attention to balls $B(x, r)$ whose radius r equals $d(x, q)$ for some $q \in P$. Henceforth in this section we only refer to this set of balls. For easing the description of the algorithm, it is convenient to add to this set of balls an element $\mathcal{I}$ whose cost is ∞. Any subset of this enlarged set of balls that includes $\mathcal{I}$ will also have a cost of ∞.

Running time. To solve an instance (Q, κ) with $\text{diam}(Q) \geq 50$, the algorithm makes $n^{O(\log n)}$ recursive calls to instances with diameter at most $\text{diam}(Q)/2$. The additional book keeping takes $n^{O(\log n)}$ time. It follows that the running time of the algorithm invoked on the original instance (P, k) is $n^{O(\log n \cdot \log \Delta)}$.

Correctness. We will show that `BC-Compute`(P, k) computes an optimal k-cover for P with high probability. We begin by noting that the base case instances (Q, κ) are solved correctly with a probability of 1. We will show by induction on $|Q|$ that any instance (Q, κ) with $|Q| \geq 2$ is optimally solved with a probability of at least $1 - \frac{|Q|-1}{n^2}$.

If the (Q, κ) instance happens to fit in one of the base cases, we are done. Otherwise, consider an optimal κ-cover OPT for Q. It is enough to show that `BC-Compute`(Q, κ) returns a κ-cover of cost at most cost(OPT) with a probability of at least $1 - \frac{|Q|-1}{n^2}$.

By Lemma 28, the probability is at least $1 - \frac{1}{n^2}$ that one of the $2 \log_2 n$ calls to `Partition`(Q) returns a partition $(Q_1, \ldots, Q_\tau)$ of Q into $\tau \geq 2$ sets such that no more than $c \log n$ balls in OPT are cut by the partition. Assuming this good event happens, fix such a partition $(Q_1, \ldots, Q_\tau)$ of Q and consider the choice of C that exactly equals the balls in OPT that are cut by the partition. The balls in OPT $\setminus\, C$ are not cut by the partition and can be partitioned into subsets $(\mathrm{OPT}_1, \ldots, \mathrm{OPT}_\tau)$ (some of these can be empty) such that for any ball $B \in \mathrm{OPT}_i$, we have $B \cap Q \subseteq Q_i$. It is easy to see that OPT_i must be an optimal $|\mathrm{OPT}_i|$-cover for Q'_i. By the induction hypothesis, `BC-Compute`$(Q'_i, |\mathrm{OPT}_i|)$ returns an $|\mathrm{OPT}_i|$-cover for Q'_i with a probability of at least $1 - \frac{|Q'_i|-1}{n^2}$ if $|Q'_i| \geq 2$ and with a probability of 1 otherwise. The probability that `BC-Compute`$(Q'_i, |\mathrm{OPT}_i|)$ returns an $|\mathrm{OPT}_i|$-cover for Q'_i for every i is at least

$$\prod_{i:|Q'_i|\geq 2} 1 - \frac{|Q'_i| - 1}{n^2} \geq \prod_i 1 - \frac{|Q_i| - 1}{n^2} \geq 1 - \frac{|Q| - 2}{n^2}.$$

Assuming this second good event also happens, it follows from an easy backwards induction on i that $\mathrm{best}(R_i, \sum_{j>i} |\mathrm{OPT}_j|)$ is a $(\sum_{j>i} |\mathrm{OPT}|_j)$-cover for R_i with cost at most $\sum_{j>i} \mathrm{cost}(\mathrm{OPT}_j)$. Thus $\mathrm{best}(R_0, \kappa - |C|)$ is an $(\kappa - |C|)$-cover for $R_0 = \sum_{i=1}^{\tau} Q'_i$ with cost at most $\sum_{i=1}^{\tau} \mathrm{cost}(\mathrm{OPT}_i)$. Thus $\mathrm{best}(R_0, \kappa - |C|) \cup C$ is a κ-cover of Q with cost at most cost(OPT). The probability of this happening is at least the product of the probabilities of the two good events we assumed, which is

at least $(1 - \frac{|Q|-1}{n^2})$. This completes the inductive step, because `BC-Compute`(Q, κ) returns the lowest cost κ-cover among the $2\log_2 n$ κ-covers that it sees.

Theorem 29. *There is a randomized algorithm that, given a set P of n points in a metric space and an integer k, runs in $n^{O(\log n \cdot \log \Delta)}$ time and returns, with probability at least $1/2$, an optimal k-cover for P. Here Δ is an upper bound on the ratio between the maximum and minimum interpoint distances within P.*

6.2 NP-hardness of Min-Cost k-Cover

A natural question is whether there is a quasipolynomial time algorithm in n for the case where the input metric has unbounded aspect ratio. This is unlikely to be the case because, as we show in this section, the general problem is NP-hard even in case of a planar metric. We give a reduction from a version of the planar 3-SAT problem - the *pn-planar* 3-SAT problem. This problem was shown to be NP-complete in [45]. Planar 3-SAT is defined as follows: Let $\Phi = (X, C)$ be an instance of 3SAT, with variable set $X = \{x_0, \ldots, x_{n-1}\}$ and clauses $C = \{c_1, \ldots, c_m\}$ such that each clause consists of exactly 3 literals. Define a *formula graph* $G_\Phi = (V, E)$ with vertex set $V = X \bigcup C$ and edges $E = E_1 \bigcup E_2$ where $E_1 = \{(x_i, x_{i+1}) | 0 \leq i \leq n-1\}$[1] and $E_2 = \{(x_i, c_j) | c_j \text{ contains } x_i \text{ or } \overline{x_i}\}$. A 3SAT formula Φ is called *planar* if the corresponding formula graph G_Φ is planar. The edge set E_1 defines a cycle on the vertices X, and thus divides the plane into exactly 2 faces. Each node $c_j \in C$ lies in exactly one of those two faces. In the *pn-planar* 3SAT problem, we have the additional

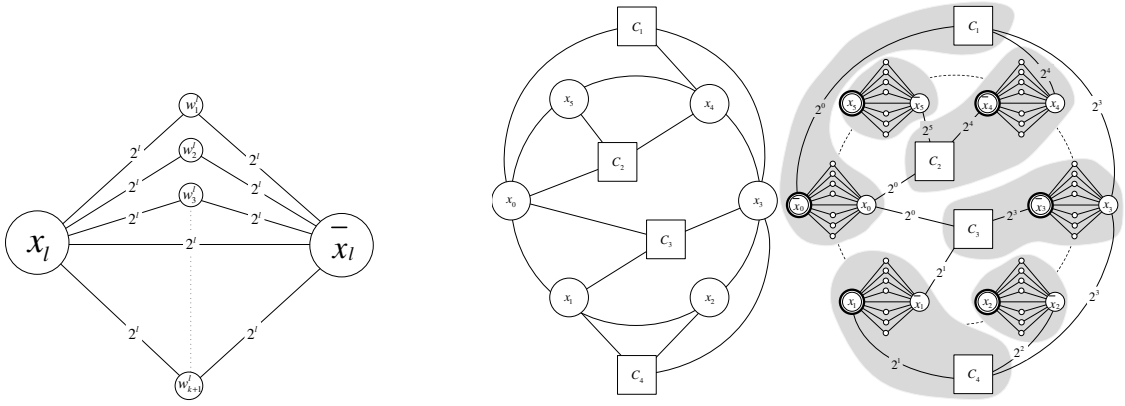
[1]Here we assume that the arithmetic wraps around i.e. $(n-1)+1 = 0$

restriction that there exists a planar drawing of G_Φ such that if c_j and $c_{j'}$ contain opposite occurrences of the same variable x_i, then they lie in opposite faces. In other words, all clauses with the literals x_i lie in one of the two faces and all clauses with $\overline{x_i}$ lie in the other face. We have to determine whether there exists an assignment of truth values to the variables in X that satisfies all the clauses in C.

We describe a simple transformation, easily seen to be effected by a polynomial time algorithm, from such a *pn-planar* 3SAT instance to an instance of finding an optimal k-cover in a metric induced by a weighted planar graph $G = (V, E)$. The transformation has the property that there is a k-cover in the metric of cost at most $2^k - 1$ if and only if the original *pn-planar* 3SAT instance is satisfiable.

We set $k = n$. The vertex set V of the graph is a union of $k + 2$ sets: (a) a set $X = \{x_0, \overline{x_0}, \ldots, x_{k-1}, \overline{x_{k-1}}\}$ that can be identified with the set of variables of the *pn-planar* 3SAT instance with each variable occurring twice - once as a positive literal and once as a negative literal, (b) a set $C = \{c_1, \ldots, c_m\}$ that can be identified with the set of clauses of the *pn-planar* 3SAT instance, and (c) sets $W^0, \ldots, W^{k-1}$, where each W^l consists of $k+1$ vertices. To obtain the edge set E, we add an edge between each vertex x_l and $\overline{x_l}$ in X with weight 2^l for $0 \leq l \leq k-1$. For each vertex $x_l \in X$ we add an edge between x_l and every vertex in W^l of weight 2^l for $0 \leq l \leq k-1$. Analogously, we add an edge between each vertex $\overline{x_l}$ and every vertex in W^l again of weight 2^l. In addition we add edges between every vertex $c_i \in C$ and every variable vertex x_l or its negation $\overline{x_l}$ whichever appears in it of weight 2^l. Note that this graph G is planar – this follows from the pn-planarity of the 3SAT instance. See Figure 6.1

for an illustration.



(a) (b)

Figure 6.1: The construction for the planar metric case. (a) The gadget for variable x_l in Φ. (b) A planar embedding for Φ and construction of the corresponding instance of k-clustering problem. All "clause-literal" edges have weight 2^l for the variable x_l. The optimal cover is highlighted with grey "blobs". $\Phi = (\neg x_0 \vee x_3 \vee x_4) \wedge (x_0 \vee \neg x_4 \vee \neg x_5) \wedge (x_0 \vee \neg x_1 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3)$. Satisfying assignment $X = (0, 1, 1, 0, 0, 1)$. Weight of the covering is exactly $2^6 - 1$

Claim 30. *Any k-cover of V whose cost is at most $2^k - 1$ includes, for each $0 \leq l \leq k-1$, a ball centered at either x_l or $\overline{x_l}$ with radius at least 2^l.*

Proof. Consider any k-cover of V and let t be the largest index such that there is no ball in the k-cover centered at either x_t or $\overline{x_t}$ and having radius at least 2^t. So for each $t+1 \leq l \leq k-1$, there is a ball B_l in the k-cover centered at either x_l or $\overline{x_l}$ and having radius at least 2^l. Since W^t has $k+1$ points in it, there is point $a \in W^t$ that is not the center of any ball in the k-cover. Let B be some ball in the k-cover that covers a. If $B = B_l$ for some $t+1 \leq l \leq k-1$, then B_l has radius at least $2^l + 2 \cdot 2^t$.

In this case the k-cover has cost at least $2^{k-1} + 2^{k-2} \cdots 2^{t+1} + 2 \cdot 2^t = 2^k$. If $B \neq B_l$ for any $t+1 \leq l \leq k-1$, then the radius of B is at least $2 \cdot 2^t$, since the distance of a from any point other than x_t and $\overline{x_t}$ is at least $2 \cdot 2^t$. Thus in this case too the k-cover has cost at least $2^{k-1} + 2^{k-2} \cdots 2^{t+1} + 2 \cdot 2^t = 2^k$. □

Now suppose the original *pn-planar* 3SAT instance is a yes instance. So there is an assignment of truth values to $x_0, \cdots, x_{k-1}$ such that all clauses in C are satisfied. Consider the set of k balls $B_0, \ldots, B_{k-1}$, where B_l is centered at x_l or $\overline{x_l}$ (whichever is satisfied by the assignment) and has radius 2^l. It is easily checked that these balls form a k-cover of V of cost $2^0 + 2^1 + \cdots 2^{k-1} = 2^k - 1$.

Now suppose the original *pn-planar* 3SAT instance is a no instance. We claim that any k-cover of V has cost strictly greater than $2^k - 1$ in this case. Suppose this is not the case and consider a k-cover of cost at most $2^k - 1$. As a consequence of the claim, such a k-cover must consist of balls $B_0, \ldots, B_{k-1}$ where B_l is centered at either x_l or $\overline{x_l}$ and has radius precisely 2^l. Since these balls must cover each vertex in C, it follows that the assignment of truth values to variables in X which comprises of x_l being true if the ball B_l is centered at x_l and false if it is centered at $\overline{x_l}$ satisfies all clauses in C. This contradicts the supposition that the original *pn-planar* 3SAT instance is a no instance.

Theorem 31. *The (decision version of the) problem of computing an optimal k-cover for an n-point planar metric (P, d) is NP-hard.*

6.3 The Doubling Metric Case

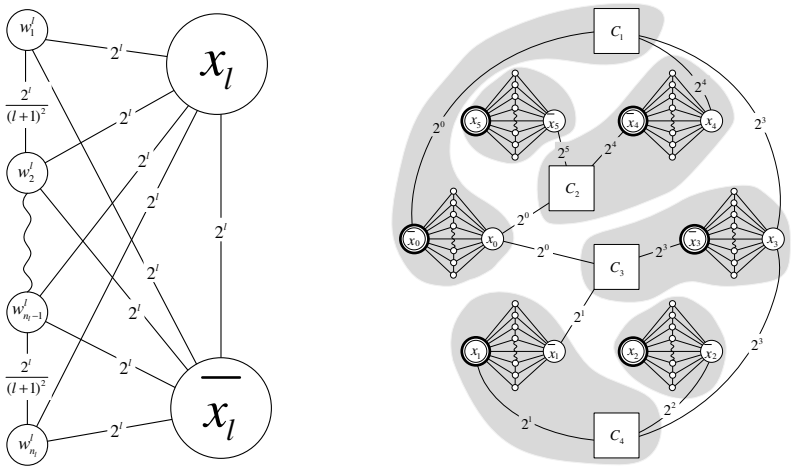
We now consider the k-cover problem when the input metric (P, d) has doubling dimension bounded by some constant $\rho \geq 0$. The doubling dimension of the metric (P, d) is said to be bounded by ρ if any ball $B(x, r)$ in (P, d) can be covered by 2^ρ balls of radius $r/2$ [42]. In this section, we show that for a large enough constant ρ, the k-cover problem for metrics of doubling dimension at most ρ is NP-hard.

The proof is by a reduction from a restricted version of 3SAT where each variable appears in at most 5 clauses [31]. Let Φ be such a 3-CNF formula with variables $x_0, \ldots, x_{n-1}$ and clauses $c_1, \ldots, c_m$. We describe a simple transformation, easily seen to be effected by a polynomial time algorithm, from such a 3SAT instance Φ to an instance of finding an optimal k-cover in a metric induced by a weighted graph $G = (V, E)$. The metric will have doubling dimension bounded by some constant. The transformation has the property that there is a k-cover in the metric of cost at most $2^k - 1$ if and only if the original 3SAT instance is satisfiable.

The transformation is similar to the one in the previous section with some modifications to ensure the doubling dimension property.

We set $k = n$. The vertex set V of the graph is a union of $k + 2$ sets: (a) a set $X = \{x_0, \overline{x_0}, \ldots, x_{k-1}, \overline{x_{k-1}}\}$ that can be identified with the set of literals in Φ, (b) a set $C = \{c_1, \ldots, c_m\}$ that can be identified with the set of clauses of Φ, and (c) sets $W^0, \ldots, W^{k-1}$, where each W^l consists of $n_l = 8(l + 1)^2 + 1$ vertices $w^l_1, \ldots, w^l_{n_l}$. To obtain the edge set E, we add an edge between x_l and $\overline{x_l}$ with weight 2^l for $0 \leq l \leq k - 1$. We add an edge between x_l and every vertex in W^l of weight 2^l

for $0 \leq l \leq k-1$. Analogously, we add an edge between $\overline{x_l}$ and every vertex in W^l again of weight 2^l. In addition we add edges between every vertex $c_i \in C$ and every literal that appears in the clause c_i. If the literal is either x_l or $\overline{x_l}$, the weight of the corresponding edge is 2^l. Finally for each $0 \leq l \leq n-1$ and each $1 \leq i \leq n_l - 1$, we add an edge of weight $2^l/(l+1)^2$ between w_i^l and w_{i+1}^l. See Figure 6.2 for an illustration of the transformation.



(a) (b)

Figure 6.2: The construction for the doubling metric case. (a) The gadget for the variable x_l in Φ. Each edge between w_i^l and w_{i+1}^l has weight exactly $2^l/(l+1)^2$ and the number of w_i^l's is $8(l+1)^2+1$. (b) A representation of an instance of k-clustering on a doubling metric constructed from an instance of Φ. All "clause-literal" edges have weight 2^l for variable x_l. The optimal cover is highlighted with grey"blobs". $\Phi = (\neg x_0 \vee x_3 \vee x_4) \wedge (x_0 \vee \neg x_4 \vee \neg x_5) \wedge (x_0 \vee \neg x_1 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3)$. Satisfying assignment $X = (0,1,1,0,0,1)$. Weight of the covering is exactly $2^6 - 1$

Lemma 32. *There is a constant $\rho \geq 0$ so that the doubling dimension of the metric induced by the graph $G = (V, E)$ is bounded by ρ.*

Proof. Let $B(x,r)$ be some ball in the metric. If $r < 1$, then either (a) the ball consists of a singleton vertex, or (b) $B(x,r) \subseteq W^l$ for some l and the subgraph of G induced by $B(x,r)$ is a path. In either case, it is easily verified that $O(1)$ balls centered within $B(x,r)$ and having radius $r/2$ cover $B(x,r)$.

We therefore consider the case $r \geq 1$. Let t be the largest integer that is at most $n-1$ such that $2^t \leq r$. For each $s \in \{t-3, t-2, t-1, t\}$, we place balls of radius $r/2$ centered at (i) $\{x_s, \overline{x_s}\} \cap B(x,r)$, (ii) clause vertices incident to x_s or $\overline{x_s}$ that are in $B(x,r)$, and (iii) $O(1)$ points of $B(x,r) \cap W^s$ so that these balls cover $B(x,r) \cap W^s$ (this is possible because $B(x,r) \cap W^s$ induces a path of length at most 2^{s+3}.) In addition, if $x \in W^l$ for some l, we place $O(1)$ balls of radius $r/2$ at points of $B(x,r) \cap W^l$ so that these balls cover $B(x,r) \cap W^l$. Finally, we place a ball of radius $r/2$ at x. Clearly, we have placed $O(1)$ balls and we will show that these cover $B(x,r)$. Let C denote the set of centers at which we have placed balls.

Let $y \in B(x,r)$ be a point that is not in C or in W^s for $s \in \{t-3, t-2, t-1, t\}$ or in W^l (if $x \in W^l$). Fix a shortest path from x to y and let x' be the last vertex on this path that is in C. We first observe that none of the internal vertices on the path from x to y is in W^q for any q. Furthermore, if $x \in W^l$ for some l, then by assumption $y \notin W^l$. Thus all edges of the subpath from x' to y have weight 2^q for some $0 \leq q \leq n-1$. No such edge can have weight 2^{t+1} or greater because $2^{t+1} > r$ if $t \leq n-2$. No such edge can have weight 2^s for $s \in \{t-3, t-2, t-1, t\}$ because otherwise the endpoint of the edge closer to y would be in C. Thus every edge on the subpath from x' to y has weight at most 2^{t-4}. It is easy to see that the subpath

contains at most 3 edges of weight 2^q for any $q \leq t-4$. Thus the weight of the subpath from x' to y is at most

$$3(2^{t-4} + 2^{t-5} + \cdots + 2^0) < 3 \cdot 2^{t-3} < 2^{t-1} < r/2.$$

So y is in the ball of radius $r/2$ centered at x'. □

Claim 33. *Any k-cover of V whose cost is at most $2^k - 1$ includes, for each $0 \leq l \leq k-1$, a ball centered at either x_l or $\overline{x_l}$ with radius at least 2^l.*

Proof. Consider any k-cover of V and let t be the largest index such that there is no ball in the k-cover centered at either x_t or $\overline{x_t}$ and having radius at least 2^t. So for each $t+1 \leq l \leq k-1$, there is a ball B_l in the k-cover centered at either x_l or $\overline{x_l}$ and having radius at least 2^l.

If some point in W^t is covered by some B_l for $t+1 \leq l \leq k-1$, then B_l has radius at least $2^l + 2 \cdot 2^t$. In this case the k-cover has cost at least $2^{k-1} + 2^{k-2} \cdots 2^{t+1} + 2 \cdot 2^t = 2^k$. If some point in W^t is covered by a ball B different from the B_l's and not centered at any of the points in W^t, then the radius of B is at least $2 \cdot 2^t$. (Note that by assumption B can't be centered at x_t or $\overline{x_t}$.) Thus in this case too the k-cover has cost at least $2^{k-1} + 2^{k-2} \cdots 2^{t+1} + 2 \cdot 2^t = 2^k$.

The only remaining case is when each point in W^t is covered by some ball centered at a point in W^t. Since there can be at most $t+1$ balls in the k-cover centered within W^t, the sum of the radii of these balls is at least

$$\frac{1}{2}\left((n_t - 1)\frac{2^t}{(t+1)^2} - (t+1)\frac{2^t}{(t+1)^2}\right) > 2 \cdot 2^t.$$

The k-cover has cost at least $2^{k-1} + 2^{k-2} \cdots 2^{t+1} + 2 \cdot 2^t = 2^k$. □

We now argue that the transformation has the property that there is a k-cover in the metric of cost at most $2^k - 1$ if and only if the original 3SAT instance Φ is satisfiable.

Suppose that Φ is satisfiable. Then we can choose for each $0 \leq l \leq k-1$ exactly one of x_l or $\overline{x_l}$ such that within each clause of Φ there is a chosen literal. Consider the set of k balls $B_0, \ldots, B_{k-1}$ where B_l has radius 2^l and is centered at x_l or $\overline{x_l}$, whichever was chosen. These balls form a k-cover of V with cost $2^k - 1$.

For the reverse direction, consider a k-cover of the target metric space of cost at most $2^k - 1$. It follows from Claim 33 that the k-cover must consist of balls $B_0, \ldots, B_{k-1}$, where B_l is centered at either x_l or $\overline{x_l}$ and has radius precisely 2^l. Let us choose the literals corresponding to the centers of these balls. For each l, we clearly choose exactly one of x_l of $\overline{x_l}$. Consider any clause vertex c. It must be covered by at least one of the balls B_l. Given the radii of the balls, the only balls that can cover c are the ones centered at literals contained in the clause. It follows that our set of chosen literals contains, for each clause in Φ, at least one of the literals contained in the clause. Thus Φ is satisfiable.

Theorem 34. *For a large enough constant $\rho \geq 0$, the (decision version of the) k-cover problem for metrics of doubling dimension at most ρ is NP-hard.*

Algorithm 6.2 `BC-Compute`(Q, κ)

If $|Q| = 0$, return the empty set.

Otherwise, if $\kappa = 0$, return $\{\mathcal{I}\}$ (not possible to cover).

Otherwise, if $\text{diam}(Q) \leq 50$, directly compute the optimal solution in polynomial time.

for all $2 \log_2 n$ iterations **do**

  Call `Partition`(Q) to obtain a partition of Q into two or more sets. Let $Q_1, \ldots, Q_\tau$ denote the nonempty sets in this collection.

  for all sets C of at most $c \log n$ balls, where c is the constant in Lemma 28 **do**

    Let Q'_i be the points in Q_i not covered by C. For each $1 \leq i \leq \tau$ and $0 \leq \kappa_1 \leq \kappa$, recursively call `BC-Compute`$(Q'_i, \kappa_1)$ and store the set returned in the local variable $\text{best}(Q'_i, \kappa_1)$.

    For $0 \leq i \leq \tau - 1$, let $R_i = \cup_{j=i+1}^{\tau} Q'_j$. Note that $R_{\tau-1} = Q'_\tau$ and $R_i = Q'_{i+1} \cup R_{i+1}$ for $0 \leq i \leq \tau - 2$.

    for all $i \leftarrow \tau - 2$ down to 0 and $0 \leq \kappa_1 \leq \kappa$, **do**

      set local variable $\text{best}(R_i, \kappa_1)$ to be the lowest cost solution among $\{\text{best}(Q'_{i+1}, \kappa') \cup \text{best}(R_{i+1}, \kappa_1 - \kappa') | 0 \leq \kappa' \leq \kappa_1\}$.

      Let S denote the lowest cost solution $\text{best}(R_0, \kappa - |C|) \cup C$ over all choices of C tried above with $|C| \leq \kappa$.

    end for

  end for

end for

Return the lowest cost solution S obtained over the $\Theta(\log n)$ trials.

CHAPTER 7
DOMINATING SET FOR DISK GRAPHS

In this chapter, we give our PTAS for minimum dominating set for disk graphs. Here, we are given a disk graph with a set $\mathcal{D}$ of n disks in the Euclidean plane, and we are interested in computing a minimum cardinality dominating set of the disk graph. The algorithm is given in Section 7.1 and the analysis of the approximation ratio is given in Section 7.2.

7.1 The Algorithm

Local Search. Call a subset of disks, $B \subseteq \mathcal{D}$, b-locally optimal if one cannot obtain a smaller dominating set by removing a subset $X \subseteq B$ of size at most b from B and replacing that with a subset of size at most $|X| - 1$ from $\mathcal{D} \setminus B$. Our algorithm will compute a b-locally optimal set of disks for $b = \frac{c}{\epsilon^2}$ where $c > 0$ is a large enough constant. Our algorithm simply begins with an arbitrary feasible set of disks (e.g. every disk in $\mathcal{D}$), and proceeds by making small local exchanges of size $b = O(\frac{1}{\epsilon^2})$, for a given $\epsilon > 0$. We stop when no further local improvements are possible.

Suppose that the solution returned is B. Finally, for reasons apparent in the analysis, we check to see if for any disk $u \in B$ there is a disk $v \in \mathcal{D}$ such that u is completely contained in $v \in \mathcal{D} \setminus B$. If such a disk exists, then simply replace u with v. We return this as our final solution and call it B.

Running Time. We will now show that the running time can be bounded by a polynomial in n. The number of swaps that the local search algorithm will make is at most n, because there are n disks and each swap strictly decreases the number of disks in the solution. For each swap, we need to check every subset of disks of size at most b which can be done in time $O(n^b)$. Recall that b is only a function of ϵ, and thus we can have the exponential dependence on b in the running time.

So we will make at most n swaps, each of which takes time $O(n^b)$. Clearly, the last step (where we check to see if a disk is contained within another) can be done in time polynomial in n, and therefore the entire running time of the algorithm is efficient with respect to n.

7.2 Approximation Ratio

In this section, we will show that our algorithm is a PTAS, thus proving the following theorem:

Theorem 35. *For any $\epsilon > 0$, there exists a polynomial time algorithm for the minimum dominating set problem on disk graphs that returns a solution whose cost is at most $(1+\epsilon)OPT$ where OPT is the cost of an optimal solution.*

Let R be the disks in an optimal solution such that no disk in R is contained by any other disk in $\mathcal{D}$ (clearly such an R must exist), and let B be the disks returned by our local search algorithm. Note that by the definition of PTAS, we need to show that $|B| \leq (1+\epsilon) \cdot |R|$. We will refer to R as the set of red disks and B as the set of blue disks. Without loss of generality, we will assume that $R \cap B = \emptyset$, i.e. there

is no disk that is both red and blue. For a disk $u \in \mathcal{D}$, we say a disk $v \in R \cup B$ is a *dominator of* u if u and v intersect. Similarly, we also say that v *dominates* u.

We must show the existence of an appropriate planar graph which relates the disks in R with the disks in B. The following lemma is the key contribution of this work:

Lemma 36. *There exists a planar graph with vertex set $R \cup B$, such that for every $d \in \mathcal{D}$, there is a disk u from amongst the red dominators of d and a disk v amongst the blue dominators of d such that $\{u, v\}$ is an edge in the graph.*

Because we only need to show the *existence* of the graph and do not actually need to construct it, we do not need to know which disks are in R. Section 7.2.1 is devoted to a proof of Lemma 36. In Section 7.2.2, we describe the argument (from [49]) that uses the lemma to show that $|B| < (1 + \epsilon)|R|$.

7.2.1 Proof of Lemma 36

In this subsection, we will show the existence of the appropriate planar graph and prove Lemma 36.

We first introduce a diagram which is commonly used in computational geometry called the *Voronoi diagram*. Given a set S of input points in the plane, the Voronoi diagram is a partitioning of the plane into cells with the following properties:

- There is exactly one $s \in S$ inside of each cell. We call such a cell $\mathbf{cell}(s)$.
- For each $x \in \mathbf{cell}(s)$, x is closer to s than it is to any other $s' \in S$.

An exception occurs if a point x lies on the boundary of two or more cells. In such a case, x is equidistant to all points in S whose cells share the boundary that x lies on. See Figure 7.1 for an illustration.

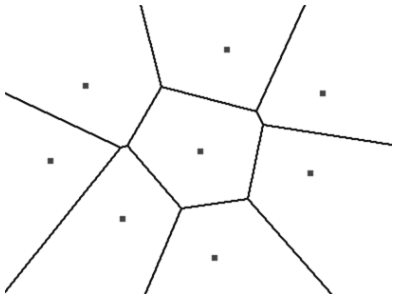


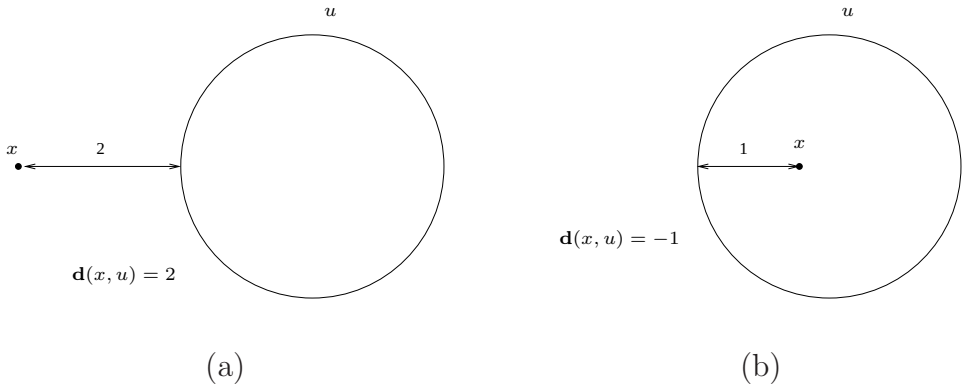
Figure 7.1: An example of a Voronoi diagram.

Weighted Voronoi Diagram. We will be using a generalization of Voronoi diagrams called a *weighted Voronoi Diagram* (WVD). Instead of defining cells with respect to a set of points, we will be defining cells with respect to red and blue disks. Recall that for Voronoi diagrams, all of the points in **cell**(s) were closer to s than to any other s'. In order to do this generalization for disks, we must define the distance between a point in the plane and a disk. Once these distances are defined, then defining the cells for the disks is similar to how they were formed in unweighted Voronoi diagrams.

These distances can be viewed as a special case as WVD's with respect to points [7] where the points are the centers of the disks and the weight of a point is the radius of the corresponding disk. And so the distance between a point and a disk

is the distance to the center of the disk minus the radius of the disk.

Let u be a disk and let x be a point in the plane. We define $\mathbf{d}(x, u) = d(x, c_u) - r_u$ where c_u is the center of u, r_u is the radius of u, and $d(x, c_u)$ is the Euclidean distance between x and c_u. Intuitively, for a point x, $\mathbf{d}(x, u)$ is the Euclidean distance from x to the boundary of u; the distance to a disk is negative for points that are strictly inside the disk. Alternatively, if $x \notin u$, then $\mathbf{d}(x, u)$ is the amount we would need to increase the radius of u so that x lies on the boundary of u; if $x \in u$, then $\mathbf{d}(x, u)$ is the negative of the amount we would need to decrease the radius of u so that x lies on the boundary of u. See Figure 7.2 for an illustration.



(a) (b)

Figure 7.2: An illustration for the distances used in our WVD. (a) $\mathbf{d}(x, u)$ when x is not in u. (b) $\mathbf{d}(x, u)$ when x is in u.

For a disk u in any collection of disks, let $\mathbf{cell}(u)$ be the set of points x in the plane such that $\mathbf{d}(x, u) \leq \mathbf{d}(x, v), u \neq v$. The resulting partition of the plane is the WVD. The cells in a WVD are star-shaped. That is, there is a point $x \in \mathbf{cell}(u)$ such that for every point $y \in cell(u)$, $\overline{xy}$ is completely contained within $\mathbf{cell}(u)$ (the point

x is the center of the disk u).

The Graph. Consider the WVD using disks in $R \cup B$. Our graph is simply the dual of the this diagram. That is, for each cell in the WVD there is a vertex, and there is an edge between two vertices if and only if their corresponding cells share a boundary in the diagram. The graph is clearly planar; the edges can easily be drawn in a way where no two edges intersect [7]. Therefore, all that remains to prove Lemma 36 is to show that for every $d \in \mathcal{D}$, there is a disk u from amongst the red dominators of d and a disk v amongst the blue dominators of d such that $\mathbf{cell}(u)$ and $\mathbf{cell}(v)$ share a boundary in the WVD (and hence their corresponding vertices in the graph have an edge connecting them).

First, we will show that for every $u \in R \cup B$, u has a non-empty cell in the WVD. That is, we will show that there is some point in the plane that is closer to u than it is to any other red or blue disk.

Lemma 37. *In the weighted Voronoi diagram of the union of red and blue disks, the cell of every disk u is nonempty. Moreover, c_u (the center of u) belongs only to* $\boldsymbol{cell}(u)$.

Proof. We first will show that c_u is in $\mathbf{cell}(u)$. Suppose for the sake of contradiction that $c_u \in \mathbf{cell}(v)$ such that $u \neq v$. This means that $\mathbf{d}(c_u, v) < \mathbf{d}(c_u, u) = d(c_u, c_u) - r_u = -r_u$. So, $-r_u > \mathbf{d}(c_u, v) = d(c_u, c_v) - r_v \Rightarrow r_v > d(c_u, c_v) + r_u$. This implies that v completely contains u, but we can show that for any two disks in $R \cup B$, it is not possible for one disk to contain the other.

First note that for two disks $u, v \in R$, it cannot be the case that u contains v or that v contains u. This would contradict the fact that R is an optimal solution, because the contained disk could be eliminated without leaving any other disks uncovered. Likewise, for two disks $u, v \in B$, it cannot be the case that u contains v or that v contains u. The local search algorithm would drop the disk that is contained within the other disk. Finally, no disk in R can contain a disk in B and vice versa because of the final replacement step of the algorithm and the fact that no disk in R is contained in any other disk in $\mathcal{D}$.

Since it is not possible for two disks to contain each other, it must be the case that $c_u \in \mathbf{cell}(u)$ for each $u \in R \cup B$, and therefore each disk u has a nonempty cell in the WVD.

Now we will show that c_u is only in $\mathbf{cell}(u)$. Since we already know $c_u \in \mathbf{cell}(u)$, it suffices to show that c_u is not on the boundary of $\mathbf{cell}(u)$. Suppose for the sake of contradiction that c_u is on the boundary of $\mathbf{cell}(u)$ and $\mathbf{cell}(v)$. We have $\mathbf{d}(c_u, v) = \mathbf{d}(c_u, u) = -r_u$. So, $-r_u = d(c_u, c_v) - r_v$ which gives us that $r_v = d(c_u, c_v) + r_u$ and thus v completely contains u. Again, this is a contradiction, and therefore it must be that c_u is only in $\mathbf{cell}(u)$. □

Because every red and blue disk has a nonempty cell in the WVD, every such disk will also have a corresponding vertex in our planar graph. We are now ready to show that for each $d \in \mathcal{D}$, there is a disk u from amongst the red dominators of d and a disk v amongst the blue dominators of d such that $\mathbf{cell}(u)$ and $\mathbf{cell}(v)$ share a boundary in the WVD. This would then imply that their corresponding vertices in

the graph share an edge, completing the proof of Lemma 36. For simplicity, if there is an edge connecting the vertex corresponding to $\mathbf{cell}(u)$ and the vertex corresponding to $\mathbf{cell}(v)$, then we will simply say there is an edge connecting u and v.

Lemma 38. *In the dual graph of the weighted Voronoi diagram for $R \cup B$, for an arbitrary input disk $u \in \mathcal{D}$, there is an edge between some red dominator of u and some blue dominator of u.*

Proof. Consider the WVD of $R \cup B$. Without loss of generality, assume $c_u \in \mathbf{cell}(r)$ for some $r \in R$. It must be the case that r is a dominator of u, because r is the closest disk in $R \cup B$ to c_u. If r does not dominate u, u is not dominated by any disk in $R \cup B$ which contradicts the fact that both R and B are dominating sets.

Let b denote a closest blue disk to c_u, that is $\mathbf{d}(c_u, b) \leq \mathbf{d}(c_u, b')$ for all other blue disks b'. Note that b must dominate u, because if it did not, then no blue disks would dominate u. This would contradict the fact that B is a dominating set. Also, note that for any disk $d \in \mathcal{D}$ such that $\mathbf{d}(c_u, d) \leq \mathbf{d}(c_u, b)$, d must intersect with u.

We will walk from c_u to c_b along the straight line segment $\overline{c_u c_b}$. The proof strategy is that during this walk, we will be crossing red cells and at some point before reaching c_b we will enter a blue cell, in particular, $\mathbf{cell}(b)$. We must have entered this cell from a red cell $\mathbf{cell}(r')$ which shares a boundary with $\mathbf{cell}(b)$, and thus $\{r', b\}$ is an edge in our planar graph. Moreover, we will argue that r' necessarily dominates u, completing the proof.

As seen in the proof of Lemma 37, $c_b \in \mathbf{cell}(b)$, and thus we will enter $\mathbf{cell}(b)$ at some point in time along our walk from c_u to c_b. Let x be the point at which

we first enter **cell**(b). Then x is on the boundary of **cell**(b) and **cell**(r') for some $r' \in R \cup B$. If $c_{r'}$ is on the line through c_u and c_b, then it must be that either r' contains b or b contains r', a contradiction. So now assume that $c_{r'}$ is not on the line through c_u and c_b. Then we have

$$\mathbf{d}(c_u, r') < d(c_u, x) + \mathbf{d}(x, r') = d(c_u, x) + \mathbf{d}(x, b) = \mathbf{d}(c_u, b)$$

Note that it must be the case that $r' \in R$ because $\mathbf{d}(c_u, r') < \mathbf{d}(c_u, b)$ and b is the closest blue disk to c_u. This also implies that r' must dominate u. See Figure 7.3 for an illustration.

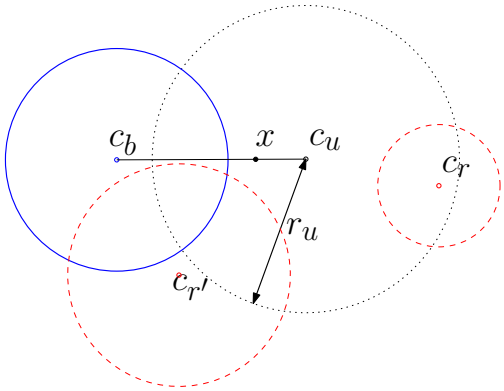


Figure 7.3: Proof of Lemma 38. The dotted disk is u with center c_u and radius r_u. The two red disks r and r' are shown as dashed disks with centers c_r and $c_{r'}$, respectively. The only blue disk b is shown as a solid disk with center c_b.

Therefore **cell**(b) and **cell**(r') share a boundary implying that the edge $\{b, r'\}$ is in our graph. Moreover, b is blue, r' is red, and both dominate u, which completes the proof. □

So we have a planar graph such that for every $d \in \mathcal{D}$, there is a disk u from amongst the red dominators of d and a disk v amongst the blue dominators of d such that $\{u, v\}$ is an edge in the graph, completing the proof of Lemma 36.

7.2.2 Proof of Theorem 35

To show $|B| \leq (1+\epsilon) \cdot |R|$, we make use of the planar graph separator theorem of Frederickson [29]. This argument is similar to the work in [14, 49] and is only given here for completeness. Given a graph $G = (V, E)$, we denote $N(V')$ for subset of the vertices V' to be the set of all vertices in V that share an edge with a vertex in V'.

Theorem 39 (Frederickson [29])**.** *There are constants $c_1, c_2, c_3 > 0$, such that for any planar graph $G = (V, E)$ with n vertices and a parameter $r \geq 1$, there is a set $X \subseteq V$ of size at most $c_1 n/\sqrt{r}$, and a partition of $V \setminus X$ into n/r sets $V_1, V_2, \ldots, V_{n/r}$, satisfying: (i) $|V_i| \leq c_2 r$, (ii) $N(V_i) \cap V_j = \emptyset$, for $i \neq j$, and (iii) $|N(V_i) \cap X| \leq c_3\sqrt{r}$.*

We will now show that $|B| \leq (1+\epsilon)|R|$; this is similar to [14, 49, 34]. Let $r \equiv b/(c_2 + c_3)$ (where b is the parameter from the local search algorithm). From Theorem 39(i),(iii), we get $|V_i \cup N(V_i)| \leq c_2 r + c_3\sqrt{r} \leq b$. Let $R_i = R \cap V_i$ and $B_i = B \cap V_i$. Due to the optimality of local search, we must have $|B_i| \leq |R_i| + |N(V_i)|$, otherwise local search can replace B_i with $R_i \cup N(V_i)$ to obtain a smaller dominating set, contradicting the local optimality of local search. This is why we require that for each $d \in \mathcal{D}$, there is a red dominator of d and a blue dominator of d with an edge in the graph. If there were no such edge, then making this swap could possibly leave

some disks without a dominator. So now we have,

$$\begin{aligned} |B| &\leq |X| + \sum_i |B_i| \leq |X| + \sum_i |R_i| + \sum_i |N(V_i)| \leq |R| + c\frac{|R|+|B|}{\sqrt{r}} \\ &\leq |R| + c'\frac{|R|+|B|}{\sqrt{b}}, \end{aligned}$$

where c and c' are positive constants. With b a large enough constant times $1/\epsilon^2$, it follows that $|B| \leq (1+\epsilon)|R|$.

7.3 Conclusion

We show that a local search algorithm is a PTAS for the minimum dominating set problem for disk graphs by proving that the dual of a weighted Voronoi diagram appropriately relates the optimal solution with a locally optimal solution. This is not the first instance in which the dual of a Voronoi diagram has led to improved algorithms. The construction of our graph is similar to the graph that Mustafa and Ray use for the *hitting set* problem for disks. In this problem, we are given a set of disks and a set of points in the plane such that each disk contains at least one of the points. The problem is to compute a minimum cardinality subset of the points so that each disk contains a point from the subset. The graph that they use to relate the optimal solution with a locally optimal solution is the *Delaunay triangulation* of the point set. The Delaunay triangulation of a point set is simply the dual of the Voronoi diagram of the point set.

CHAPTER 8
CONCLUSION AND OPEN PROBLEMS

In this document, we have presented several algorithms for geometric special cases of the set cover problem. Each problem has is unique in what makes it difficult and how geometry can be used to overcome these difficulties. That being said, there is much work that is left to be done in this area. We conclude the document with a discussion of open problems related to our work.

8.1 Decomposing Coverings

Our result for convex polygons and the work of Pálvölgyi [53] for concave polygons settles the problem for translates of polygons. The major open problem is to obtain results for disks, even in the case when all of the disks have unit radius. The techniques for obtaining $\Omega(k)$ covers for convex polygons involve reducing the problem to coloring points inside of wedges. We are able to understand the complications involving the interactions of the wedges, and we are able to color the points in a clever way to obtain our result. This largely has to do with the fact that there are only a constant number of "types" of wedges; however, we do not seem to have the tools to handle the interactions of the disks.

8.2 The Sensor Cover Problem

We give the first constant factor approximations for the Restricted Strip Cover problem and for the Planar Sensor Cover problem. Recall that our algorithm com-

putes a schedule of duration at least $\frac{L}{5}$ where L is the load of the problem and thus is also at least $\frac{OPT}{5}$ where OPT is the duration of an optimal schedule. Buchsbaum et al. showed that RSC is NP-hard; however, a PTAS has not been ruled out. That being said, the example in Figure 1.4 rules out the possibility of computing a schedule of duration $\frac{L}{1+\epsilon}$, and thus any PTAS for RSC must be with respect to OPT.

The status of the Planar Sensor Cover problem is essentially identical to the Restricted Strip Cover problem. We give a constant factor approximation, the problem is NP-hard, and the possibility of a PTAS has not been ruled out. The example for RSC in Figure 1.4 can easily be extended into an example in the plane to show that any PTAS for Planar Sensor Cover cannot be with respect to L.

8.3 Clustering to Minimize the Sum of Radii

Our work on metric clustering to minimize the sum of radii and the work of Gibson et al. [33] for the geometric version of the problem are very interesting from a theoretical perspective. The problem seems to be extremely similar to many other variants of clustering which are NP-hard, yet the geometric version can be solved exactly in polynomial time and the metric version can be solved exactly in quasi-polynomial time when the aspect ratio is a polynomial in the number of input points. While these results have great theoretical impact, the running times are too large to be of practical interest. It would be interesting to see if the separability properties of the problem that provide the structure to develop interesting theoretical algorithms can lead to the possibility of exact algorithms with practical running times.

8.4 Dominating Set for Disk Graphs

We show that a simple local search algorithm is a PTAS for the minimum dominating set problem for disk graphs. It would be interesting to get sublogarithmic factor approximation algorithms for minimum dominating set for ball graphs in three dimensions. Our planar result is influenced by the results of Chan and Har-Peled [14] and Mustafa and Ray [49] who use separator theorems for planar graphs to show that local search is a PTAS for certain problems if there exists a planar graph which appropriately relates the optimal solution with a locally optimal solution. The requirement that the graph must be planar is fairly restrictive, so it would be very interesting to see if similar results could be obtained for a less-restrictive class of graphs.

REFERENCES

[1] Zoë Abrams, Ashish Goel, and Serge Plotkin. Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In *IPSN '04: Proceedings of the third international symposium on Information processing in sensor networks*, pages 424–432, New York, NY, USA, 2004. ACM.

[2] Greg Aloupis, Jean Cardinal, Sébastien Collette, Stefan Langerman, David Orden, and Pedro Ramos. Decomposition of multiple coverings into more parts. In *SODA '09: Proceedings of the Nineteenth Annual ACM -SIAM Symposium on Discrete Algorithms*, pages 302–310, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[3] Greg Aloupis, Jean Cardinal, Sébastien Collette, Stefan Langerman, and Shakhar Smorodinsky. Coloring geometric range spaces. *Discrete Comput. Geom.*, 41(2):348–362, 2009.

[4] Helmut Alt, Esther M. Arkin, Hervé Brönnimann, Jeff Erickson, Sándor P. Fekete, Christian Knauer, Jonathan Lenchner, Joseph S. B. Mitchell, and Kim Whittlesey. Minimum-cost coverage of point sets by disks. In Nina Amenta and Otfried Cheong, editors, *Symposium on Computational Geometry*, pages 449–458. ACM, 2006.

[5] Christoph Ambühl, Thomas Erlebach, Matús Mihalák, and Marc Nunkesser. Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs. In Josep Díaz, Klaus Jansen, José D. P. Rolim, and Uri Zwick, editors, *APPROX-RANDOM*, volume 4110 of *Lecture Notes in Computer Science*, pages 3–14. Springer, 2006.

[6] Boris Aronov, Esther Ezra, and Micha Sharir. Small-size epsilon-nets for axis-parallel rectangles and boxes. In Michael Mitzenmacher, editor, *STOC*, pages 639–648. ACM, 2009.

[7] Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, 1991.

[8] Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *FOCS*, pages 184–193, 1996.

[9] Mihir Bellare, Shafi Goldwasser, Carsten Lund, and Alexander Russeli. Efficient probabilistically checkable proofs and applications to approximations. In *STOC*, pages 294–304, 1993.

[10] Vittorio Bilò, Ioannis Caragiannis, Christos Kaklamanis, and Panagiotis Kanellopoulos. Geometric clustering to minimize the sum of cluster sizes. In Gerth Stølting Brodal and Stefano Leonardi, editors, *ESA*, volume 3669 of *Lecture Notes in Computer Science*, pages 460–471. Springer, 2005.

[11] Adam L. Buchsbaum, Alon Efrat, Shaili Jain, Suresh Venkatasubramanian, and Ke Yi. Restricted strip covering and the sensor cover problem. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1056–1063, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[12] Adam L. Buchsbaum, Howard J. Karloff, Claire Kenyon, Nick Reingold, and Mikkel Thorup. Opt versus load in dynamic storage allocation. *SIAM J. Comput.*, 33(3):632–646, 2004.

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

[14] Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. In Hershberger and Fogel [36], pages 333–340.

[15] Moses Charikar and Rina Panigrahy. Clustering to minimize the sum of cluster diameters. *J. Comput. Syst. Sci.*, 68(2):417–441, 2004.

[16] Miroslav Chlebík and Janka Chlebíková. Approximation hardness of dominating set problems. In Susanne Albers and Tomasz Radzik, editors, *ESA*, volume 3221 of *Lecture Notes in Computer Science*, pages 192–203. Springer, 2004.

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

[18] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990.

[19] Sridhar Dasika, Sarma Vrudhula, Kaviraj Chopra, and R. Srinivasan. A framework for battery-aware sensor management. In *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, page 20962, Washington, DC, USA, 2004. IEEE Computer Society.

[20] Wenceslas Fernandez de la Vega and Claire Kenyon. A randomized approximation scheme for metric max-cut. *J. Comput. Syst. Sci.*, 63(4):531–541, 2001.

[21] Amol Deshpande, Samir Khuller, Azarakhsh Malekian, and Mohammed Toossi. Energy efficient monitoring in sensor networks. In Laber et al. [43], pages 436–448.

[22] Irit Dinur, Klaus Jansen, Joseph Naor, and José D. P. Rolim, editors. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21-23, 2009. Proceedings*, volume 5687 of *Lecture Notes in Computer Science*. Springer, 2009.

[23] Srinivas Doddi, Madhav V. Marathe, S. S. Ravi, David Scot Taylor, and Peter Widmayer. Approximation algorithms for clustering to minimize the sum of diameters. *Nord. J. Comput.*, 7(3):185–203, 2000.

[24] Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM J. Comput.*, 34(6):1302–1323, 2005.

[25] Thomas Erlebach and Erik Jan van Leeuwen. Domination in geometric intersection graphs. In Laber et al. [43], pages 747–758.

[26] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *STOC*, pages 448–455. ACM, 2003.

[27] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[28] Uriel Feige, Magnús M. Halldórsson, Guy Kortsarz, and Aravind Srinivasan. Approximating the domatic number. *SIAM J. Comput.*, 32(1):172–195, 2003.

[29] Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM J. Comput.*, 16(6):1004–1022, 1987.

[30] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[31] M.R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[32] Jordan Gergov. Algorithms for compile-time memory optimization. In *SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 907–908, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[33] Matt Gibson, Gaurav Kanade, Erik Krohn, Imran A. Pirwani, and Kasturi R. Varadarajan. On clustering to minimize the sum of radii. In Shang-Teng Huang, editor, *SODA*, pages 819–825. SIAM, 2008.

[34] Matt Gibson, Gaurav Kanade, Erik Krohn, and Kasturi R. Varadarajan. An approximation scheme for terrain guarding. In Dinur et al. [22], pages 140–148.

[35] Sariel Har-Peled. Being fat and friendly is not enough. *CoRR*, abs/0908.2369, 2009.

[36] John Hershberger and Efi Fogel, editors. *Proceedings of the 25th ACM Symposium on Computational Geometry, Aarhus, Denmark, June 8-10, 2009*. ACM, 2009.

[37] Dorit S. Hochbaum and David B. Shmoys. A best possible approximation algorithm for the k-center problem. *Math. Oper. Res.*, 10:180–184, 1985.

[38] Harry B. Hunt III, Madhav V. Marathe, Venkatesh Radhakrishnan, S. S. Ravi, Daniel J. Rosenkrantz, and Richard Edwin Stearns. Nc-approximation schemes for np- and pspace-hard problems for geometric graphs. *J. Algorithms*, 26(2):238–274, 1998.

[39] David S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9(3):256–278, 1974.

[40] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. part II: The p-medians. *SIAM J. Appl. Math.*, 37:539–560, 1982.

[41] Claire Kenyon and Eric Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Math. Oper. Res.*, 25(4):645–656, 2000.

[42] Robert Krauthgamer and James R. Lee. Navigating nets: simple algorithms for proximity search. In J. Ian Munro, editor, *SODA*, pages 798–807. SIAM, 2004.

[43] Eduardo Sany Laber, Claudson F. Bornstein, Loana Tito Nogueira, and Luerbio Faria, editors. *LATIN 2008: Theoretical Informatics, 8th Latin American Symposium, Búzios, Brazil, April 7-11, 2008, Proceedings*, volume 4957 of *Lecture Notes in Computer Science*. Springer, 2008.

[44] Nissan Lev-Tov and David Peleg. Polynomial time approximation schemes for base station coverage with minimum total radii. *Computer Networks*, 47(4):489–501, 2005.

[45] David Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[46] László Lovász. On the ratio of the optimal integeral and fractional covers. *Disc. Math.*, 13:383–390, 1975.

[47] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994.

[48] P. Mani and János Pach. Decomposition problems for multiple coverings with unit balls. *Manuscript*, 1986.

[49] Nabil H. Mustafa and Saurabh Ray. Ptas for geometric hitting set problems via local search. In Hershberger and Fogel [36], pages 17–22.

[50] Tim Nieberg, Johann Hurink, and Walter Kern. Approximation schemes for wireless networks. *ACM Transactions on Algorithms*, 4(4), 2008.

[51] János Pach. Covering the plane with convex polygons. *Discrete & Computational Geometry*, 1:73–81, 1986.

[52] János Pach and Géza Tóth. Decomposition of multiple coverings into many parts. *Comput. Geom.*, 42(2):127–133, 2009.

[53] Dömötör Pálvölgyi. Indecomposable coverings with concave polygons. *Discrete and Computational Geometry*, 2009.

[54] Dömötör Pálvölgyi and Géza Tóth. Convex polygons are cover-decomposable. *Discrete and Computational Geometry*, 2008.

[55] Saurav Pandit, Sriram V. Pemmaraju, and Kasturi R. Varadarajan. Approximation algorithms for domatic partitions of unit disk graphs. In Dinur et al. [22], pages 312–325.

[56] Sriram V. Pemmaraju and Imran A. Pirwani. Energy conservation via domatic partitions. In *MobiHoc '06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 143–154, New York, NY, USA, 2006. ACM.

[57] Mark A. Perillo and Wendi B. Heinzelman. Optimal sensor management under energy and reliability constraints. *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, 3:1621–1626 vol.3, 16-20 March 2003.

[58] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *STOC*, pages 475–484, 1997.

[59] Sasha Slijepcevic and Miodrag Potkonjak. Power efficient organization of wireless sensor networks. *Communications, 2001. ICC 2001. IEEE International Conference on*, 2:472–476 vol.2, 2001.

[60] Gábor Tardos and Géza Tóth. Multiple coverings of the plane with triangles. *Discrete & Computational Geometry*, 38(2):443–450, 2007.

[61] Kasturi R. Varadarajan. Epsilon nets and union complexity. In Hershberger and Fogel [36], pages 11–16.

[62] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.