THE PLANAR COLLECTION DEPOTS LOCATION PROBLEM

by

Jeff Sember
B.Sc., Simon Fraser University, 2004

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in the School of Computing Science

© Jeff Sember 2006
SIMON FRASER UNIVERSITY
Spring 2006

All rights reserved. This work may not be reproduced in whole or in part, by photocopy or other means, without the permission of the author.
APPROVAL

Name: Jeff Sember
Degree: Master of Science
Title of thesis: The Planar Collection Depots Location Problem

Examining Committee:
Dr. G. Tardos
Chair

Dr. B. K. Bhattacharya
Senior Supervisor

Dr. R. Krishnamurti
Supervisor

Dr. C. A. Wang
External Examiner
Professor, Department of Computer Science
Memorial University of Newfoundland

Date Approved: Apr 11, 2006
DECLARATION OF
PARTIAL COPYRIGHT LICENCE

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection, and, without changing the content, to translate the thesis/project or extended essays, if technically possible, to any medium or format for the purpose of preservation of the digital work.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author's written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

Simon Fraser University Library
Burnaby, BC, Canada
Abstract

In this thesis we examine the collection depots location problem, which is an extension of the classical facility center location problem. Besides \( n \) customers, a set of \( p \) collection depots are given, and the service of a customer consists of the travel of a server to the customer and return back to the center via a collection depot. We show that the problem can be transformed to \( O(p^2n^2) \) number of different classical facility location problems by using Voronoi diagrams. This is the first such non-trivial bound for this problem. We present fast 2-approximation algorithms for both the 1-median and 1-center versions of the problem. We also show how some of the techniques developed in this thesis can be applied to facility location problems involving line barriers.
To my parents
Acknowledgments

I thank Binay Bhattacharya for his advice and encouragement during my time at Simon Fraser University.

I am also grateful for the support provided by the Natural Sciences and Engineering Research Council of Canada.
# Contents

Approval .................................................. ii  
Abstract .................................................. iii  
Dedication ................................................ iv  
Acknowledgments ........................................... v  
Contents .................................................. vi  
List of Figures .......................................... ix  

## 1 Introduction ........................................ 1  
1.1 Facility Location Problems .............................. 2  
1.2 The Collection Depots Location Problem ............... 2  
1.3 Metrics ................................................ 3  
1.4 Previous results ....................................... 5  
1.5 Thesis Organization .................................... 7  

## 2 Geometry ............................................. 8  
2.1 The $L_2$ metric: Ellipse sets .......................... 8  
2.2 The complexity of an ellipse set ....................... 10  
2.3 Constructing an ellipse set ............................ 14  
2.4 The $L_1$ metric ....................................... 15  

## 3 Voronoi Diagrams ..................................... 17  
3.1 Applications of Voronoi diagrams ...................... 18
3.2 Algorithms for constructing Voronoi diagrams ........................................ 19
3.3 The Voronoi diagram for a customer ......................................................... 19
3.4 Additively weighted Voronoi diagrams ..................................................... 23
3.5 Algorithms for additively weighted Voronoi diagrams ............................. 25
3.6 The $L_1$ metric ....................................................................................... 25

4 Applying Voronoi diagrams to the depots problem .......................................... 27
4.1 Feasible assignments ................................................................................. 27
4.2 The MinMax depots problem ................................................................. 29
4.3 The MinSum depots problem .................................................................. 31
4.4 The $L_1$ metric ....................................................................................... 31

5 The Circle Sweep algorithm ........................................................................... 33
5.1 Motivation ............................................................................................... 33
5.2 Data structures ......................................................................................... 35
5.3 Events ....................................................................................................... 37
5.4 Running time ............................................................................................ 40
5.5 Lower Bounds for Generating Voronoi Diagrams .................................... 40
5.6 The $L_1$ metric ....................................................................................... 46

6 Approximation algorithms ............................................................................... 48
6.1 The classical Euclidean weighted 1-center problem ................................. 48
6.2 A MinMax depots problem approximation algorithm ............................. 50
6.3 A MinSum depots problem approximation algorithm ............................. 53
6.4 The $L_1$ metric ....................................................................................... 55

7 A Randomized MinMax depots algorithm ....................................................... 56
7.1 Implementing Solve ................................................................................ 57
7.1.1 Discrete approach .............................................................................. 57
7.1.2 Parametric search ............................................................................... 59
7.2 Running time analysis ............................................................................... 59
7.3 The $L_1$ metric ....................................................................................... 61
# List of Figures

2.1 Ellipse set with bounding circle. ........................................... 10
2.2 Ellipses with one common focus. ........................................... 12
2.3 Octagon set, with bounding square, and redundant depots identified. ... 16

3.1 A point set and its Voronoi diagram. ....................................... 18
3.2 The partitioning edge for two depots, with the customer on the opposite arm of the hyperbola. ......................................................... 21
3.3 The $L_1$ Voronoi diagram for a customer. ................................... 26

4.1 Voronoi diagram for one customer. ........................................... 28
4.2 Voronoi diagrams for two customers. ......................................... 29
4.3 Worst case situation. ............................................................. 29
4.4 $O(pn^2)$ feasible assignments in $L_1$ metric. ......................... 32

5.1 Voronoi diagram with ellipse sets. .......................................... 35
5.2 Voronoi diagram construction reduced to sorting problem. .............. 41
5.3 Rays associated with unbounded edges ...................................... 43
5.4 A Voronoi diagram and its dual graph. ...................................... 44
5.5 The dual graph is a triangulation of the polygon formed from the depots. 46

6.1 Proving MinMax approximation bound is tight. ........................... 52
6.2 Proving MinSum approximation bound is tight. ............................ 54

7.1 If $|\Gamma| \geq 2$, $s^*$ must occur on the boundary of at least two ellipses .... 58
7.2 $\lambda_{C,D} = 4$. ................................................................. 60
7.3 $\lambda_{C,D} = 5$. ................................................................. 61
7.4 $\lambda_{C,D} = 4$, $L_1$ metric. .................................................. 62
8.1 Voronoi diagram for customer, barrier problem. 65
8.2 Worst case situation, barrier problem. 65
8.3 Polygon partitioned by barriers into convex rooms, with customers and doors. 67
# List of Algorithms

<table>
<thead>
<tr>
<th></th>
<th>Algorithm</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CircleSweep($c, D$)</td>
<td>38</td>
</tr>
<tr>
<td>2</td>
<td>FindRegion($F, c, d, r$)</td>
<td>39</td>
</tr>
<tr>
<td>3</td>
<td>PredictEdgeEvents($B_{a,b}, F, Q, r$)</td>
<td>39</td>
</tr>
<tr>
<td>4</td>
<td>LeadingPoint($c, B_{a,b}, r$)</td>
<td>39</td>
</tr>
<tr>
<td>5</td>
<td>FindCenter($C$)</td>
<td>49</td>
</tr>
<tr>
<td>6</td>
<td>Extend($C, \Gamma, s$)</td>
<td>49</td>
</tr>
<tr>
<td>7</td>
<td>SolutionCovers($s, c$)</td>
<td>49</td>
</tr>
<tr>
<td>8</td>
<td>Solve($C$)</td>
<td>49</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Suppose a company is about to start offering a hazardous waste transportation service. The service will consist of dispatching a truck from the service center, travelling to a customer's site, filling the truck with a load of hazardous materials, delivering the materials to a waste treatment depot, and returning empty to the service center. Each truck must be cleaned by a special machine upon each return to the service center. The longer a trip was, the more detergent is required to clean the truck. The machine is automated, and uses a preset amount of detergent with each cleaning. An executive hires two analysts to study the business model. The first determines that the major expense of the business will be in purchasing the detergent, while the second finds that the major expense of the business will be in paying the truck drivers, who are paid in proportion to each kilometer driven. The executive knows the locations of the customers and the treatment depots, and must decide where to locate the service center for the transport service. If the first analyst is to be believed, the detergent used by the cleaning machine must be minimal, while still sufficient to clean a truck after a long trip. Locating the facility so the longest trip is minimized is the best choice; then the amount of detergent for the cleaning machine can be preset to handle this longest trip. On the other hand, if the second analyst is correct, the detergent cost is not important; the best location for the service center is that which minimizes the total distance travelled. This scenario is an example of the collection depots location problem, a type of facility location problem and the topic of this thesis, which is based in part on [4].
CHAPTER 1. INTRODUCTION

1.1 Facility Location Problems

In a facility location problem, the goal is to locate one or more facilities to service a set of clients, and to do so in a way that optimizes some objective function. Such problems can be single facility problems, in which the location of a single facility is to be determined; or they can be multiple facility problems, in which several facilities need to be positioned.

This thesis will be concerned with the single facility collection depots location problem in the plane.

1.2 The Collection Depots Location Problem

We are given a number of points in the \( \mathbb{R}^2 \) plane, consisting of a set of \( n \) customers \( C = \{c_1, \ldots, c_n\} \) and \( p \) collection depots \( D = \{d_1, \ldots, d_p\} \). Each customer \( c_i \) is associated with a positive weight factor \( w_i \). We are to then choose a location for the service center \( s \). A trip consists of a journey from \( s \) to a customer \( c_i \), from \( c_i \) to some depot \( d_j \), and from \( d_j \) back to \( s \). A trip must be made for each customer. The choice of which depot to use for a customer is made so as to minimize the round trip distance; as a result, some depots may be used by more than one customer, while others may not be used by any customers. The cost of a trip is the total distance travelled, multiplied by the customer’s weight factor. If the weights are all equal, we say the problem is unweighted.

The choice of where to locate the service center \( s \) is made with respect to one of two goals. In the MinMax depots problem, the aim is to minimize the maximum cost of any single customer’s trip. The objective function to be minimized is

\[
F(s) = \max_{i=1,\ldots,n} w_i \cdot \{\text{dist}(s, c_i) + \min_{j=1,\ldots,p} \{\text{dist}(c_i, d_j) + \text{dist}(d_j, s)\}\} \quad (1.1)
\]

where \( \text{dist}(a,b) \) represents the distance between points \( a \) and \( b \) according to a particular distance metric (see section 1.3). Alternatively, in the MinSum depots problem, the aim is to minimize the total cost of all customers’ trips, expressed by the objective function

\[
G(s) = \sum_{i=1}^{n} w_i \cdot \{\text{dist}(s, c_i) + \min_{j=1,\ldots,p} \{\text{dist}(c_i, d_j) + \text{dist}(d_j, s)\}\} \quad (1.2)
\]

This is equivalent to minimizing the average trip cost.

In the hazardous waste trucking business discussed at the start of the chapter, the first analyst is concerned with the MinMax depots problem, while the second is faced with the MinSum depots problem.
Other examples of practical situations where the collection depots location problem applies are given in [13]. These include a septic tank cleaning service, in which an empty truck travels from the service center to a customer, fills up, then drives to some depot to empty out before returning to the service center. Another example is an ambulance service, where an ambulance travels from a garage to an accident scene, loads a patient, delivers him or her to the nearest hospital, and returns to the garage.

1.3 Metrics

A metric is a function that assigns a distance (a nonnegative real value) to a pair of points. In this thesis, we will be considering these metrics in the $\mathbb{R}^2$ plane:

$$L_2((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (1.3)$$
$$L_1((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2| \quad (1.4)$$
$$L_\infty((x_1, y_1), (x_2, y_2)) = \max(|x_1 - x_2|, |y_1 - y_2|) \quad (1.5)$$

The $L_2$ metric measures standard Euclidean distance. The $L_1$ metric is also known as the taxicab, Manhattan, or rectilinear metric, since it is a measure of distance that results when movement is restricted along vectors parallel to the horizontal and vertical axes. The $L_\infty$ metric is also known as the Chebyshev metric.

In many problems from computational geometry, the choice of a metric can have a significant effect on the complexity of the problem and the approach taken by a particular solution; the collection depots location problem is no exception. We will see that the running times, the complexity of data structures, and the complexity of calculations are all affected by the choice of metric.

We will now show that a solution of the collection depots location problem under the $L_1$ metric can be easily converted to a solution of the problem under the $L_\infty$ metric.

Lemma 1.3.1. The planar collection depots location problem under the $L_\infty$ metric can be solved by any algorithm that solves the problem under the $L_1$ metric, at an additional cost that is linear in the size of the input and output.

Proof We apply a transformation to the sets of $L_\infty$ customers and depots to yield corresponding $L_1$ sets, which we then solve using the $L_1$ algorithm. We then apply the inverse transformation to any service center locations returned. This transformation (due to [23])
has the property that it preserves distances between the two spaces, where the distances are measured using the metric appropriate to the space. Given a point \((x, y)\) in the \(L_\infty\) metric space, we transform it to a point \((x', y')\) in the \(L_1\) metric space:

\[
\begin{align*}
x' &= \frac{1}{2}(x - y) \\
y' &= \frac{1}{2}(x + y)
\end{align*}
\]

(1.6)

(1.7)

The inverse of this transformation is applied to any solution points:

\[
\begin{align*}
x &= x' + y' \\
y &= -x' + y'
\end{align*}
\]

These transformations can clearly be performed in \(O(1)\) time for each input and output point. To complete the proof, we need only show that a distance between two points under the \(L_\infty\) metric is equal to the distance between the transformed points under the \(L_1\) metric.

Let \((x_1, y_1), (x_2, y_2)\) be the two points under the \(L_\infty\) metric. By (1.5), their distance is the distance between the transformed points under the \(L_1\) metric (applying (1.4)) is

\[
|x_1' - x_2'| + |y_1' - y_2'|
\]

substituting (1.6) and (1.7) yields

\[
\left|\frac{1}{2}(x_1 - y_1) - \frac{1}{2}(x_2 - y_2)\right| + \left|\frac{1}{2}(x_1 + y_1) - \frac{1}{2}(x_2 + y_2)\right|
\]

which, after rearranging terms, is

\[
\frac{1}{2}((x_1 - x_2) + (y_1 - y_2)) + |(x_1 - x_2) - (y_1 - y_2)|
\]

If we now apply the identity

\[
\max(|a|, |b|) = \frac{1}{2}(|a + b| + |a - b|)
\]

we will get (1.8).

The transform from lemma 1.3.1 can be applied to convert planar \(L_1\) diagrams to planar \(L_\infty\) diagrams, which is a useful point to remember with regard to Voronoi diagrams for the collection depots location problem; these are introduced in chapter 3. From this point on, the thesis will discuss only the \(L_1\) and \(L_2\) metrics, since lemma 1.3.1 allows us to ignore further consideration of the \(L_\infty\) metric.
1.4 Previous results

We will discuss in this section the history of the collection depots location problem, including results presented in previous works. We begin with a discussion of the classical MinSum problem, a problem from the field of facility location that is closely related to the MinSum collection depots location problem.

The classical MinSum problem (which is also known as the Weber problem, the Fermat problem, the Fermat-Toricelli problem, the Steiner problem, and others; see [12]), is to find the point \( s \in \mathbb{R}^2 \) which minimizes

\[
W(s) = \sum_{i=1}^{n} w_i \cdot \text{dist}(c_i, s)
\]  

where \( C = \{c_1, \ldots, c_n\} \) is a set of points in \( \mathbb{R}^2 \), each associated with a nonnegative weight \( w_i \). Under the \( L_1 \) metric, (1.9) is separable into horizontal and vertical components, and the problem is easily solved, in linear time, by calculating the (one-dimensional) median of the projections of the weighted points to each axis [29]. In the \( L_2 \) metric, (1.9) is not exactly solvable for \( n > 3 \) points [2], though many practical numerical methods exist. Most common among these is an iterative technique known as the \textit{Weiszfeld procedure}, a form of one-point iteration [36]. The \textit{Varignon frame} is a mechanical device that has been used in practice to solve the classical MinSum problem [12]. It consists of a board representing the \( \mathbb{R}^2 \) plane, a set of actual weights that hang below the board, and a string from each weight that extends up through a hole in the board at the point corresponding to the weight. The strings are connected at a common knot which moves across the board (assuming massless and frictionless strings) to settle at the point \( s \).

Drezner and Wesolowsky [13] introduced the collection depots location problem, and focused on the MinSum version of the problem. The authors pointed out that if the depots are excluded from each round trip, then the MinSum depots problem is equivalent to the classical MinSum problem, since the cost of each round trip is simply twice the cost of travel from the service center to a customer. We note that in this case, the MinMax depots problem reduces to the \textit{weighted 1-center problem}. We investigate this in chapter 6, where it is presented in relation to our approximation algorithm for the MinMax depots problem.

Drezner and Wesolowsky also examined the case when the choice of a depot for each customer is fixed, making the cost of travel from each customer to its associated depot a constant. The MinSum version of the problem then reduces to the classical MinSum
problem, since the cost of the remaining legs are equivalent to the weighted distance between
the service center and a fixed point in the plane. The MinMax version of the problem reduces
to the round trip location model, which we will refer to as the round trip problem. In this
problem, round trips are constructed between a common point \( s \) and pairs of points \( (p_i, q_i) \).
Point \( s \) is located to minimize the maximum length of any such round trip. The round trip
problem (for \( L_1 \) distances) was introduced by Chan and Hearn [8], and further work on the
problem is detailed in [13].

Drezner and Wesolowsky presented properties of the solution to the MinSum depots
problem and an iterative procedure that converges to local minima. They examined both
\( L_2 \) and \( L_1 \) versions of the problem, as well as the MinSum depots problem on a line.

Tamir and Halman [35] studied various versions of the MinMax depots problem, with
the additional assumption that the choice of depots available to each customer is restricted
to a subset of \( D \). We will refer to this as the restricted collection depots problem. For the
\( L_2 \) MinMax depots problem, they have shown that the optimal service center location \( s \)
must lie within a set of ellipses associated with each customer and depot, and describe an
algorithm that runs in \( O(p^2 n^2 \log^3(pn)) \) time. We investigate this algorithm in detail in
section 4.2. In the \( L_1 \) metric, they pointed out that the optimal service center location lies
within an octagon associated with each customer, and proved that an intersection of unions
of such octagons has a lower complexity than an intersection of unions of the ellipses from
the \( L_2 \) case. They used this fact to derive an \( O(pn \log^3(pn)) \) algorithm for the \( L_1 \) MinMax
depots problem.

Tamir and Halman also discussed two one-way versions, in which each trip consists of
just two 'legs'. In the depot one-way problem, a trip starts at the service center, travels
to the customer, then to a depot. In the customer one-way problem, a trip starts at the
service center, travels to a depot, then to a customer. In the depot one-way problem, the
choice of depot for each customer is simply the nearest depot to the customer, and the
problem reduces to the weighted 1-center problem, as described in the previous section.
In the customer one-way problem, the ellipses of the standard round trip version become
circles. While this can also be solved in \( O(p^2 n^2 \log^3(pn)) \) time using an algorithm similar
to that for the standard problem, they show how properties of the problem allow a solution
to be found in \( O(pn^2 \text{polylog}(pn)) \) time. Tamir and Halman also examined path and tree
network versions of the problem.

Other papers have examined the collection depots location problem on networks. These
include [6], in which properties of solutions to both the MinMax and MinSum depots problems are described; [5], which investigates tree networks; and [7], which studies the MinSum depots problem with multiple facilities on a network and describes techniques for solving the problem, as well as properties of its solutions.

1.5 Thesis Organization

In this chapter, we have been introduced to the collection depots location problem, a particular type of facility location problem. We have also investigated the $L_2$, $L_1$, and $L_\infty$ metrics, three methods of measuring distances that will be applied to the problem, and we reviewed previous work on the problem.

In chapter 2, we examine how ellipses, octagons, and other geometry objects relate to the collection depots location problem. We prove that a union of ellipses that share a common focus has linear complexity, which improves on that found in [35]. In chapters 3 and 4, we see how a certain class of Voronoi diagram can be used to solve the problem, and solve an open problem posed by [13] by proving that at most $O(p^2n^2)$ feasible assignments of depots to customers are possible, and that this bound is tight. We further show that these assignments can be generated in $O(p^2n^2 \lg(pn))$ time. In chapter 5 we present an optimal algorithm for constructing these Voronoi diagrams, and prove that the algorithm is optimal even in the cases when the Voronoi sites (depots) are given in sorted order by distance from the customer, or by polar angle around the customer.

We present fast approximation algorithms for both the MinMax and MinSum depots problems in chapter 6. We present a randomized algorithm for the MinMax depots problem in chapter 7, and analyze its running time. In chapter 8, we show how some of the techniques we use to solve the collection depots location problem can be used to improve existing solutions to a type of facility location problem involving linear barriers, and solve an extension of this problem involving a set of rooms with walls and doors. Finally, in chapter 9, we summarize our results, and consider areas of possible future research.
Chapter 2

Geometry

In this chapter, we investigate the role of certain geometry primitives in the collection depots problem, and how they relate to previous solutions of the problem. Unless otherwise stated, we assume all distances are $L_2$.

**Definition** The round trip distance for an ordered set of points $(a_1, \ldots, a_k)$ is

$$\text{dist}(a_1, a_2) + \text{dist}(a_2, a_3) + \ldots + \text{dist}(a_k, a_1),$$

and is denoted $\text{rtd}(a_1, \ldots, a_k)$.

### 2.1 The $L_2$ metric: Ellipse sets

Consider a single customer $c$ and an upper bound $r$ on the length of any round trip involving $c$. If a particular depot $d_j$ can participate in such a round trip, then from (1.1) we have

$$\text{rtd}(s, c) + \text{dist}(d_j, s) \leq r. \quad (2.1)$$

We wish to determine where a service center $s$ can be located that doesn't violate the bound $r$ for the customer. Note that for fixed $c$, $d_j$, and $r$, $\text{dist}(c, d_j)$ is a constant, and (2.1) becomes

$$\text{dist}(s, c) + \text{dist}(d_j, s) \leq k, \quad (2.2)$$

where $k$ is a constant. If $k$ is less than the distance between the customer and depot, (2.2) has no solution $s$. Otherwise, as observed in [35], (2.2) is the equation of an ellipse with foci...
at \( c \) and \( d_j \), with round trip distance \( r = k + \text{dist}(c, d_j) \); we will denote this ellipse \( E_{c,d_j,r} \), or when the customer location and round trip distance are clear from the context, \( E_j \).

Each depot \( d_j \) is associated with some ellipse \( E_{c,d_j,r} \), or cannot participate in any round trips; in this case, we assume \( E_{c,d_j,r} = \emptyset \). The set of all possible service center locations for the customer \( c \) that allow a trip length not exceeding \( r \) is then

\[
\bigcup_{j=1}^{p} E_{c,d_j,r}.
\]

This is a union of a number of ellipses, each with \( c \) as one of the two foci. We will refer to this construct as an ellipse set for customer \( c \) and round trip length \( r \), and will denote it \( S_{c,r} \) (or \( S_r \), if \( c \) is clear from the context).

Ellipses and ellipse sets are fundamental to the material in this chapter, so we will investigate their properties.

**Definition** The radius of ellipse \( E = E_{c,d_j,r} \) is

\[
r_j = \text{dist}(c, s) + \text{dist}(s, d_j),
\]

where \( s \) is any point on the boundary of \( E \). We denote this quantity \( r_j \).

**Definition** The line through the foci of an ellipse is its semimajor axis, or simply its axis.

**Definition** The polar angle of a depot \( d \), denoted \( \theta(d) \), is the angle that the ray from \( c \) through \( d \) makes with the positive \( x \)-axis, where the origin is placed at \( c \).

**Definition** The far point of ellipse \( E = E_{c,d,r} \) is the point on the boundary of \( E \) that is on the axis of \( E \) and is farthest from \( c \). The near point of \( E \) is the corresponding point that is nearest to \( c \).

**Lemma 2.1.1.** A circle can be drawn around an ellipse set \( S_{c,r} \) that contains each component ellipse \( E \) of the set, and will contain the far point of \( E \) on its boundary. The radius of this circle is \( r/2 \).

**Proof** Consider the ray \( R \) from \( c \) that contains a depot \( d_j \). If \( d_j = c \), then \( E = E_{c,d_j,r} \) is equal to the bounding circle. Otherwise, the farthest point \( p \in R \) from \( c \) that can be part of a round trip with length \( \text{rtd}(c, d, p) \leq r \) is the point \( p_j \) at distance \( r/2 \) from \( c \). This is the far point of \( E \), and is thus on the boundary of the circle \( C \) with origin at \( c \) and radius \( r/2 \). Every other point of \( E \) is closer to \( c \), and is thus inside the circle. \( \blacksquare \)
In the figures of this thesis, we will denote customers with squares and depots with triangles. See figure 2.1 for an example of an ellipse set. Note that the diameter of an ellipse set's bounding circle is the upper bound on the round trip distance involving the set's customer.

A special situation occurs in a set \( S_{c,r} \) when a depot \( d \) lies on the ellipse set's bounding circle. The distance from \( c \) to \( d \) is thus \( r/2 \), and ellipse \( E_d \) degenerates to a line segment between these two points.

### 2.2 The complexity of an ellipse set

The complexity of an ellipse set is the number of vertices of its boundary, which is composed of elliptic arcs from its component ellipses. We will call these elliptic arcs segments. An ellipse set vertex occurs at the interface between two segments.

Throughout this section, we assume that the depots and customer are distinct, and we also assume that each depot has a distinct polar angle. (If two ellipses have equal polar angles, the two depots will lie on the same ray from the customer. The nearer depot's ellipse will contain the farther's ellipse, and we say that the nearer depot dominates the farther depot.)
Lemma 2.2.1. Every ellipse $E_i$ in a set $S_r$ must contribute at least one boundary segment to the boundary of $S_r$.

Proof Let $f$ be the far point of $E_i$. By definition, $f \in E_i$. Assume some $E_j$ contains $f$ as well. Since $\theta(d_j) \neq \theta(d_i)$, $rtd(c,d_j,f) > rtd(c,d_i,f)$. But every ellipse in the set has the same round trip distance, so this is a contradiction. Since only $E_i$ contains $f$, there must be at least one segment in the set’s boundary derived from $E_i$.

Lemma 2.2.2. For ellipses $E_i$ and $E_j$, with $\text{dist}(c,d_i) \leq \text{dist}(c,d_j)$, the near point of $E_j$ is strictly inside $E_i$.

Proof Assume the near point $q$ of $E_j$ is not interior to $E_i$. Then
\[
\text{dist}(c,d_i) + \text{dist}(d_i,q) + \text{dist}(q,c) \geq r ,
\]
\[
\text{dist}(c,d_j) + \text{dist}(d_j,q) + \text{dist}(q,c) = r .
\]
Combining these two yields
\[
\text{dist}(c,d_j) + \text{dist}(d_j,q) \leq \text{dist}(c,d_i) + \text{dist}(d_i,q) . \tag{2.3}
\]
By the triangle inequality,
\[
\text{dist}(d_i,q) \leq \text{dist}(c,d_i) + \text{dist}(q,c) .
\]
Since each depot has a unique polar angle, $d_i$ cannot lie on the ray from $c$ through $d_j$, and we have a strict inequality
\[
\text{dist}(d_i,q) < \text{dist}(c,d_i) + \text{dist}(q,c) .
\]
By the ordering of the depots by distance from $c$,
\[
\text{dist}(d_i,q) < \text{dist}(c,d_j) + \text{dist}(q,c) .
\]
Since $q$ is the near point of $E_j$,
\[
\text{dist}(d_i,q) < \text{dist}(c,d_j) + \text{dist}(q,c) = \text{dist}(d_j,q) , \tag{2.4}
\]
so combining (2.3) with (2.4) yields
\[
\text{dist}(c,d_j) < \text{dist}(c,d_i) ,
\]
a contradiction. □
We will need the following lemma in subsequent proofs.

**Lemma 2.2.3.** The boundaries of distinct ellipses $A$ and $B$ that have exactly one focus in common will intersect in at most two points.

**Proof** Place the axes so the common focus is at the origin, and $u$, the other focus of $A$, is on the positive $x$-axis. Without loss of generality, assume that the radius of $A$ is at least as great as that of $B$ (i.e. $r_A \geq r_B$), and that $v$, the other focus of $B$, is on or above the $x$-axis. See figure 2.2.

Assume by way of contradiction that there are more than two points of intersection between the boundaries of $A$ and $B$. Any such intersection point $q$ must satisfy:

$$\text{dist}(q, O) + \text{dist}(q, u) = r_A,$$

$$\text{dist}(q, O) + \text{dist}(q, v) = r_B.$$

Subtracting the second from the first yields

$$\text{dist}(q, u) - \text{dist}(q, v) = r_A - r_B,$$

which is the equation for a hyperbolic arm $H$ with foci at $u$ and $v$.

If $r_A = r_B$, $H$ will be a line bisecting the segment $\overline{uv}$; since a line can intersect any ellipse in at most two points, and all intersection points $q$ must lie on this line, we have a contradiction.
Otherwise, \( v \) will be the nearer of the two foci to any point on \( H \). Consider \( \tilde{H} \), the other arm of the hyperbola with foci \( u \) and \( v \). \( H \) and \( \tilde{H} \) partition the plane into three regions, which we can label the region on or inside \( H \), the region on or inside \( \tilde{H} \), and the region outside both \( H \) and \( \tilde{H} \).

Let \( I_H \) (resp. \( I_{\tilde{H}} \)) be the number of points of intersection between \( H \) (resp. \( \tilde{H} \)) and the boundary of \( A \). Since a hyperbola can intersect an ellipse in at most four points, we have \( I_H + I_{\tilde{H}} \leq 4 \). Since all points of intersection between the ellipse boundaries must lie on \( H \), and we have assumed there are more than two such points, \( I_H > 2 \). This implies that \( I_{\tilde{H}} \in \{0, 1\} \). If \( I_{\tilde{H}} = 0 \), then since \( u \) lies in the region inside \( \tilde{H} \), and \( \tilde{H} \) is unbounded, the entire boundary of \( A \) must lie strictly inside this region. Thus no point of \( A \) can then intersect \( H \), and \( I_H = 0 \), a contradiction. If \( I_{\tilde{H}} = 1 \), then since \( u \) lies in the region inside \( \tilde{H} \), \( A \) must be inside and tangent to \( \tilde{H} \); so again, no point on the boundary of \( A \) can intersect \( H \), and we have a contradiction.

**Lemma 2.2.4.** When the ellipse for depot \( d_j \) is added to an ellipse set \( S_{c,r} \), where \( d_j \) is no closer to \( c \) than any previously added ellipse's depot \( (d_1 \ldots d_{j-1}) \), the number of vertices in the boundary of the resulting ellipse set \( S'_{c,r} \) will be no more than two plus that of \( S_{c,r} \).

**Proof** Consider the intersection of \( E_j \) with any previously-added ellipse \( E_i \). Since both ellipses have the customer as one of their two foci, by lemma 2.2.3 the portion of the boundary of \( E_j \) common to \( E_i \) is a single segment. By lemma 2.2.2, this segment must contain the near point of \( E_j \) in its interior. To determine the points on the boundary of \( E_j \) that are not already in the ellipse set, we subtract the union of those points that are already in the set. This is a union of \( j - 1 \) segments of \( E_j \), each of which must contain the near point of \( E_j \) (by lemma 2.2.2). The complement of this union is thus a single segment, which when inserted into the boundary of \( S_{c,r} \) can increase the vertex count by at most two.

Lemma 2.2.3 allows us to apply Theorem 3.1 from [19] to show that an ellipse set of \( p > 2 \) depots has at most \( 6p - 12 \) vertices, but we can improve on this bound.

**Theorem 2.2.5.** An ellipse set of \( p \) depots has at most \( 2p - 1 \) vertices.

**Proof** The proof is by induction on the number of depots. Assume the depots \( P \) are ordered by nondecreasing distance from the customer, and that no depots are outside the
ellipse set's bounding circle. For \( p = 1 \), the ellipse set is a single ellipse, with no vertices. By lemma 2.2.4, adding additional depots increases this value by at most two per depot.

2.3 Constructing an ellipse set

**Theorem 2.3.1.** An ellipse set for \( p \) depots can be constructed in \( O(p \lg p) \) time, using \( O(p) \) space.

**Proof** We first sort the depots into nondecreasing distance from the customer. We construct a sorted list of ellipse set segments, sorted by their angle around \( c \). We process each ellipse \( E \) in sorted order. By the proof of lemma 2.2.4, exactly one segment of \( E \) will be added to the ellipse set. We search for the insertion point of this segment by looking for the existing segment that intersects the axis of \( E \). We then traverse to both sides of this segment, deleting any segments that are completely inside \( E \), and stopping at the segment that crosses \( E \). This crossing segment and \( E \) are both clipped at this point. When this traversal procedure is complete for both sides, the remaining segment of \( E \) is inserted into the sorted list.

We maintain the sorted list of segments in a data structure that supports \( O(\lg n) \) accesses: insertions, deletions, searches, and traversals to adjacent elements. A red/black tree is one such data structure.

Each new ellipse requires \( O(1) \) accesses for everything except the deletion of existing segments. We can assign an amortized cost to each access to show that the total number of accesses performed for the entire ellipse set is \( O(p) \). There are four types of operations performed on segments: a new segment can be inserted, and existing segments can be deleted, clipped, or split into two segments. Note that each operation requires \( O(1) \) accesses. The number of segments increases by one for an insertion and split operations, decreases by one for a deletion operation, and remains the same for a clip operation. The total number of segments created during construction is not more than \( 2p \), since each segment can be inserted just once, possibly splitting an existing segment while doing so. The total number of deletion operations can thus not exceed \( 2p \).

The running time for the construction of the ellipse set is then \( O(p \lg p) \) for the initial sorting of ellipses, \( O((a+b) \lg p) \) for the segment operations (where \( a \) represents the number of non-deletion operations, and \( b \) the deletion operations). Since \( a = O(p) \), and \( b = O(2p) \),
the total running time is $O(p \lg p)$. Finally, we note that a red/black tree can perform this construction using $O(p)$ space.

In chapter 3, we will see how an algorithm for constructing a variant of the Voronoi diagram related to the collection depots location problem yields a much simpler proof of theorems 2.2.5 and 2.3.1.

2.4 The $L_1$ metric

As noted in [35], under the $L_1$ metric, points satisfying equation (2.1) are bounded by a (possibly degenerate) octagon whose sides have slopes of 0, $\infty$, 1, or $-1$. The set of possible service center locations for customer $c$ that allow a trip length not exceeding $r$ is therefore a union of $p$ such octagons, which we will refer to as the customer's octagon set. Tamir and Halman showed that the arrangement of the octagons in such a set has complexity of $O(p)$ (Section 3.2.2.2 in [35]). Thus the boundary of an octagon set has $O(p)$ complexity. Using an approach very similar to that of section 2.3, an octagon set for $p$ depots can be constructed in $O(p \lg p)$ time, using $O(p)$ space.

In the $L_1$ case, it is common for one depot to render another redundant. Consider depot $d_1$ that lies within the minimum bounding rectangle of customer $c$ and a second depot $d_2$. Any shortest round trip involving $c$ and $d_2$ can be modified, without increasing its length, to pass through $d_1$ as well; thus $d_2$ can be ignored as a candidate depot for $c$. Identifying and eliminating these redundant depots can be done as a side effect of the octagon set construction algorithm without affecting its $O(p \lg p)$ running time. See figure 2.3 for an example octagon set. Note that every octagon in an octagon set will lie within a bounding square whose diagonals lie on the coordinate axes.
Figure 2.3: Octagon set, with bounding square, and redundant depots identified.
Chapter 3

Voronoi Diagrams

In this chapter, we review Voronoi diagrams, how they can be used to solve problems from computational geometry, and methods used in their construction (a more detailed discussion of Voronoi diagrams, algorithms, and their uses can be found in [1]). We then examine how a particular subclass of Voronoi diagrams apply to the collection depots location problem.

Let $S$ be a set of distinct points, or sites, in the $\mathbb{R}^2$ plane, and $p$ be an element of $S$. The set of points in the plane which are at least as close to $p$ as to any other site forms the Voronoi region of $p$, and is denoted $U(p)$. In formal terms,

$$U(p) = \{x \in \mathbb{R}^2 \mid \forall q \in S \text{ dist}(x,p) \leq \text{dist}(x,q)\} \quad (3.1)$$

Unless stated otherwise, all distances in this chapter will be $L_2$ distances.

The set of all points in the plane that are members of more than one Voronoi region form the Voronoi diagram of $S$, and is denoted $V_S$.

$$V_S = \{x \in \mathbb{R}^2 \mid \exists_{p,q \in S, p \neq q} x \in U(p) \cap U(q)\}$$

The points in the plane which belong to at least two Voronoi regions form the edges of $V_S$, while those that belong to three or more Voronoi regions form the vertices of $V_S$.

Voronoi diagrams can be generalized in several ways. The dimensional space, which for our purposes is $\mathbb{R}^2$, can be extended to other dimensions. The sites, or generators [27], can be structures other than points, including lines, polygons, or other subsets of the space. Finally, the distance function can be varied. Figure 3.1 is an example of a Voronoi diagram, in $\mathbb{R}^2$, of a set of (point) sites using the $L_2$ metric. We will refer to this variety of Voronoi diagram as an ordinary Voronoi diagram [27].
3.1 Applications of Voronoi diagrams

Voronoi diagrams can be used to solve a variety of problems from computational geometry. Perhaps the most common example is the post office problem [11], where the task is to determine which post office is closest to a particular customer. If a Voronoi diagram is constructed with the post offices as sites, then this question can be answered for a customer by determining which site's Voronoi region contains the customer. Such a query operation can be performed for \( n \) sites in \( O(\lg n) \) time, if an additional structure [15] is constructed along with the Voronoi diagram. This additional structure can be constructed in \( O(n) \) time using \( O(n) \) space, which does not affect the \( O(n \lg n) \) time required to build the diagram itself [32].

Examples of Voronoi diagrams being used to solve facility location problems are found in [27]. These include the largest empty circle problem, where the goal is to find the point in a bounded region whose distance to the nearest site is greatest. This might be used, for example, to determine where a new grocery store could be placed, so that it is as far away as possible from existing stores [28]. The bottleneck problem, where the goal is to find the two sites in \( S \) which are closest together, is another example. A simple way to solve this problem is to test each possible pair of sites in \( O(n^2) \) time. A more efficient method is to
construct and examine a Voronoi diagram, a task which can be completed in $O(n \lg n)$ time.

Navigating a robot through an environment is an example of path planning. If a robot is circular, the environment is planar, and obstacles are points, then to avoid colliding with obstacles, the robot should remain on the edges of the Voronoi diagram constructed from the environment [28].

### 3.2 Algorithms for constructing Voronoi diagrams

There are three main types of algorithm for constructing Voronoi diagrams.

Incremental algorithms build a diagram by adding one site at a time, and have $O(n^2)$ worst-case running times.

Divide-and-conquer algorithms recursively divide the sites into two smaller subsets, construct diagrams recursively for each, then merge the two diagrams together. The first such algorithm with optimal worst-case behaviour was developed by Shamos and Hoey [32]. Their algorithm has an $O(n \lg n)$ running time, but the merge steps can be difficult to implement [16].

Plane sweep methods to construct Voronoi diagrams were first employed by Fortune [16]. Like divide-and-conquer techniques, they have $O(n \lg n)$ running times, and have the advantage of simplicity; however, they can suffer from numerical sensitivity issues. Guibas and Stolfi [17] presented a more intuitive interpretation of Fortune's algorithm as the sweep of a set of cones in $\mathbb{R}^3$ by a sweep plane inclined parallel to the slope of the cones.

### 3.3 The Voronoi diagram for a customer

In this section, we investigate the Voronoi diagram that is associated with a single customer in an instance of the collection depots location problem.

**Definition** The *minimal round trip distance* for a customer $c$, a point $q$, and a set of depots $D$ in the $\mathbb{R}^2$ plane is the length of the shortest round trip between the customer, the point, and any one depot $d \in D$. In formal terms, it is equal to:

\[
\min_{d \in D} \operatorname{rtd}(c, q, d)
\]

For fixed $c$ and $q$, we say a round trip for a depot $d \in D$ is *minimal* if its distance equals the minimal round trip distance.
We would like to partition the \( \mathbb{R}^2 \) plane into regions that satisfy the following two properties. First, each region should be associated with a single depot. Second, for any point within the region, the round trip between that point, the customer, and the region’s depot is minimal.

We can consider the round trip distance between a point in the plane, a customer, and a depot as a Voronoi diagram’s distance function. This is not a simple distance between two points, since the round trip involves three points. We will see that this distance function produces a particular generalization of Voronoi diagram.

From this point on, we will use the notation \( U(d) \) to refer to the Voronoi region of a depot \( d \) associated with a particular customer \( c \), and \( V_{c,D} \) to refer to the Voronoi diagram for customer \( c \) and depots \( D \).

Consider the set of points that have minimal round trip distances achievable with either of two depots. If such a set is nonempty, it will form an edge of our desired Voronoi diagram. Since these edges serve to partition a region of the plane between two depots, we may call them partitioning edges or partitioning curves (since the edges may not be linear). We will now investigate the structure of such an edge.

Let \( d_1 \) and \( d_2 \) be depots in \( D \). If a Voronoi edge separates regions \( U(d_1) \) and \( U(d_2) \), then any point \( q \) on that edge satisfies

\[
\text{dist}(q, c) + \text{dist}(c, d_1) + \text{dist}(d_1, q) = \text{dist}(q, c) + \text{dist}(c, d_2) + \text{dist}(d_2, q)
\]

(3.2)

We can simplify and rearrange (3.2) to get

\[
\text{dist}(d_1, q) - \text{dist}(d_2, q) = \text{dist}(c, d_2) - \text{dist}(c, d_1)
\]

(3.3)

For fixed \( c, d_1, \) and \( d_2 \), the right hand side of (3.3) is a constant, so points satisfying (3.3) will lie on a hyperbola. The equation of this hyperbola is:

\[
|\text{dist}(d_1, q) - \text{dist}(d_2, q)| = |\text{dist}(c, d_2) - \text{dist}(c, d_1)|
\]

(3.4)

**Lemma 3.3.1.** A Voronoi edge which separates regions \( U(d_1) \) and \( U(d_2) \) will lie on a hyperbola, and if the hyperbola consists of two separate arms, one arm will contain the customer and the other will contain the edge.

**Proof** If \( c \) is equidistant to \( d_1 \) and \( d_2 \), then the right hand side of (3.4) is zero, the hyperbola does not have two separate arms, and we’re done. Otherwise, setting \( q \) to \( c \) will satisfy (3.4),
so \( c \) must lie on some arm. Without loss of generality, assume \( c \) is closer to \( d_1 \). Then the right hand side of (3.3) is positive, and any point \( q \) on the edge must be further from \( d_1 \) than \( d_2 \). Since \( c \) is closer to \( d_1 \) than \( d_2 \), \( q \neq c \); thus the edge cannot lie on the same arm as \( c \).

Figure 3.2 shows an example of a partitioning edge for two depots.

![Figure 3.2: The partitioning edge for two depots, with the customer on the opposite arm of the hyperbola.](image)

We will now examine some properties of a depot's Voronoi region, and develop some lemmas to help bound the complexity of \( V_{c,D} \).

**Lemma 3.3.2.** Each \( U(d) \) is unbounded.

**Proof** Consider the ray from \( d \) that is coincident with, and parallel to, the ray from \( c \) through \( d \). Clearly every point \( q \) on this ray belongs to \( U(d) \), since the line segment from \( c \) to \( q \) contains \( d \), and is thus a minimal round trip for a service center located at \( q \). \( U(d) \) therefore contains this unbounded ray.

**Lemma 3.3.3.** \( U(d) \) has zero area iff there exists some other depot \( d' \) that lies on the line segment between \( c \) and \( d \).

**Proof** Let \( L \) be the line segment between \( c \) and \( d \). If there exists \( d' \in L \), assume there exists some point \( q \in U(d) \) whose minimal round trip through \( d \) is strictly less than that through any other depot. Since this minimal round trip must pass through \( d' \), this is a
contradiction; thus every point \( q \in U(d) \) must lie on its boundary, so \( U(d) \) has zero area. Observe that the boundary of \( U(d) \) has the form of a ray from \( d' \) through \( d \).

If \( U(d) \) has zero area, assume no other depot \( d' \) lies on \( L \). Consider \( q \) at \( d \). \( \text{rtd}(c, q, d) = 2|L| \), and by the triangle inequality, \( \text{rtd}(c, q, d') > 2|L| \) for every other \( d' \). Thus \( q \) is not on the boundary of \( U(d) \), there must exist a locus of points around \( d \) with this property as well, and \( U(d) \) has nonzero area; a contradiction.

**Definition** If the Voronoi region for depot \( d_1 \) contains that of \( d_2 \), then \( d_1 \) dominates \( d_2 \).

If depot \( d_1 \) dominates \( d_2 \) for customer \( c \), \( d_2 \) can be ignored, since \( d_2 \) cannot admit a better optimal solution than \( d_1 \). \( d_2 \) can thus be left out of the Voronoi diagram for \( c \). Note that dominated depots can be determined in \( O(p \log p) \) time by a two-step process. First, the depots are sorted lexicographically: by polar angle around \( c \), and by distance from \( c \) if their polar angles are the same. Second, a linear scan is performed on the sorted list to determine the dominated depots.

**Definition** A region \( R \) is star-shaped if there exists a point \( f \) within \( R \) such that for every point \( x \) in \( R \), the line segment from \( f \) to \( x \) lies within \( R \). We also say that \( R \) is star-shaped from \( f \).

**Lemma 3.3.4.** Each \( U(d) \) is star-shaped from \( d \).

**Proof** \( U(d) \) is the intersection of some number of unbounded convex regions of \( \mathbb{R}^2 \), where by lemma 3.3.1 the boundary of each region is a hyperbolic arm. Since \( d \) is a focus of each of these arms, each convex region is star-shaped from \( d \); thus an intersection of some number of them must be as well.

We are now ready to establish a bound on the complexity of a Voronoi diagram for a customer.

**Lemma 3.3.5.** The complexity of \( V_{c,D} \) is \( O(|D|) \).

**Proof** To simplify the proof, we assume that the depots and customer are in general position (no three points are collinear). The proof can be extended, if necessary, to relax this restriction.

Since no three points are collinear, by lemma 3.3.3, each depot lies strictly inside its region; thus each depot can occur in only one region. We also note that every region contains
some depot: every point in $\mathbb{R}^2$ must have a minimal round trip involving some depot and therefore belongs to that depot’s region, and by lemma 3.3.4 each region is connected. Thus $f = |D|$. The rest of the proof is essentially the same as that of the complexity of ordinary Voronoi diagrams presented in [28], which is based on Euler’s formula for planar graphs.

We now return to the definition of a Voronoi region to investigate the type of Voronoi diagram generated for a customer. The Voronoi region for a depot $d$ is, by expanding (3.1),

$$U(d_1) = \{ x \in \mathbb{R}^2 \mid \forall_{d_2 \neq d_1} \, \text{rtd}(x, c, d_1) \leq \text{rtd}(x, c, d_2) \}.$$

Eliminating the common trip leg gives us

$$U(d_1) = \{ x \in \mathbb{R}^2 \mid \forall_{d_2 \neq d_1} \, \text{dist}(d_1, x) + \text{dist}(c, d_1) \leq \text{dist}(d_2, x) + \text{dist}(c, d_2) \}.$$

Since $\text{dist}(c, d_1)$ and $\text{dist}(c, d_2)$ are constants, this amounts to adding a constant value to the standard distance function. This produces an additively weighted Voronoi diagram.

### 3.4 Additively weighted Voronoi diagrams

Voronoi diagrams have been used to model the growth of crystals. If crystals start growing (in the $\mathbb{R}^2$ plane) around fixed sites, they will meet at Voronoi edges. If the crystals’ growth rates are the same, and they all start growing at the same time, the crystal boundaries will form an ordinary Voronoi diagram. If the growth rates remain equal, but they are allowed to start growing at different times, their boundaries will form an additively weighted Voronoi diagram. Extended to $\mathbb{R}^3$, this has been termed the Johnson-Mehl model [1] of crystal growth.

In the collection depots problem, sites are represented by depots, so we will use the two terms interchangeably.

**Lemma 3.4.1.** Each edge of an additively weighted Voronoi diagram lies on the arm of a hyperbola.

**Proof** If an edge exists between the regions of depots $d_1$ and $d_2$ with respective additive weights $w_1$ and $w_2$, each point $x$ on the edge will satisfy

$$\text{dist}(x, d_1) + w_1 = \text{dist}(x, d_2) + w_2$$
which after rearranging yields
\[
\text{dist}(x, d_1) - \text{dist}(x, d_2) = w_2 - w_1
\] (3.5)

Since the right hand side of (3.5) is constant, we can apply the same reasoning as with (3.3) to show that this is the equation of an arm of a hyperbola with foci at \(d_1\) and \(d_2\).

It is easy to show that with nonnegative weights, the arm containing the edge will be the one closest to the depot with the larger weight. In the collection depots location problem, this means the arm is closest to the depot that is furthest from the customer.

Additively weighted Voronoi diagrams can have regions that are empty. For example, if there are only two depots, \(d_1\) and \(d_2\), and they have corresponding weights \(w_1 = 0\) and \(w_2 = \text{dist}(d_1, d_2) + \epsilon\) for some \(\epsilon > 0\), then any point \(x\) which is within \(U(d_2)\) must satisfy
\[
\text{dist}(x, d_2) + \text{dist}(d_1, d_2) + \epsilon \leq \text{dist}(x, d_1) ,
\]
which violates the triangle inequality. We say that depot \(d_2\) is trivial [18].

**Lemma 3.4.2.** In the collection depots location problem, no depots are trivial.

**Proof** Assume by way of contradiction that for some customer \(c\), depot \(d\) is trivial. Then for every point \(q \in \mathbb{R}^2\), there is some other depot \(d'\) that satisfies
\[
\text{dist}(q, c) + \text{dist}(c, d') + \text{dist}(d', q) < \text{dist}(q, c) + \text{dist}(c, d) + \text{dist}(d, q) .
\] (3.6)

If we choose \(q\) to be \(d\), then by the triangle inequality, (3.6) is false.

We can relate additively weighted Voronoi diagrams to generalized Voronoi diagrams in the following way [18]. First, we note that translating the depot weights by a constant doesn’t change the structure of the diagram. Let \(w'\) be the largest weight associated with a set of depots \(D\). Assign new weights \(W^*\) to each depot, where \(w_i^* = w' - w_i\). Note that each of the new weights is nonnegative.

We can then define a distance function between a point \(x\) in the plane and a depot \(d_i\):
\[
\text{dist}^*(x, d_i) = \text{dist}(x, d_i) - w_i^* .
\]

Note that this is the distance of a point in the plane from a circle of radius \(w_i^*\), centered at \(d_i\). An additively weighted Voronoi diagram is therefore a Voronoi diagram of circular sites, where the original sites are the circle centers, and the new weights \(W^*\) are the radii of the circles.
3.5 Algorithms for additively weighted Voronoi diagrams

Early algorithms for constructing additively weighted Voronoi diagrams were based on this view of sites as circles [14, 22]. A problem arises when circles overlap, and these early algorithms could not deal with such non-disjoint sites. The first $O(n \lg n)$ algorithm for constructing general additively weighted Voronoi diagrams was presented, though not well developed, by Fortune [16].

The Voronoi diagram for a customer in the collection depots location problem is a restricted type of additively weighted Voronoi diagram. In chapter 5, we present an optimal algorithm for its construction.

3.6 The $L_1$ metric

In the $L_1$ metric, equation (3.3) typically describes a curve consisting of up to three linear sections, with slopes of $\{0, \infty, 1, -1\}$; but as noted in [9, 21], it may not describe a curve, but rather a curve connected to one or two unbounded square regions. To simplify $L_1$ Voronoi diagrams, an arbitrary curve that satisfies equation (3.3) can be chosen as the bisector by replacing each unbounded square region with a ray. We call such portions of the curve that have been arbitrarily chosen effective edges.

As with the $L_2$ case, the complexity of the Voronoi diagram in the $L_1$ metric is linear; the proof of lemma 3.3.5 can be easily adapted.

Figure 3.3 is an $L_1$ Voronoi diagram for a customer. Effective edges are shown as dashed lines, and redundant depots are drawn in a lighter gray than the other depots.
Figure 3.3: The $L_1$ Voronoi diagram for a customer.
Chapter 4

Applying Voronoi diagrams to the depots problem

In this chapter, we examine how collection depots Voronoi diagrams can be employed to solve both the MinMax and MinSum depots problems. Our general approach will involve constructing Voronoi diagrams for a set of customers, merging these diagrams together, and generating a set of assignments of depots to customers, which can then be used to solve the collection depots problem. Unless stated otherwise, we will assume all distances are $L_2$.

4.1 Feasible assignments

Note that in the collection depots location problem, as the service center is moved, the optimal choice of depot for each customer will vary. Let $I_s$ denote the assignment vector of length $n$ where $I_s[i]$ indicates the optimal depot assignment for customer $c_i$ when the service facility is at $s$. We call $I_s$ a feasible assignment vector, or just a feasible assignment.

Equations (1.1) and (1.2) can be rewritten as

$$F(s) = \max_{i=1,...,n} \left( w_i \cdot \left\{ \text{dist}(s, c_i) + \{\text{dist}(c_i, I_s[i]) + \text{dist}(I_s[i], s)\} \right\},
$$

$$G(s) = \sum_{i=1}^{n} w_i \cdot \left\{ \text{dist}(s, c_i) + \{\text{dist}(c_i, I_s[i]) + \text{dist}(I_s[i], s)\} \right\}. \tag{4.1}$$
Note that the feasible assignment vector is the same for a particular $s$ for both the objective functions $F(s)$ and $G(s)$.

It is natural to ask how many different feasible assignments exist for any placement of the facility in the plane. An obvious upper bound is $O(p^n)$, but tighter bounds should exist. Drezner and Wesolowsky [13] left this question open. We will prove a tight bound for this problem for the $L_2$ metric.

**Theorem 4.1.1.** At most $O(p^2n^2)$ different feasible assignments of $p$ depots to $n$ customers are possible for any choice of a service center in the $L_2$ metric.

**Proof** Since $V_{c,D}$ for each customer has $O(p)$ complexity, and any edge in one such diagram can intersect every edge in every other diagram $O(1)$ times, there are at most $O((pn)^2)$ regions in the merged diagrams. For any point in a particular region, the depots assignments for the customers of $C$ remains the same. 

**Lemma 4.1.2.** All different feasible assignments of depots can be computed in $O(p^2n^2 \lg (pn))$ time.

**Proof** By lemma 5.4.1, each $V_{c,D}$ can be constructed in $O(p \lg p)$ time. Generating all feasible assignments is dominated by the time spent merging the diagrams, which can be done in $O(p^2n^2 \lg (pn))$ time using the standard plane sweep technique.

It is possible to construct an example to show that the bound of theorem 4.1.1 is tight. First, we start with a vertical line of depots. A single customer $c$ to the right of the lowest depot produces $V_{c,D}$ of figure 4.1.

![Figure 4.1: Voronoi diagram for one customer.](image)
Adding a second customer to the right of the highest depot produces figure 4.2. The idea can be extended by adding additional customers to the right of the existing customers (figure 4.3). In this way, we can generate an example whose number of feasible assignments of depots is $\Theta(p^2n^2)$.

4.2 The MinMax depots problem

Tamir and Halman [35] presented an $O(p^2n^2 \log^3(pn))$ algorithm for the MinMax restricted collection depots problem. We describe our approach for the unrestricted collection depots problem, in which each customer is allowed to use any of the depots of $D$. The algorithm
can also be applied without any change to the restricted version.

The algorithm is based on the parametric approach of Megiddo [25] which requires an efficient parallel implementation for the following decision problem (which we will call the covering problem): Determine whether there exists a facility location such that the maximum weighted round trip distance of the customers is at most r.

**Definition** The weighted round-trip distance r for a customer c is r/wc, and is denoted rc.

Recall that Ec,dj,rc is the ellipse consisting of those service center locations that can participate with customer c and depot dj in a round trip of length not exceeding rc. The union of the p depots gives us the ellipse set Sc,rc. For the covering problem, we ask the question: Is ∩c∈C Sc,rc empty? It was argued in [35] that the boundary of Sc,rc can have $O(p^{2\alpha(p)})$ vertices and elliptical arcs where $\alpha(p)$ is the functional inverse of the Ackermann's function. We have shown (theorem 2.2.5) that its complexity is actually $O(p)$.

As described in [35], whether $\cap_{c\in C} S_{c,rc}$ is non-empty can be tested in $O(p^2 n^3 \log (pn))$ (section 6 in Sharir and Agarwal [33]). The optimal value of the MinMax collection depots problem is the smallest r of the covering problem for which $\cap_{c\in C} S_{c,rc}$ is non-empty. For this we apply the parametric approach of Megiddo [25].

**Theorem 4.2.1.** The optimal solution to the MinMax collection depots problem can be computed in $O(p^2 n^3 \log^2 (pn))$ time.

One observation we make is that by lemma 2.3.1, n ellipse sets can be generated, one for each customer, in $O(pn \log p)$ time. The sweep of these ellipse sets will generate fewer intersection events than a sweep of all their component ellipses.

It is an open problem of whether an intersection of n ellipse sets of p depots can have a complexity of $\Theta(p^2 n^2)$. While Tamir and Halman [34] have achieved this complexity for the restricted collection depots problem, their technique does not extend to the unrestricted version of the problem, and we conjecture that this complexity is $o(p^2 n^2)$ for the unrestricted case.

As noted in section 1.4, when the assignment of depots to customers is fixed, the MinMax depots problem reduces to the round trip problem. This suggests another approach to solving the MinMax depots problem: we generate all feasible assignments for the customers, solve the round trip problem for each assignment, and choose the best of these solutions. In chapter 7, we will show how the round trip problem can be solved in $O(n^4)$ time. Thus
Theorem 4.2.2. The MinMax depots problem can be solved in $O(p^2n^2\lg(pn) + p^2n^6)$ time.

The running time of this approach appears to be quite inferior to that of theorem 4.2.1, but it will be used to some advantage in chapter 7.

4.3 The MinSum depots problem

It was observed in [13] that equation (4.1) can be rewritten as follows:

$$G(s) = \sum_{i=1}^{n} w_i \cdot \text{dist}(s, c_i)$$

$$+ \sum_{i=1}^{n} w_i \cdot \text{dist}(c_i, I_s[i])$$

$$+ \sum_{i=1}^{n} w_i \cdot \text{dist}(I_s[i], s)$$

$$= G_1(s) + G_2(s) + G_3(s)$$

For a given feasible assignment vector $I_s$, $G_2(s)$ is constant, so minimizing $G(s)$ is the same as minimizing $G_1(s) + G_3(s)$ which is the classical MinSum problem of at most $2n$ points (note that this maximum is reached only if each customer is assigned to a unique depot). By lemma 4.1.2, all feasible assignments $I_s$ can be determined in $O(n\lg(pn))$ time; therefore

Theorem 4.3.1. The MinSum collection depots problem can be solved in $O(p^2n^2(T(n) + \lg(pn)))$ time, where $T(n)$ is the time it takes to solve the classical MinSum problem of $O(n)$ points.

4.4 The $L_1$ metric

It is an open question of whether the bound of theorem 4.1.1 is tight under the $L_1$ metric. Figure 4.4 displays a configuration of customers and depots that achieves a complexity of $O(pn^2)$ feasible assignments.

In this figure, one set of $O(n)$ customers (in the center of the diagram) together with $O(p)$ depots create a sequence of 'zigzag' edges, while another set of $O(n)$ customers (at the bottom of the diagram) with two depots create a vertical column of edges that intersect the zigzag edges, producing an arrangement with $O(pn^2)$ complexity.
Figure 4.4: $O(pn^2)$ feasible assignments in $L_1$ metric.
Chapter 5

The Circle Sweep algorithm

In this chapter, we present the Circle Sweep algorithm, which produces collection depots Voronoi diagrams. It takes advantage of the fact that the additive weight for each depot is the depot’s distance from the customer to produce the diagram in a more intuitive manner than Fortune’s algorithm, while still achieving the same optimal time and space complexity.

We will assume all distances are $L_2$, until section 5.6, where we will discuss a version of the algorithm modified for $L_1$ distances.

5.1 Motivation

The algorithm is motivated by the relationship between the ellipse set for a customer and its Voronoi diagram. Recall from section 2.1 that $S_{c,r}$ is the ellipse set for customer $c$ and round trip length $r$.

Lemma 5.1.1. The boundary of ellipse set $S_{c,r}$ contains exactly those points $q$ that satisfy

$$\min_{d \in D} rtd(c, d, q) = r$$

Proof Any point $q$ inside the set’s boundary must be inside the boundary of some ellipse $E_{c,d,r}$, so $rtd(c, d, q) < r$. Any point $q$ that is outside the set’s boundary must be outside of all its component ellipses, so $\forall_{d \in D} rtd(c, d, q) > r$. \qed

Lemma 5.1.2. Every point in the $\mathbb{R}^2$ plane is contained in the boundary of $S_{c,r}$, for exactly one $r \geq 0$. 

33
Proof Every point \( q \in \mathbb{R}^2 \) has some minimal round trip distance \( r \), and by lemma 5.1.1, \( S_{c,r} \) contains \( q \), and is the only ellipse set that contains \( q \).

Voronoi edges contain exactly those points that belong to two or more Voronoi regions, which leads to the following definition and lemmas.

Definition An interface point in the boundary of an ellipse set is a point located at the intersection of two of the boundary's elliptic arcs.

Lemma 5.1.3. Every point in a Voronoi edge lies on an interface point of ellipse set \( S_{c,r} \) for some \( r \).

Proof By lemma 5.1.1, all points with minimal round trip distance \( r \) will lie on the boundary of \( S_{c,r} \). For a point \( q \) to be on a Voronoi edge, there must be two or more depots admitting an optimal round trip distance \( r \). Thus there must exist ellipses \( E_1 = E_{c,d_1,r} \) and \( E_2 = E_{c,d_2,r} \) such that \( q \) lies on the boundaries of both \( E_1 \) and \( E_2 \), as well as being on the boundary of \( S_{c,r} \):

\[
q \in \partial E_1 \cap \partial E_2 \cap \partial S_{c,r}
\]

Clearly \( p \) can only exist where two ellipses intersect on the boundary of \( S_{c,r} \).

Lemma 5.1.4. The interface points of ellipse set \( S_{c,r} \) will trace out the collection depots Voronoi diagram, as \( r \) ranges from \( 0 \) to \( +\infty \).

Proof Follows from lemma 5.1.3.

Figure 5.1 shows a Voronoi diagram overlaid with ellipse set boundaries for several values of \( r \).

Instead of using a sweep line or plane, Circle Sweep uses a sweep circle, which represents the radius of the ellipse set's bounding circle. As the radius of the sweep circle increases, the algorithm keeps track of the hyperbolic arcs representing the paths of individual interface points which form the Voronoi edges, and generates Voronoi vertices when two such arcs intersect.

For the sake of simplicity, we assume that the degree of each Voronoi vertex is three, and that no Voronoi vertex occurs on the horizontal line containing the customer. The first assumption ensures that each edge that we construct will have nonzero length (and thus two distinct endpoints). The second avoids any difficulties in predicting wrap events, discussed later. The algorithm could be modified, if necessary, to relax these restrictions.
5.2 Data structures

The algorithm uses the standard data structures for a plane sweep, an event queue and frontier. The event queue $Q$ maintains a list of events, sorted in nondecreasing order according to the radius that the sweep circle will have when the event is to be processed. The frontier $F$ maintains a list of edges, which are portions of the curve partitioning the plane between two depots. The edge (or bisector) separating depot $d_a$ from $d_b$ is denoted $B_{a,b}$. Each bisector has a list ($\text{VertList}$) of known vertices. When this list contains two vertices, the bisector is added to the diagram. We examine some properties of these curves before describing the edge data structure in more detail.

**Lemma 5.2.1.** If $B_{a,b}$ is a bisector for depot $d_a$, depot $d_b$, and customer $c$, where $\text{dist}(c, d_a) \leq \text{dist}(c, d_b)$, and $L$ is the line segment between $c$ and $d_b$, then the intersection $I = L \cap B_{a,b}$ is a single point $q^*$. Furthermore, the minimum round trip distance between $c$ and either depot for points on $B_{a,b}$ is achieved at $q = q^*$, and this distance strictly increases as $q$ moves away from $q^*$ in either direction along $B_{a,b}$.

**Proof** First, we will prove that $I$ consists of a single point $q^*$. Note that $\text{rtd}(c, d_b, e)$ is the same for any point $e \in L$; let this quantity equal $r^*$. Note also that for any point $e' \notin L$, $\text{rtd}(c, d_b, e') > r^*$. Consider ellipses $E_a = E_{c,d_a, r^*}$ and $E_b = E_{c,d_b, r^*}$. $E_b$ is degenerate, and
is equal to L. Since L starts inside $E_a$ and ends on or outside $E_a$ (since $d_b$ is not closer to c than $d_a$), the boundaries of $E_a$ and $E_b$ will intersect in exactly one point; call this point $q^*$. Next, we will show that $q^*$ is the only point $q$ on $B_{a,b}$ satisfying

$$r_t(c, d_a, q) = r_t(c, d_b, q) = r^*$$

Another way of describing $B_{a,b}$ is as the set of points of intersection of $\partial E_{c,d_a,r} \cap \partial E_{c,d_b,r}$, for all values of $r$. All points on $B_{a,b}$ not equal to $q^*$ must come from ellipse intersections for values $r > r^*$, so $q^*$ is the only point on $B_{a,b}$ achieving round trip distance $r^*$.

Finally, we examine the behaviour of $r$ as we move away from $q^*$ along $B_{a,b}$. If c lies on the $x$-axis to the left of $d_a$, then the polar angles the depots are the same, and $B_{a,b}$ is a degenerate hyperbolic arm in the form of a ray from $d_b$ in the direction of the positive $x$-axis. Clearly, moving further to the right along this ray strictly increases the round trip distance.

If the polar angles of the depots are different, then for $r > r^*$, by lemma 2.2.3, the ellipse boundaries intersect at exactly two points, and by lemmas 2.2.1 and 2.2.2 these points must lie on opposite sides of L. Thus the round trip distances for points on a particular side of $B_{a,b}$ are pairwise distinct. Since each side of $B_{a,b}$ is continuous, the distances for these points are strictly increasing.

**Definition** The leading point of an edge $B_{a,b}$ for sweep circle radius $r$ is the point $q$ on $B_{a,b}$ that satisfies $r_t(c, d_a, q) = 2r$.

Edges are stored in the frontier sorted by the polar angles of their leading points. To ensure that the leading point of an edge is unique, a bisector is split into two edges at the point $q^*$ (see lemma 5.2.1). By convention, edge $B_{a,b}$ describes a hyperbolic arm with depot $d_a$ to its right and depot $d_b$ to its left, with the direction of the arm determined by the direction of movement of the leading point of the edge along the arm as the sweep circle radius increases. An example of an ordered subsequence of edges in the frontier is

$$\ldots B_{a,b}, B_{b,c}, B_{c,a}, B_{a,d}, \ldots$$

Note that neighboring edges have matching depots, and a depot's region can occur more than once in the frontier.

36
5.3 Events

The Circle Sweep algorithm manipulates three types of event. Each event is a tuple, and the first element of the tuple is the sweep circle radius value the event is to be sorted by.

- **SITE (radius \( r \), depot \( d_n \)).** This event adds a new depot \( d_n \) to the diagram. At the start of the algorithm, the queue is populated with a SITE event for each depot. When a SITE event is processed, the existing depot \( d_e \) of the region which contains \( d_n \) is determined. The bisector \( H \) of these two depots is constructed, and split into two edges \( B_{e,n} \) and \( B_{n,e} \) at point \( q^* \). These edges are added to the frontier, and INTERSECT and WRAP events are predicted for each.

- **INTERSECT (radius \( r \), point \( q \), right edge \( B_{a,b} \), left edge \( B_{b,c} \)).** Whenever the frontier is modified, either by inserting, deleting, or changing the relative positions of edges, an INTERSECT event is predicted for edges that have become neighbors and will intersect at some point outside the current sweep circle. When an INTERSECT event is processed, the two edges involved are examined to see if they are still neighbors. It is possible that the frontier has been modified since the prediction was made to the extent that the edges are no longer neighbors; if so, the event is treated as a 'false alarm'. Otherwise, the intersection point is added to each edge's vertex list, and the two edges are removed from the frontier. A new edge \( B_{b,c} \) representing the interface between the two regions that are now touching is added to the frontier.

- **WRAP (radius \( r \), edge \( H \).)** The frontier maintains a linear list of edges sorted by polar angle, yet the sweep circle represents a closed curve. WRAP events are predicted and processed to move edges whose leading points have crossed from one side of the \( \pm \pi \) ray to the other.

When the event queue is empty, any edges remaining in the frontier will never intersect, and thus are unbounded Voronoi edges. For each of these, a point on the hyperbolic arm representing a point at infinity is stored as the final vertex.

When a SITE event is processed, two edges are created from a hyperbolic arm, and the point on this arm nearest the customer is stored as the initial vertex for both edges. In the Voronoi diagram, these two edges are actually connected, and no vertex exists. If necessary, these spurious vertices and edges (which are \( O(n) \) in number) can be detected and removed in \( O(n) \) time at the end of the algorithm.
Algorithm 1: CircleSweep(c, D)

Input: Customer c, depots D
Output: Voronoi diagram for c

V ← ∅
F ← ∅
Q ← ∅

foreach d ∈ D do
    add SITE(dist(c, d), d) to Q

SITE(r_{first}, d_{first}) ← ExtractMin(Q)

while Q not empty do
    E ← ExtractMin(Q)

    if E = SITE(r, n) then
        if F is empty then
            e ← d_{first}
        else
            e ← FindRegion(F, c, n, r)
        end
        construct H bisecting d_c and d_n
        R ← ray from c through n
        v ← R ∩ H
        B_{e,n} ← portion of H to right of R
        B_{n,e} ← portion of H to left of R
        add vertex v to B_{e,n}, B_{n,e}
        add B_{e,n}, B_{n,e} to V and F
        PredictEdgeEvents(B_{e,n}, F, Q, r)
        PredictEdgeEvents(B_{n,e}, F, Q, r)
    end
    else if E = INTERSECT(r, v, B_{a,b}, B_{b,c}) then
        if B_{a,b}, B_{b,c} are still neighbors then
            construct B_{a,c}
            add vertex v to B_{a,c}
            add B_{a,c} to V
            replace B_{a,b} and B_{b,c} in F with B_{a,c}
            PredictEdgeEvents(B_{a,c}, F, Q, r)
        else if E = WRAP(r, H) then
            if H is still in F then
                move H to opposite end of F
        end
    end

foreach B in F do
    add point on B at infinity as vertex of B

return V
**Function** `FindRegion(F, c, d, r)`

**Input:** Frontier $F$, customer $c$, new depot $d$, sweep circle radius $r$

**Output:** Depot of region containing $d$

Search $F$ for neighboring arms $B_{a,b}$, $B_{b,c}$ with $p \leftarrow \text{LeadingPoint}(c, B_{a,b}, r), q \leftarrow \text{LeadingPoint}(c, B_{b,c}, r)$, such that $\Theta(p) \leq \Theta(d) \leq \Theta(q)$.

return $b$

---

**Procedure** `PredictEdgeEvents(B_{a,b}, F, Q, r)`

**Input:** Edge $H$, frontier $F$, event queue $Q$, sweep circle radius $r$

- $B_{prev} \leftarrow$ edge preceding $B_{a,b}$ in $F$
- $B_{next} \leftarrow$ edge following $B_{a,b}$ in $F$

if $B_{prev}$ exists and is not the same as $B_{a,b}$ then

- $s \leftarrow B_{prev} \cap B_{a,b}$
  - if $s \neq \emptyset$ and $s$ is not in $B_{a,b}.\text{VertList}$ then
    - add $\text{INTERSECT}(\text{rtd}(c, d_a, s)/2, s, B_{prev}, B_{a,b})$ to $Q$

if $B_{next}$ exists and is not the same as $H$ then

- $t \leftarrow B_{a,b} \cap B_{next}$
  - if $t \neq \emptyset$ and $t$ is not in $B_{a,b}.\text{VertList}$ then
    - add $\text{INTERSECT}(\text{rtd}(c, d_b, t)/2, t, B_{a,b}, B_{next})$ to $Q$

$R \leftarrow$ ray from $c$ that makes polar angle $\pm \pi$

- $u \leftarrow B_{a,b} \cap R$

if $u$ exists then

- $r' \leftarrow \text{rtd}(c, d_a, u)/2$
  - if $r' > r$ then
    - add $\text{WRAP}(r', B_{a,b})$ to $Q$

---

**Function** `LeadingPoint(c, B_{a,b}, r)`

**Input:** Customer $c$, edge $B_{a,b}$, sweep circle radius $r$

**Output:** Point of intersection of bisector with sweep circle

- $p \leftarrow$ the point of intersection between ellipse $E_{c,a,2r}$ and bisector $B_{a,b}$ such that $\text{rtd}(c, p, a) = 2r$

return $p$
5.4 Running time

By analyzing the number of types of each event that can be generated, we can place a bound on the running time of the Circle Sweep algorithm.

A total of \( p \) SITE events are generated, one for each depot. Each SITE event can immediately spawn up to two INTERSECT events. When an INTERSECT event is processed, if it produces a Voronoi vertex, an additional two INTERSECT events can be produced. An edge can produce at most one WRAP event, since it can cross the \( \pm \pi \) ray at most once. Since each bisector is associated with at most two edges, and there are \( O(p) \) Voronoi vertices and \( O(p) \) Voronoi edges, the total number of events processed by the algorithm is \( O(p) \).

If the event queue and frontier are maintained in a data structure supporting \( O(\lg n) \) insertions and deletions (such as a red/black tree), the Circle Sweep algorithm will execute in \( O(p \lg p) \) time. Therefore

**Lemma 5.4.1.** The Circle Sweep algorithm constructs \( V_c \) in \( O(p \lg p) \) time and \( O(p) \) space.

In chapter 2, we presented a proof for the complexity of ellipse sets (lemma 2.2.5), as well as a proof of the complexity of any algorithm for their construction (lemma 2.3.1). We can use the Circle Sweep algorithm to derive much simpler proofs of these two lemmas.

**Theorem 5.4.2.** The complexity of an ellipse set of \( p \) depots is \( O(p) \).

**Proof** By lemma 3.3.5, \( V_{c,D} \) has \( O(p) \) edges. If we start the Circle Sweep algorithm, and interrupt it when the sweep circle has radius \( r/2 \), the frontier \( F \) will contain a list of edges from which the set of ellipse segments comprising \( S_{c,r} \) can be easily extracted. Since \( V_{c,D} \) has \( O(p) \) edges, \( F \) has \( O(p) \) edges; thus there are \( O(p) \) segments in \( S_{c,r} \).

**Theorem 5.4.3.** An ellipse set of \( p \) depots can be constructed in \( O(p \lg p) \) time.

**Proof** By lemma 5.4.1, the frontier \( F \) of the proof of lemma 5.4.2 can be constructed in \( O(p \lg p) \) time. Extracting the ellipse segments by iterating through \( F \) can be done in \( O(p \lg p) \) time as well.

5.5 Lower Bounds for Generating Voronoi Diagrams

We can prove that \( O(p \lg p) \) is a tight bound on the running time for generating \( V_c \), by showing how the problem reduces to the problem of sorting. We will rely upon this well-known theorem related to sorting (see, for example, [30]):
Theorem 5.5.1. Any algorithm for comparison sorting $n$ numbers requires $[\lg n!]$ comparisons, and runs in $\Omega(n \lg n)$ time.

Lemma 5.5.2. Any algorithm for generating $V_c$ for $p$ arbitrary depots has execution time $\Omega(p \lg p)$.

Proof The proof is by reduction from sorting a sequence of numbers $\{t_1, \ldots, t_p\}$. Assume some algorithm $F$ exists which can construct $V_c$ in $o(p \lg p)$ time. First, translate the numbers so each $t_i > 0$. Construct a depot with coordinates $(t_i, t_i^2)$ for each $i$, and place the customer at $(0,0)$. Apply $F$ to construct $V_c$ with from these depots. $V_c$ will appear in a form similar to figure 5.2, and its underlying graph will consist of a sequence of $p - 2$ nodes, $p - 3$ edges connecting the nodes in a path, and $p$ unbounded edges (two from the nodes at the ends of the path, and one from each of those in its interior). These nodes can be traversed in $O(p)$ time; their (inverse translated) $x$-coordinates will yield the sort of the original sequence. Thus the time to sort $p$ numbers can be done in $o(p \lg p) + O(p) = o(p \lg p)$ time, a contradiction of theorem 5.5.1.

Figure 5.2: Voronoi diagram construction reduced to sorting problem.

We now address the question of whether certain types of additional information provided as input can allow the Voronoi diagram to be constructed in $o(n \lg n)$ time. The approach taken here follows that of [31]. We will show that if the additional information allows
$V_c$ to be constructed in $o(p \log p)$ time, then sorting $p$ numbers unrelated to the additional information can be done in $o(p \log p)$ time as well, which is a contradiction.

In this section, it will help in our analysis if we associate each edge of $V_c$ with two oppositely directed half edges from the graph of $V_c$.

**Definition** An unbounded edge of $V_c$ is a half edge of $V_c$ which is directed towards the vertex at infinity.

**Lemma 5.5.3.** All diagrams $V_c$ will have $\Theta(p)$ unbounded edges.

**Proof** By lemma 3.3.2, each depot's region is unbounded, and each depot must therefore contribute at least one distinct unbounded edge to $V_c$ (each unbounded edge between two depots, incident with the vertex at infinity, can be associated with the depot to its right); thus in $V_c$ there are at least $\Omega(p)$ unbounded edges, and by lemma 3.3.5, there are at most $O(p)$ edges in total.  

Consider the unbounded edges incident with the vertex at infinity. There exists some circle centered at the customer with finite radius such that the circle properly intersects each edge $E_i$ at point $q_i$, and beyond the circle no two edges intersect. We will call the ray from $c$ through $q_i$ the ray of edge $E_i$; see figure 5.3. Note that each such ray has a unique polar angle with respect to $c$. We will refer to this polar angle as the polar angle of edge $E_i$.

**Lemma 5.5.4.** Given $V_c$ and $G = \{E_1, \ldots, E_k\}$, the sequence of unbounded edges of $V_c$ sorted by polar angle, the sequence of depots $D_\theta = \{d_1, \ldots, d_p\}$ sorted by polar angle around $c$ can be determined in $O(p)$ time.

**Proof** Consider each pair of consecutive unbounded edges $E_i, E_j \in G$. These edges will border the unbounded region for some depot $d$. We next examine the wedge-shaped section of the plane swept out as the ray of $E_i$ rotates to that of $E_j$. The wedges for each pair of consecutive edges in $G$ form a partition of the plane, so only one such wedge can contain $d$. Every depot must appear in one such wedge, and by lemma 3.3.2, no wedge can contain more than one depot. Note that the wedges may outnumber the depots; in figure 5.3, depot $a$ is associated with two wedges, only one of which contains $a$. The sequence $D_\theta$ can be extracted from $G$ by processing each consecutive pair of unbounded edges $(E_i, E_j)$, testing
if their associated depot \( d \) lies within their wedge, and adding \( d \) to \( D_\theta \) if so. Processing each pair in this way can be done in constant time, thus \( D_\theta \) can be generated in \( O(p) \) time.

The Voronoi diagram \( V_c \) may not be connected (see figure 5.3), so generating the sorted sequence of unbounded edges \( G \) in linear time cannot be performed by a simple traversal of the boundary of \( V_c \). The dual graph of the Voronoi diagram provides a solution to this problem.

**Definition** The dual graph of \( V_c \), denoted \( V_c^* \), is constructed so that faces in \( V_c \) (which are associated with depots) are mapped to nodes in \( V_c^* \), and two nodes are adjacent if their corresponding faces share a Voronoi edge. A Voronoi diagram and its dual graph are shown in figure 5.4.

Some properties of \( V_c^* \) are easy to derive.

**Lemma 5.5.5.** \( V_c^* \) is connected.

**Proof** Every region of \( V_c \) contains exactly one depot, so any path between depots \( i \) and \( j \) in the plane must cross some \( k > 0 \) sequence of region boundaries whose duals are edges in \( V_c^* \). Thus a path exists in \( V_c^* \) between all pairs of nodes \( i \) and \( j \).
Lemma 5.5.6. Every node in $V_c^*$ is on the outer face of $V_c^*$.

Proof Assume by way of contradiction that some node $k \in V_c^*$ is not on the outer face of $V_c^*$. Then none of the edges incident with $k$ are adjacent to the outer face of $V_c^*$, so the face of $k$ in $V_c$ does not share an edge with the vertex at infinity in $V_c$. This implies that the face of $k$ in $V_c$ is bounded, contradicting lemma 3.3.2.

Lemma 5.5.7. The sequence of unbounded edges $G = \{E_1, \ldots, E_k\}$ sorted by polar angle can be constructed in $O(p)$ time if $V_c^*$ is provided.

Proof By lemma 5.5.5, $V_c^*$ is connected; and since every unbounded edge of $V_c$ will appear as a boundary edge of $V_c^*$, $G$ can be produced in linear time by traversing the boundary of $V_c^*$.

Lemma 5.5.8. The sequence of depots $D_\theta = \{d_1, \ldots, d_p\}$ sorted by polar angle around $c$ can be extracted from $V_c$ in linear time.

Proof $V_c^*$ can be constructed from $V_c$ in linear time. By lemma 5.5.7, the sequence of unbounded edges $G$ can be constructed from $V_c^*$ in linear time. Finally, by lemma 5.5.4, $D_\theta$ can be constructed from $V_c$ and $G$ in linear time.
Theorem 5.5.9. Any algorithm for generating Voronoi diagrams for depots $D = \{d_1, \ldots, d_p\}$ which are sorted by distance from $c$ has execution time $\Omega(p \log p)$.

Proof Assume by way of contradiction that some such algorithm $F$ exists, and that it has execution time $o(p \log p)$. Let $T$ be a sequence of reals $\{t_1, \ldots, t_p\}$ where $\forall i : 0 \leq t_i \leq 1$. Place the customer at the origin, and construct an ordered set of depots $D = \{d_1, \ldots, d_p\}$ such that $d_i = (2^i, 2^i t_i)$. The distance of each depot from the customer is $d_i = 2^i \sqrt{1 + t_i^2}$, and

$$2^i \leq |d_i| \leq 2^i \sqrt{2} < 2^{i+1},$$

thus the depots $D$ are ordered by distance from the customer. We can apply algorithm $F$ to $D$ to construct $V_c$ in $o(p \log p)$ time. By lemma 5.5.8, $D_\theta$ can be extracted from $V_c$ in linear time. Let $\theta_i$ be the polar angle of $d_i$. Note that $\forall i : 0 \leq \theta_i \leq \pi/4$, and that $\tan \theta_i$ increases as $\theta_i$ in this range. For each $d'_i = (x, y) \in D_\theta$, we set $t'_i = \tan \theta_i = y/x$. Constructing $D$, constructing $V_c$, extracting $D_\theta$ from $V_c$, and extracting $T' = \{t'_1, \ldots, t'_p\}$ can each be done in $o(p \log p)$ time. Thus $T'$, the sort of $T$, has been constructed in $o(p \log p)$ time; a contradiction.

Theorem 5.5.9 indicates that even if the depots are presorted by distance from the customer, constructing the Voronoi diagram still requires $\Omega(p \log p)$ time. We can show that presenting the depots sorted by polar angle doesn’t improve the running time either.

Theorem 5.5.10. Any algorithm for generating Voronoi diagrams for depots $D = \{d_1, \ldots, d_p\}$ which are sorted by polar angle around $c$ has execution time $\Omega(p \log p)$.

Proof Assume by way of contradiction that some such algorithm $F$ exists, and that it has execution time $o(n \log n)$. Let $D$ be a set of depots evenly spaced on a circle around $c$, sorted by their polar angle. Observe that the depots form a convex polygon, which we will denote $Y_D$. Let $D'$ be the depots after being slightly perturbed so that no vertex in the Voronoi diagram $V_{c,D'}$ has degree greater than three, and so that the depots still form a convex polygon $Y_{D'}$ with vertices having the same ordering as in $Y_D$.

$V^{*}_{c,D'}$, the dual graph of the voronoi diagram of $D'$, will appear as a triangulation of $Y_{D'}$. Figure 5.5 is an example of a perturbed set of eight depots, and the resulting triangulated octagon. If we perturb the locations of the depots by moving them slightly nearer or farther from $c$, we can produce, in a deterministic fashion, any desired triangulation of $Y_D$. The
number of triangulations of a convex polygon of $n$ vertices is given by the Catalan number $C_{n-2}$, where $C_n = \frac{1}{n+1} \binom{2n}{n}$; thus there are $\Omega(n!)$ different graphs underlying the possible Voronoi diagrams for $n$ depots.

If we construct the dual graph of the Voronoi diagram for each of two sets of depots with identical polar angle orderings, then compare these graphs for equivalence, we can answer the question of Element Uniqueness which by [3] has an $\Omega(n \lg n)$ lower bound. Since constructing the dual graph for $V_c$ can be done in linear time, we conclude that constructing $V_c$ when the depots are sorted by polar angle requires $\Omega(n \lg n)$ time; thus algorithm $F$ cannot exist.

We conclude this section by noting that it is an open question of whether $V_c$ can be constructed in $o(p \lg p)$ time if both the depots sorted by distance and depots sorted by polar angle are given as input.

### 5.6 The $L_1$ metric

The Circle Sweep algorithm can be modified to produce $L_1$ collection depot Voronoi diagrams. We omit the details, but make the following observations:

- Under the $L_1$ metric, the sweep circle becomes a sweep square (see figure 2.3).
Recall from section 3.6 that the partitioning edges are no longer hyperbolic arm sections, but instead consist of up to three linear sections. This can complicate the algorithm logic (for example, it is possible for portions of two edges to coincide), but the math operations are simpler.

The detection of redundant depots can be performed as a side effect of the diagram construction.

We conclude by noting that any algorithm for the construction of $L_1$ Voronoi diagrams must take $\Omega(n \lg n)$ time, since it is easy to produce a diagram analogous to figure 5.2.
Chapter 6

Approximation algorithms

An approximation algorithm is one that provides a solution that may be suboptimal, but does so with a better runtime. Such algorithms are often used to generate approximate solutions in polynomial time for problems whose only known exact solutions require exponential running times. For an introduction to the subject, see [10].

The exact algorithms proposed for the MinMax and MinSum problems are expensive for large $p$ and $n$. We present here fast approximation algorithms for both the MinMax and MinSum collection depots problems.

We first investigate a problem that will be used in the MinMax approximation algorithm.

6.1 The classical Euclidean weighted 1-center problem

In the (classical Euclidean) weighted 1-center problem, we are given a set of customers $C = \{c_1, \ldots, c_n\}$, each associated with positive weight $w_i$, and seek to minimize

$$W(s) = \max_{i=1, \ldots, n} w_i \cdot \text{dist}(s, c_i).$$

The MinMax depots problem reduces to this problem if a depot is located at every customer (or, alternatively, if the round trips need not include a depot). While a deterministic linear-time algorithm for this problem exists [26], a simple and practical randomized algorithm with linear expected running time can be developed using a technique that will be used in chapter 7 as well.

Our algorithm, FindCenter, is a generalization of Welzl's algorithm for the smallest enclosing disk problem [37]. Formally, we are given a set of points $C = \{c_1, \ldots, c_n\}$ and
Algorithm 5: FindCenter(C)

Input: Set of items C
Output: Solution

permute C into random order
return Extend(C, 0, 0)

Function Extend(C, Γ, s)

Input: Set of items C, determining set Γ, current solution s
Output: Solution

for i ← 1...|C| do
  if SolutionCovers(s, c_i) = FALSE then
    Γ' ← Γ ∪ {c_i}
    s ← Extend({c_1, ..., c_{i-1}}, Γ', Solve(Γ'))
  return s

Function SolutionCovers(s, c)

Input: Solution s, item c
Output: TRUE if item c is covered by solution s

test if solution s covers item c, and return TRUE if so

Function Solve(C)

Input: Set of items C
Output: Solution

calculate and return the 1-center of the items C
need to minimize
\[ U(s) = \max_{i=1, \ldots, n} \text{dist}(s, c_i). \]

Note that the weighted 1-center problem reduces to this problem if all customer weights are equal.

For the weighted 1-center problem, a candidate solution consists of the 1-center (a point in \( \mathbb{R}^2 \)), and a radius (the maximum weighted distance to any customer). Testing a new point in \text{SolutionCovers} is simply a matter of seeing if the weighted distance between the point and the 1-center exceeds the radius, an \( O(1) \) operation. It can be shown that at most three customers will comprise the determining set for this problem, which implies an \( O(1) \) running time for \text{Solve}. Thus the expected running time for Welzl's algorithm applied to this problem is \( O(n) \), the same as that for the smallest enclosing disk problem.

### 6.2 A MinMax depots problem approximation algorithm

We now present a fast approximation algorithm for the MinMax depots problem, which will rely on finding a solution to the weighted 1-center problem described in the previous section.

**Definition** A *bounding circle* for a customer \( c_i \) and maximum weighted round trip distance \( r \) is the circle centered at \( c_i \) with radius \( r/(2 \cdot w_i) \), and is denoted \( C_{i,r} \); see figure 2.1.

Lemma 2.1.1 implies that if the value of the objective function (1.1) for some service center \( s \) does not exceed some value \( r \), then for every customer \( c_i \), \( s \) must lie within \( C_{i,r} \).

**Definition** The *approximation distance* is
\[ \bar{r} = \min_r \left\{ \cap_{1 \leq i \leq n} C_{i,r} \neq \emptyset \right\}. \]

**Definition** A *MinMax approximation center* is any point
\[ \bar{s} \in \cap_{1 \leq i \leq n} C_{i,\bar{r}}. \]

**Lemma 6.2.1.** If the customers are distinct, \( \bar{s} \) is unique.

**Proof** Assume two distinct points \( \bar{s}_1 \) and \( \bar{s}_2 \) exist. Both points are inside every circle, and since circles are convex, every point on the line segment between them is also inside every circle. Therefore, there exists a point which is in the interior of every circle, so \( \bar{r} \) is not minimal; a contradiction. \( \square \)
Lemma 6.2.2. If \( V_1 \) is an ordered set of points, and \( V_2 \) is \( V_1 \) with one or more points removed, then
\[
\rtd(V_2) \leq \rtd(V_1).
\]

Proof By the triangle inequality.  

Let \( I_s \) denote the set of optimal depots assigned to customers for service center location \( s \). Let \( s^* \) be a location that minimizes equation (1.1).

Lemma 6.2.3. \( \bar{r} \leq F(s^*) \).

Proof Assume \( \bar{r} > F(s^*) \). By (1.1), there exists a vector \( I_{s^*} \) of customer-to-depot assignments that satisfies
\[
\forall 1 \leq i \leq n \ w_i \cdot \rtd(s^*, c_i, I_{s^*}[i]) \leq F(s^*) < \bar{r}.
\]

By lemma 6.2.2,
\[
\forall 1 \leq i \leq n \ w_i \cdot \rtd(s^*, c_i) < \bar{r},
\]
which implies
\[
\forall 1 \leq i \leq n \ \dist(s^*, c_i) < \bar{r}/(2 \cdot w_i).
\]
This means every customer \( c_i \) is strictly inside \( C_{i,\bar{r}} \), so \( \bar{r} \) is not minimal; a contradiction.  

We are now ready to prove the approximation bound.

Theorem 6.2.4. Choosing \( \tilde{s} \) as the service center location is a 2-approximation algorithm for the MinMax collection depots problem.

Proof Let \( d_j = I_{s^*}[i] \) for each \( c_i \). By equation (1.1),
\[
F(\tilde{s}) \leq \max_{i=1,\ldots,n} w_i \cdot \rtd(c_i, d_j, \tilde{s}).
\]

By lemma 6.2.2,
\[
\max_{i=1,\ldots,n} w_i \cdot \rtd(c_i, d_j, \tilde{s}) \leq \max_{i=1,\ldots,n} w_i \cdot \rtd(c_i, d_j, s^*, c_i, \tilde{s})
\]
\[
\leq \max_{i=1,\ldots,n} w_i \cdot \rtd(c_i, d_j, s^*) + \max_{i=1,\ldots,n} w_i \cdot \rtd(c_i, \tilde{s}).
\]

51
By equation (1.1),
\[
\max_{i=1,\ldots,n} w_i \cdot \text{rtd}(c_i, d_j, s^*) + \max_{i=1,\ldots,n} w_i \cdot \text{dist}(c_i, \tilde{s}) \leq F(s^*) + \max_{i=1,\ldots,n} 2 \cdot w_i \cdot \text{dist}(c_i, \tilde{s}),
\]
and by the definition of \( \tilde{s} \),
\[
F(s^*) + \max_{i=1,\ldots,n} 2 \cdot w_i \cdot \text{dist}(c_i, \tilde{s}) = F(s^*) + \hat{\tau}.
\]
By lemma 6.2.3,
\[
F(s^*) + \hat{\tau} \leq 2 \cdot F(s^*).
\]
Thus \( F(\hat{s}) \leq 2 \cdot F(s^*) \).

We now prove that this approximation bound is "tight" by showing how to construct a set of customers and depots for which the ratio is arbitrarily close to 2. Consider customer \( c_1 \) at \((0, 0)\), \( c_2 \) at \((1, 0)\), and a single depot at \([1+t, 0]\) with \( t > 0 \). Let \( c_1 \) have weight 1 and \( c_2 \) have weight \((1 + t)/t\); see figure 6.1.

Figure 6.1: Proving MinMax approximation bound is tight.

Clearly one possible choice for \( s^* \) is \((1 + t, 0)\), so by equation (1.1),
\[
F(s^*) = \max \{1 \cdot 2(1 + t), (1 + t)/t \cdot 2t\} = 2(1 + t).
\]

We can calculate \( \tilde{s} \) as being the point \((u, 0)\) between \( c_1 \) and \( c_2 \) where the weighted round trip distance from each customer is the same:
\[
1 \cdot u = \frac{(1 + t)/t \cdot (1 - u)}{u} = \frac{1}{2u + 1}.
\]
By equation (1.1),
\[
F(\tilde{s}) = \max \{1 \cdot 2(1 + t), (1 + t)/t \cdot 2(1 - u + t)\} = \frac{4(1 + t)^2}{1 + 2t}.
\]
The approximation ratio is thus

\[ F(\hat{s})/\tilde{F}(s^*) = \frac{2(1 + t)}{1 + 2t} , \]

which approaches 2 as \( t \) approaches 0.

The MinMax approximation center \( \hat{s} \) is the weighted 1-center of the customers, ignoring the depots. As discussed in section 6.1, this can be calculated in \( O(n) \) time. Therefore

**Lemma 6.2.5.** A 2-approximation to the MinMax depots problem can be determined in \( O(n) \) time.

We point out that \( \hat{s} \) is still a 2-approximation bound for the MinMax restricted collection depots problem. In this case, the vector \( I_{s^*} \) may change, but the proof of theorem 6.2.4 is still valid.

### 6.3 A MinSum depots problem approximation algorithm

If an assignment of a depot to each customer is fixed, the collection depots location problem is equivalent to the round trip problem. It was pointed out in [13] that under the MinSum objective, this problem reduces to the classical MinSum problem. We will base our approximation algorithm for the MinSum depots problem on the round trip problem.

Let \( t^* \) be a location that minimizes (1.2). \( I_s \) is defined as in the previous section, since the optimal assignment of depots to customers for fixed \( s \) is the same for both MinMax and MinSum versions of the problem.

**Definition** The MinSum approximation center is the optimal service center location for the MinSum depots problem where each customer uses its nearest depot, and is denoted \( \hat{t} \).

Let \( K \) be an assignment vector of length \( n \) where \( K[i] \) is the nearest depot to customer \( c_i \).

**Theorem 6.3.1.** Choosing \( \hat{t} \) as the service center location is a 2-approximation algorithm for the MinSum collection depots problem.

**Proof** If \( s \) is a point in the plane, the shortest round trip distance involving \( s \), some customer \( c \), and any depot is no shorter than the round trip distance involving just \( c \) and the depot
closest to \( c \). We use this fact in the last inequality below.

\[
G(\tilde{t}) = \sum_{i=1}^{n} \left( w_{i} \cdot rtd(\tilde{t}, \tilde{t}) \right) \quad \text{(equation (1.2))}
\]

\[
\leq \sum_{i=1}^{n} \left( w_{i} \cdot rtd(\tilde{t}, K[i], t^*) \right) \quad \text{(by optimality of \( t^* \))}
\]

\[
\leq \sum_{i=1}^{n} \left( w_{i} \cdot rtd(\tilde{t}, K[i], t^*) \right) \quad \text{(by lemma (6.2.2))}
\]

\[
= \sum_{i=1}^{n} \left( w_{i} \cdot (rtd(c_i, K[i]) + rtd(\tilde{t}, K[i], t^*)) \right) \quad \text{(by optimality of \( K[i] \) w.r.t. \( K \))}
\]

\[
= 2 \cdot G(t^*) \quad \text{(equation (1.2))}
\]

This approximation bound is also tight. Consider customers \( c_1 \) at \((1,0)\) and \( c_2 \) at \((2,0)\), and depots \( d_1 \) at \((q,0)\) and \( d_2 \) at \((2,0)\) with \(0 < q < 1\). Let \( c_1 \) have weight \( 1 + q \) and \( c_2 \) have weight \( 1 \); see figure 6.2.

![Figure 6.2: Proving MinSum approximation bound is tight.](image)

Since \( c_1 \) is closest to \( d_1 \), and \( c_2 \) is closest to \( d_2 \), the approximation algorithm will place \( \tilde{t} \) at the point \((u,0)\) that minimizes (from equation (1.2)):

\[
\left\{ \begin{array}{ll}
(1 + q) \cdot 2(1 - q) + 2(2 - u) & \text{if } q \leq u < 1, \\
(1 + q) \cdot 2(u - q) + 1 \cdot 2(2 - u) & \text{if } 1 \leq u \leq 2.
\end{array} \right.
\]

The point \( \tilde{t} \) is therefore fixed at \((1,0)\) for any valid \( q \).

\( t^* \) is set to \((u,0)\) that minimizes:

\[
\left\{ \begin{array}{ll}
(1 + q) \cdot 2(1 - q) + 1 \cdot 2(2 - u) & \text{if } q \leq u < 1, \\
(1 + q) \cdot 2(u - q) + 1 \cdot 2(2 - u) & \text{if } 1 \leq u < (1 + q)/2, \\
(1 + q) \cdot 2(1) + 1 \cdot 2(2 - u) & \text{if } (1 + q)/2 \leq u \leq 2.
\end{array} \right.
\]

\( t^* \) is therefore fixed at \((2,0)\) for any valid \( q \).

As \( q \) approaches 0, \( G(t^*) \) is \( 2(1+q)(1-q) + 2 \), while \( G(\tilde{t}) \) is \( 2(1+q)(1-q) + 2 \), so the approximation ratio \( G(\tilde{t})/G(t^*) \) is \( \frac{2(1+q)(1-q) + 2}{2(1+q)(1-q) + 2} \) which approaches 2 as \( q \) approaches 0.
Lemma 6.3.2. \( \tilde{t} \) can be calculated in \( O((n + p) \lg p) \) time plus the time it takes to solve the classical MinSum problem of \( O(n) \) points.

Proof The assignment vector \( K \) can be determined in \( O((n + p) \lg p) \) time by first constructing a standard Voronoi diagram with depots as sites, then querying the diagram for each customer. By using the same reasoning found in the proof of theorem 4.3.1, once \( K \) is known, \( \tilde{t} \) can be determined by solving the classical MinSum problem of at most \( 2n \) points.

The same technique can be used to produce a 2-approximation bound for the MinSum restricted collection depots problem. The only changes required are to restrict the assignment vectors \( I_s \) and \( K \) to use the appropriate subsets of \( D \) for each customer, and to change the definition of \( \tilde{t} \) accordingly. The proof of theorem 6.3.1 is otherwise unchanged.

6.4 The \( L_1 \) metric

The approximation algorithms presented in the previous sections can be used, with minor modifications, for the \( L_1 \) metric as well. For the MinMax depots problem, we set \( \tilde{s} \) to the \( L_1 \) weighted 1-center of the customers, which can be calculated in linear time [24]. The proofs of lemma 6.2.3 and theorem 6.2.4 can be slightly modified to prove that \( \tilde{s} \) gives a 2-approximation for the MinMax depots problem. For the MinSum depots problem, we follow the same steps as in section 6.3 by assigning each customer to its nearest depot, then solving an \( L_1 \) classical MinSum problem of \( O(n) \) points to calculate \( \tilde{t} \). As noted in section 1.4, this can be done in \( O(n) \) time. Assigning depots to customers therefore dominates the running time, which is \( O((n + p) \lg p) \). The proof for theorem 6.3.1 can be easily adapted for the \( L_1 \) metric. Note that the approximation bounds are tight in this metric as well, since the examples given in each of the previous sections are restricted to a line.
Chapter 7

A Randomized MinMax depots algorithm

In this chapter, we will show how the FindCenter algorithm described in section 6.1 can be modified to solve the MinMax depots problem. We will then examine its expected running time.

Recall that to apply the FindCenter algorithm to a particular type of center problem, we need to define what a solution is, as well as the functions SolutionCovers and Solve. For the MinMax depots problem, a solution will consist of a center (a point representing the service center location), and a distance (the maximum weighted round trip distance for any customer). SolutionCovers takes a solution and a customer as input, and determines if the customer can participate in a round trip with the center and some depot. Since the center and customer are known at this point, the best choice for a depot can be found by querying the customer’s Voronoi diagram. If the weighted round trip distance between the customer, center, and the depot is not greater than the solution distance, SolutionCovers returns TRUE. Solve takes as input \( \Gamma \), a subset of the customers, and \( D \), the complete set of depots, and finds the optimal solution that is guaranteed to cover the subset of customers. We will discuss two approaches to solving this problem.
7.1 Implementing Solve

7.1.1 Discrete approach

In the first approach, we use the same method of section 4.2 to enumerate all feasible assignments of depots to customers in $\Gamma$, then test each set of assignments for optimality.

Let $p = |D|$ and $\lambda = |\Gamma|$, and let $s^*$ represent the solution center. By theorem 4.1.1, there are $O(p^2\lambda^2)$ feasible assignments, which by lemma 4.1.2 can be generated in $O(p^2\lambda^2 \lg(p\lambda))$ time.

For each set of assignments of depots to customers, determining $s^*$ is the round trip problem (see section 1.4). We now present a solution to this problem.

We first derive some properties of $s^*$. Let $d_i$ refer to the depot that has been assigned to customer $c_i$. Recall from chapter 2 that for a solution with distance $r$, the set of possible solution centers that can participate in a weighted round trip with customer $c_i$ and depot $d_i$ is the ellipse $E_{c_i,d_i,r/w_i}$. In this section, for brevity, we will denote this ellipse $E_{c_i}$.

Lemma 7.1.1. If $\Gamma = \{c_1\}$, some optimal $s^*$ must occur at $c_1$.

Proof Since only one customer determines the solution, the optimal service center can appear anywhere in the degenerate ellipse $E_{c_1}$; certainly $c_1$ is one such point.

Lemma 7.1.2. If $\Gamma = \{c_1, c_2\}$, some optimal $s^*$ must satisfy lemma 7.1.1 for some single customer of $\Gamma$; otherwise (see figures 7.1.a and 7.1.b), an optimal $s^*$ must occur on the boundaries of both $E_{c_1}$ and $E_{c_2}$ (for some optimal $r^*$), and in addition, if neither ellipse is degenerate, $s^*$ is the only point of intersection between them.

Proof Assume $E_{c_1}$ is degenerate. Either $E_{c_1}$ is contained in $E_{c_2}$, in which case any point satisfying lemma 7.1.1 for $\{c_1\}$ will cover $c_2$ as well, or part of $E_{c_1}$ lies outside $E_{c_2}$, in which case $s^*$ can occur at a point of intersection of their boundaries.

If neither ellipse is degenerate, then clearly one ellipse cannot contain the other, or else both radii could decrease and still leave a point of intersection. By the same reasoning, their intersection cannot consist of a region with nonzero area, since decreasing the radii could still leave a point of intersection; thus $s^*$ must be a single point of tangency between the two ellipses.
Lemma 7.1.3. If $\Gamma = \{c_1, \ldots, c_\lambda\}$ with $\lambda \geq 3$, then some optimal $s^*$ must occur on the boundaries of three non-degenerate customer ellipses (figure 7.1.c), or must satisfy the conditions of lemma 7.1.2 for some pair of customers in $\Gamma$.

Proof If some ellipse $E_{c_i}$ is degenerate, let $M$ represent the portion of $E_{c_i}$ that can contain $s^*$. Either $M = E_{c_i}$, in which case lemma 7.1.1 is satisfied for $\{c_i\}$, or there is some other ellipse $E_{c_j}$ intersecting an endpoint of $M$, in which case lemma 7.1.2 is satisfied for $\{c_i, c_j\}$.

Otherwise, no ellipses are degenerate. If $s^*$ occurs on the boundary of three or more ellipses, we’re done. If $s^*$ occurs on the boundary of less than two ellipses, then all the ellipse radii can shrink, and $s^*$ is not optimal. If $s^*$ occurs on the boundary of exactly two ellipses $E_{c_i}$ and $E_{c_j}$, then neither of the two radii can shrink, else $s^*$ is not optimal; thus lemma 7.1.2 is satisfied for $\{c_i, c_j\}$ (figure 7.1.b).

The above lemmas allow us to find an optimal solution by testing a discrete set of points. Lemma 7.1.1 implies that each of the $\lambda$ customer locations are candidates. Lemma 7.1.2 implies that we should also examine, for each $\binom{\lambda}{2}$ pairs of customers, the single point of tangency (for nondegenerate ellipses) and a single point of intersection (for a degenerate ellipse). Lemma 7.1.3 implies that we should also examine, for each $\binom{\lambda}{3}$ customer triples, the point of intersection of each customer’s ellipse boundary (where $r$ is minimal). Generating a candidate solution center and distance can be done in $O(1)$ time (though the algebra is rather involved in the last case), and there are $O(\lambda + 2\binom{\lambda}{2} + \binom{\lambda}{3}) = O(\lambda^3)$ candidates.

Once a candidate solution has been generated, testing if it covers all of the customer/depot pairs can be done in $O(\lambda)$ time. Therefore

Lemma 7.1.4. The round trip problem for $n$ customers and $p$ depots can be solved in $O(n^4)$ time.
Finding the optimal solution for a particular assignment of depots to the customers \( \Gamma \) can thus be done in \( O(\lambda^4) \) time.

By theorem 4.2.2, we can solve the MinMax depots problem for \( \lambda \) customers and \( p \) depots in \( O(p^2\lambda^2 \lg(p\lambda) + p^2\lambda^6) \) time, so this is the running time for \( \text{Solve} \). Note that for very small \( \lambda \), its running time is \( O(p^2 \lg p) \).

### 7.1.2 Parametric search

The second approach we present for implementing \( \text{Solve} \) is to use the plane sweep and parametric search combination proposed by Tamir and Halman [35], which we discussed in section 4.2. This has a running time of \( O(p^2\lambda^2 \lg^3(p\lambda)) \).

### 7.2 Running time analysis

The expected running time of the \( \text{FindCenter} \) algorithm is linear in the number of items, or customers in this case. The expected number of calls to \( \text{SolutionCovers} \) and \( \text{Solve} \) is thus \( O(n) \). To determine the best choice of depot for a particular customer and service center (required in \( \text{SolutionCovers} \)), we can construct, in \( O(pm \lg p) \) time, a set of \( n \) Voronoi diagrams. Each \( \text{SolutionCovers} \) call can then execute in \( O(\lg p) \) time.

The efficiency of the \( \text{FindCenter} \) algorithm is conditioned on the size of the determining set of customers being small. For both the smallest enclosing disk problem and the weighted 1-center problem, for instance, the size of the determining set is at most three.

For the MinMax depots problem, the maximum size of this set (which we will denote by \( \lambda_{C,D} \)) is an open problem.

**Conjecture 7.2.1.** \( \lambda_{C,D} \) is bounded by some small integer.

It is easy to construct examples where \( \lambda_{C,D} = 4 \); see figure 7.2. Figure 7.3 is an example where \( \lambda_{C,D} = 5 \). We have not found an example where \( \lambda > 5 \). We have run the \( \text{FindCenter} \) algorithm with many randomly generated customers and depots, and it has never encountered an example where \( \lambda > 5 \) (\( \lambda = 4 \) is common, and \( \lambda = 5 \) occurs infrequently).

In these figures, the customers are labelled \( a, \ldots, e \), and the optimal service center location is labelled \( s^* \). There exist nonempty regions of intersection between every \( (\frac{\lambda}{\lambda-1}) \) of customer ellipse sets; these are labelled \(-q\), where \( q \) is the excluded customer. If customer
$c_q$ is removed from $C$, $s^*$ will move to the interior of $-q$, and the value of equation (1.1) will decrease.

If conjecture 7.2.1 is true, then $\lambda$ is bounded by a constant, and the running time for \texttt{Solve} using the first approach (section 7.1.1) is $O(p^2 \lg p)$, while for the second approach (section 7.1.2) it is $O(p^2 \lg^3 p)$.

The expected running time for \texttt{FindCenter} is $T_1 + n(T_2 + T_3)$, where $T_1$ is the time to construct the Voronoi diagrams for the customers, $T_2$ is the time for each call to \texttt{SolutionCovers}, and $T_3$ is the time for each call to \texttt{Solve} (using the first approach). The total expected running time is thus

$$O(T_1 + n(T_2 + T_3)) = O(pn \lg p + n(p^2 \lg p)) = O(p^2 n \lg p)$$

If the conjecture is true, the expected running time of the algorithm is linear in the number of customers. In addition, if the number of customers is much larger than the number of depots (which could be expected, for example, if customers are households and depots are hospitals), then the expected running time of the algorithm is linear.
7.3 The $L_1$ metric

It is an open question, even in the $L_1$ metric, of whether $\lambda_{C,D}$ is bounded by some small integer. Figure 7.4 is an example where $\lambda_{C,D} = 4$. We have not found an example where $\lambda > 4$ in this metric.
Figure 7.4: $\lambda_{C,D} = 4$, $L_1$ metric.
Chapter 8

Additional applications

In this chapter, we will show how the technique of merging Voronoi diagrams to generate a list of feasible depot assignments can be applied to problems involving line barriers.

8.1 The MinSum barrier problem

Klamroth [20] investigated the classical MinSum problem in the presence of a single line barrier, which can represent a river, highway, or other border.

Let \( \{c_1, \ldots, c_n\} \) represent existing facilities (which we will refer to as customers) in the plane, each associated with a positive weight \( w_i \). Without loss of generality, we assume the barrier is a horizontal line with a set \( D = \{d_1, \ldots, d_p\} \) of points on the line representing passages across the barrier.

The objective function to be minimized is

\[
G(s) = \sum_{i=1}^{n} w_i \cdot \text{dist}'(s, c_i)
\]  

(8.1)

where the distance function \( \text{dist}'(a, b) \) is defined as the shortest path from \( a \) to \( b \) that doesn’t cross the barrier except possibly at one of the passages. Finding \( s \) which minimizes (8.1) is the MinSum barrier problem.

Klamroth points out that the optimal \( s \) exists on one of the two sides of the barrier, so the solution to the problem can be viewed as the best of the solutions of the subproblems to either side. To solve a subproblem, partition the customers into two sets \( C^1 \) and \( C^2 \) according to which side of the barrier they lie on. Then the optimal \( s \) will be that which
minimizes one of

\[ G^{(1)}(s) = \sum_{c \in C^1} w(c) \cdot \text{dist}(s, c) \]
\[ + \sum_{c \in C^2} w(c) \cdot \left\{ \min_{j=1, \ldots, p} \{ \text{dist}(s, d_j) + \text{dist}(d_j, c) \} \right\} \]  \hspace{1cm} (8.2)

\[ G^{(2)}(s) = \sum_{c \in C^2} w(c) \cdot \text{dist}(s, c) \]
\[ + \sum_{c \in C^1} w(c) \cdot \left\{ \min_{j=1, \ldots, p} \{ \text{dist}(s, d_j) + \text{dist}(d_j, c) \} \right\} \]  \hspace{1cm} (8.3)

Once the assignments of passages to those customers lying on the opposite side of the barrier from \( s \) are made, the final distance terms in (8.2) and (8.3) become constants, and the problem reduces to the classical MinSum problem.

Klamroth presented an algorithm which reduces the MinSum barrier problem with \( p \) passages to \( O(\binom{n+p-1}{p-1}) \) classical MinSum subproblems. This is an improvement over a simple enumeration of all possible selection of passages, which produces \( O(p^n) \) subproblems, yet is still exponential in \( p \).

To improve these results, we construct, for each customer \( c \), a Voronoi diagram associating each point \( s \) in the halfplane opposite the customer with the passage \( d_i \) which minimizes \( \text{dist}'(c, s) \). We restrict the diagram to the halfplane on the opposite side of the barrier from \( c \), since these are the only points for which passage assignments are necessary.

The partitioning curve between regions for passages \( d_1 \) and \( d_2 \) are the points \( q \) which satisfy

\[ \text{dist}(c, d_1) + \text{dist}(d_1, q) = \text{dist}(c, d_2) + \text{dist}(d_2, q) \]

which can be rearranged to form equation (3.3). Thus the Voronoi diagram we seek is exactly that portion of \( V_{c,D} \) which lies on the appropriate side of the barrier.

A Voronoi diagram for \( p \) barriers will consist of \( p-1 \) nonintersecting hyperbolic arcs, as figure 8.1 shows.

**Theorem 8.1.1.** The MinSum barrier problem with \( n \) customers and \( p \) passages can be solved in \( O(p^2n^2(T(n) + \log(pn))) \) time, where \( T(n) \) is the time it takes to solve the classical MinSum problem of \( O(n) \) points.
Proof Since the Voronoi diagram for the barrier problem is a subset of $V_{c,D}$, we can directly apply theorems 4.1.1 and 4.1.2 to show that at most $O(p^2n^2)$ different feasible assignments of passages to customers are possible. Each such set of assignments reduces the line barrier problem to the classical MinSum problem. Equations (8.2) and (8.3) imply that the number of facilities in each subproblem is at most $n$.

The $O(p^2n^2)$ bound on the number of feasible assignments in the proof of theorem 8.1.1 is tight. Using an approach similar to that employed in section 4.1, we can construct the example of figure 8.2.

Figure 8.1: Voronoi diagram for customer, barrier problem.

Figure 8.2: Worst case situation, barrier problem.
8.2 The MinMax barrier problem

If we replace equation (8.1) with the objective function

\[ F(s) = \max_{i=1,...,n} w_i \cdot \text{dist}'(s, c_i) \]

we get the MinMax barrier problem.

As with the MinSum barrier problem, the approach taken to solve the MinMax barrier problem is to search on both sides of the barrier and choose the best of the two solutions.

To find \( s \) on a particular side of the barrier, the problem is very similar to the MinMax depots problem of section 4.2. Instead of ellipse sets, here we are concerned with circle sets: a union of circles for each customer. If the customer is on the same side of the barrier as \( s \), this union will consist of a single circle centered at the customer; otherwise, it will consist of a set of at most \( p \) circles centered at each passage. Each circle in a set has a radius that is additively weighted by the (negative) distance of its center from the customer, and is multiplicatively weighted by the inverse of the customer's weight. In both cases, the circles are cropped to exclude the halfplane which doesn't contain \( s \).

A circle set represents exactly those points in the halfplane of \( s \) that satisfy \( \text{dist}'(q, c_i) \leq r/w_i \), for customer \( c_i \) (with weight \( w_i \)) and maximum weighted distance \( r \). We will denote this circle set \( C_{c,r} \).

**Lemma 8.2.1.** The complexity of a circle set of \( p \) passages is \( O(p) \).

**Proof** If we add circles to the set in nonincreasing order of radius, it is easy to see that each additional circle can add at most one circular arc and one horizontal segment to the union's boundary.

**Theorem 8.2.2.** The MinMax barrier problem with \( n \) customers and \( p \) passages can be solved in \( O(p^2n^2 \lg^3(pn)) \) time.

**Proof** The problem reduces to finding the smallest value \( r \) for which \( \bigcap_{c \in C} C_{c,r} \) is not empty. By lemma 8.2.1, the complexity of each customer's circle set is \( O(p) \); thus the intersection of \( n \) such sets has \( O(p^2n^2) \) complexity in the worst case, and the optimal \( r \) can be found using the same plane sweep and parametric search combination employed to solve the MinMax depots problem. We conjecture that the above \( O(p^2n^2) \) bound is not tight, since the sets are composed of circles, and in the case of the set being consisting of more than one circle, their centers are fixed at the passages.
8.3 The room problem

In this section, we investigate an extension of the barrier problem. Consider $n$ customers inside a polygon that has been partitioned into $m$ smaller polygons by a set of linear barriers; see figure 8.3. In this context, we call the smaller polygons rooms, the barriers walls, and the $p$ passages doors. We assume each room is convex. (The proposed algorithm can also be extended to handle nonconvex rooms; the details are tedious, but straightforward.)

![Polygon partitioned by barriers into convex rooms, with customers and doors.](image)

Figure 8.3: Polygon partitioned by barriers into convex rooms, with customers and doors.

We wish to find the location within a particular room that minimizes

$$G(s) = \sum_{i=1}^{n} w_i \cdot \text{dist}'(s, c_i)$$

(8.4)

which is the same as equation (8.1), but with the distance $\text{dist}'(a, b)$ representing the shortest path from $a$ to $b$ that doesn't cross any walls at points other than doors. We will call the task of finding a location $s$ within a particular room that minimizes equation (8.4) the room problem.

We assume that there exists a network $N$ with $p$ nodes, with edges between two nodes if the corresponding doors share a room. In the worst case, the network $N$ has $O(p^2)$ edges. The convexity of the rooms ensures that the distance represented by each edge is simply the Euclidean distance between the nodes. It follows that the edges do not need to be explicitly stored, since each door is on the boundary of two rooms, and an edge is implied between this door and every door in either of these two rooms.

**Theorem 8.3.1.** The room problem with $n$ customers and $p$ doors can be solved in $O(p^2 n^2 (T(n) + \lg(pn)))$ time, where $T(n)$ is the time it takes to solve the classical MinSum problem of $O(n)$ points.
Proof For every customer \( c_i \), we determine the single-source shortest paths from \( c_i \) to every door. This can be done as follows. We first augment the network \( N \) by adding the vertex \( c_i \) and the edges from \( c_i \) in a room to all the doors of the room. We then solve the single source shortest path problem in the augmented network with \( c_i \) as the source node. Dijkstra's algorithm [10] can be applied to solve the problem in \( O(p^2) \) time.

We repeat the process for each customer in \( C \). Therefore, in \( O(n \cdot p^2) \) time, we can compute the distance of all the customers to all the doors of the rooms. Since the edges do not need to be explicitly stored, the storage space requirement is \( O(n \cdot p) \).

Let \( h \) be the room containing \( s \). As in the previous section, we partition the customers \( C \) into two sets: \( C_1 \), the customers in room \( h \), and \( C_2 \), those not in room \( h \). We also construct, for each room \( h \), the set of doors that exist in \( h \), and denote this set \( D^h \) (with \( p(h) = |D^h| \)).

Equation (8.4) becomes

\[
G(s) = \sum_{c \in C_1} w(c) \cdot \text{dist}(s, c) + \sum_{c \in C_2} w(c) \cdot \left\{ \min_{j=1,\ldots,p(h)} \{ \text{dist}(s, d^h_j) + \text{dist}'(d^h_j, c) \} \right\}
\]

Observe that for fixed room \( h \), \( \text{dist}'(d^h_j, c) \) is a constant precomputed by the single source shortest path algorithm, and can be used as an additive weight for customer \( c \) when we construct a Voronoi diagram with the doors from room \( h \) as sites.

We can construct this diagram for a customer and room in \( O(p \lg p) \) time, once the additive weights have been determined. We then proceed as in the previous section, merging the \( n \) diagrams together to get \( O(p^2 n^2) \) sets of feasible assignments of doors to customers. Each of these sets reduces the problem to the classical MinSum problem of \( n \) points.

One observation we make is that the Voronoi diagrams constructed for the customers in the room problem are not the same as those for the collection depots problem, since the additive weights for the doors are not simply the distance of each door from some common point. Instead, they are general additively weighted Voronoi diagrams (which can still be constructed in \( O(p \lg p) \) time using Fortune's algorithm [16]). We can apply lemma 3.4.1 to show that each region in such a diagram is star-shaped from its site, in a manner analogous to that of the proof of lemma 3.3.4. We can then construct a proof similar to that of lemma 3.3.5 to show that an additively weighted Voronoi diagram of \( p \) sites has \( O(p) \) complexity, which ensures that any arrangement of \( n \) such diagrams has \( O(p^2 n^2) \) complexity.
Note that we can find the optimal $s$ over all rooms by choosing the best solution from $m$ room problems.

The MinMax version of the room problem is very similar to the MinMax barrier problem, and can be solved with the same approach (and running time) given in the proof of theorem 8.2.2.
Chapter 9

Summary

In this thesis, we have studied both the MinMax and MinSum variants of the planar collection depots location problem. We have examined these problems in the standard Euclidean ($L_2$) metric, as well as in the rectilinear ($L_1$) metric.

In the next sections, we describe our results, and discuss areas for further research.

9.1 Results

In chapter 2, we examined the role that ellipses, octagons, and hyperbolas play in the collection depots location problem. We have proven a linear complexity for a union of ellipses with a common focus, which improves the bound of [35].

Every customer is associated with a particular type of Voronoi diagram, with the depots as sites. In chapter 3, we proved that they are a restricted type of additively weighted Voronoi diagram, and that they have linear complexity. In chapter 5, we developed the Circle Sweep algorithm which can construct them in optimal time and space in both $L_2$ and $L_1$ metrics. We also proved that this algorithm is optimal even if the depots are already sorted either by angle around the customer or by distance from the customer.

In chapter 4, we used these Voronoi diagrams to enumerate candidate solutions for the collection depots location problem. We solved an open problem posed by [13] by proving that at most $O(p^2 n^2)$ different feasible assignments of depots are possible for any choice of service center in $L_2$ or $L_1$ metrics, and in the case of the $L_2$ metric, we have shown that this bound is tight. We also showed how, using Voronoi diagrams, these assignments can be generated in time $O(p^2 n^2 \lg(pn))$. 
For the first time, we have solved the MinSum depots problem by showing that by using Voronoi diagrams, it can be reduced in $O(p^2n^2 \log(pn))$ time to $O(p^2n^2)$ classical MinSum problems of $O(n)$ points (section 4.3).

In chapter 6, we presented fast 2-approximation algorithms for both the MinMax and MinSum collection depots problems, for both the $L_2$ and $L_1$ metrics; to the best of our knowledge, these are the first approximation algorithms given for the continuous collection depots location problem.

We have developed an abstract algorithm, FindCenter, as a generalization of that of Welzl [37], which can be used to efficiently solve 1-center problems in which the size of the determining set of the solutions is small. In section 4.2, we presented a method for solving the MinMax depots problem that is simpler than that of [35] (which uses parametric search). We conjectured in section 7.2 that the size of any determining set for the MinMax depots problem is bounded by a small integer. If this is so, then by using the aforementioned algorithm as a subroutine, we can solve the MinMax depots problem in expected $O(p^2n \log p)$ time.

In chapter 8, we showed how our method of generating feasible assignments from merged Voronoi diagrams can be used to improve the solution posed by Klamroth [20] for the barrier problem. We also introduced the room problem, and showed how it can be reduced to a set of classical MinSum problems using the same technique. In addition, we showed that the MinMax barrier and room problems can be solved by using the same technique used to solve the MinMax depots problem.

\section*{9.2 Future research}

It is an open problem of whether the intersection of customer ellipse sets has complexity of $o(p^2n^2)$ for the unrestricted collection depots problem.

It is also an open problem of whether a constant bound exists on the number of customers that determine the solution to the MinMax depots problem (as well as the MinMax versions of the barrier and room problems). A constant bound must be established to prove conjecture 7.2.1, which in turn will prove that the MinMax depots problem can be solved in expected time $O(p^2n \log p)$.

We conjecture that the bound of theorem 4.1.1 on the number of feasible assignments to customers in the collection depots location problem is not tight in the $L_1$ metric.

71
Bibliography


[34] A. Tamir and N. Halman. Private communication, 2005.

