

Algorithms for Geometric Covering and Piercing Problems

by

Robert Fraser

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2012

© Robert Fraser 2012

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

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

Abstract

This thesis involves the study of a range of geometric covering and piercing problems, where the unifying thread is approximation using disks. While some of the problems addressed in this work are solved exactly with polynomial time algorithms, many problems are shown to be at least NP-hard. For the latter, approximation algorithms are the best that we can do in polynomial time assuming that $P \neq NP$.

One of the best known problems involving unit disks is the Discrete Unit Disk Cover (DUDC) problem, in which the input consists of a set of points \mathcal{P} and a set of unit disks in the plane \mathcal{D} , and the objective is to compute a subset of the disks of minimum cardinality which covers all of the points. Another perspective on the problem is to consider the centre points (denoted \mathcal{Q}) of the disks \mathcal{D} as an approximating set of points for \mathcal{P} . An optimal solution to DUDC provides a minimal cardinality subset $\mathcal{Q}^* \subseteq \mathcal{Q}$ so that each point in \mathcal{P} is within unit distance of a point in \mathcal{Q}^* . In order to approximate the general DUDC problem, we also examine several restricted variants.

In the Line-Separable Discrete Unit Disk Cover (LSDUDC) problem, \mathcal{P} and \mathcal{Q} are separated by a line in the plane. We write that ℓ^- is the half-plane defined by ℓ containing \mathcal{P} , and ℓ^+ is the half-plane containing \mathcal{Q} . LSDUDC may be solved exactly in $O(mn + n \log n)$ time using a greedy algorithm. We augment this result by describing a 2-approximate solution for the Assisted LSDUDC problem, where the union of all disks centred in ℓ^+ covers all points in \mathcal{P} , but we consider using disks centred in ℓ^- as well to try to improve the solution. Next, we describe the Within-Strip Discrete Unit Disk Cover (WSDUDC) problem, where \mathcal{P} and \mathcal{Q} are confined to a strip of the plane of height h . We show that this problem is NP-complete, and we provide a range of approximation algorithms for the problem with trade-offs between the approximation factor and running time.

We outline approximation algorithms for the general DUDC problem which make use of the algorithms for LSDUDC and WSDUDC. These results provide the fastest known approximation algorithms for DUDC. As with the WSDUDC results, we present a set of algorithms in which better approximation factors may be had at the expense of greater running time, ranging from a 15-approximate algorithm which runs in $O(m^6n + n \log n)$ time to a 18-approximate algorithm which runs in $O(mn + m \log m + n \log n)$ time.

The next problems that we study are Hausdorff Core problems. These problems accept an input polygon P , and we seek a convex polygon Q which is fully contained in P and minimizes the Hausdorff distance between P and Q . Interestingly, we show that this problem may be reduced to that of computing the minimum radius of disk, call it k_{OPT} , so that a convex polygon Q contained in P intersects all disks of radius k_{OPT} centred on the vertices of P . We begin by describing a polynomial time algorithm for the simple case where P has only a single reflex vertex. On general polygons, we provide a parameterized algorithm which performs a parametric search on the possible values of k_{OPT} . The solution to the decision version of the problem, i.e. determining whether there exists a Hausdorff Core for P given k_{OPT} , requires some novel insights. We also describe an FPTAS for the decision version of the Hausdorff Core problem.

Finally, we study Generalized Minimum Spanning Tree (GMST) problems, where the input consists of imprecise vertices, and the objective is to select a single point from each imprecise vertex in order to optimize the weight of the MST over the points. In keeping with one of the themes of the thesis, we begin by using disks as the imprecise vertices. We show that the minimization and maximization versions of this problem are NP-hard, and we describe some

parameterized and approximation algorithms. Finally, we look at the case where the imprecise vertices consist of just two vertices each, and we show that the minimization version of the problem (which we call 2-GMST) remains NP-hard, even in the plane. We also provide an algorithm to solve the 2-GMST problem exactly if the combinatorial structure of the optimal solution is known.

We identify a number of open problems in this thesis that are worthy of further study. Among them:

- Is the Assisted LSDUDC problem NP-complete?
- Can the WSDUDC results be used to obtain an improved PTAS for DUDC?
- Are there classes of polygons for which the determination of the Hausdorff Core is easy?
- Is there a PTAS for the maximum weight GMST problem on (unit) disks?
- Is there a combinatorial approximation algorithm for the 2-GMST problem (particularly with an approximation factor under 4)?

Acknowledgments

Many of the topics of research in this thesis were initiated within research discussions with members of the Algorithms and Complexity group at the University of Waterloo and visitors to the University who attended the meetings. As a result, I have had the good fortune to work with many talented and brilliant researchers. I have published portions of this work with Francisco Claude, Gautam Das, Reza Dorrigiv, Stephane Durocher, Arash Farzan, Meng He, Shahin Kamali, Akitoshi Kawamura, Alejandro López-Ortiz, Ian Munro, Patrick Nicholson, Bradford Nickerson, Alejandro Salinger, Diego Seco and Matthew Skala, and I greatly appreciate their contributions to this thesis. In particular, I thank my lab-mates Ale, Pat and Tuna for many helpful impromptu discussions.

My thesis committee provided a great number of insightful comments, suggestions and corrections that substantially improved the quality of this work. Thank you Professors Thomas Erlebach, Jochen Koenemann, Anna Lubiw and Ian Munro for your careful reading and hard work. Many details of this thesis were included or improved upon based on comments provided by anonymous reviewers.

My supervisor Professor Alejandro López-Ortiz has provided countless hours of guidance throughout my Ph.D., both in academic and personal contexts. I greatly appreciate all of the time and effort that Alex has invested in my education.

Valuable guidance and corrections for sections of Chapter 2 were provided by Joseph Cheriyan and Raju Jampani. The initial work on the Discrete Unit Disk Cover problem came from discussions with Sarel Har-Peled on a preliminary version of the problem. Paz Carmi was kind enough to share his insights on the problem, and discuss details of his results on the problem from [29]. Diego Arroyuelo and Barbara Macdonald participated in early discussions of the Hausdorff Core problem, and Joseph Cheriyan, Nathan Krislock, and Marcel Silva participated in helpful discussions on optimization problems.

Funding for portions of this research was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), partially under the NSERC Strategic Grant on Optimal Data Structures for Organization and Retrieval of Spatial Data.

I wish to thank my family for their support and encouragement through so many years of study. My parents have unwaveringly supported my meandering path through different universities and fields of study, and I cannot adequately express my gratitude. Thank you Mom, Dad, Wes, Bev, Jim and Rollie. My children have tried their best to give me time to work when I needed it, and both have contributed to this work in their own ways! Finally, I must thank my wonderful, intelligent, and beautiful wife Kelly. She has provided more support than I could have ever asked for. She has taken care of so many little things and big things, ensured that our children are being well raised, and dealt with a husband that was often absent in mind if not body. She has shared in my stress and joy, and given me support every day. I love you Kelly, and I look forward to all of our adventures to come!

Dedication

This thesis is dedicated to my wonderful children Calvin and Alice. You are my greatest treasures.

Contents

List of Tables	xv
List of Figures	xvii
Definitions	xxii
1 Introduction	1
1.1 Organization of the Thesis	2
2 Discrete Unit Disk Cover Essentials	7
2.1 Related Work	9
2.1.1 General Geometric Set Cover	10
2.1.2 Duality for Discrete Unit Disk Cover	12
2.1.3 PTAS for Discrete Unit Disk Cover	12
2.1.4 ϵ -nets	14
2.1.5 Minimum Dominating Set on Unit Disk Graphs	15
2.1.6 Minimum Geometric (Unit) Disk Cover	16
2.1.7 Discrete k -Center	18
2.1.8 Other Discrete Geometric Covering Problems	18
2.2 Specialized Settings	18
2.2.1 Line-Separable Discrete Unit Disk Cover	18
2.2.2 Assisted Line-Separable Discrete Unit Disk Cover	20
2.2.3 Strip-Separable Discrete Unit Disk Cover	21
2.3 38-Approximate DUDC Algorithm	24
2.3.1 The BOX Algorithm	25

3	(Assisted) Line-Separable Discrete Unit Disk Cover	27
3.1	Greedy Algorithm for LSDUDC	27
3.1.1	Correctness of GREEDY	29
3.2	GREEDY-GRAPH	32
3.2.1	Correctness of GREEDY-GRAPH	32
3.3	Assisted LSDUDC	33
3.4	Conclusions and Future Work	35
4	Within-Strip DUDC	37
4.1	6-Approximation Algorithm for $h \leq \frac{1}{\sqrt{2}}$	38
4.2	$3\lceil 1/\sqrt{1-h^2} \rceil$ -Approximation Algorithm for $h < 1$	42
4.2.1	Covering $\mathcal{P}_{\overline{\mathcal{R}}}$	43
4.2.2	Covering $\mathcal{P}_{\mathcal{R}}$	44
4.2.3	Combining Solutions for $\mathcal{P}_{\overline{\mathcal{R}}}$ and $\mathcal{P}_{\mathcal{R}}$	45
4.3	4-Approximation Algorithm for $h \leq 2\sqrt{2}/3$	46
4.4	3-Approximation Algorithm for $h \leq 4/5$	47
4.5	NP-Completeness of WSDUDC	48
4.5.1	Gadgets	54
4.6	Conclusions and Future Work	57
5	Approximation Algorithms for DUDC	59
5.1	22-Approximate DUDC Solution	59
5.2	15-Approximate DUDC Solution	60
5.2.1	Outside Strip Discrete Unit Disk Cover	60
5.2.2	The Discrete Unit Disk Cover Algorithm	63
5.3	Conclusions and Future Work	64
6	Constrained Polygonal Vertex Cover and the Hausdorff Core	65
6.1	Paradigms for Approximating Polygons and Polygonal Chains	66
6.2	Related Work	67
6.2.1	Feature-Based Approximation	67
6.2.2	Mathematical Approximation	67
6.2.3	Error Tolerance-Based Approximation	68
6.2.4	Chain Coding Scheme	70
6.2.5	Two-Strip Solution	70

6.2.6	Error Metrics	70
6.2.7	Constrained Approximation of Polygonal Curves	72
6.2.8	Constrained Approximation of Polygons	73
6.2.9	LP-type Problems	75
6.2.10	Davenport-Schinzel Sequences	76
6.2.11	The Constrained Euclidean 1-Centre	77
6.3	The Hausdorff Core	78
6.3.1	Definitions	78
6.3.2	Hausdorff Core Properties	79
7	Hausdorff Core of a Single Reflex Vertex Polygon	83
7.1	Balancing a Line	84
7.2	Computing the Hausdorff Distance	84
7.3	Finding the Optimal Solution	87
7.3.1	Expressing the Distance to Each Point	87
7.3.2	Minimizing the Maximum Distance	88
7.4	Conclusions and Future Work	89
7.5	Closed Form Hausdorff Core Solutions	92
7.5.1	Both Points Inside the Polygon	92
7.5.2	Both Points on the Polygon Boundary	92
7.5.3	One Point on the Polygon Boundary	95
8	Approximation Algorithms for Generalized Hausdorff Core	97
8.1	Algorithmic Challenges of the Hausdorff Core Problem	99
8.2	Discretization of the Problem	100
8.3	The Minimization Problem	103
8.4	Running Time and Space Requirements	104
8.5	An FPTAS for the Hausdorff Core Decision Problem	104
8.6	Conclusions and Future Work	105
9	MST with Neighborhoods	107
9.1	Related Work	108
9.2	Algebraic Complexity	110
9.2.1	Euclidean MST Problems are Sum-of-Square-Roots-Hard	110
9.3	MAX-MSTN	111

9.3.1	1/2-Approximation Algorithm	111
9.3.2	$\left(1 - \frac{2}{k+4}\right)$ -Approximation Algorithm	111
9.3.3	NP-Hardness of MAX-MSTN	112
9.4	MSTN	126
9.4.1	3-Approximation Algorithm	126
9.4.2	$(1 + 2/k)$ -Approximation Algorithm	128
9.4.3	NP-Hardness of MSTN	129
9.5	Conclusions and Future Work	135
10	2-Generalized MST	137
10.1	Related Work	137
10.2	Minimum Spanning Trees on Intervals	138
10.3	One-Dimensional 2-GMST and MAX-2-GMST	139
10.3.1	2-GMST Solution	139
10.3.2	MAX-2-GMST Solution	139
10.4	NP-Hardness of 2-GMST	140
10.4.1	Gadgets	140
10.4.2	Weight of the MST	144
10.5	Approximation Algorithms for 2-GMST	147
10.6	Problems with Known Topology	148
10.6.1	2-GMST Algorithm	150
10.7	Conclusions and Future Work	150
11	Conclusions	151
	Bibliography	153
	Index	165

List of Tables

2.1	MDS Algorithms	16
2.2	MCDS Algorithms	16
2.3	DUDC Settings	24
4.1	WSDUDC Results	38
9.1	Imprecise Convex Hull Algorithms	109

List of Figures

1.1	A MBES Survey	2
2.1	A bathymetric application for DUDC	8
2.2	Performance of DUDC algorithms	11
2.3	Line-separable discrete unit disk piercing	19
2.4	DUDC grid decomposition of [29]	25
3.1	Line-Separable Discrete Unit Disk Cover example	28
3.2	SIMPLIFICATION is not enough	29
3.3	Proof of correctness of GREEDY	31
4.1	Cases for the WSDUDC algorithm	41
4.2	Gaps and intervals in WSDUDC	43
4.3	Illustration of Lemma 4.4	45
4.4	The grid for the WSDUDC NP-hardness reduction	52
4.5	An upper bound on d_{disk}	53
4.6	The gadgets for the WSDUDC NP-hardness reduction	54
4.7	Gadget to insert a disk into a wire	55
4.8	Bounding the size of d_{disk}	57
5.1	Set membership in OSDUDC	61
6.1	MACS vs. Hausdorff distance	66
6.2	The Douglas-Peucker technique	69

6.3	Hausdorff distance vs. Fréchet distance	71
6.4	Inclusion and enclosure problems	74
6.5	Chassery and Coeurjolly algorithm counterexample	75
6.6	The upper envelope of functions	77
6.7	Different 1-centres	78
6.8	Illustration of Lemma 6.3	80
6.9	A surprising Hausdorff Core	81
7.1	Finding the cut line for a single reflex vertex polygon	83
7.2	Decomposition of a polygon with a cut line	84
7.3	The closest point is not on ℓ	85
7.4	Example Hausdorff solution on a single reflex vertex polygon	90
7.5	Distance plot for Figure 7.4.	91
8.1	Finding a Hausdorff Core by shrinking disks	99
8.2	Two disconnected solution intervals	100
8.3	A single line solution	101
8.4	Finding the minimum interval length	102
8.5	Decomposition of disks for dynamic programming solution	103
9.1	1/2-Approximation Algorithm for MAX-MSTN	111
9.2	Reduction from 3-SAT to planar 3-SAT	113
9.3	The zigzag solution for a chain of disks	114
9.4	The reflection effect	115
9.5	Examples of bad path choices	116
9.6	Canonical angles of the chain	118
9.7	MAX-MSTN variable gadgets	119
9.8	MAX-MSTN clause gadgets	120
9.9	Paths through the MAX-MSTN clause gadget.	122
9.10	Case 1 of MAX-MSTN reduction	125
9.11	Case 2 of MAX-MSTN reduction	125
9.12	Minimum weight MSTN constructions	127
9.13	MSTN variable gadget	130
9.14	Shortest path to 2 disks	131
9.15	Connecting the clause wire to the interior of the gadget	134

10.1	Points are not equivalent to intervals for imprecise MST problems	139
10.2	Example construction for 2-GMST reduction	141
10.3	Gadgets for 2-GMST reduction	143
10.4	Correctness of the crossing gadget	146
10.5	A difficult problem for 2-GMST	148
10.6	Another difficult problem for 2-GMST	149

Definitions

For clarity and consistency, we include a list of definitions and notation that we use throughout the thesis. Terms relevant to one section only are defined therein.

- \overline{ab} - a line segment with endpoints a and b .
- \vec{ab} - a ray with starting point a and incident upon b .
- \overleftrightarrow{ab} - a line incident upon points a and b .
- $C(p, r)$ - a disk of radius r centred at point p .
- $CH(P)$ - the convex hull of P .
- \mathcal{D} - a set of disks in the plane. $\mathcal{D}_{\mathcal{Q}}$ is the set of disks centred at the set of points \mathcal{Q} .
- D - a disk, usually with unit radius unless specified otherwise. q_i is the centre point of D_i .
- ∂D - the boundary of the disk D .
- $\text{dist}(p, q)$ - the Euclidean distance from point p to point q .
- P - a polygon in 2D, or possibly a polytope in higher dimensions.
- ∂P - the boundary of the polygon P . If necessary, this is used to distinguish the edges of the polygon from the region of the plane enclosed by P .
- P_V - the set of vertices of the polygon P .
- \mathcal{P}, \mathcal{Q} - sets of points.

The problems that we discuss in this thesis are the kind that you could explain to a layperson with nothing more than a pen, a napkin, and a pitcher of beer (this has been empirically verified). However, many of the solutions to these problems require advanced techniques, and a number of the solutions are approximation algorithms because of the complexities inherent in the exact versions of these problems. Aspects of each problem that we discuss remain open, and we outline directions for further research at the conclusion of each chapter. People are often drawn to problems in computational geometry because of the natural formulations, particularly when set in the plane. The aim of this thesis is to present some beautiful problems from computational geometry, to outline solutions or approximation algorithms for them, and hopefully to inspire the reader to engage in similar research.

While the research described in this thesis is related to geometric covering and piercing problems, these problems initially arose from bathymetric applications (i.e. mapping of the seafloor). Bathymetric data attracts interest for a broad number of reasons, whether for navigational purposes, educational applications, or just general curiosity. There are a number of publicly available tools for visualizing the topography of the ocean floor: Google Maps [84] has integrated a coarse description into their tool, NASA produced the HoloGlobe video [151], and Canada's NRC provides the Canadian Marine Multibeam Bathymetric Data set in their Geoscience Data Repository [127] for similar educational purposes. Much of the bathymetric data contained in general interest packages such as these is inferred using surface elevations obtained by satellites [149]. None of these packages would serve usefully for navigation (most include explicit statements to this effect). Tools such as the LOTS Browser developed by CARIS [28] use data measured by more precise means, and are produced specifically for maritime navigation.

We are witnessing a rapid increase in the number and quality of data collection instruments being produced and used, and so the volume of data available for many scientific applications is exploding. Over the last three decades, the challenge in bathymetry has shifted from making do with sparse data to finding methods for handling massive amounts of it; this is primarily due to advances in multibeam echosounder (MBES) technology [24],¹ see Figure 1.1. The seafloor terrain is defined by points, and we wish to consider recent data points as more accurate than

¹The majority of data is collected using soundings, although there exist data points that were produced by the lead line method. This technique involves dropping a lead weight tied to a rope over the side of a boat, and measuring how much rope goes over the side. One of Canada's most famous shipwrecks, the S.S. Fitzgerald, was using charts based largely on lead line data from the early 1800's when it sank in 1975 [155, p. 119].

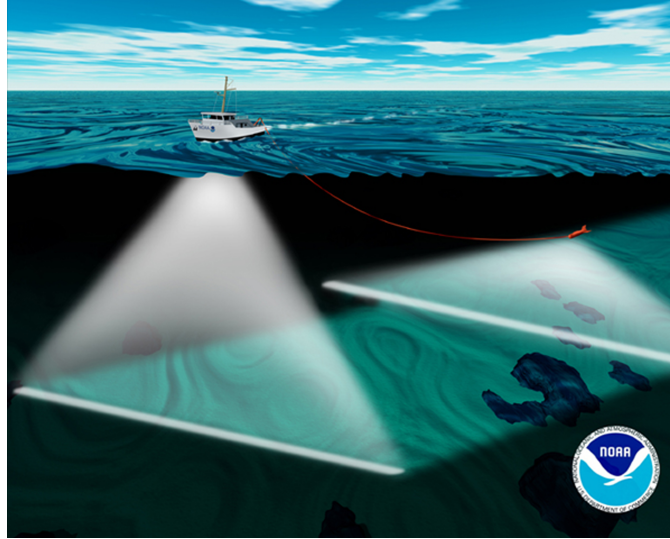


Figure 1.1: An illustration of a ship conducting a multibeam echosounder (MBES) survey. Image used with permission from NOAA [126].

older data. However, it is often desirable to maintain all old data as well, since there is some natural variability in seafloor terrain. We consider the question of how to update the points in a working set of the data when given a new survey, and we propose a model in which one wishes to maintain a minimal set of old points, while ensuring that every point in the new data set is within some unit distance of an old point. This is an instance of the Discrete Unit Disk Cover problem, which we study in detail in Chapters 2-5.

Consider a navigator who wishes to find the fastest route to a destination, while possibly visiting some waypoints en route. We assume the navigator has access to a complete set of the bathymetric data available for the relevant regions through which a possible course will pass. We wish to study approximations of the terrain which would facilitate a data structure that supports the efficient computation of such a path under various constraints imposed on this setting, such as space efficiency, query speed, and primarily the fact that it is unacceptable (for insurance purposes) to run aground of a known shallow region contained in the database. Collision detection algorithms may use a hierarchical data structure employing boxes or other convex regions to approximate the obstacles in the terrain [26, 27]. To this end, we would like to identify convex regions with minimal distance to the contour lines of the terrain. This problem inspired the Hausdorff Core problem, which we study in Chapters 6-8.

Finally, we investigate problem models where some of the difficulties originate from noisy or imprecise data. We provide a discussion of optimization problems for the Minimum Spanning Tree problem on various models of imprecision in Chapters 9 and 10. Although these problems were not inspired by any bathymetric application, they work in a model which is a natural generalization of the geometric covering problems of the previous chapters.

1.1 Organization of the Thesis

Chapter 2: Discrete Unit Disk Cover Essentials. We begin with a detailed study of the Discrete Unit Disk Cover (DUDC) problem, a well-studied geometric covering problem. Formally,

given a set \mathcal{D} of m unit disks and a set \mathcal{P} of n points in the plane, the DUDC problem is to select a minimum cardinality subset $\mathcal{D}^* \subseteq \mathcal{D}$ to cover \mathcal{P} . In this chapter, we examine the significant body of past work on the DUDC problem, and also discuss several related problems and restricted settings. We describe several paradigms that are used to achieve approximation algorithms for problems in computational geometry, and we discuss how these may apply to the DUDC problem. We restrict the DUDC problem in a number of ways, and we discuss related work on line-separated and strip-restricted settings.

Chapter 3: (Assisted) Line-Separable Discrete Unit Disk Cover. In this chapter, we study a restricted setting of DUDC in which there exists a line separating the centres of the disks from the points to be covered. For this Line-Separable Discrete Unit Disk Cover (LSDUDC) problem, we outline an $O(m^2n)$ time algorithm which finds an exact solution. We next present a method to find a 2-approximate cover for the Assisted LSDUDC (A-LSDUDC) problem, in which the disks may be on either side of the line defined for the LSDUDC problem. This is different from the general DUDC problem, because for the A-LSDUDC problem there always exists at least one candidate solution consisting only of disks whose centres are separated from the points to be covered by a line. In the general DUDC problem, this is not necessarily the case. The A-LSDUDC approximation algorithm runs in $O(mn + m \log m + n \log n)$ time. The results in this chapter include:

- An exact $O(m^2n)$ time algorithm for LSDUDC. The previous best similar result was a 2-approximation algorithm, and a subsequent result has improved this to $O(mn + n \log n)$ time.
- A 2-approximation algorithm running in $O(mn + m \log m + n \log n)$ time for A-LSDUDC. The best previous result is a 4-approximation algorithm.

Chapter 4: Within-Strip DUDC. We study another restricted DUDC setting in this chapter, in which the points and disk centres are confined to a strip of the plane. We call this the Within-Strip Discrete Unit Disk Cover (WSDUDC) problem. We describe a range of approximation algorithms for the problem, ranging from 3 to 6 factor approximations which apply for strips of varying heights ranging up to $2\sqrt{2}/3$ units, as well as a general scheme for any strip with less than unit height. The fastest algorithm (the 6-approximation algorithm, which runs in $O(mn + m \log m + n \log n)$ time) divides the strip into squares and uses the LSDUDC and A-LSDUDC results of the previous chapter to find the cover. The 3-approximation algorithm divides the strip into subproblems so that the disks used in one subproblem may overlap those in at most two others, and each subproblem is solved optimally. Unfortunately, the best known algorithm for one of these subproblems requires $O(m^6n + n \log n)$ time, and so this is the running time of the 3-approximate WSDUDC algorithm. We conclude the chapter with a proof demonstrating that the WSDUDC problem is NP-complete on strips of any fixed height. There are no known previous results on this problem, but we discuss results on similar problems.

The results in this chapter consist of the NP-completeness proof of WSDUDC, as well as a set of approximation algorithms for the DUDC problem:

- A 6-approximate algorithm requiring $O(mn + m \log m + n \log n)$ time.
- A general $3\lceil 1/\sqrt{1-h^2} \rceil$ -approximate algorithm running in $O(m^4n + n \log n)$ time.

- A 4-approximate algorithm taking $O(m^4n + n \log n)$ time.
- A 3-approximate algorithm running in $O(m^6n + n \log n)$ time.

Chapter 5: Approximation Algorithms for DUDC. We describe approximation algorithms for the DUDC problem in this chapter. We begin by showing how the LSDUDC results of Chapter 3 may be applied to provide an algorithm with an approximation factor of 22, which runs in $O(m^2n^4)$ time. Next, we describe an algorithm which uses the WSDUDC algorithms of Chapter 4 to obtain, among others, a 15-approximate algorithm for DUDC with a running time of $O(m^6n + n \log n)$, and an 18-factor algorithm which runs in $O(mn + m \log m + n \log n)$ time. The best previous results were a 38-approximate algorithm which runs in $O(m^2n^4)$ time, and a PTAS whose fastest known running time is $O(m^{65}n)$, realized when $\varepsilon \geq 2$. Because the running time of the PTAS is so large, interest remains to determine faster approximation algorithms for the DUDC problem.

In this chapter, we describe the following DUDC approximation algorithms:

- A 22-approximate algorithm taking $O(m^2n^4)$ time. This algorithm uses a grid-based decomposition of the plane.
- A $(12 + c)$ -approximation algorithm running in T time, where c and T are the approximation factor and running time respectively of the WSDUDC algorithm (see the previous chapter) used as a subroutine in the DUDC algorithm. These algorithms use a strip-based decomposition of the plane.

Chapter 6: Constrained Polygonal Vertex Cover and the Hausdorff Core. In this chapter we study various techniques that are used to approximate polygons and polygonal curves. There are a variety of techniques which have been developed to address a range of error metrics. Our interest lies in a particular generalization of disk covering, in which we wish to determine whether there exists a convex polygon Q so that points in a set \mathcal{P} are within unit distance of Q . While this formulation is not challenging, it becomes much more interesting if it is additionally required that Q is contained in a simple polygon P with vertex set \mathcal{P} . We refer to such a Q as the Hausdorff Core of P , and we define an optimization version of the Hausdorff Core problem in which the Hausdorff distance between P and Q is minimized. We develop a number of properties of the Hausdorff Core problem that are useful for our algorithms, and then in Chapters 7 and 8 we study the problem on two classes of polygons.

Chapter 7: Hausdorff Core of a Single Reflex Vertex Polygon. In this chapter, P is restricted to the set of simple polygons containing a single reflex vertex. We describe a Hausdorff Core algorithm that computes the optimal solution in $O(n^3)$ time. The algorithm essentially consists of determining the optimal rotation of a line incident upon the reflex vertex, where the line defines the convex polygon Q by cutting P . There are no previous results on this problem. The running time of our algorithm is based on the worst-case complexity of the upper envelope of the set of functions defined by the distance from each vertex of P to Q for each possible position of the line incident upon the reflex vertex. We describe a closed-form expression for the solution, although it is rather large and complicated.

Chapter 8: Approximation Algorithms for Generalized Hausdorff Core. In this chapter, we begin by outlining a parameterized algorithm for the decision version of the Hausdorff Core problem on simple polygons using dynamic programming. We centre disks on each vertex of P , and the disks are divided into a set of discrete segments. We search for a set of segments consisting of one segment from each disk so that all segments possess mutual strong visibility within P , and we demonstrate that such a set provides an approximate Hausdorff Core. By performing a parametric search on the possible values of the Hausdorff distance (which corresponds to the radii of the disks used in the decision algorithm), this extends to a parameterized algorithm for the optimization version of the problem, running in $O(\log(\varepsilon^{-1})(n^3 + n^2\varepsilon^{-6}))$ time. We demonstrate that a similar algorithm may be used to derive an FPTAS for the decision version of the Hausdorff Core problem as well. There is no previous work on this problem to our knowledge.

The algorithms described in this chapter are:

- A parameterized algorithm for the decision version of the problem, which finds a solution with Hausdorff distance $(k + d_{\text{rad}} \cdot \varepsilon)$ in $O(n^3 + n^2\varepsilon^{-6})$ time, where k_{OPT} is the value of the optimal solution and d_{rad} is the distance from a constrained 1-centre of P to the most distant vertex in P .
- A parameterized algorithm for the optimization version of the problem, which finds a solution with the same error bound as the decision version of the problem, but the running time requires an additional $O(\log(\varepsilon^{-1}))$ factor on the running time to perform a parametric (binary) search on the possible values of the optimal solution.
- A $(1 + \varepsilon)$ -approximate algorithm (FPTAS) for the decision version of the problem, taking $O(n^3 + n^2\varepsilon^{-6})$ time. This approach requires only minor modifications to the parameterized decision algorithm, but is not amenable to a parametric search as done for the parameterized algorithm for the optimization problem.

Chapter 9: MST with Neighborhoods. We generalize our setting further by looking at computational geometry problems in which the solution is only required to have points within a given distance of the input points. We call this model *computational geometry with imprecise data*. In this chapter, we study the Euclidean Minimum Spanning Tree (MST) problem within this model, where one point is chosen from each disk and the MST is computed over the selected set of points. We consider the objectives of both minimizing the weight of the MST over the input, known as the Minimum Spanning Tree with Neighborhoods (MSTN) problem, and we introduce the maximum weight version of the problem, called the MAX-MSTN problem. We prove that both the minimization and maximization versions of the problem are NP-hard even when the disks are disjoint, and we provide hardness of approximation results for both by demonstrating that there can be no FPTAS unless $P=NP$. There is some previous work on the MSTN problem, including a PTAS when the disks are restricted to unit radius. There was a previously published NP-hardness proof for MSTN, but it was found faulty by the original authors. Finally there was a previously published 3-approximation algorithm for MSTN on unit disks, but it was also faulty and we correct some of the bugs in this chapter.

We describe some approximation and parameterized algorithms for the MSTN and MAX-MSTN problems:

- A 1/2-approximation algorithm for MAX-MSTN.

- A parameterized algorithm for MAX-MSTN problem, which finds a $\left(1 - \frac{2}{k+4}\right)$ -approximate result by simply computing the MST of the centres of the disks. The parameter k is the *separability parameter*, and it is defined as the ratio of the smallest pairwise distance between any disks to the largest radius of any disk in the input.
- An asymptotic 3-approximation algorithm for MSTN on unit disks.
- A parameterized algorithm for MSTN with an approximation factor of $\left(1 + \frac{2}{k}\right)$, where k is again the separability parameter.

Chapter 10: 2-Generalized MST. We extend the problems of the previous chapter by allowing the areas of imprecision to be discrete sets of points rather than disks, which is known as the Generalized Minimum Spanning Tree (GMST) problem. We study a restricted version of GMST, where all sets of imprecise vertices have cardinality two, and the points in each set are vertically aligned. We define the 2-GMST and MAX-2-GMST problems, where the objective is to minimize or maximize the weight of the MST respectively under this constrained model of imprecision. We describe simple algorithms for both problems in one dimension, we provide a proof of NP-hardness for 2-GMST in two dimensions, and then we show that there is no FPTAS for 2-GMST unless P=NP. The only known previous result is a 4-approximate algorithm for 2-GMST based on a more general technique.

The results in this chapter include:

- A linear time exact 2-GMST algorithm and a constant time exact algorithm for the MAX-2-GMST problem in one dimension.
- A linear time recursive algorithm for solving 2-GMST if the topology (combinatorial structure) of the optimal solution is known.

Discrete Unit Disk Cover Essentials

Our interest in the Discrete Unit Disk Cover (DUDC) problem arose from data management problems upon terrains, with bathymetric applications in particular¹. Suppose that we have a survey of a terrain with data represented as points in the xy -plane, and the height data is stored as an attribute of each point. Call this point set \mathcal{Q} . Given a new survey of the same area, we obtain a new point set \mathcal{P} . We wish to update our data set by treating the new data set \mathcal{P} as the standard, but we wish to maintain some of the old data for completeness. Our restrictions are that each new point in \mathcal{P} must have at least one old point from \mathcal{Q} within unit distance, and that a minimum number of points from \mathcal{Q} are maintained (see Figure 2.1). This problem is an instance of the *Discrete Unit Disk Cover* (DUDC) problem, which is known to be NP-complete.

Definition 2.1. The Discrete Unit Disk Cover (DUDC) Problem: *Given a set \mathcal{P} of n points and a set \mathcal{D} of m unit radius disks in the plane, where a disk $D_i \in \mathcal{D}$ is centred at a point q_i in a set \mathcal{Q} , the Discrete Unit Disk Cover problem is to find a set $\mathcal{D}^* \subseteq \mathcal{D}$ of minimum cardinality such that \mathcal{D}^* covers \mathcal{P} .*

We say that a disk of radius r centred at q (written $C(q, r)$) covers a point p if the Euclidean distance between p and q is no greater than r , i.e. $\text{dist}(p, q) \leq r$.

Recent interest in specific geometric set cover problems is partly motivated by applications in wireless networking or facility location problems in operations research, e.g. [153, 164]. In particular, when wireless clients and servers are modelled as points in the plane and the range of wireless transmission is assumed to be constant in all directions (say one unit), the resulting region is a disk of unit radius centred on the point representing the wireless transmitting device. Under this model, sender a successfully transmits a wireless message to receiver b if and only if point b is covered by the unit disk centred at point a . This model applies more generally to a variety of facility location problems for which the Euclidean distance between clients and facilities cannot exceed a given radius, and clients and candidate facility locations are represented by discrete sets of points. Examples include:

- selecting locations for wireless servers from a set of candidate locations to cover a set of wireless clients,

¹Elements of this chapter have appeared in [39], [40], and [73].

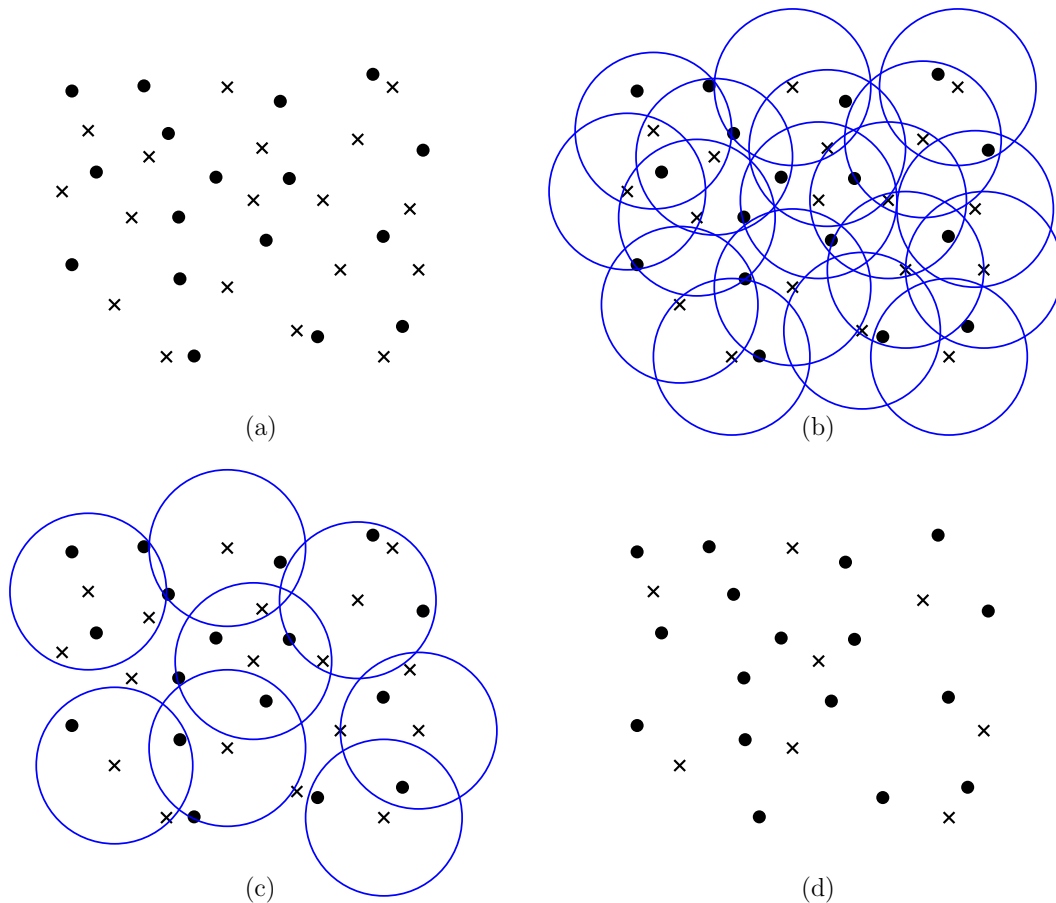


Figure 2.1: A bathymetric application for DUDC. (a) Suppose there are two surveys of the seafloor, one of new points \mathcal{P} (drawn as filled circles), and an older survey \mathcal{Q} (drawn as \times symbols). We would like to keep a minimal number of the old points, while ensuring that every new point is within some fixed distance (call it unit distance) of an old point. (b) We draw disks centred on the points of \mathcal{Q} to show all points of \mathcal{P} within unit distance of each point in \mathcal{Q} . (c) A DUDC solution provides a minimal number of disks which covers \mathcal{P} . (d) Removing the extra points and disks gives the solution.

- positioning a fleet of water bombers at airports such that every active forest fire is within a given maximum distance of a water bomber,
- selecting a set of weather radar antennae to cover a set of cities, and
- selecting locations for anti-ballistic defenses from a set of candidate locations to cover strategic sites².

These problems can be modelled by the DUDC problem. DUDC is NP-complete³ [99, 72], and the general set cover problem (i.e. the covering sets are unrestricted) is not approximable within a factor of $c \log n$, for some constant c , assuming $P \neq NP$ [137] (also, unless $NP \subseteq DTIME(n^{O(\log \log n)})$,

²Johnson [99] cheekily remarked that this would be the type of application used in a grant proposal if the funding were coming from military rather than civilian sources.

³To be precise, the Discrete Unit Disk Cover problem is NP-complete in dimensions $d > 1$, but is solvable in linear time in $d = 1$ [93].

i.e., all problems in NP can be solved in $n^{O(\log \log n)}$ time) [67]). In this thesis, we are often interested in finding approximation algorithms for hard problems.

Definition 2.2. Approximation Algorithm: *An algorithm ALG_A is a $f(n)$ -approximation algorithm for a minimization problem A if given any instance of A with size n , which has an optimal solution of size $|\text{OPT}|$, ALG_A returns a solution of size at most $f(n) \cdot |\text{OPT}|$. If A is a maximization problem, ALG_A must return a solution of size at least $f(n) \cdot |\text{OPT}|$. We may say that a $f(n)$ -approximation algorithm is an algorithm with approximation factor $f(n)$.*

A PTAS (Polynomial Time Approximation Scheme) is the strongest type of approximation algorithm.

Definition 2.3. Polynomial Time Approximation Scheme (PTAS): *A PTAS is a family of approximation algorithms for a minimization problem A , so that for any constant $\varepsilon > 0$, there exists a $1 + \varepsilon$ -approximation algorithm for A , where the running time of the algorithm is bounded asymptotically by a function which is polynomial in the size of A . Analogously, for a maximization problem, a PTAS is a family of $1 - \varepsilon$ -approximation algorithms with the same properties.*

2.1 Related Work

There are many known approximation algorithms for DUDC, using a variety of techniques. Obviously the general approximation algorithms for set cover (e.g. [36]) directly provide an $O(\log n)$ approximation algorithm. There is a series of constant factor approximation algorithms and a PTAS, mostly published within the past few years. The PTAS of [121] is applicable only for $0 < \varepsilon \leq 2$ (see Section 2.1.3), and so the running time of the PTAS is prohibitively large for any approximation factor. For this reason research has continued on the DUDC problem, specifically oriented toward determining approximation algorithms with tractable running times. We summarize the known results below, and a graphical representation of the set of Pareto optimal approximation algorithms with respect to running time versus approximation factors is given in Figure 2.2.

- $O(1)$ -approximation, Brönnimann and Goodrich, 1995 [23]. This was the first constant-factor approximation algorithm for DUDC. It is based on ε -nets, as discussed in Section 2.1.4.
- 108-approximation, Călinescu et al., 2004 [42]. This algorithm worked using a decomposition of the problem into that of using unit disks to cover points contained in an equilateral triangle with unit length sides. A 6-approximate algorithm is given for covering each triangle, and then it is shown that a unit disk covers points in no more than 18 triangles⁴ for a total approximation factor of 108.
- 72-approximation, Ambühl et al., 2006 [8] / Narayanappa and Vojtěchovský, 2006 [125]. The first result divides the plane into squares of size $1/\sqrt{2} \times 1/\sqrt{2}$, and provides a factor 2 approximation for covering points in up to 36 such squares. This approach also applies to weighted DUDC. The second result improves on that of [42] by tiling the plane with unit size hexagons.

⁴We list the approximation factor for [42] as 108, although the original publication presented the factor as 102. Narayanappa and Vojtěchovský [125] noticed a small bug in the analysis, specifically that a disk may cover points in 18 rather than 17 triangles, and corrected the factor to that listed here.

- 38-approximation, Carmi et al., 2007 [29]. This algorithm decomposes the plane into squares of size $3/2 \times 3/2$, and uses a variety of techniques to cover points in the square using disks centred both inside and outside the square. This is discussed in detail in Section 2.3.
- 22-approximation, Claude et al., 2010 [39, 40]. This result improves on [29] primarily with an improved Line-Separable DUDC algorithm, as discussed in Chapter 3.
- $(1+\varepsilon)$ -approximation, Mustafa and Ray, 2010 [121]. This is the best PTAS known for DUDC, and is based on local search. We discuss the PTAS in Section 2.1.3.
- 18-approximation, Das et al., 2011 [46, 47]. By using a decomposition similar to that of the 72-approximate algorithm of Ambühl et al. [8] and the ideas of Claude et al. [39] for Line-Separable DUDC, a slightly improved approximation factor is obtained, and the running time is significantly improved. This result is discussed in Chapters 4 and 5.
- 15-approximation, Fraser and López-Ortiz, 2012 [73]. This work examines the Within-Strip DUDC problem (Chapter 4), and uses the results with the decomposition of Das et al. [47] to obtain the lowest known approximation factor for a DUDC approximation algorithm with a tractable running time, as discussed in Chapter 5.

Another PTAS for DUDC was presented by Liao and Hu [110]. The PTAS is based on the assumption that the point set \mathcal{P} is found on the integer grid, and so there is a constant maximum cardinality of a subset of \mathcal{P} that may be covered by any particular disk. Despite the simplifying assumption, their running time does not improve upon that achieved by Mustafa and Ray. While Brönnimann and Goodrich [23] described the first small ε -net for unit disks, Pyrga and Ray [135] gave an analysis showing smaller nets for a generalized setting. The Discrete Unit Disk Cover problem (this name was coined by Carmi et al. [29]) has been given some other names as well, such as Geometric Set Cover on Unit Disks [64], Geometric Hitting Set on Unit Disks [122], and the Unit Disk Covering problem [125]. We adhere to the label *Discrete Unit Disk Cover* because it was the first name in the literature that clearly distinguishes the problem from the Minimum Geometric Disk Cover problem, where the disk centres are unrestricted (see Section 2.1.6).

2.1.1 General Geometric Set Cover

In this section, we discuss a generalization of Discrete Unit Disk Cover. This section is included as related work because techniques for Geometric Set Cover apply to DUDC, but we do not use the results of this section in the thesis. DUDC is an instance of set cover over a range space $S = (X, \mathcal{R})$ where X is defined as a set of points in \mathbb{R}^2 , and \mathcal{R} corresponds to a set of fixed disks of unit radius, whose union covers the entire point set X . The goal of Discrete Unit Disk Cover is to determine a set of disks $\mathcal{D} \subseteq \mathcal{R}$ of minimum cardinality so that all of the points in X remain covered. There are two formulations of this type of problem that are commonly used: a general Set Cover Problem (SCP) or the dual Hitting Set Problem (HSP). These problems may be defined over any range space.

Definition 2.4. Range Space [23]: Let $S = (X, \mathcal{R})$, where S is a range space over the points X and the ranges (sets) \mathcal{R} , where each range in \mathcal{R} contains a subset of X : $R_i \subseteq X, \forall R_i \in \mathcal{R}$, i.e. $\mathcal{R} \subseteq 2^X$.

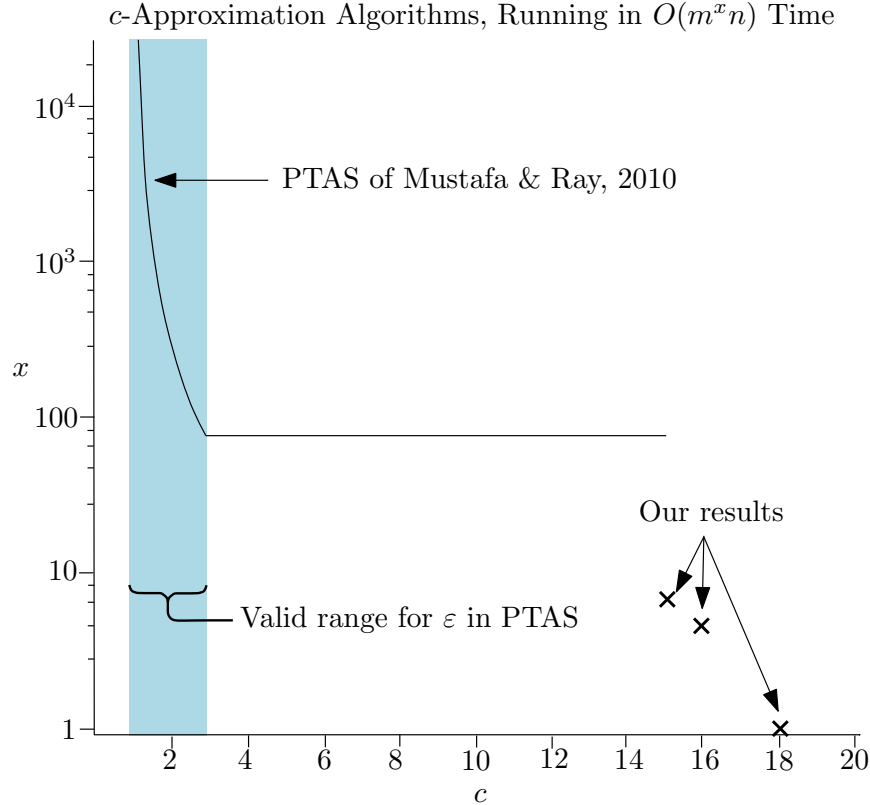


Figure 2.2: We summarize the best known algorithms for the DUDC problem. The shaded region indicates the range where ε is valid in the PTAS [121]. The points indicate our results. Note that the vertical axis is a log scale.

The term ‘range space’ is essentially synonymous with ‘set system’, so that $S = (X, \mathcal{R})$ may correspond to a system where X is the set of all items and \mathcal{R} is the family of subsets. Additionally, a graph is a range space: a graph $G = (V, E)$ maps to a range space when $X = V$ and $\mathcal{R} = E$. For this work, we are considering geometric range spaces, where X corresponds to a set of points, and \mathcal{R} is a set of geometric objects (such as disks), each of which may contain or cover some subset of the points in X . For our purposes, the term ‘range space’ may be considered synonymous with ‘set system’ or ‘hypergraph’.

Definition 2.5. Set Cover Problem [41, p.1033]: *Given a range space $S = (X, \mathcal{R})$, the set cover problem is to find a subset $\mathcal{C} \subseteq \mathcal{R}$ of minimum cardinality so that all elements of X are covered by \mathcal{C} , i.e. $X = (\cup_{R \in \mathcal{C}} R) \cup X$.*

Definition 2.6. Hitting Set Problem [23]: *Given a range space $S = (X, \mathcal{R})$, the hitting set problem is to find a subset $H \subseteq X$ of minimum cardinality so that all sets of \mathcal{R} are pierced by elements of H , i.e. $H \cap R \neq \emptyset, \forall R \in \mathcal{R}$. A set R is pierced by an element p if $p \in R$.*

Since the problems are dual, an optimal solution for the Set Cover problem is an optimal solution for the Hitting Set problem in the dual setting and vice versa. In this thesis, we use the problems interchangeably, as some problems are easier to discuss in one setting or the other.

The set cover problem and hitting set problems are NP-hard [99], so the research community has directed its energy to the determination of approximation algorithms for these problems. The

best approximation factor that one can hope for in the general setting is $\Theta(\log n)$ [36], using a simple greedy technique (e.g. Algorithm 2.1). Basically, given a set cover problem over a range space $S = (X, \mathcal{R})$, we choose the set $R \in \mathcal{R}$ with maximum cardinality. After removing R and $X \cap R$ from the problem, i.e. $\mathcal{R} \leftarrow \mathcal{R} \setminus R$, $X \leftarrow X \setminus (X \cap R)$, we proceed to the next iteration. For general set cover problems, this is the optimal approximation technique [67]. For a thorough treatment of this approach, see Vazirani [157, p.109] or Cormen et al. [41, p.1033].

Algorithm 2.1 GREEDY-SET-COVER(X, \mathcal{R}) [36]

```

1: Input: A set of items  $X$  and a range space  $\mathcal{R}$ 
2: Output: A range space  $\mathcal{C}$  which covers  $X$ 
3: // Note that  $w_R$  is the weight of  $R$  (in DUDC  $w_R = 1$  for each disk)
4:  $\mathcal{C} \leftarrow \emptyset$ 
5: while  $\exists R \in \mathcal{R}, R \neq \emptyset$  do
6:   Find  $R_{\max} \in \mathcal{R}$ , where  $|R_{\max}| = \max_{R \in \mathcal{R}} |R|/w_R$ 
7:    $\mathcal{C} \leftarrow \mathcal{C} \cup R_{\max}$ 
8:    $\forall R \in \mathcal{R}, R \leftarrow R \setminus R_{\max}$ 
9: end while
10: return  $\mathcal{C}$ 

```

2.1.2 Duality for Discrete Unit Disk Cover

For Discrete Unit Disk Cover, it is often convenient to work in the dual hitting set version of the problem. We call this the Discrete Unit Disk Piercing (DUDP) problem, and it is particularly useful because an optimal solution for DUDC is equivalent to an optimal solution for DUDP. Recall that the input for the DUDC problem is a set of points \mathcal{P} and a set of unit disks \mathcal{D} with centre points \mathcal{Q} . The dual DUDP formulation is a set of unit disks $\mathcal{D}_{\mathcal{P}}$ with centre points \mathcal{P} , and the set of points \mathcal{Q} . An optimal solution to the DUDP problem is a minimum cardinality set of points $\mathcal{Q}^* \subseteq \mathcal{Q}$ where each disk in $\mathcal{D}_{\mathcal{P}}$ contains at least one point in \mathcal{Q}^* .

Lemma 2.1. *Given an instance of DUDC with input points \mathcal{P} , input disks \mathcal{D} (where \mathcal{Q} is the set of centre points of \mathcal{D}), and an optimal solution \mathcal{D}^* , the dual DUDP problem on the instance is solved optimally by \mathcal{Q}^* , the set of centre points of \mathcal{D}^* .*

Proof. The lemma depends on the fact that both the primal DUDC problem and the dual DUDP problem use unit disks. Any pair of points $(p \in \mathcal{P}, q \in \mathcal{Q})$ where $\text{dist}(p, q) \leq 1$ have the property that the disk centred at q in the primal covers p , and the disk centred at p in the dual is pierced by q . Therefore, the set of points covered by a disk D in the primal is exactly the set of centre points of the disks pierced by q (the centre point of D) in the dual. \square

2.1.3 PTAS for Discrete Unit Disk Cover

Mustafa and Ray [121, 123] presented a $(1 + \varepsilon)$ -approximation algorithm (PTAS) for the Hitting Set problem on geometric sets which is based on the idea of local search. In much of the literature, discussions on this topic are generalized to include pseudo-disks or objects that satisfy certain locality conditions, but for clarity of exposition we restrict ourselves to unit radius disks as the

range objects. Having said that, all technical details from this section are generalizable to pseudo-disks (as they were originally presented in this context) [121]. A set of *pseudo-disks* is a family of objects such that for any set of three points in the plane, at most one member of the family is incident upon the three points. Generally, pseudo-disks are considered to include sets of disks, or the set of all homothets (scalings and/or translations) of a fixed oval [118].

Since we are interested in DUDC, we convert the parameter notation to be consistent with those used in this thesis. Therefore, for our purposes, the number of points in the hitting set formulation is m , because this is the number of disks in the DUDC setting. Similarly, n is the number of disks in the hitting set problem.

Local Search

Local search is essentially the idea of making optimal choices in each iteration of an algorithm using only the information available at the time, which can lead to overall suboptimal solutions. In the context of the PTAS of Mustafa and Ray [121], the local search is a bounded exhaustive search so that the total work performed may be bounded by a polynomial. The search looks for sets of points in the current solution that may be replaced by any subset while maintaining a hitting set. In this setting, they use a technique called k -level search, where each set of k points in the hitting set solution is compared with each subset of X with size $k - 1$ to see if they may be substituted and remain a hitting set. Generally, the smallest point set initially known to be a hitting set is the entire point set X , so that is the starting state. For each iteration of the algorithm, the size of the hitting set is reduced by one by checking $\binom{m}{k} \binom{m}{k-1}$ possible local improvements. Therefore, when the local search finishes, we know that no k points can be replaced by any $k - 1$ points to produce a hitting set. Clearly, if we allow $k = m$, then we will find the optimal solution, but may require superpolynomial time. The PTAS is obtained by choosing $k = O(\varepsilon^{-2})$:

Theorem 2.1. [121] *The Discrete Unit Disk Cover problem defined over $S = (X, \mathcal{R})$ may be solved using a $(c/\varepsilon)^2$ -level search to obtain a hitting set $H \subseteq X$ of size less than or equal to $(1 + \varepsilon) \cdot |H^*|$, where H^* is the optimal hitting set solution.*

Limitations of Local Search

The PTAS runs in $O(m^{2(c/\varepsilon)^2+1}n)$ time, where $c \leq 4\gamma$ [121]. Their γ value can be bounded from above by $2\sqrt{2}$ [76, 105]. They perform a k -level search, where $k = (c/\varepsilon)^2 \geq 4\gamma^2$, so $\varepsilon \leq 2$. The fastest operation of this algorithm is obtained by setting $\varepsilon = 2$ for a 3-approximation algorithm, and this will run in $O(m^{2 \cdot (8\sqrt{2}/2)^2+1}n) = O(m^{65}n)$ time in the worst case. Clearly, this algorithm will not be practical for large values of m (or even $m = 2$). However, it is possible that a lower running time may be obtained through better bounding of the constant factors or improvements to their algorithm, or that the running time on real world data never approaches this worst case bound.

For future work, the original authors conjecture that it may be possible to do (c/ε) -level local search, rather than $(c/\varepsilon)^2$ [121]. The impact of such an improvement on the running time would be significant; the worst case could be bounded by $O(m^{8\sqrt{2}+1}n)$ for a 3-approximation algorithm.

2.1.4 ε -nets for Geometric Set Cover Problems

ε -nets have useful applications in computational geometry; their use has facilitated the construction of approximation algorithms including one of the constant factor approximation algorithms for Discrete Unit Disk Cover.

Suppose we are given a range space $S = (X, \mathcal{R})$, where X is a set of points and \mathcal{R} is a family of subsets of X . Given S and a parameter ε , where $0 \leq \varepsilon \leq 1$, an ε -net is a set $N \subseteq X$ such that for each set $R \in \mathcal{R}$ where $|R| > \varepsilon|X|$, we have that $R \cap N \neq \emptyset$ [23, 118]. Intuitively, ε -nets are hitting sets which are defined over all ranges in the range space with some minimum size. We present a definition of ε -nets using the setting of range spaces (although it is easily generalized):

Definition 2.7. ε -net [89]: *Given a range space $S = (X, \mathcal{R})$ and a constant $\varepsilon \in [0, 1]$, a set $N \subseteq X$ is an ε -net if every set $R \in \mathcal{R}_\varepsilon$ contains a point from N (i.e., $R \cap N \neq \emptyset, \forall R \in \mathcal{R}_\varepsilon$), where $\mathcal{R}_\varepsilon = \{R \in \mathcal{R} \mid |R|/|X| > \varepsilon\}$.*

In simple terms, the ε -net N must intersect each range in \mathcal{R} of cardinality greater than a threshold equal to ε . If we set $\varepsilon = 0$, then finding an ε -net N of minimum cardinality is precisely the hitting set problem. Haussler and Welzl [89] discuss the problem in terms of points in \mathbb{R}^d space and the set of ranges consisting of the set of half-spaces. If all of the points in X are extreme (X defines a simplex in \mathbb{R}^d), then for $\varepsilon = 0$, the smallest ε -net (0-net) is X itself [89]. However, things are much more interesting for $\varepsilon > 0$, where they claim that given any finite point set $X \in \mathbb{R}^d$, the set of half-spaces H_d^+ , and a constant $\varepsilon > 0$, there exists an ε -net N for $S = (X, H_d^+)$ such that $|N| \leq O(d/\varepsilon \log d/\varepsilon)$ [89].

Set Cover with ε -nets

Much of work in the area has focused on identifying bounds on the size of ε -nets given a particular application, since the existence of small ε -nets provides smaller approximation factors. For the family of circular disks, it is known that for any finite point set P there exists an ε -net of size $O(1/\varepsilon)$ [118] (this may be generalized to pseudo-disks in \mathbb{R}^2 [135]). For general problems with finite VC-dimension (see [88, Chapter 5] for a nice introduction to VC-dimension), the $O(d/\varepsilon \log d/\varepsilon)$ bound of [89] was improved to $O(d/\varepsilon \log 1/\varepsilon)$ [21], and later refined to $(1 + o(1))(d/\varepsilon \log 1/\varepsilon)$ [104], where it stands now. Matoušek et al. [118] conjectured that geometric range spaces may typically admit ε -nets of size $O(d/\varepsilon)$ (later $O(1/\varepsilon)$ by Aronov et al. [10]), which may be contrasted with the lower bound of $\Omega(d/\varepsilon \log 1/\varepsilon)$ proven for general problems of finite VC-dimension [104]. Given an ε -net of size $O(1/\varepsilon \phi(1/\varepsilon))$ for some function ϕ , the techniques of Brönnimann and Goodrich [23] provide a hitting set (or set cover) with approximation factor $O(\phi(\text{OPT}))$ [10]. Therefore, establishing ε -nets of size $O(1/\varepsilon \log \log 1/\varepsilon)$ gives an approximation factor of $O(\log \log \text{OPT})$, and ε -nets of size $O(1/\varepsilon)$ for a problem provide a constant factor approximation algorithm.

Given a range space with constant VC-dimension, such as a set of disks, a set cover with approximation factor $O(\log |\text{OPT}|)$ may be found in polynomial time, where $|\text{OPT}|$ is the size of the optimal minimum set cover [23] (not to be confused with the $O(\log |X|)$ approximation factor of Algorithm 2.1). Since the approximation bound on general set cover is $\Theta(\log |X|)$ [67], this parameterized version of set cover allows for better approximations for set cover problems on range spaces with bounded VC-dimension. For a nice derivation of small ε -nets for a range space, see the example of axis parallel rectangles in \mathbb{R}^2 , as derived by Aronov et al. [10].

Limitations of ε -nets

While ε -nets yield constant factor approximation algorithms for problems with ε -nets of size $O(1/\varepsilon)$, the constant in the approximation factor is dependent on the (potentially large) constants hidden in the size of the ε -net. Therefore, it is conjectured that a direct application of ε -nets cannot be used to create a PTAS for geometric hitting set problems [123].

2.1.5 Minimum Dominating Set on Unit Disk Graphs

In the Minimum Dominating Set on Unit Disk Graphs problem, the input consists of a graph $G = (\mathcal{P}, E)$, where \mathcal{P} consists of a set of points in the plane, and the edge (p_i, p_j) for $p_i, p_j \in \mathcal{P}$ exists in E if and only if $\text{dist}(p_i, p_j) \leq 1$. In other words, given a unit disk D_i centred at p_i , the point p_j is covered by D_i , and vice-versa (i.e. $p_i \in D_j$). The unit disk graph cover problem asks for the minimum dominating set \mathcal{P}^* over G , so that $\forall p_i \in \mathcal{P}$, if $p_i \notin \mathcal{P}^*$, then $\exists e = (p_i, p_j) \in E$ such that $p_j \in \mathcal{P}^*$. Alternatively, the problem may be formulated as DUDC where $\mathcal{P} = \mathcal{Q}$:

Definition 2.8. Unit Disk Graph Cover Problem: *Given a set \mathcal{P} of n points and the set \mathcal{D} of n unit radius disks in the plane, where each disk $D_i \in \mathcal{D}$ is centred at point $p_i \in \mathcal{P}$, the Unit Disk Graph Cover problem is to find a set $\mathcal{D}^* \subseteq \mathcal{D}$ of minimum cardinality such that \mathcal{D}^* covers \mathcal{P} .*

There is an alternate (and equivalent) formulation on the unit disk graph, where there is an edge between points p_i and p_j if and only if unit disks centred at p_i and p_j have a non-empty area of intersection:

Definition 2.9. Minimum Dominating Set on Unit Disk Graphs: *Given a set \mathcal{P} of n points in the plane, we define the graph $G_2 = (\mathcal{P}, E_2)$, where $(p_i, p_j) \in E_2$ if and only if $\text{dist}(p_i, p_j) \leq 2$. The Minimum Dominating Set on Unit Disk Graphs (MDS) problem is to find a set $\mathcal{P}^* \subseteq \mathcal{P}$ of minimum cardinality such that \mathcal{P}^* covers E_2 .*

In their seminal work on unit disk graphs, Clark et al. [37] called the Definition 2.8 version of the problem the containment model, and the latter definition corresponds to the intersection model of unit disk graphs. They noted that a graph under the containment model can be converted to the intersection model by halving the radius of the disks, while doubling the radius performs the conversion in the other direction (a more formal discussion may be found in [63]). Therefore, we may consider the models interchangeable. Interestingly, the proof of NP-completeness of the Minimum Dominating Set on Unit Disk Graphs problem [117] predates the formal unit disk graph definition by nearly a decade!

One of the applications of the problem is for ad-hoc wireless networks. Many papers also consider the minimum weight *connected* dominating set, where the solution found for Definition 2.9 is the graph $G'_2 = (\mathcal{P}', E_2)$, where G'_2 is connected. Lichtenstein [111] proved that the geometric minimum weight connected dominating set is also NP-complete.

There is a significant set of results on the unweighted Minimum Dominating Set (MDS) and Minimum Connected Dominating Set (MCDS) problems on unit disk graphs, as shown in Tables 2.1 and 2.2. The first result in the area, by Marathe et al. [116], was very simple: first compute any maximal independent set on the graph, and then show that this forms a dominating set within some factor of the optimal. To connect the graph, they simply showed that no more than one vertex needs to be added for each vertex in the independent set. This provides a very simple

and fast $O(n)$ time approximation algorithm. The algorithms of [79], [158] and [162] use similar ideas although they are more rigorous in the sense that they operate in a distributed setting. For our purposes, they are principally superior due to better bounding of the factors between the solutions and the size of a maximal independent set, and so they also run in $O(n)$ time. The recent result of De et al. [49] provides the smallest constant factor approximation algorithm outside of a PTAS, but the running time of their approach is relatively high.

Table 2.1: Approximation Factors of MDS Algorithms

Approximation Factor	Running Time	Reference
5	$O(n)$	Marathe et al., 1995 [116]
$(1 + \varepsilon)$	-	Hunt III et al., 1998 [95]
$(1 + \varepsilon)$	-	Nieberg and Hurink, 2006 [128]
4, 3	$O(n^9), O(n^{18})$	De et al., 2011 [49]
$44/9 = 4.8\bar{8}$	$O(n \log n)$	Da Fonseca et al., 2012 [43]

Table 2.2: Approximation Factors of MCDS Algorithms

Approximation Factor	Reference
10	Marathe et al., 1995 [116]
8	Wan et al., 2002 [158]
$(1 + \varepsilon)$	Cheng et al., 2003 [34]
7.8	Wu et al., 2006 [162]
$6.91 \cdot \text{OPT} + 16.58$	Funke et al., 2006 [79]

There exists a PTAS for the basic version of the problem, i.e. for the setting in Definition 2.8 [95], as well as for the variation where the solution is required to consist of a connected graph (MCDS) [34]. Each PTAS is based on the shifting strategy described in Section 2.1.6. Furthermore, there exists a PTAS for the problem if a planar embedding of the graph is not provided [128].

2.1.6 Minimum Geometric (Unit) Disk Cover

In the Minimum Geometric Disk Cover (GDC) problem, the input consists of a set of points in the plane, and the problem is to find a set of disks of radius r of minimum cardinality whose union covers the points. Naturally, the problem may be scaled so that $r = 1$, to be similar to the DUDC definition.

Definition 2.10. The Minimum Geometric (Unit) Disk Cover (GDC) Problem: *Given a set \mathcal{P} of n points in the plane, the Minimum Geometric Disk Cover problem is to find a set of unit disks \mathcal{D}^* of minimum cardinality such that \mathcal{D}^* covers \mathcal{P} .*

Unlike the Discrete Unit Disk Cover problem (Definition 2.1), the disk centres are not constrained to be selected from an input set, but rather may be centred at arbitrary points of the

plane. This problem is NP-hard [72, 152] and has a PTAS solution [83, 93]. There are many possible generalizations of the GDC problem; see Clarkson and Varadarajan [38] for a discussion of geometric set cover problems. The PTAS for the GDC problem is particularly noteworthy, because it introduced the shifting strategy for geometric approximation algorithms. It is worth studying the technique to determine whether a similar scheme would apply to the problems addressed in this thesis (no such approach is known).

Geometric PTAS Using the Shifting Strategy

The PTAS for the minimum GDC problem is based on the idea that the problem may be solved exactly in polynomial time when the problem is confined to a square of constant size. Our discussion follows the presentation of [93]⁵. We consider the context of covering with disks, but the PTAS applies to a more general class of geometric objects, including squares, rectangles, and α -fat objects.

To begin with, a bounding box I is found so that $\mathcal{P} \subset I$. I is divided into vertical strips of width 2 so that, if strips are chosen well, every disk will intersect exactly one boundary line between any two strips. Next, the strips are grouped so that each set of l consecutive strips form a broader strip of width $2l$, where $l \in \mathbb{N}$ is called the *shifting parameter*. There are l possible ways to form these broader strips, and these *shift partitions* are labelled with S_1, S_2, \dots, S_l .

Suppose that an algorithm A can compute the cover \mathcal{D}' in one of these broad strips in polynomial time with an approximation factor r_A , i.e. $\mathcal{D}' \leq r_A \cdot |\mathcal{D}_{\text{opt}}|$, where \mathcal{D}_{opt} is the optimal solution. Let \mathcal{D}_i be the set of disks obtained by taking the union of applying A to each strip in the shift partition S_i , and let $\mathcal{D}_A = \min_{i \in \{1, \dots, l\}} |\mathcal{D}_i|$. The approximation factor of this minimal solution, call it $r_{\mathcal{D}_A}$, is given by the *Shifting Lemma* [93, Lemma 2.1]:

$$r_{\mathcal{D}_A} \leq r_A (1 + 1/l).$$

The algorithm A is simply a recursive application of this technique: divide each strip of width $2l$ into squares sized $2l \times 2l$, and with the same shifting technique we can find an approximate cover for a strip using algorithm B to cover the squares, so that A has an approximation factor $r_A \leq r_B (1 + 1/l)$, where r_B is the approximation factor of B . The optimal cover for a square may be solved (so $r_B = 1$) with brute force in $O\left(n_s^{2(l\sqrt{2})^2}\right)$ time, where n_s is the number of points in the square. Therefore, $r_A \leq (1 + 1/l)$, and $r_{\mathcal{D}_A} \leq (1 + 1/l)^2$. The total running time of the approximation algorithm in the plane is $O\left(4l^2 (2n)^{4l^2+1}\right)$. Note that this scheme may be applied recursively in higher dimensions as well.

The primary criterion for applying this approximation scheme is that the boundary of the subproblems are independent for each of the shift partitions. Specifically, there may exist disks in a given shift partition that intersect the boundary between two adjacent broad strips. These disks are called the *boundary disks* for this shift partition. The set of boundary disks in one shift partition is independent from those in all other shift partitions because the narrow strips have a width of at least two units⁶. For this reason, the approximation factor is at most $(1 + 1/l)^2 = 4$, and the shifting parameter l cannot be greater than n , so the approximation factor must be at

⁵See Baker [15] for similar results on graphs.

⁶Strips of width less than 2 may not have this property, and so we would not be able to apply the Shifting Lemma. This is the case for our work in Chapter 4.

least $(1 + 1/n)^2$. Gonzalez [83] improved the running time of the PTAS by presenting a faster deterministic algorithm for computing the optimal cover of the points in a strip of general width, i.e. algorithm A discussed previously. This also produces an approximation bound of $(1 + 1/l)^{d-1}$ for the d dimensional version of the problem, rather than the $(1 + 1/l)^d$ bound of [93].

2.1.7 Discrete k -Center

Another problem with a similar feel to DUDC is the Discrete Euclidean k -Centre problem: given a set \mathcal{Q} of m points in the plane, a set \mathcal{P} of n points in the plane, and an integer k , find a set of k disks centred on points in \mathcal{Q} whose union covers \mathcal{P} such that the radius of the largest disk is minimized. Observe that the set \mathcal{P} has a Discrete Unit Disk Cover solution consisting of k disks centred on points in \mathcal{Q} if and only if \mathcal{P} has a Discrete k -Centre centred on points in \mathcal{Q} with radius at most one. This problem is NP-hard if k is an input variable [2]. When k is fixed, Hwang et al. [96] give an $m^{O(\sqrt{k})}$ -time algorithm, and Agarwal and Procopiuc [1] give an $m^{O(k^{1-1/d})}$ -time algorithm for points in \mathbb{R}^d .

2.1.8 Other Discrete Geometric Covering Problems

A natural extension of the DUDC problem is to consider other shapes such as squares or α -fat objects. The PTAS of Mustafa and Ray [121] applies to these settings (see Section 2.1.3). Somewhat surprisingly, since this approach does not seem to work for DUDC, a PTAS for the weighted version of the Discrete Unit *Square* Cover problem [64] was found based on the shifting strategy discussed in Section 2.1.6.

2.2 Specialized Settings

The approximation algorithms that have been used for the DUDC problem often employ restricted versions of the general problem as subcases, and the restricted versions are quite interesting in their own regard. We introduce several of these settings here, and examine two in detail in subsequent chapters.

2.2.1 Line-Separable Discrete Unit Disk Cover

Perhaps the most basic restricted DUDC setting is the *Line-Separable Discrete Unit Disk Cover* (LSDUDC) problem, where \mathcal{P} and \mathcal{Q} may be separated by a line ℓ .

Definition 2.11. The Line-Separable Discrete Unit Disk Cover (LSDUDC) Problem: *Given a set \mathcal{P} of n points, a set \mathcal{D} of unit radius disks in the plane, where each disk $D_i \in \mathcal{D}$ is centred at a point $q_i \in \mathcal{Q}$, $|\mathcal{Q}| = |\mathcal{D}| = m$, and a line ℓ defining half-planes ℓ^+ and ℓ^- such that $\mathcal{P} \subset \ell^-$ and $\mathcal{Q} \subset \ell^+$, the Line-Separable Discrete Unit Disk Cover problem is to find a set $\mathcal{D}^* \subseteq \mathcal{D}$ of minimum cardinality such that \mathcal{D}^* covers \mathcal{P} .*

We may arbitrarily set ℓ to be horizontal, and furthermore, we may have the set \mathcal{Q} lie in the region above the line ℓ (call this the half-plane ℓ^+). The LSDUDC problem is an important subproblem that is applied multiple times in several of the general DUDC approximation algorithms.

A similar problem to LSDUDC is studied in [42, 45], but their setting has the centres of the disks within a specified unit disk and the points to be covered are outside that disk.

The first appearance of some form of LSDUDC was presented by Călinescu et al. [42, Algorithm 2]. In their setting, the points to be covered (\mathcal{P}) all lie in one quadrant of the plane, and are outside of a unit disk A which is centred upon the origin, and the points \mathcal{Q} (the centre points of \mathcal{D}) are all contained in A . They determine the upper envelope⁷ of \mathcal{D} in the quadrant, and then greedily use the disks to cover the points in \mathcal{Q} to obtain a 2-approximate solution which runs in $O((n+m)\log(n+m))$ time. Carmi et al. [29] use the same approach to obtain a 2-approximation for the version of LSDUDC described in Definition 2.11. The first exact algorithm known to solve LSDUDC was designed for the Strip-Separated Discrete Unit Disk Cover problem (discussed in Section 2.2.3), a more general problem than LSDUDC.

In [39], an LSDUDC algorithm was shown which improves upon the algorithm presented in Chapter 3. We provide the algorithm and a sketch of its operation here, because it is currently the best known LSDUDC algorithm. The problem is addressed in the dual setting: rather than seeking a subset of disks $\mathcal{D}^* \subseteq \mathcal{D}$ which covers all of the points in \mathcal{P} , we are seeking a minimum cardinality subset of points $\mathcal{Q}^* \subseteq \mathcal{Q}$ such that each disk in $\mathcal{D}^{\mathcal{P}}$ (the set of unit disks centred at the points in \mathcal{P}) is pierced by at least one point in \mathcal{Q}^* , as shown in Figure 2.3.

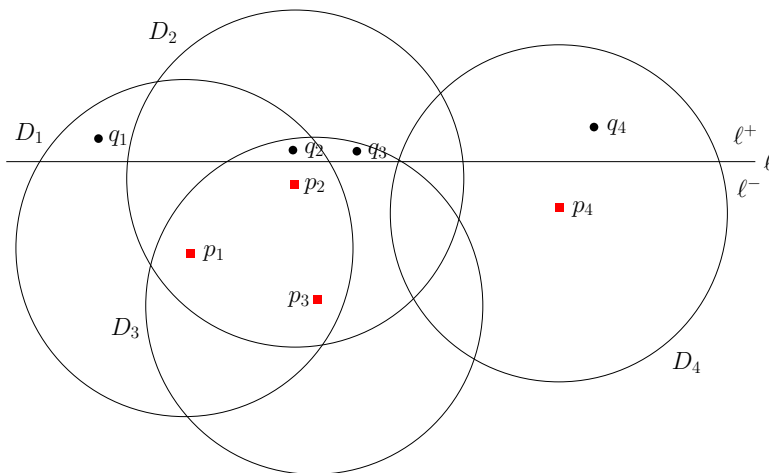


Figure 2.3: An example of the dual disk piercing setting of LSDUDC. In the dual (shown), we place disks centred on the rectangular points, and seek the minimum number of round points so that each disk is pierced by at least one point. In the primal disk covering setting, we would seek the minimum number of disks centred at the round points to cover all of the rectangular points. The solutions to both problems are the same: $\{q_2, q_4\}$ is an optimal solution for the piercing problem, and disks centred at these points form an optimal LSDUDC solution in the primal setting.

Theorem 2.2. [39] *The LSDUDC problem may be solved exactly in $O(mn + n \log n)$ time.*

Algorithm 2.2 covers all the points in \mathcal{P} using a minimum number of unit radius disks centred at points in \mathcal{Q} . Note that covering all the points in \mathcal{P} by the minimum number of unit disks centred at points in \mathcal{Q} implies each disk $D_i^{\mathcal{P}}$ (centred at $p_i \in \mathcal{P}$) contains at least one point in \mathcal{Q} . Let $D_i^{\mathcal{P}}$ be the disk of unit radius centred at the point $p_i \in \mathcal{P}$, and let l_i and r_i be the leftmost

⁷The upper envelope is also known as the skyline. See Figure 6.6 for an explanation.

and rightmost intersection points of $D_i^{\mathcal{P}}$ with the horizontal line ℓ , respectively. Without loss of generality, rename the points in \mathcal{P} and disks in $\mathcal{D}^{\mathcal{P}}$ based on the left intersection points from left to right order, i.e. $\forall p_i, p_j \in \mathcal{P}$, l_i is left of l_j if and only if $i < j$. Let ℓ^+ be the region above the line ℓ and ℓ^- be the region below the line ℓ . Let $C_i \subseteq \mathcal{Q}$ be the set of points covered by the disk $D_i^{\mathcal{P}}$ centred at the point $p_i \in \mathcal{P}$.

Algorithm 2.2 LSDUDC($\mathcal{Q}, \mathcal{D}^{\mathcal{P}}, \ell$)

```

1: Input: A set  $\mathcal{Q}$  of points in  $\ell^+$ , and a set  $\mathcal{D}^{\mathcal{P}}$  of unit disks centred in  $\ell^-$ 
2: Output: A set  $\mathcal{Q}^*$  of points piercing all disks in  $\mathcal{D}^{\mathcal{P}}$ 
3:  $\mathcal{Q}^* \leftarrow \emptyset$ 
4: Sort  $\mathcal{D}^{\mathcal{P}}$  left to right by the points  $l_i$ 
5:  $C \leftarrow D_1^{\mathcal{P}} \cap \mathcal{Q}$ 
6: for  $i := 2$  to  $n$  do
7:    $C_i \leftarrow D_i^{\mathcal{P}} \cap \mathcal{Q}$ 
8:    $C' \leftarrow C_i \cap C$ 
9:   if  $C' = \emptyset$  then
10:     $s \leftarrow$  right-most point in  $C$ 
11:     $\mathcal{Q}^* \leftarrow \mathcal{Q}^* \cup \{s\}$ 
12:     $C \leftarrow C_i$ 
13:   else
14:     $C \leftarrow C'$ 
15:   end if
16: end for
17: return  $\mathcal{Q}^*$ 

```

For the proof of correctness, please refer to [39]. In Chapter 3, we study the LSDUDC problem in greater detail, and we present a different exact algorithm for the problem.

2.2.2 Assisted Line-Separable Discrete Unit Disk Cover

If we generalize the LSDUDC problem by allowing disk centres to be on both sides of the line, we have an instance of the Assisted Line-Separable Discrete Unit Disk Cover (A-LSDUDC) problem.

Definition 2.12. The Assisted Line-Separable Discrete Unit Disk Cover (A-LSDUDC) Problem: Given a set \mathcal{P} of n points, a set \mathcal{D} of unit radius disks in the plane, where each disk $D_i \in \mathcal{D}$ is centred at a point $q_i \in \mathcal{Q}$, $|\mathcal{Q}| = |\mathcal{D}| = m$, and a line ℓ defining half-planes ℓ^+ and ℓ^- such that $\mathcal{P} \subset \ell^-$, and every point $p \in \mathcal{P}$ is covered by some D_i where $q_i \in \ell^+$ (although points in \mathcal{Q} may be found in ℓ^- as well), the Assisted Line-Separable Discrete Unit Disk Cover problem is to find a set $\mathcal{D}^* \subseteq \mathcal{D}$ of minimum cardinality such that \mathcal{D}^* covers \mathcal{P} .

In other words, we are given a set of disks $\mathcal{D} = \mathcal{D}_L \cup \mathcal{D}_U$. The disks in \mathcal{D}_U have centres in the half-plane ℓ^+ above a line ℓ , and disks in the set $\mathcal{D}_L (= \mathcal{D} \setminus \mathcal{D}_U)$ have centres below ℓ in the half-plane ℓ^- . We are also given a set of points \mathcal{P} in ℓ^- which is entirely covered by \mathcal{D}_U , so that $\mathcal{P} = \mathcal{P} \cap (\cup \mathcal{D}_U)$. The goal of the A-LSDUDC problem is to obtain a set $\mathcal{D}^* \subseteq \mathcal{D}$ of minimum cardinality such that every point in \mathcal{P} is covered by a disk in \mathcal{D}^* . Existing approaches to A-LSDUDC first solve the LSDUDC problem on the instance using \mathcal{D}_U , and then use the disks of \mathcal{D}_L to improve the cover.

Carmi et al. [29] use the fact that their 2-approximation to the LSDUDC problem consists of disks forming the lower boundary of \mathcal{D}_U , which is defined as the *semi-chain*.

Definition 2.13. Semi-chain [29]: *The semi-chain \mathcal{S} is the ordered (from left to right) set of all lower circular arcs below the line ℓ of the disks in \mathcal{D}_U . The set of indices associated with \mathcal{S} forms a consecutive set of indices $i, i+1, \dots, j$ for $i \leq j$. An interval from i to j is an interval cell and it is denoted by $\text{icell}(i, j)$. Let B denote the region $\ell^- \cap (\cup_{i=1}^m D_i)$, where $(D_i \in \mathcal{D}_U)$, which corresponds to the region below ℓ contained by all of the circular arcs in \mathcal{S} .*

Definition 2.14. Assisted Cover [29]: *Consider a unit disk $\hat{D}_L \in \mathcal{D}_L$ which intersects B . Given an interval cell $\text{icell}(i, j)$, if the set $\{D_i, D_j, \hat{D}_L\}$ covers all the points covered by the disks in the interval cell, then this new set is called an assisting set for the interval $[i, j]$. In the special case where $j = i + 1$, $\{D_i, D_j\}$ forms the assisting set of the interval $[i, j]$. The assisting set $\{D_i, D_j, \hat{D}_L\}$ is said to contain a left assisting pair, which is simply the set $\{D_i, \hat{D}_L\}$. In special cases where an assisting set is composed of only one or two disks, the leftmost individual disk is considered a left assisting pair for these purposes. Finally, an assisted cover is simply the family of these left assisting pairs which together form a cover of the points in \mathcal{P} .*

We provide details for the 2-approximate Minimum Assisted Cover (MAC) procedure of Carmi et al. [29] in Algorithm 2.3. The technique is based on the idea of covering intervals, where the disks cover intervals of the points when the points are sorted properly. The algorithm takes as input the set of points \mathcal{P} and a separating line ℓ , along with \mathcal{D}_L and \mathcal{D}'_U , where \mathcal{D}'_U is an LSDUDC solution on \mathcal{P} . The disks in \mathcal{D}'_U are sorted by their left-most intersection point with ℓ , and then the points are sorted so that any point in the set covered by D_i^U is before those covered by D_{i+1}^U , where $D_i^U, D_{i+1}^U \in \mathcal{D}'_U$. Note that this sorted order is not necessarily unique. The algorithm walks along the ordered set of points as long as some disk can cover all points seen since the last disk was added to the solution. Once a last such point is found, any disk covering all points in this interval is added to the solution set. This procedure is repeated until all points are covered.

The algorithm runs in $O(mn + m \log m)$ time. Sorting the disks takes $O(m \log m)$ time, and the first nested loops (to sort the points) take $O(mn)$ time. In the second set of nested loops, we iterate through the set of points and check for the set of disks covering the points, which again takes $O(mn)$ time.

The correctness of the algorithm follows from the order imposed upon the points. If there are no assisting disks, this algorithm returns the set of disks \mathcal{D}'_U . If there exists an assisting disk, it is found by greedily covering the set of points in the interval cell.

2.2.3 Strip-Separable Discrete Unit Disk Cover

The *Strip-Separable Discrete Unit Disk Cover* (SSDUDC) problem restricts \mathcal{P} to a strip defined by parallel lines ℓ_1 and ℓ_2 , and \mathcal{Q} must be located entirely outside of the strip.

Definition 2.15. Strip-Separable Discrete Unit Disk Cover (SSDUDC) Problem: *Suppose we are given a set \mathcal{P} of n points, a set \mathcal{D} of unit radius disks in the plane, where each disk $D \in \mathcal{D}$ is centred at a point in a set \mathcal{Q} of m points, and parallel lines ℓ_1 and ℓ_2 such that $\ell_1 \subset \ell_2^+$ and $\ell_2 \subset \ell_1^-$. Furthermore, ℓ_1 and ℓ_2 define a strip $s = \ell_1^- \cap \ell_2^+$ such that $\mathcal{P} \subset s$ and $\mathcal{Q} \cap s = \emptyset$. The Strip-Separable Discrete Unit Disk Cover problem is to find a set $\mathcal{D}^* \subseteq \mathcal{D}$ of minimum cardinality such that \mathcal{D}^* covers \mathcal{P} .*

Algorithm 2.3 $\text{MAC}(\mathcal{P}, \mathcal{D}'_U, \mathcal{D}_L, \ell)$

```
1: Input: A set of points  $\mathcal{P}$  below  $\ell$ , a set of disks  $\mathcal{D}'_U$  centred above  $\ell$  where  $\mathcal{P} \cap (\cup \mathcal{D}'_U) = \mathcal{P}$ ,  
   and a set of disks  $\mathcal{D}_L$  centred below  $\ell$ .  
2: Output: A set of disks  $\mathcal{D}^* \subset \mathcal{D}'_U \cup \mathcal{D}_L$  covering  $\mathcal{P}$ .  
3: Sort  $\mathcal{D}'_U$  by left-most intersection with  $\ell$   
4:  $\mathcal{P}' \leftarrow \mathcal{P}, k \leftarrow 0$   
5: // The first loops build  $\mathcal{P}^{\text{sort}}$ : points are sorted in  $\mathcal{P}$  so that points covered by  $D_i^U$  are before  
   those covered by  $D_{i+1}^U$ , where  $D_i^U, D_{i+1}^U \in \mathcal{D}'_U$   
6: for  $i := 1$  to  $|\mathcal{D}'_U|$  do  
7:   for  $j := 1$  to  $|\mathcal{P}'|$  do  
8:      $p \leftarrow p_j \in \mathcal{P}'$   
9:     if  $p \in D_i^U$  and  $p \notin D_{i+1}^U$  then  
10:       $p_k^{\text{sort}} \leftarrow p, k \leftarrow k + 1$   
11:       $\mathcal{P}' \leftarrow \mathcal{P}' \setminus \{p\}$   
12:     end if  
13:   end for  
14: end for  
15:  $\mathcal{D}^* \leftarrow \emptyset, i \leftarrow 0, \mathcal{P} \leftarrow \mathcal{P}^{\text{sort}}$   
16: while  $i < |\mathcal{P}|$  do  
17:    $\mathcal{D}' \leftarrow \mathcal{D}'_U \cup \mathcal{D}_L$  // For each point, get all disks to consider again  
18:   while  $|\mathcal{D}'| > 0$  and  $i < |\mathcal{P}|$  do  
19:     while  $p_i \in \mathcal{D}^*$  and  $i < |\mathcal{P}|$  do  
20:        $i \leftarrow i + 1$  // If the current point is already covered, skip it  
21:     end while  
22:     for  $j := 1$  to  $|\mathcal{D}'|$  do  
23:       if  $p_i \notin D'_j$  then  
24:          $\mathcal{D}' \leftarrow \mathcal{D}' \setminus \{D'_j\}$  // Remove all disks not covering the current point  
25:       end if  
26:     end for  
27:      $D_s \leftarrow D'_1 \in \mathcal{D}'$  // Take an arbitrary disk (every disk in  $\mathcal{D}'$  covers all points seen since  
   last disk was added to  $\mathcal{D}^*$ )  
28:      $i \leftarrow i + 1$   
29:   end while  
30:    $\mathcal{D}^* \leftarrow \mathcal{D}^* \cup \{D_s\}$   
31: end while  
32: return  $\mathcal{D}^*$ 
```

The SSDUDC problem was introduced by Ambühl et al. [8, Lemma 1], and they propose a dynamic programming algorithm for a weighted version of the problem. We present a description of their technique with a time bound of $O(m^4n + n \log n)$ in Algorithm 2.4. Note that this is a generalized version of the LSDUDC problem, and so an SSDUDC algorithm may also be applied to that setting. However, the best known running time for SSDUDC exceeds that of LSDUDC by a significant margin. In Algorithm 2.4, we make use of the function $\text{neq}(j, j')$, which evaluates to 1 if $j \neq j'$, and 0 otherwise. Assume that the lines ℓ_1 and ℓ_2 are horizontal. As described in Definition 2.15, all points in \mathcal{P} are contained in the strip defined by the lines, and all disks in \mathcal{D} are centred outside of the strip. We denote the weight of disk D_i by w_{D_i} , and the weights of the lines are $w_{\ell_1} = w_{\ell_2} = 0$

Algorithm 2.4 SSDUDC($\mathcal{P}, \mathcal{D}, \ell_1, \ell_2$)

```

1: Input: A set of points  $\mathcal{P}$  lying between  $\ell_1$  and  $\ell_2$ , a set of disks  $\mathcal{D}$  whose centres are separable
   from  $\mathcal{P}$  by  $\ell_1$  or  $\ell_2$ .
2: Output:  $\mathcal{D}^* \subseteq \mathcal{D}$ , a minimum cardinality set of disks covering  $\mathcal{P}$ .
3: Sort  $\mathcal{P}$  left-to-right
4: Divide  $\mathcal{D}$  into  $\mathcal{D}^U$  and  $\mathcal{D}^L$  (disks centred above  $\ell_1$  and below  $\ell_2$  respectively)
5:  $\mathcal{D}^U \leftarrow \mathcal{D}^U \cup \{\ell_1\}$ ,  $\mathcal{D}^L \leftarrow \mathcal{D}^L \cup \{\ell_2\}$ 
6: for  $i := 1$  to  $n$  do
7:   for  $j := 1$  to  $|\mathcal{D}^U|$  do
8:     for  $k := 1$  to  $|\mathcal{D}^L|$  do
9:        $T[i, j, k] \leftarrow \infty$  // Base case (i.e. if  $p_i \notin D_j^U \cup D_k^L$ )
10:      if  $p_i \in D_j^U \cup D_k^L$  then
11:        if  $i = 1$  then
12:           $T[1, j, k] \leftarrow w_{D_j^U} + w_{D_k^L}$  // Initialize the DP table
13:        else
14:          for  $j' := 1$  to  $|\mathcal{D}^U|$  do
15:            for  $k' := 1$  to  $|\mathcal{D}^L|$  do
16:               $T[i, j, k] \leftarrow \min\{T[i, j, k], T[i - 1, j', k'] + \text{neq}(j, j') \cdot w_{D_j^U} + \text{neq}(k, k') \cdot w_{D_k^L}\}$ 
17:            end for
18:          end for
19:        end if
20:      end if
21:    end for
22:  end for
23: end for
24: // Backtrack from minimum weight cover on  $p_n$  to find  $\mathcal{D}^*$ 
25: return  $\mathcal{D}^*$ 

```

Theorem 2.3. *SSDUDC may be solved exactly in $O(m^4n + n \log n)$ time on weighted disks.*

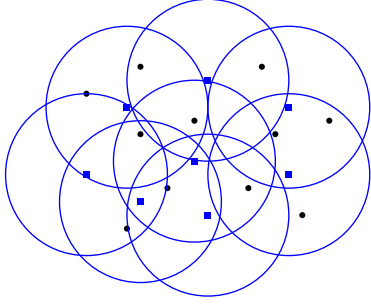
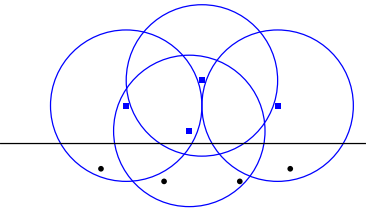
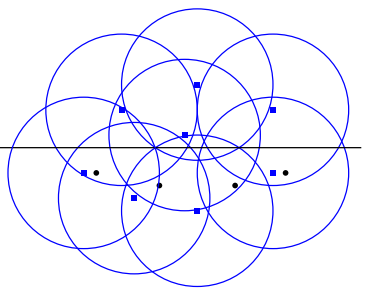
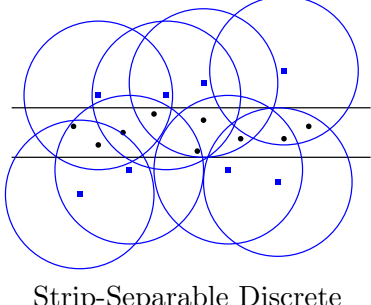
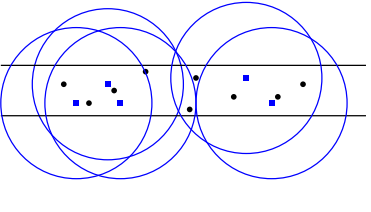
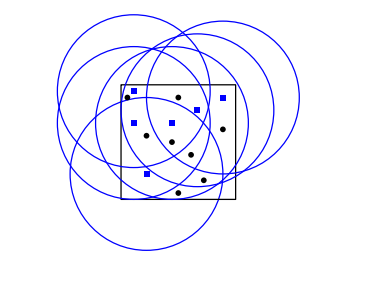
This algorithm consists of five nested for loops and the initial sorting operation; the latter contributes $O(n \log n)$ to the final running time. The outer loop may iterate $O(n)$ times, while the inner four loops may iterate $O(m)$ times each. All other work performed in the loops may be done in constant time, so the total running time of this implementation is $O(m^4n + n \log n)$.

The dual version of the SSDUDC problem has also been studied. In the Double-Sided Disk Cover (DSDC) problem, disks centred in a strip are used to cover points outside of the strip; this

also has an exact dynamic programming solution [163].

There are enough variants of the DUDC problem used in this thesis that a reference table (Table 2.3) describing the settings is warranted.

Table 2.3: A summary of the settings for DUDC used in this thesis.

 <p>Discrete Unit Disk Cover (DUDC), Definition 2.1, page 7.</p>	 <p>Line Separable Discrete Unit Disk Cover (LSDUDC), Definition 2.11, page 18.</p>	 <p>Assisted Line Separable Discrete Unit Disk Cover (A-LSDUDC), Definition 2.12, page 20.</p>
 <p>Strip-Separable Discrete Unit Disk Cover (SSDUDC), Definition 2.15, page 21.</p>	 <p>Within Strip Discrete Unit Disk Cover (WSDUDC), Definition 4.1, page 37.</p>	 <p>BOX Problem, Section 2.3.1.</p>

2.3 38-Approximate DUDC Algorithm

Carmi et al. [29] developed a 38-approximate DUDC algorithm (Algorithm 2.5) which was the inspiration for our work. The algorithm begins by imposing a $3/2 \times 3/2$ grid upon the plane (Figure 2.4). Upon every gridline, the A-LSDUDC approximation algorithm is run on each side, and then each square of the grid is covered using a 6-approximate algorithm for covering points in a $3/2 \times 3/2$ square with points centred in the same square. This latter algorithm, which we call BOX, is rather involved as it requires a check for nine different cases which enumerate various configurations of points and disk centres in the square (we provide more details in Section 2.3.1). The A-LSDUDC algorithm used was a 4-approximation. This distinguishes their algorithm from the 22-approximate algorithm discussed in Section 5.1, which uses the 2-approximate algorithm for A-LSDUDC discussed in Section 3.3. For the approximation factor of their algorithm, they observed that any disk may be used by a single invocation of the BOX algorithm, and twice by

A-LSUDUC for each of the four lines bounding the $3/2 \times 3/2$ square containing the disk (once in the primary sense and once in the assisting sense for each line). Since the first application is for a 6-approximation and each of the latter is for a 4-approximation, the total approximation factor of their algorithm is at most $6 + 8 \times 4 = 38$.

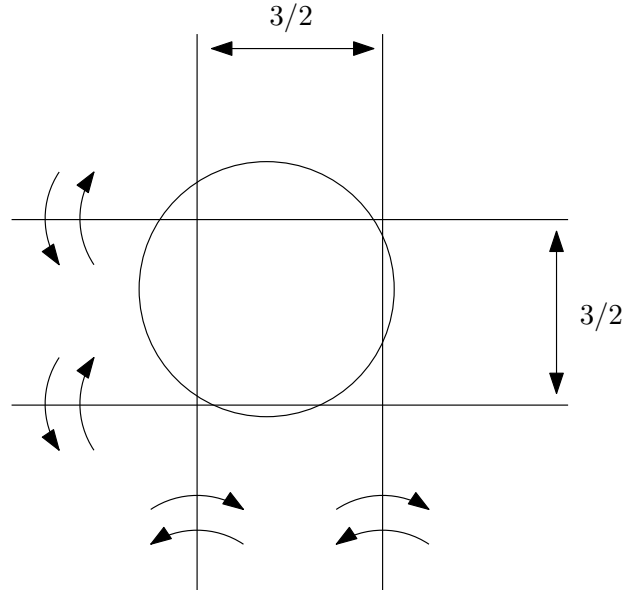


Figure 2.4: The 38-approximate algorithm of Carmi et al. [29] divides the plane into a grid with distance $3/2$ between grid lines. Each disk may be used by the 6-approximate BOX algorithm (Section 2.3.1) to cover the square containing the disk centre, and it may be used in eight applications of the A-LSUDUC algorithm (indicated by the arrows).

2.3.1 The BOX Algorithm

The BOX subroutine of the Carmi et al. [29] technique finds a 6-approximation for the DUDC of all disk centres and points contained in each of the $3/2 \times 3/2$ squares of the grid. Their technique is based on the application of a subset of nine properties depending on where the disk centres are located (the analysis has not been published in entirety, as [29] included only two of the cases). First, they determine whether a solution exists using one or two centres by brute force, which is easily done in $O(m^2n)$ time. Failing this, they divide the square into 9 squares of size $1/2 \times 1/2$, and the setting is classified according to the nine cases based on the positions of the points to be covered and disks centres within these squares. The determination of which properties may be applied can be done in $O(m)$ time, and there are only two expensive steps that may be used in any of the procedures, each of which may only be used a constant number of times. First is the assisted LSDUDC technique, whose running time is $O(mn + m \log m + n \log n)$. The second technique that may be required is to determine the optimal disk cover of a set of points using centres contained in one of the $1/2 \times 1/2$ squares, which can be solved in $O(m^2n^4)$ time using the technique presented in [108]. The centre of each disk can only be contained in one square, and so this operation is never performed twice for any given disk. Therefore, the complete DUDC algorithm achieves worst-case performance when all of the disk centres in the plane are confined to a single $1/2 \times 1/2$ square, so that the $O(m^2n^4)$ operation is performed over the entire data set.

Algorithm 2.5 DUDC-38(\mathcal{P}, \mathcal{D})

- 1: **Input:** A set of points \mathcal{P} and a set of unit disks \mathcal{D} .
 - 2: **Output:** $\mathcal{D}^* \subseteq \mathcal{D}$, a set of disks covering \mathcal{P} with a 38-factor approximation.
 - 3: Apply $3/2 \times 3/2$ axis-aligned grid to the problem area
 - 4: $\mathcal{D}^* \leftarrow \emptyset$
 - 5: **for** Each vertical line ℓ_i in the grid **do**
 - 6: $\{\mathcal{P}_L, \mathcal{P}_R\} \subseteq \mathcal{P}$ (Divide points into sets left and right of line ℓ_i)
 - 7: $\{\mathcal{D}_L, \mathcal{D}_R\} \subseteq \mathcal{D}$ (Divide disks into sets centred left and right of line ℓ_i)
 - 8: $\mathcal{D}^* \leftarrow \mathcal{D}^* \cup \text{UA-LSDUDC}(\mathcal{P}_L, \mathcal{D}_R, \ell_i)$ (Algorithm 3.3, page 34)
 - 9: $\mathcal{D}^* \leftarrow \mathcal{D}^* \cup \text{UA-LSDUDC}(\mathcal{P}_R, \mathcal{D}_L, \ell_i)$
 - 10: **end for**
 - 11: **for** Each horizontal line ℓ_i in the grid **do**
 - 12: $\{\mathcal{P}_L, \mathcal{P}_U\} \subseteq \mathcal{P}$ (Divide points into sets below and above line ℓ_i)
 - 13: $\{\mathcal{D}_L, \mathcal{D}_U\} \subseteq \mathcal{D}$ (Divide disks into sets centred below and above line ℓ_i)
 - 14: $\mathcal{D}^* \leftarrow \mathcal{D}^* \cup \text{UA-LSDUDC}(\mathcal{P}_L, \mathcal{D}_U, \ell_i)$
 - 15: $\mathcal{D}^* \leftarrow \mathcal{D}^* \cup \text{UA-LSDUDC}(\mathcal{P}_U, \mathcal{D}_L, \ell_i)$
 - 16: **end for**
 - 17: **for** Each $3/2 \times 3/2$ box B_i in the grid **do**
 - 18: $\mathcal{P}_I \subseteq \mathcal{P}$ (Find points located inside box B_i)
 - 19: $\mathcal{D}_I \subseteq \mathcal{D}$ (Find disks centred inside box B_i)
 - 20: $\mathcal{D}^* \leftarrow \mathcal{D}^* \cup \text{UBOX}(\mathcal{P}_I, \mathcal{D}_I)$
 - 21: **end for**
 - 22: **return** \mathcal{D}^*
-

(Assisted) Line-Separable Discrete Unit Disk Cover

In this chapter, we describe a polynomial-time algorithm for the Line-Separable Discrete Unit Disk Cover (LSDUDC) problem¹. We extend this algorithm by demonstrating that the minimum assisted cover (MAC) algorithm (Algorithm 2.3) for LSDUDC may be applied to provide a 2-approximate solution to the Assisted Line-Separated Discrete Unit Disk Cover (A-LSDUDC) problem. To begin, we present a greedy LSDUDC algorithm for which the correctness is straightforward to verify. We refine the approach with a graph-based algorithm with an $O(m^2n)$ worst-case running time, and we use the greedy algorithm to establish the correctness of this approach. Recall that subsequent work [39] has shown an $O(mn + n \log n)$ time algorithm for LSDUDC, and so we use that algorithm for our A-LSDUDC results.

3.1 Greedy Algorithm for LSDUDC

In the LSDUDC problem (see Definition 2.11 on page 18), we have two sets of points in the plane $\mathcal{P} = \{p_1, \dots, p_n\}$ and $\mathcal{Q} = \{q_1, \dots, q_m\}$ separated by a line ℓ . Let D_i denote the unit disk centred at q_i , for $i \in \{1, \dots, m\}$, and let \mathcal{D} denote the set of these disks. We use q_i and D_i interchangeably, e.g., our solution can be considered both as a set of points (a subset of \mathcal{Q}) and as a set of disks. Further, when we discuss the intersection of a line with a disk, we are referring to the intersection of the line with ∂D_i , the boundary of the disk.

We want to find a subset $\mathcal{Q}^* \subseteq \mathcal{Q}$ of minimum cardinality such that all points of \mathcal{P} are covered by the set of unit disks \mathcal{D}^* centred at the points of \mathcal{Q}^* . An instance of the problem is shown in Figure 3.1. Without loss of generality we assume that ℓ is a horizontal line and points of \mathcal{Q} are above ℓ .

Theorem 3.1. *Given sets \mathcal{P} of n points and \mathcal{Q} of m points in the plane, where \mathcal{P} and \mathcal{Q} can be separated by a line ℓ , and each $q \in \mathcal{Q}$ is the centre point of a unit disk $D \in \mathcal{D}$, the LSDUDC problem can be solved exactly in $O(m^2n)$ time.*

We present an iterative algorithm for the LSDUDC problem in Algorithm 3.1. During the execution of the algorithm, it may be determined that a disk $D \in \mathcal{D}$ should be added to the solution or that it is not relevant for the remainder of the computation of the solution set. When

¹Elements of this chapter have appeared in [39] and [40].

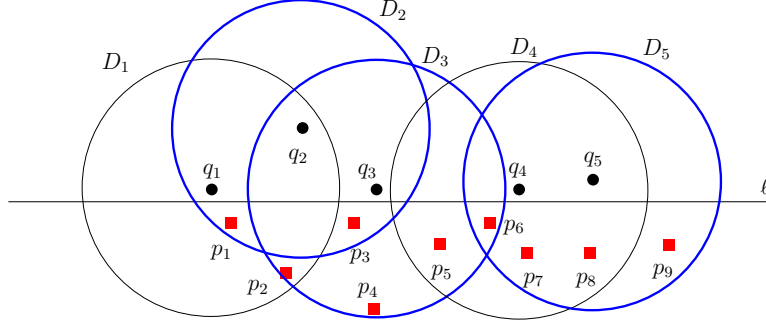


Figure 3.1: An instance of the Line-Separable Discrete Unit Disk Cover (LSDUDC) problem. The set $\mathcal{D}^* = \{D_2, D_3, D_5\}$ (the bold disks) forms an optimal solution.

this occurs, we remove disk D from the problem. Similarly, we remove a point $p \in \mathcal{P}$ if this point is not relevant for the remainder of the computation (i.e., point p is covered by a disk in the partial solution already constructed). Our algorithm relies on the following three observations:

1. If a disk D_i covers no points from \mathcal{P} , we remove it.
2. If a disk D_i is *dominated* by a disk D_j , then we can remove D_i from the problem instance. Disk D_j dominates D_i if it covers all points of \mathcal{P} covered by D_i , i.e. $(D_i \cap \mathcal{P}) \subseteq (D_j \cap \mathcal{P})$. If two disks cover the same subset of points from \mathcal{P} , we designate the dominating disk as that whose left intersection with ℓ is rightmost.
3. If a point $p_1 \in \mathcal{P}$ is only covered by a disk D_i , then D_i must be part of the solution. We remove D_i together with all points of \mathcal{P} covered by D_i from the problem sets.

These three observations give us three SIMPLIFICATION rules, formalized in Procedure 3.2. The idea is to apply these rules to as many disks as possible in order to simplify the problem. For example, consider the problem instance shown in Figure 3.1. Initially all disks cover points and no disk dominates any other, so neither the first or second rules are applicable. Disk D_3 is the only disk that covers p_4 and, similarly, disk D_5 is the only disk that covers p_9 . Thus we add D_3 and D_5 to the (initially empty) solution and remove them from the problem set, together with the points that are covered by the disks, namely $\{p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$. Now disk D_4 covers no point and can be removed. There is only one remaining point (p_1) and it is covered by the two remaining disks (D_1 and D_2). According to our convention, D_1 is dominated by D_2 and is removed. Now D_2 is the only disk covering p_1 . We add D_2 to the solution and remove D_2 and p_1 . No disks or points remain and we are done. Thus the SIMPLIFICATION rules suffice for this instance and give an optimal solution $\{D_2, D_3, D_5\}$. This example also illustrates that an optimal solution is not necessarily unique, as $\{D_1, D_3, D_5\}$ is also an optimal solution. In general, however, these SIMPLIFICATION rules do not suffice to obtain an optimal solution. As shown in Figure 3.2, if given only disks D_1, D_2 and D_3 and points p_1, p_2 and p_3 , then no point $p \in \mathcal{P}$ is covered by only one disk and no disk dominates any other.

We augment the SIMPLIFICATION rules with a greedy step to solve the LSDUDC problem. We order the disks so that l_i , the left intersection of D_i with ℓ , is to the left of l_{i+1} . We say that D_i precedes D_{i+1} in the ordering (the disks in Figures 3.1 and 3.2 follow this convention). We present the algorithm, GREEDY, in Algorithm 3.1. It works by first applying the SIMPLIFICATION rules as many times as possible, and then it finds the first remaining disk in the left-to-right order,

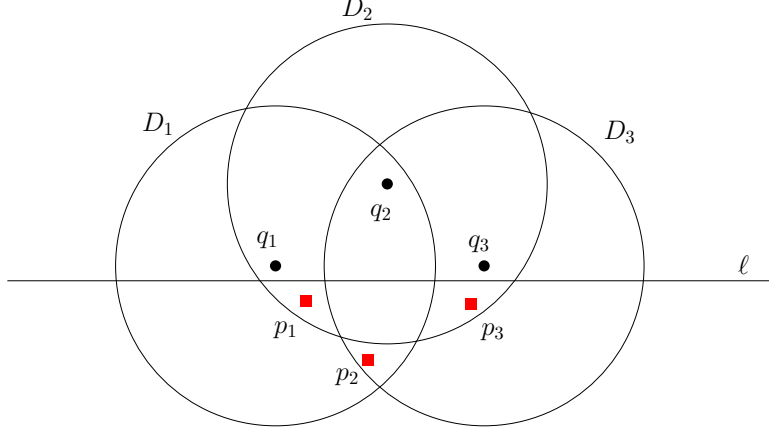


Figure 3.2: An example to demonstrate that the set of SIMPLIFICATION rules do not suffice to obtain an optimal solution. No point $p \in \mathcal{P}$ is covered by only one disk and no disk dominates any other.

say D_j . We add D_j to our solution and remove D_j from \mathcal{D} and all points covered by D_j from \mathcal{P} . We exhaustively apply the SIMPLIFICATION rules followed by the greedy step, repeating these in turn until all disks have been removed. Since we remove at least one disk at each greedy step, the algorithm terminates after at most m iterations. See Algorithm 3.1 for the corresponding pseudocode.

Algorithm 3.1 GREEDY ($\mathcal{P}, \mathcal{D}, \ell$)

- 1: **Input:** A set of points \mathcal{P} and a set of disks \mathcal{D} , where \mathcal{P} and the centres of \mathcal{D} are separable by ℓ .
 - 2: **Output:** $\mathcal{D}^* \subseteq \mathcal{D}$, a minimum cardinality set of disks covering \mathcal{P} .
 - 3: $\mathcal{D} \leftarrow \text{sortLeftToRight}(\mathcal{D})$ // Sort in increasing order of left intersection with ℓ
 - 4: $\mathcal{D}^* \leftarrow \emptyset$
 - 5: **while** $\mathcal{D} \neq \emptyset$ **do**
 - 6: SIMPLIFICATION ($\mathcal{P}, \mathcal{D}, \mathcal{D}^*$) // may modify \mathcal{P} , \mathcal{D} , and \mathcal{D}^* (Procedure 3.2)
 - 7: $D_\ell \leftarrow$ leftmost disk in \mathcal{D}
 - 8: $\mathcal{D}^* \leftarrow \mathcal{D}^* \cup \{D_\ell\}$
 - 9: $\mathcal{D} \leftarrow \mathcal{D} \setminus D_\ell$
 - 10: $\mathcal{P}' \leftarrow \{p \in \mathcal{P} \mid p \text{ is contained in } D_\ell\}$
 - 11: $\mathcal{P} \leftarrow \mathcal{P} \setminus \mathcal{P}'$
 - 12: **end while**
 - 13: **return** \mathcal{D}^*
-

3.1.1 Correctness of GREEDY

We now establish the correctness of Algorithm 3.1 by proving that GREEDY gives a minimum LSDUDC solution. Assume for the sake of contradiction that there is an algorithm OPT that finds a cover with fewer disks than GREEDY. Let D_1 be the first disk in the ordering that is selected by GREEDY but not by OPT. Let C_1 be the set of points in \mathcal{P} that are covered by D_1 (we consider only the remaining points and disks, i.e., those that have not been removed by the algorithm). First assume that C_1 is covered by a single disk D_0 in the solution of OPT. Since

Procedure 3.2 SIMPLIFICATION ($\mathcal{P}, \mathcal{D}, \mathcal{D}^*$)

```
1:  $m \leftarrow \text{true}$  // Iterate as long as modifications are possible
2: while  $m$  do
3:    $m \leftarrow \text{false}$ 
4:   for  $i := 1$  to  $|\mathcal{D}|$  do
5:     if  $D_i \cap \mathcal{P} = \emptyset$  then
6:        $\mathcal{D} \leftarrow \mathcal{D} \setminus \{D_i\}$ 
7:        $m \leftarrow \text{true}$ 
8:     end if
9:     for  $j := 1$  to  $|\mathcal{D}|$  do
10:      if  $i \neq j$  and  $D_i \cap \mathcal{P} \subseteq D_j \cap \mathcal{P}$  then
11:         $\mathcal{D} \leftarrow \mathcal{D} \setminus \{D_i\}$ 
12:         $m \leftarrow \text{true}$ 
13:      end if
14:    end for
15:  end for
16:  for  $i := 1$  to  $|\mathcal{P}|$  do
17:     $\mathcal{D}_{p_i} = \{D \in \mathcal{D} \mid p_i \in D\}$ 
18:    if  $|\mathcal{D}_{p_i}| = 1$  then
19:       $D' \leftarrow D \in \mathcal{D}_{p_i}$ 
20:       $\mathcal{D}^* \leftarrow \mathcal{D}^* \cup \{D'\}$ 
21:       $\mathcal{D} \leftarrow \mathcal{D} \setminus \{D'\}$ 
22:       $\mathcal{P} \leftarrow \mathcal{P} \setminus (\mathcal{P} \cap D')$ 
23:       $m \leftarrow \text{true}$ 
24:    end if
25:  end for
26: end while
```

D_1 is not removed in the SIMPLIFICATION step, it is not dominated by any other disk. Thus the only possibility is that D_0 and D_1 cover exactly the same set of (remaining) points (i.e., set C_1) and D_0 precedes D_1 in the ordering. In this case, we replace D_0 with D_1 in OPT, pushing the first difference between the solution of GREEDY and OPT to the right. Otherwise, C_1 is covered by at least two disks in the solution of OPT. Let D_2 and D_3 be two disks in the solution of OPT such that each of them cover a strict subset of C_1 . Without loss of generality assume that D_2 precedes D_3 in the ordering. We prove that $D_1 \cup D_3$ covers all points of \mathcal{P} covered by $D_2 \cup D_3$.

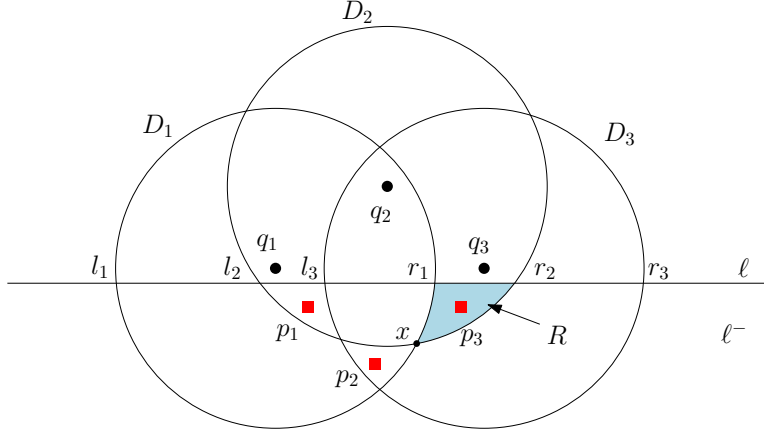


Figure 3.3: Illustration of the proof of correctness of GREEDY. If D_1 is the first disk selected by GREEDY and not by OPT, then OPT must have D_2 and D_3 in its solution.

Let l_i and r_i denote the respective left and right intersection points of the boundary of the unit disk D_i with the line ℓ , for $i \in \{1, 2, 3\}$. If D_2 precedes D_1 in the ordering, D_1 dominates D_2 (otherwise, GREEDY would select D_2 and not D_1 at this step). In this case we replace D_2 with D_1 in OPT, pushing the difference between the two algorithms to the right. Hence we are left with the case in which D_1 precedes D_2 and D_2 precedes D_3 in the ordering. Thus, the points are ordered $l_1, l_2, l_3, r_1, r_2, r_3$ along line ℓ (see Figure 3.3). Note that we cannot have a pair of disks nested above ℓ , i.e. l_i and r_i cannot appear between l_j and r_j for any i and j , otherwise the nested disk is dominated by the other. Furthermore, we know that $(D_1 \cap D_3) \setminus D_2 \neq \emptyset$, and we define $R = (D_2 \setminus D_1) \cap \ell^-$. It suffices to prove that R is completely contained in D_3 .

Proposition 3.1. *Region R is contained in disk D_3 .*

Proof. Since points r_1 and r_2 both lie between l_3 and r_3 on line ℓ , both points r_1 and r_2 are in disk D_3 . Let x denote the rightmost point of the intersection of ∂D_1 and ∂D_2 (the boundaries of disks D_1 and D_2). Observe that x lies on the boundary of region $(D_1 \cap D_3) \setminus D_2$. Consequently, $x \in D_3$. Since the boundary of R consists of arcs of unit disks joining the points x , r_1 , and r_2 , it follows that R is contained in the 1-hull of $\{x, r_1, r_2\}$, where the 1-hull of $\{x, r_1, r_2\}$ is the intersection of all unit disks that contain $\{x, r_1, r_2\}$, denoted $1-H(\{x, r_1, r_2\})$. Since $\{x, r_1, r_2\} \subseteq D_3$, it follows that $R \subseteq 1-H(\{x, r_1, r_2\}) \subseteq D_3$. \square

Therefore, by replacing D_2 with D_1 in the OPT solution, we will have a feasible solution with the same number of disks. This pushes the first difference between the solution of GREEDY and OPT to the right. By iterating this argument we prove that the solution returned by GREEDY uses the same number of disks as OPT and therefore GREEDY is an optimal algorithm. We describe an implementation of the algorithm on graphs in the following section.

3.2 GREEDY-GRAPH

We construct a graph $G = (V, E)$, where each vertex $v_i \in V$ corresponds to disk D_i for $i \in \{1, \dots, m\}$ (recall that D_i is the i^{th} disk sorted according to its left intersection with ℓ). We also associate a counter c_{v_i} to each vertex v_i that stores the number of points in \mathcal{P} contained in disk D_i that have not yet been covered by the algorithm. Similarly, we associate with each edge $e = (v_{i_1}, v_{i_2})$ a counter c_e that represents the number of points contained in $D_{i_1} \cap D_{i_2}$. This graph can be constructed in $O(m^2n)$ time by checking which points are contained in the intersection of each pair of disks, adding the corresponding edges, and updating the vertex and edge counters. The algorithm GREEDY-GRAPH starts by traversing the vertices in order v_1, v_2, \dots, v_m . At each vertex v_i , there are three possible cases:

- *Rule 1:* The counter c_{v_i} is 0; in this case D_i does not contain any points or is dominated by a set of disks that has already been added to the solution. This disk will not be in the solution set, so we can dismiss this vertex and continue with the next. This is analogous to the first SIMPLIFICATION rule.
- *Rule 2:* There is an edge $e = (v_i, v_k)$, $k > i$, such that $c_e = c_{v_i}$; in this case we know that D_i is dominated by disk D_k . Again, we dismiss this vertex and continue. This corresponds to an application of the second SIMPLIFICATION rule.
- *Rule 3:* Every edge $e = (v_i, v_k)$, $k > i$, satisfies $c_e < c_{v_i}$, meaning that disk D_i is not dominated by any disk to its right. In this case we add D_i to the solution set and we eliminate all remaining points contained by this disk from the graph. We continue with the next vertex in the graph. This is an application of the third rule of SIMPLIFICATION and the greedy step.

In order to identify the appropriate case above we traverse the adjacency list of each vertex we visit. This requires $O(m)$ time in the worst case. When a disk is added to the solution in the third case, all points covered by the disk must be eliminated. Consider the elimination of a point p in disk D_i . Let $N(v_i) = \{v_k \mid c_{(v_i, v_k)} > 0\}$. For all $v_k \in N(v_i)$, we decrease c_{v_k} and $c_{(v_i, v_k)}$ by one. In addition, for each pair of elements $\{v_{k_1}, v_{k_2}\} \subseteq N(v_i)$, we check whether the point is covered by both disks. If so, we decrease $c_{(v_{k_1}, v_{k_2})}$ by one. This can take at most $O(m^2)$ time per point, thus the time required for eliminating all points is bounded by $O(m^2n)$ time. Since the time required to construct the graph is $O(m^2n)$, the GREEDY-GRAPH algorithm takes $O(m^2n)$ time.

3.2.1 Correctness of GREEDY-GRAPH

We now demonstrate that the GREEDY-GRAPH algorithm is optimal by showing that the set of disks returned by this algorithm has the same cardinality as that returned by the GREEDY algorithm presented in Algorithm 3.1.

Lemma 3.1. *If \mathcal{D}' is the disk cover returned by GREEDY-GRAPH, and \mathcal{D}^* is the disk cover returned by GREEDY, then $|\mathcal{D}'| = |\mathcal{D}^*|$.*

Proof. Assume for the sake of contradiction that $|\mathcal{D}'| \neq |\mathcal{D}^*|$. Recall that GREEDY is optimal, therefore $|\mathcal{D}^*|$ and $|\mathcal{D}'|$ can only differ if $|\mathcal{D}'| > |\mathcal{D}^*|$. Let D_1 be the first disk in the left-to-right

order that is present in the solution of GREEDY-GRAPH, and not in the solution of GREEDY. At some point during its execution, GREEDY must have decided to discard disk D_i . The only mechanisms in GREEDY for discarding disks are the first and second SIMPLIFICATION rules. Recall that the first rule removes a disk if it contains no points, and the second rule discards a disk if it is dominated by some other disk. We now show that for any of the following possible events, GREEDY-GRAPH will discard the same disk D_1 .

- **Empty** - Suppose D_1 contains no points. In this case, GREEDY-GRAPH will find that $c_{v_1} = 0$. Therefore, D_1 will be discarded by Rule 1, in contradiction to our assumption.
- **Dominance (right)** - Now suppose D_1 is dominated by some disk to the right, D_r . In this case, we will encounter D_1 first during our walk, and we will have that $c_{v_1} = c_{(v_1, v_r)}$. Therefore, GREEDY-GRAPH will remove D_1 by Rule 2, in contradiction to our assumption.
- **Dominance (left)** - Suppose D_1 is dominated by some disk to the left, D_l . In this case, we will have encountered D_l first during our walk. There are two possible cases in this scenario:
 - (i) If $c_{v_\ell} > c_{(v_\ell, v_k)}$ for all D_k , D_l is added to \mathcal{D}' by Rule 3 of GREEDY-GRAPH. All points covered by D_l are removed, leaving no points covered by D_1 . This is now an instance of the Empty case.
 - (ii) Otherwise, $c_{v_\ell} = c_{(v_\ell, v_k)}$ for some D_k . This means that D_l is dominated by D_k . GREEDY-GRAPH would discard D_l by Rule 3. By transitivity, D_k also dominates D_1 . If D_k is to the right of D_1 , then this is now an instance of Dominance (right), and thus we reach a contradiction. If D_k is to the left of D_1 , then this is again an instance of Dominance (left), so we apply this same argument recursively. The recursion stops either when we reach an instance of Dominance (right) or case (i) of Dominance (left).

□

We have shown that the solution of GREEDY-GRAPH has the same cardinality as the solution of GREEDY, and since GREEDY is optimal, so is GREEDY-GRAPH.

3.3 Assisted LSDUDC

We now show that an exact algorithm for the LSDUDC problem may be used to achieve a 2-approximation algorithm for the Assisted Line Separated Discrete Unit Disk Cover (A-LSDUDC) problem (Definition 2.12, page 20). The approximation algorithm is based on an adaptation of the 4-approximation algorithm of Carmi et al. [29].

Theorem 3.2. *The A-LSDUDC problem has a factor 2 approximation algorithm which runs in $O(mn + m \log m + n \log n)$ time.*

Our solution does not necessarily use disks that contribute to the semi-chain \mathcal{S} (Definition 2.13, page 21). Instead, we first solve the LSDUDC problem optimally using Algorithm 2.2 on the set of disks \mathcal{D}_U to obtain a disk set \mathcal{D}_U^* . Let $\mathcal{D}_U^* = \{D_1^*, D_2^*, \dots, D_{|\mathcal{D}_U^*|}^*\}$ be the ordered set of unit disks from left to right based on the left intersection point of the disks in \mathcal{D}_U^* with ℓ . We

Algorithm 3.3 A-LSDUDC($\mathcal{P}, \mathcal{D}, \ell$)

- 1: **Input:** A set of points \mathcal{P} contained in one half-plane defined by ℓ , and a set of unit disks \mathcal{D} .
 - 2: **Output:** $\mathcal{D}' \subseteq \mathcal{D}$, a set of disks covering \mathcal{P} with a 2-factor approximation.
 - 3: Assign disks of \mathcal{D} to \mathcal{D}_U and \mathcal{D}_L , where \mathcal{D}_L and \mathcal{P} are in the same half-plane of ℓ
 - 4: $\mathcal{D}'' \leftarrow \text{LSDUDC}(\mathcal{P}, \mathcal{D}_U, \ell)$
 - 5: $\mathcal{D}' \leftarrow \text{MAC}(\mathcal{P}, \mathcal{D}'', \mathcal{D}_L, \ell)$
 - 6: **return** \mathcal{D}'
-

use the greedy MAC algorithm (Algorithm 2.3) over the sets \mathcal{D}_U^* and \mathcal{D}_L to obtain an improved solution \mathcal{D}' for covering points in \mathcal{P} .

Now we wish to compare the cardinality of the solution found by A-LSDUDC, call it \mathcal{D}' , with that of the global minimum disk cover \mathcal{D}^* . Consider the upper and lower components of the solutions \mathcal{D}' and \mathcal{D}^* , i.e., $\mathcal{D}'_U = \mathcal{D}' \cap \mathcal{D}_U$, $\mathcal{D}'_L = \mathcal{D}' \cap \mathcal{D}_L$, $\mathcal{D}^*_U = \mathcal{D}^* \cap \mathcal{D}_U$, and $\mathcal{D}^*_L = \mathcal{D}^* \cap \mathcal{L}$. Let \mathcal{U}^* be the optimal LSDUDC solution, i.e. $\mathcal{U}^* = \text{LSDUDC}(\mathcal{P}, \mathcal{D}_U, \ell)$. Note that $|\mathcal{D}^*| \leq |\mathcal{D}'|$ since \mathcal{D}^* is the global minimum. Similarly, since \mathcal{D}' is the minimum assisted cover based on \mathcal{U}^* , it follows that $|\mathcal{D}'| = |\mathcal{D}'_U| + |\mathcal{D}'_L| \leq |\text{ac}(\mathcal{U}^*, \mathcal{D}^*_L)| + |\mathcal{D}^*_L|$, where $\text{ac}(\mathcal{U}^*, \mathcal{D}^*_L)$ is the smallest subset of \mathcal{U}^* that forms an assisted cover with \mathcal{D}^*_L .

Now we will show that $2|\mathcal{D}^*_U| \geq |\text{ac}(\mathcal{U}^*, \mathcal{D}^*_L)|$. Given a disk D in \mathcal{D}^*_U , there are two cases: either D lies above the lower boundary of $\text{ac}(\mathcal{U}^*, \mathcal{D}^*_L)$, i.e., D is contained in the union of all the disks in $\text{ac}(\mathcal{U}^*, \mathcal{D}^*_L)$, or D contains one or more arc segments of the lower boundary of $\text{ac}(\mathcal{U}^*, \mathcal{D}^*_L)$. In the first case, Carmi et al. [29] show that at most two disks in $\text{ac}(\mathcal{U}^*, \mathcal{D}^*_L)$ suffice to cover D and, hence, for every such disk in the global optimum solution \mathcal{D}^* there are at most two disks in $\text{ac}(\mathcal{U}^*, \mathcal{D}^*_L)$. In the second case, let \mathcal{V} denote the subset of disks that have lower boundary segments that are contained in D . The set of arc segments of the disks in \mathcal{V} consists of, from left to right, a partially-covered arc segment of the lower boundary, followed by zero or more fully-covered arc segments, and then a partially-covered arc segment. Let \mathcal{W} denote the disks whose arcs are partially covered together with D . \mathcal{W} dominates \mathcal{V} and hence there is at most one arc of the lower boundary fully contained in D ; otherwise replacing \mathcal{V} with \mathcal{W} results in a cover of smaller cardinality, deriving a contradiction, since $\mathcal{V} \subset \mathcal{U}^*$, and \mathcal{U}^* is the optimal LSDUDC solution. Recall that all disks in \mathcal{V} and \mathcal{D}_U are centred above ℓ , and all points in \mathcal{P} are below ℓ . Furthermore, observe that the partially-covered arc disks must contain points not contained in the fully-covered disk; otherwise they can also be eliminated while reducing the cardinality of the cover. As those disks contain other points, each of the disks is partially covered by at least one other disk in \mathcal{D}^* . We arbitrarily associate each disk covered more than once with its leftmost disk in \mathcal{D}^* . Thus, of the (at most) three disks in \mathcal{V} , at most two are associated with D . In sum, in either case each disk in \mathcal{D}^*_U has at most two associated disks in $\text{ac}(\mathcal{U}^*, \mathcal{D}^*_L)$ from which it follows that $2|\mathcal{D}^*_U| \geq |\text{ac}(\mathcal{U}^*, \mathcal{D}^*_L)|$. Hence,

$$\begin{aligned} 2|\mathcal{D}^*| &= 2(|\mathcal{D}^*_U| + |\mathcal{D}^*_L|) \\ &\geq 2|\mathcal{D}^*_U| + |\mathcal{D}^*_L| \\ &\geq |\text{ac}(\mathcal{U}^*, \mathcal{D}^*_L)| + |\mathcal{D}^*_L| \\ &\geq |\mathcal{D}'_U| + |\mathcal{D}'_L| \\ &\geq |\mathcal{D}'|, \end{aligned}$$

which gives the approximation factor of two as desired.

The running time of the A-LSDUDC algorithm is determined by the running times of the LSDUDC and MAC algorithms that are used. The LSDUDC algorithm described in Section 2.2.1 runs in $O(mn + n \log n)$ time, and the MAC algorithm (Algorithm 2.3) runs in $O(mn + m \log m)$ time as well. Therefore, the overall running time is $O(mn + m \log m + n \log n)$ time.

3.4 Conclusions and Future Work

The algorithms presented in this chapter provide an exact LSDUDC algorithm and a 2-approximate algorithm for A-LSDUDC, which run in $O(m^2n)$ time and $O(mn + m \log m + n \log n)$ time respectively.

The A-LSDUDC problem is still open, in the sense that there is no polynomial time algorithm and no NP-completeness proof. It would be interesting to see a hardness proof for A-LSDUDC, but an exact polynomial time algorithm would be exciting. The latter result would immediately improve the DUDC algorithms presented in Chapter 5 of this thesis.

Within-Strip Discrete Unit Disk Cover

In this chapter, we consider a restricted version of DUDC in which both the points and disk centres are confined to a strip of the plane¹. We are motivated by the problem of updating an area of seafloor terrain following a survey by a ship. A survey is typically performed by towing an echosounder behind a ship, which results in a swath of data points along the path of the ship. If the ship performs this task while in transit, the surveyed area forms a strip along the seafloor. We wish to update the seafloor data set by treating the new data set \mathcal{P} as the standard, while maintaining representative points from the old data set \mathcal{Q} for completeness. Our goal is to find a minimum cardinality set $\mathcal{Q}^* \subseteq \mathcal{Q}$ so that each point in \mathcal{P} is within a unit distance of a point in \mathcal{Q}^* . This is an instance of the *Within-Strip Discrete Unit Disk Cover* (WSDUDC) problem.

To begin, we use a fixed partitioning scheme which divides the strip into squares to provide a 6-approximate algorithm for solving the Within-Strip Discrete Unit Disk Cover problem on strips of height $h \leq 1/\sqrt{2}$ (≈ 0.71), which runs in $O(mn + m \log m + n \log n)$ time. Next, we refine this idea with a general $3\lceil 1/\sqrt{1-h^2} \rceil$ -approximate algorithm for strips of height $h < 1$, which runs in $O(m^4n + n \log n)$ time. Given a strip of height at most $2\sqrt{2}/3$ (≈ 0.94), a 4-approximate solution is given which improves upon the general algorithm by checking for simple redundancy while still running in $O(m^4n + n \log n)$ time. For a strip of height at most $4/5 = 0.8$, an $O(m^6n + n \log n)$ time 3-approximate solution is provided which uses dynamic programming to solve all sub-problems optimally (note that using the general $3\lceil 1/\sqrt{1-h^2} \rceil$ -approximate algorithm on strips of height $2\sqrt{2}/3$ or $4/5$ would produce a 6-approximation). These results are summarized in Table 4.1. To conclude, we show that WSDUDC is NP-complete for a strip of height $h > 0$, and so no exact polynomial time algorithm is possible for the problem unless P=NP.

In the WSDUDC problem, the input consists of a set of m unit disks \mathcal{D} with centre points \mathcal{Q} , and a set of n points \mathcal{P} , all of which lie in the Euclidean plane. We define the *strip* s of height h as the region of the plane between two parallel lines ℓ_1 and ℓ_2 , where $\mathcal{Q} \cap s = \mathcal{Q}$ and $\mathcal{P} \cap s = \mathcal{P}$. We assume that we are provided with the lines ℓ_1 and ℓ_2 ; alternatively, a minimum width strip may be computed (e.g. [154]). For our discussion, we assume that the strip is horizontal. We wish to determine the minimum cardinality set of disks $\mathcal{D}^* \subseteq \mathcal{D}$ such that $\mathcal{P} \cap (\cup \mathcal{D}^*) = \mathcal{P}$.

Definition 4.1. The Within-Strip Discrete Unit Disk Cover (WSDUDC) Problem: *Given a set \mathcal{P} of n points, a set \mathcal{D} of unit radius disks in the plane, where each disk $D \in \mathcal{D}$ is centred at a point in a set \mathcal{Q} of m points, and parallel lines ℓ_1 and ℓ_2 such that $\ell_1 \subset \ell_2^+$ and $\ell_2 \subset \ell_1^-$, which*

¹Elements of this chapter have appeared in [46], [47] and [73].

Table 4.1: Approximation Factors and Running Times of WSDUDC Algorithms

Approximation Factor	Running Time	Height of Strip	Section
6	$O(mn + m \log m + n \log n)$	$\leq 1/\sqrt{2}$	4.1
$3\lceil 1/\sqrt{1-h^2} \rceil$	$O(m^4n + n \log n)$	≤ 1	4.2
4	$O(m^4n + n \log n)$	$\leq 2\sqrt{2}/3$	4.3
3	$O(m^6n + n \log n)$	$\leq 4/5$	4.4

define a strip $s = \ell_1^- \cap \ell_2^+$ of height h such that $\mathcal{P} \subset s$ and $\mathcal{Q} \subset s$, the *Within-Strip Discrete Unit Disk Cover* problem is to find a set $\mathcal{D}^* \subseteq \mathcal{D}$ of minimum cardinality such that \mathcal{D}^* covers \mathcal{P} .

This is a seemingly simpler context than the general Discrete Unit Disk Cover problem (DUDC), which is only distinguished from WSDUDC by the fact that DUDC has no strip confining the positions of the points and disks. This chapter addresses an interesting question regarding the hardness of the general DUDC problem: while it is known that the problem of covering points in a strip with disks centred outside the strip is polynomial time solvable [8] (see SSDUDC in Section 2.2.3 for details), it remained to be seen whether WSDUDC is NP-hard for a strip of fixed height.

An implication of a polynomial time algorithm for WSDUDC for strips of any fixed height would be a simple PTAS for DUDC, using the shifting techniques of Hochbaum and Maass [93], as described in Section 2.1.6. The PTAS for DUDC [121], as discussed in Section 2.1.3, uses fundamentally different techniques.

The notion of decomposing a problem into strip-based subproblems is natural, since an exact algorithm or PTAS for the subproblem can potentially be used to derive a general PTAS using the shifting strategy. For example, the PTAS for the Minimum Geometric Unit Disk Cover problem (Section 2.1.6) operates by dividing the problem into strips [93]. The maximum independent set of a unit disk graph may be found in polynomial time if the setting is confined to a strip of fixed height [119]. Geometric set cover on unit squares (precisely WSDUDC, except the disks are replaced with axis-aligned unit squares) may be solved optimally in $n^{O(k)}$ time when confined to strips of height k [64]. Considering these results, the hardness of WSDUDC is somewhat surprising.

In Section 2.2.3, we discussed the Strip-Separated Discrete Unit Disk Cover (SSDUDC) problem. In that setting, the input consists of a set of points \mathcal{P} located in a strip in the plane, like WSDUDC, but the set of unit disk centres \mathcal{Q} lies strictly outside of the strip rather than in the strip. We described an $O(m^4n + n \log n)$ dynamic programming algorithm for SSDUDC in Algorithm 2.4, which we use in several of the WSDUDC algorithms in this chapter.

4.1 A 6-Approximation Algorithm for $h \leq \frac{1}{\sqrt{2}}$

For this algorithm, the points \mathcal{P} and disk centres \mathcal{Q} are confined to a horizontal strip s , where the height h of s is at most $\frac{1}{\sqrt{2}}$. Without loss of generality, assume that the length of s is $\frac{k}{\sqrt{2}}$ for some positive integer k .

We partition the strip s into k disjoint squares of size $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$ by introducing vertical line segments L_0, L_1, \dots, L_k at intervals of $\frac{1}{\sqrt{2}}$. L_0 and L_k form the left and right vertical boundaries of the strip. We denote the set of squares as $\Sigma = \{\sigma_1, \dots, \sigma_k\}$ in left to right order. Let $\mathcal{D}_i^\sigma (\subseteq \mathcal{D})$ be the set of disks centred in the square σ_i , and $\mathcal{P}_i^\sigma (\subseteq \mathcal{P})$ be the set of points in the square σ_i . A *gap* is the union of a maximal sequence of consecutive squares $\sigma_i, \dots, \sigma_j$ such that no disk is centred inside any square of the gap, i.e. $\forall i' \in \{i, \dots, j\}, \mathcal{D}_{i'}^\sigma = \emptyset$. Let G_1, \dots, G_ℓ be the gaps in left to right order in the strip, where G_i is bounded by the vertical line segments L_b and L_e ($0 \leq b < e \leq k$). We now describe the WSDUDC algorithm.

To begin, we cover all points within all gaps G_1, \dots, G_ℓ . To cover points in the gap G_i , we first apply the A-LSDUDC algorithm (Algorithm 3.3) based on the line L_b to cover the points in G_i that are covered by disks centred left of L_b (although the resulting cover may consist of disks centred on either side of L_b , since we use A-LSDUDC). Next, the best known LSDUDC algorithm (Algorithm 2.2) is used on the line L_e to cover any remaining points in G_i . Note that since the latter points are only covered by disks centred right of L_e , the LSDUDC algorithm computes a globally optimal cover of these points. All points in G_i are covered by the above two algorithms.

In the second step of the algorithm, we scan each square in Σ in order from left to right. For any individual square σ_i , if there are points in \mathcal{P}_i^σ not covered by any disk from the solution set found in the first step, then any disk from \mathcal{D}_i^σ may be used to cover all points in \mathcal{P}_i^σ . One such disk is chosen arbitrarily and added to the solution set. The 6-approximate WSDUDC algorithm for \mathcal{P} is described in Algorithm 4.1.

Lemma 4.1. *Algorithm 4.1 never calls A-LSDUDC with respect to two consecutive lines L_{i-1} and L_i for $i \in \{1, 2, \dots, k\}$. Furthermore, the algorithm never calls LSDUDC with respect to two consecutive lines L_{i-1} and L_i for $i \in \{1, 2, \dots, k\}$.*

Proof. The A-LSDUDC algorithm is only ever applied with respect to a line which is the left boundary of a gap (line 24 of Algorithm 4.1). Say without loss of generality one such line is L_{i-1} . By definition, this means that $\mathcal{D}_{i-1}^\sigma \neq \emptyset$ and $\mathcal{D}_i^\sigma = \emptyset$, which together imply that neither L_{i-2} nor L_i can form the left boundary of a gap. Therefore, A-LSDUDC is never applied to two consecutive lines.

Analogously, the LSDUDC algorithm is only ever applied to a line which forms the right boundary of a gap (line 18 of Algorithm 4.1), say without loss of generality one such line is L_{i-1} . It follows that in this case $\mathcal{D}_{i-1}^\sigma = \emptyset$ and $\mathcal{D}_i^\sigma \neq \emptyset$, which together imply that neither L_{i-2} nor L_i can form the right boundary of a gap. Therefore, Algorithm 4.1 never applies LSDUDC to two consecutive lines. \square

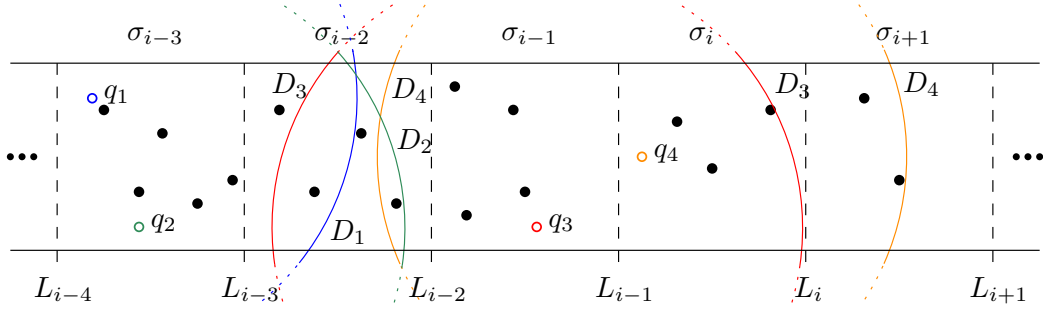
Theorem 4.1. *Algorithm 4.1 produces a 6-factor approximation for the WSDUDC problem in $O(mn + m \log m + n \log n)$ time.*

Proof. To prove that Algorithm 4.1 gives a 6-factor approximation, we consider four cases which describe all of the various ways that disks from \mathcal{D}_i^σ may be used. The first three cases are illustrated in Figure 4.1, and the fourth is trivial.

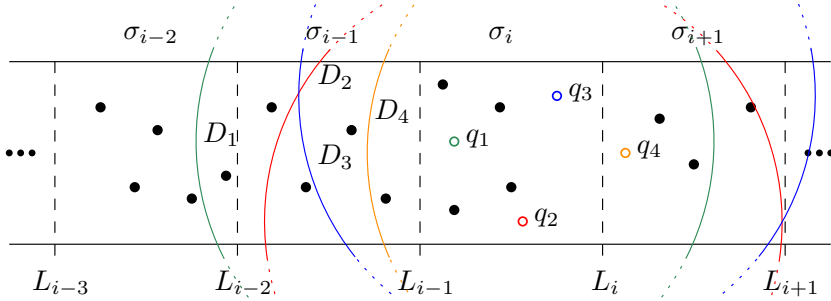
Case 1: Used by A-LSDUDC on line L_j , where $j < i$. Since the distance between the lines L_{i-1} and L_{i-3} is less than 2, a disk $D \in \mathcal{D}_i^\sigma$ may cover points that are covered by disks with centres left of the lines L_{i-1}, L_{i-2} and L_{i-3} . This suggests that the disk D may be used as an assisting disk in solutions to the A-LSDUDC algorithm with respect to the lines L_{i-1}, L_{i-2} and

Algorithm 4.1 WSDUDC-6($\mathcal{P}, \mathcal{D}, s$)

- 1: **Input:** A set \mathcal{P} of points in a horizontal strip s of dimensions $\frac{1}{\sqrt{2}} \times \frac{k}{\sqrt{2}}$, and a set \mathcal{D} of unit disks centred in s .
- 2: **Output:** A set $\mathcal{D}_2^* \subseteq \mathcal{D}$, which covers \mathcal{P} with a factor 6 approximation.
- 3: $\mathcal{D}_{-1}^\sigma \leftarrow \emptyset, \mathcal{D}_0^\sigma \leftarrow \emptyset, \mathcal{P}_{-1}^\sigma \leftarrow \emptyset, \mathcal{P}_0^\sigma \leftarrow \emptyset, \mathcal{P}_{k+1}^\sigma \leftarrow \emptyset, \mathcal{P}_{k+2}^\sigma \leftarrow \emptyset, \mathcal{D}_2^* \leftarrow \emptyset$
- 4: **for** $i := 1$ **to** k **do**
- 5: Compute the sets \mathcal{D}_i^σ and \mathcal{P}_i^σ
- 6: **end for**
- 7: $i \leftarrow 1$
- 8: **while** $i \leq k$ **do**
- 9: // Begin by covering the gaps
- 10: **while** $i \leq k$ **and** $\mathcal{D}_i^\sigma = \emptyset$ **do**
- 11: $i \leftarrow i + 1$ // Find the end of the gap
- 12: **end while**
- 13: **if** $\mathcal{D}_{i-2}^\sigma = \emptyset$ **then**
- 14: $\mathcal{P}_{\text{left}} \leftarrow (\mathcal{P}_{i-2}^\sigma \cup \mathcal{P}_{i-1}^\sigma) \cap (\mathcal{D}_i^\sigma \cup \mathcal{D}_{i+1}^\sigma)$
- 15: **else**
- 16: $\mathcal{P}_{\text{left}} \leftarrow \mathcal{P}_{i-1}^\sigma \cap (\mathcal{D}_i^\sigma \cup \mathcal{D}_{i+1}^\sigma)$
- 17: **end if**
- 18: $\mathcal{D}' \leftarrow \text{LSDUDC}(\mathcal{P}_{\text{left}}, \mathcal{D}_i^\sigma \cup \mathcal{D}_{i+1}^\sigma, L_{i-1})$ (Algorithm 2.2)
- 19: $\mathcal{D}_2^* \leftarrow \mathcal{D}_2^* \cup \mathcal{D}', i \leftarrow i + 1$
- 20: **while** $i \leq k$ **and** $\mathcal{D}_i^\sigma \neq \emptyset$ **do**
- 21: $i \leftarrow i + 1$ // Find the start of the next gap
- 22: **end while**
- 23: $\mathcal{P}_{\text{right}} \leftarrow (\mathcal{P}_i^\sigma \cup \mathcal{P}_{i+1}^\sigma) \cap (\mathcal{D}_{i-2}^\sigma \cup \mathcal{D}_{i-1}^\sigma)$
- 24: $\mathcal{D}' \leftarrow \text{A-LSDUDC}(\mathcal{P}_{\text{right}}, \mathcal{D}, L_{i-1})$ (Algorithm 3.3)
- 25: $\mathcal{D}_2^* \leftarrow \mathcal{D}_2^* \cup \mathcal{D}', i \leftarrow i + 1$
- 26: **end while**
- 27: **for** $j := 1$ **to** k **do**
- 28: // Now cover the remainder
- 29: **if** $\mathcal{P}_j^\sigma \setminus \mathcal{P}_j^\sigma \cap \mathcal{D}_2^* \neq \emptyset$ **then**
- 30: $\mathcal{D}_2^* \leftarrow \mathcal{D}_2^* \cup \{D\}$, where $D \in \mathcal{D}_j^\sigma$
- 31: **end if**
- 32: **end for**
- 33: **return** \mathcal{D}_2^*



(a) Cases 1 and 2. The disk D_4 is in \mathcal{D}_i^σ , and it may be used by A-LSDUDC on L_i , or on L_{i-3} as an assisting disk. σ_{i-2} and σ_{i+1} are gaps in this strip.



(b) Case 3. All disks shown are considered by the LSDUDC algorithm on L_{i-1} . σ_{i-2} and σ_{i-1} form a gap in this strip.

Figure 4.1: Cases for the 6-approximate WSDUDC algorithm.

L_{i-3} , but in fact, A-LSDUDC can only be used on one of these lines. Since $\mathcal{D}_i^\sigma \neq \emptyset$ ($D \in \mathcal{D}_i^\sigma$), Algorithm 4.1 would not call A-LSDUDC with respect to the line L_{i-1} . Lemma 4.1 states that Algorithm 4.1 never calls the A-LSDUDC algorithm with respect to two consecutive lines, and so D may appear in the solution of A-LSDUDC with respect to either L_{i-2} or L_{i-3} .

Case 2: Used by A-LSDUDC on line L_j , where $j \geq i$. Since the distance between the lines L_i and L_{i+1} is less than 1, a disk $D \in \mathcal{D}_i^\sigma$ may cover points in σ_{i+1} and σ_{i+2} . Therefore, the disk D may appear in a solution to the A-LSDUDC algorithm with respect to the lines L_i and L_{i+1} . Again, Lemma 4.1 established that Algorithm 4.1 never calls the A-LSDUDC algorithm with respect to two consecutive lines. Therefore, D may appear in the solution of A-LSDUDC with respect to either L_i or L_{i+1} .

Case 3: Used by LSDUDC. Since the distance between the lines L_{i-1} and L_{i-2} is less than 1, a disk $D \in \mathcal{D}_i^\sigma$ may cover points in σ_{i-1} and σ_{i-2} . Therefore, the disk D may appear in a solution to the LSDUDC algorithm with respect to the lines L_{i-1} and L_{i-2} . However, by Lemma 4.1, Algorithm 4.1 never calls the LSDUDC algorithm with respect to two consecutive lines. Therefore, D may appear in the solution of LSDUDC with respect to either L_{i-1} or L_{i-2} .

Case 4: Used to cover \mathcal{P}_i^σ . A disk $D \in \mathcal{D}_i^\sigma$ may be selected in the solution if some points in \mathcal{P}_i^σ are not covered by any disk in the solutions of the A-LSDUDC and LSDUDC algorithms (line 30 of Algorithm 4.1). Since the size of all squares σ_i ($1 \leq i \leq k$) is $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$, one disk centred anywhere in σ_i is sufficient to cover all the points in \mathcal{P}_i^σ .

There are two basic properties that are fundamental to our analysis. First, a disk can cover points in at most four squares. Second, Case 4 is only ever used for a particular square if no disk

was used for Cases 1-3 (since any disk centred in σ_i covers all points in \mathcal{P}_i^σ).

Suppose that there are four squares $\sigma_{i-2}, \dots, \sigma_{i+1}$, where all points $\mathcal{P}_{i-2}^\sigma \cup \mathcal{P}_{i-1}^\sigma \cup \mathcal{P}_i^\sigma \cup \mathcal{P}_{i+1}^\sigma$ may be covered by a single disk from \mathcal{D}_i^σ . We may choose poorly so that Case 4 is applied to each square, and so a disk is chosen from each of $\mathcal{D}_{i-2}^\sigma, \mathcal{D}_{i-1}^\sigma, \mathcal{D}_i^\sigma$, and \mathcal{D}_{i+1}^σ . In this example, we have used four disks to cover this set of points rather than one, so the approximation factor of Algorithm 4.1 is at least 4.

By substituting some of these squares with gaps, we can force disks from \mathcal{D}_i^σ to fall into Cases 1-3 to increase the approximation factor. Specifically, suppose there exists a gap G_j which consists of a single square lying to the left of σ_i , and a gap G_{j+1} which begins to the right of σ_i so that the left boundary lines of G_j and G_{j+1} may be intersected by disks from \mathcal{D}_i^σ . In this setting, disks from \mathcal{D}_i^σ could be used in Cases 1 and 3 to cover the points of G_j , and also by Case 2 to cover the points of G_{j+1} . These applications lead to at least a 5-approximation, since Cases 1 and 2 lead to 2-approximations and Case 3 is an optimal algorithm.

Disks from \mathcal{D}_i^σ may also cover points in two other squares outside of these gaps, say $\sigma_{i'}$ and $\sigma_{i''}$, $i' < i < i''$. If these squares are adjacent to σ_i , then if applicable, their disks are used in the same computations of A-LSDUDC and LSDUDC as the disks of \mathcal{D}_i^σ , so this would have no effect on the approximation factor. However, it is possible that $\sigma_{i'}$ and $\sigma_{i''}$ fall into Case 4, and that a set of disks of equal cardinality for the solutions to Cases 1-3 would also cover all the points in $\sigma_{i'}$ and $\sigma_{i''}$. Therefore, the overall approximation factor is five times the size of optimal set of disks, plus two. Note that since disks from \mathcal{D}_i^σ are covering points from five squares in this setting (specifically $\sigma_{i-2}, \dots, \sigma_{i+2}$), the optimal solution requires at least two disks to cover the same set of points. Let $|\text{OPT}|$ be the size of the optimal solution, then note that $5|\text{OPT}| + 2 \leq 6|\text{OPT}|$ when $|\text{OPT}| \geq 2$, so we have our bound. If all disks from \mathcal{D}_i^σ cover points from at most four squares, then the approximation factor of 6 is immediate. Thus, the approximation factor of the lemma follows.

The time complexity of the algorithm follows from the fact that each disk in \mathcal{D} can participate a constant number of times in the A-LSDUDC and LSDUDC algorithms and the running time of both the A-LSDUDC and LSDUDC algorithms may be bounded by $O(mn + m \log m + n \log n)$ (see Section 2.2.1). \square

4.2 A General $3\lceil 1/\sqrt{1-h^2} \rceil$ -Approximation Algorithm for $h < 1$

In this section, we present an algorithm for approximating the optimal WSDUDC solution on strips of a general height $h < 1$. The algorithms of Sections 4.3 and 4.4 extend these ideas with refinements which achieve better approximation factors in narrower strips.

Theorem 4.2. *Given a strip of height $h < 1$, we may find a $3\lceil 1/\sqrt{1-h^2} \rceil$ -approximation to the WSDUDC problem in $O(m^4n + n \log n)$ time.*

We define the set of rectangles \mathcal{R}° , where $R_i^\circ \in \mathcal{R}^\circ$ is the largest rectangle of height $2h$ which may be covered by $D_i \in \mathcal{D}$, and the strip s has height h and is assumed to be horizontal. Furthermore, we use a set of rectangles \mathcal{R} of height h , defined as $R_i = R_i^\circ \cap s$, $R_i \in \mathcal{R}$, $R_i^\circ \in \mathcal{R}^\circ$. In other words, a rectangle in \mathcal{R} is defined as the portion of a rectangle in \mathcal{R}° which lies inside the strip. In Figure 4.2, the shaded rectangles are those in \mathcal{R} .

Observation 4.1. *Suppose we are given a strip of height $h < 1$ and a unit disk D whose centre lies in the strip. R° is defined as the rectangle of height $2h$ and width $k = 2\sqrt{1-h^2}$ which is circumscribed by D . If a point q is covered by R° , then D also covers q . Furthermore, R° covers the entire height of the strip.*

We divide the set of points \mathcal{P} into two sets $\mathcal{P} = \mathcal{P}_{\mathcal{R}} \cup \mathcal{P}_{\overline{\mathcal{R}}}$, where $\mathcal{P}_{\mathcal{R}}$ is the set of points covered by the set of rectangles \mathcal{R} , and $\mathcal{P}_{\overline{\mathcal{R}}} = \mathcal{P} \setminus \mathcal{P}_{\mathcal{R}}$, i.e. those points covered by \mathcal{D} but not \mathcal{R} . The approximation algorithm proceeds in two stages to compute the cover: first the points in $\mathcal{P}_{\overline{\mathcal{R}}}$ are covered, and then the remaining uncovered points in $\mathcal{P}_{\mathcal{R}}$ are covered. We refer to the points in $\mathcal{P}_{\overline{\mathcal{R}}}$ as occurring in the *gaps* of the strip, and the points in $\mathcal{P}_{\mathcal{R}}$ are in the *intervals* (see Figure 4.2). In our discussion, we assume that $h > 0$, so that the width of the rectangles is $k = 2\sqrt{1-h^2} < 2$.²

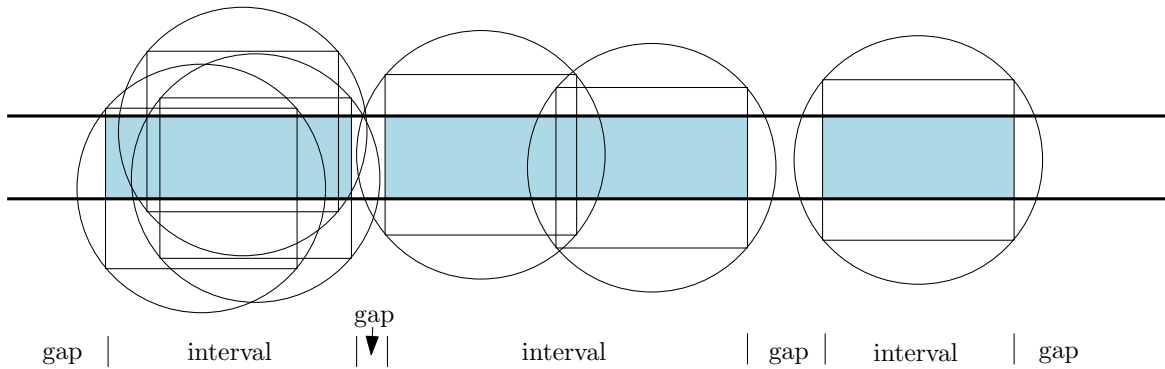


Figure 4.2: Intervals are continuous segments of the strip covered by the rectangles in \mathcal{R} , and gaps are the segments of the strip outside of the intervals.

4.2.1 Covering $\mathcal{P}_{\overline{\mathcal{R}}}$

The set $\mathcal{P}_{\overline{\mathcal{R}}}$ may be viewed as existing in the gaps lying between the rectangles in the set \mathcal{R} . The centres of all disks are separated from the points in $\mathcal{P}_{\overline{\mathcal{R}}}$ by vertical lines (those of the gap boundaries). For each gap of the strip, the points are covered optimally with the $O(m^4n + n \log n)$ time algorithm for SSDUDC (Algorithm 2.4). While points in each gap are covered optimally, we may lose optimality when we combine these solutions³. Recall that rectangles have width $k = 2\sqrt{1-h^2}$. There is a rectangle corresponding to each disk, and so no disk centre lies within a distance of $k/2$ from any gap. By interleaving rectangles of width k with gaps of some width ε , a disk may cover points in $2\lceil 1/k - 1/2 \rceil$ gaps as ε tends to 0. To see this, consider the right side of a disk D_i , where R_i defines an interval of width $k/2$ on this right side. Since D_i has unit radius, $\lceil (1 - k/2)/k \rceil$ additional rectangles (and gaps, one to the left of each rectangle) may be at least partially covered by D_i to the right. For example, given $2/5 < k \leq 2/3$, disks may cover points in four gaps, and for $2/3 < k < 2$, disks may cover points in two. Thus, the union of the solutions for each gap has an approximation factor of $2\lceil 1/k - 1/2 \rceil$ for covering $\mathcal{P}_{\overline{\mathcal{R}}}$.

²If $h = 0$, all points and disk centres are collinear, and $\mathcal{P}_{\overline{\mathcal{R}}}$ is empty. This setting is solved optimally by the GREEDY-RECTANGLES algorithm detailed in Section 4.2.2.

³Covering the points in the union of the gaps cannot be solved optimally in general. The hardness proof for WSDUDC (Section 4.5) only has points in gaps.

4.2.2 Covering $\mathcal{P}_{\mathcal{R}}$

To cover the points which remain after the previous step, we iteratively add the right-most rectangle that covers the left-most remaining point to the solution, as detailed in Algorithm 4.2 (GREEDY-RECTANGLES).

Algorithm 4.2 GREEDY-RECTANGLES($\mathcal{R}, \mathcal{P}_{\mathcal{R}}$)

```

1: Input: A set of points  $\mathcal{P}_{\mathcal{R}}$  and a set of rectangles  $\mathcal{R}$ .
2: Output:  $\mathcal{R}'$ , a set of rectangles covering  $\mathcal{P}_{\mathcal{R}}$ .
3:  $\mathcal{R}' \leftarrow \emptyset$ 
4: Sort  $\mathcal{P}_{\mathcal{R}}$  by x-coordinate, sort  $\mathcal{R}$  by left boundary
5: while  $\mathcal{P}_{\mathcal{R}} \neq \emptyset$  do
6:    $p_{\ell} \leftarrow$  left-most point in  $\mathcal{P}_{\mathcal{R}}$ 
7:    $R_r \leftarrow$  right-most rectangle in  $\mathcal{R}$  covering  $p_{\ell}$ 
8:    $\mathcal{R}' = \mathcal{R}' \cup R_r$ 
9:    $\mathcal{P}_{\mathcal{R}} = \mathcal{P}_{\mathcal{R}} \setminus (R_r \cap \mathcal{P}_{\mathcal{R}})$ 
10: end while
11: return  $\mathcal{R}'$ 

```

Lemma 4.2. *A rectangle R'_i selected by GREEDY-RECTANGLES may overlap another rectangle R'_{i-1} (the previous rectangle chosen) by $k - \varepsilon$, for any $\varepsilon > 0$.*

Proof. The left rectangle R'_{i-1} covers a point p_{ℓ} which is to the left of the right rectangle R'_i , which in turn covers a point p_r to the right of R'_{i-1} . By pushing p_{ℓ} to the left edge of R'_{i-1} and p_r to the right edge of R'_i , the rectangles may be overlapped so that R'_{i-1} lies ε to the left of R'_i , and both are chosen. \square

Lemma 4.3. *Let $\mathcal{R}' = \{R'_1, \dots, R'_{|\mathcal{R}'|}\}$ be the set of rectangles found by GREEDY-RECTANGLES, indexed from left to right so that $\forall i, j, i < j \leftrightarrow \text{left}(R'_i, R'_j)$ where $\text{left}(R'_i, R'_j)$ indicates that R'_i is left of R'_j . Then no two non-consecutive rectangles intersect, i.e. $\forall i, j, j > i + 1 \rightarrow R'_i \cap R'_j = \emptyset$.*

Proof. Suppose that this is not the case and R'_i and R'_j intersect although there exist rectangles $R'_{i+1}, \dots, R'_{j-1} \in \mathcal{R}'$, where $i + 1 \leq j - 1$. R'_j is to the right of each of these intermediate rectangles, and R'_j covers all points to the right of R'_i which these rectangles cover. Let \mathcal{P}' be the set of points covered by the rectangles $R'_{i+1}, \dots, R'_{j-1}$. \mathcal{P}' is covered entirely by $R'_i \cup R'_j$, and none of $R'_{i+1}, \dots, R'_{j-1}$ are the right-most rectangle covering a point to the right of R'_i , so they would not be chosen by GREEDY-RECTANGLES. \square

Lemma 4.4. *GREEDY-RECTANGLES computes a cover of $\mathcal{P}_{\mathcal{R}}$ with an approximation factor of $3\lceil 1/k - 1/2 \rceil$ times the optimal solution.*

Proof. Consider the maximum number of rectangles in the GREEDY-RECTANGLES solution that may be replaced by a single disk D_i in the strip. One of the rectangles available to the algorithm is $R_i \subset R_i^{\circ}$, where R_i° is circumscribed by D_i . By Lemma 4.2, there may be another rectangle ε to the left or right of R_i which will be selected by the algorithm, and so the approximation factor is at least 2. It may be possible to pack additional pairs of nearly overlapping rectangles as densely as permitted by Lemma 4.3 so that the points covered by these rectangles are also covered by D_i .

Since all disks have unit radius and R_i° is circumscribed, each side of D_i can potentially cover all points covered by at most $2\lceil(1 - k/2)/k\rceil - 1$ additional rectangles (see Figure 4.3). This analysis is similar to Section 4.2.1, but now all rectangles are paired except for the right-most one (in a right-most pair, the region covered only by the right rectangle cannot be covered at all by D_i since we consider the *pairs* to have width k , i.e. $\varepsilon = 0$). Thus, the total approximation factor is $4\lceil 1/k - 1/2\rceil$. \square

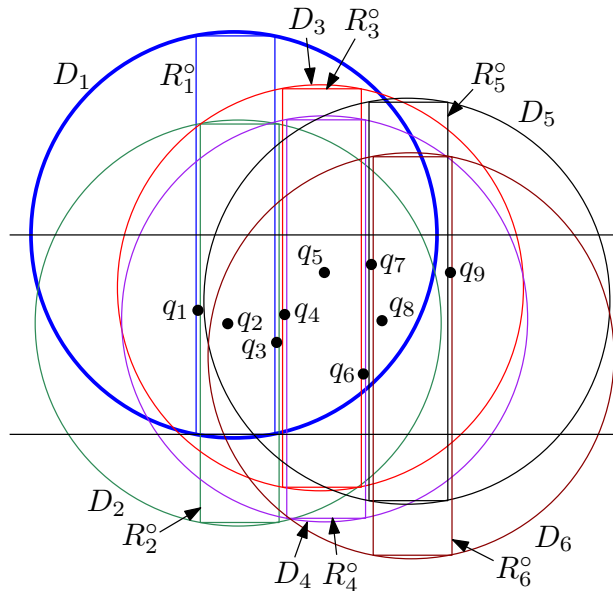


Figure 4.3: An illustration of Lemma 4.4. We wish to know the number of rectangles in which a disk, say D_1 , may possibly cover points. R_1° may be paired with another rectangle as described in Lemma 4.2, we illustrate this with R_2° . We now wish to determine how many distinct rectangles may be covered to the right of this pair (the case for the left side is analogous). In this case, we have $h = 0.97$, and so $2\lceil(1 - k/2)/k\rceil - 1 = 3$. We see this with $R_3^\circ, R_4^\circ, R_5^\circ$, as D_1 covers all points in these rectangles. It is possible that D_1 covers points contained in a rectangle paired with the right-most rectangle in this set (e.g., $q_8 \in D_1 \cap R_6^\circ$), but such points are covered by R_5° as well.

GREEDY-RECTANGLES requires both the set of rectangles \mathcal{R} and the set of points $\mathcal{P}_{\mathcal{R}}$ to be sorted in left to right order. The sorted lists are each walked through a single time, so the total running time is $O(m \log m + n \log n)$.

4.2.3 Combining Solutions for $\mathcal{P}_{\overline{\mathcal{R}}}$ and $\mathcal{P}_{\mathcal{R}}$

Recall that the approximation factor for covering the entire set of $\mathcal{P}_{\overline{\mathcal{R}}}$ is $2\lceil 1/k - 1/2\rceil$ and $4\lceil 1/k - 1/2\rceil$ for covering $\mathcal{P}_{\mathcal{R}}$, where k is the width of the rectangles. We simply sum these factors to get an overall approximation factor of $6\lceil 1/k - 1/2\rceil < 3\lceil 1/\sqrt{1 - h^2}\rceil$ for strips of arbitrary height $h < 1$. The running time is $O(m^4n + n \log n)$, dominated by the SSDUDC algorithm used to cover $\mathcal{P}_{\overline{\mathcal{R}}}$. The WSDUDC algorithm is summarized in Algorithm 4.3.

Algorithm 4.3 WSDUDC-GEN($\mathcal{P}, \mathcal{D}, s$)

- 1: **Input:** A set \mathcal{P} of points in a horizontal strip s of height $h < 1$, and a set \mathcal{D} of unit disks centred in s .
 - 2: **Output:** A set $\mathcal{D}^* \subseteq \mathcal{D}$, which covers \mathcal{P} with a factor $3\lceil 1/\sqrt{1-h^2} \rceil$ approximation.
 - 3: $\mathcal{D}^* \leftarrow \emptyset$
 - 4: $\mathcal{R}^\circ \leftarrow$ set of rectangles of height $2h$ circumscribed by \mathcal{D}
 - 5: $\mathcal{R} \leftarrow \mathcal{R}^\circ \cap s$
 - 6: Sort \mathcal{R} left to right by left boundary of each rectangle
 - 7: $\mathcal{G} \leftarrow$ set of gaps in $\mathcal{R} \cap s$
 - 8: **for** $i := 1$ **to** $|\mathcal{G}|$ **do**
 - 9: $\mathcal{P}_i = G_i \cap \mathcal{P}$ // Find the set of points in the current gap
 - 10: $\mathcal{D}' \leftarrow$ SSDUDC($\mathcal{P}_i, \mathcal{D}, G_i$) (Algorithm 2.4)
 - 11: $\mathcal{D}^* \leftarrow \mathcal{D}^* \cup \mathcal{D}'$
 - 12: $\mathcal{P} \leftarrow \mathcal{P} \setminus (\mathcal{P} \cap \mathcal{D}')$
 - 13: **end for**
 - 14: $\mathcal{R}' \leftarrow$ GREEDY-RECTANGLES($\mathcal{R}, \mathcal{P}_{\mathcal{R}}$) (Algorithm 4.2)
 - 15: $\mathcal{D}^* \leftarrow \mathcal{D}^* \cup \mathcal{D}'$ // \mathcal{D}' is the set of disks corresponding to \mathcal{R}'
 - 16: **return** \mathcal{D}^*
-

4.3 4-Approximation Algorithm for $h \leq 2\sqrt{2}/3$

The general algorithm for covering $\mathcal{P}_{\mathcal{R}}$ presented in the previous section has an approximation factor of 4 when $k \geq 2/3$ ($h \leq 2\sqrt{2}/3$). However, for each pair of consecutive rectangles R'_{i-1} and R'_i found by GREEDY-RECTANGLES, we can determine whether one or two disks are required to cover the points covered by the pair of rectangles. In other words, we determine whether there exists a disk D_j such that $(R'_{i-1} \cup R'_i) \cap \mathcal{P} \subseteq D_j \cap \mathcal{P}$. To do so, we run through \mathcal{R}' in order, and check whether the current pair may be replaced by any disk in \mathcal{D} . This way, we are assured that exactly two disks are required to cover the points contained in any consecutive pair of rectangles in our solution for $\mathcal{P}_{\mathcal{R}}$.

Theorem 4.3. *If $h \leq 2\sqrt{2}/3$, we can improve the approximation factor for WSDUDC to 4 in $O(m^4n + n \log n)$ time.*

Consider a disk $D_i \in \mathcal{D}^*$, which may or may not be a member of our refined solution set. D_i may intersect at most four rectangles in \mathcal{R}' . Every consecutive pair of rectangles in \mathcal{R}' now requires at least two disks, so at least two disks are required to cover any four consecutive rectangles. Therefore, the overall approximation factor is two. This operation will scan m disks for every possible disk to remove from the solution, so the operation takes $O(m^2n)$ time⁴. We outline the improvements in Algorithm 4.4.

We now have a 2-approximate algorithm for $\mathcal{P}_{\mathcal{R}}$ when $k \geq 2/3$, and we may solve each gap of $\mathcal{P}_{\overline{\mathcal{R}}}$ optimally. We show that we may combine these solutions to achieve an overall 4-approximate algorithm for WSDUDC, when the strip is of height less than $2\sqrt{2}/3$. For the purposes of counting, we may assume that the disks forming the cover for each gap are equally distributed amongst the neighbouring intervals for both the approximate solution and the optimal one. We

⁴It would not be difficult to improve on the running time of this operation, but this is not a bottleneck on the overall running time.

Algorithm 4.4 WSDUDC-4($\mathcal{P}, \mathcal{D}, s$)

```
1: Input: A set  $\mathcal{P}$  of points in a horizontal strip  $s$  of height  $h \leq 2\sqrt{2}/3$ , and a set  $\mathcal{D}$  of unit disks centred in  $s$ .
2: Output: A set  $\mathcal{D}^* \subseteq \mathcal{D}$ , which covers  $\mathcal{P}$  with a factor 4 approximation.
3: Lines 3-14 are identical to Algorithm 4.3
4:  $R_p \leftarrow R_1 \in \mathcal{R}'$  // Assume the rectangles are stored left to right in  $\mathcal{R}'$ 
5:  $\mathcal{D}' \leftarrow \emptyset$ 
6: for  $i := 2$  to  $|\mathcal{R}'|$  do
7:    $R_c \leftarrow R_i \in \mathcal{R}'$ 
8:   for  $j := 1$  to  $|\mathcal{D}|$  do
9:     if  $(R_p \cup R_c) \cap \mathcal{P} \subseteq D_j \cap \mathcal{P}$  then
10:       $R_p, R_c \leftarrow D_j$  // A bit abusive, but to simplify the algorithm description we allow  $R_p$ 
      and  $R_c$  to be a rectangle or a disk
11:       $j \leftarrow |\mathcal{D}| + 1$ 
12:     end if
13:   end for
14:   if  $D_p \notin \mathcal{D}'$  then
15:      $\mathcal{D}' \leftarrow \mathcal{D}' \cup \{D_p\}$  //  $D_p$  is the disk corresponding to  $R_p$ 
16:   end if
17:    $R_p \leftarrow R_c$ 
18: end for
19: return  $\mathcal{D}^* \cup \mathcal{D}'$ 
```

are not interested in the worst-case approximation factor in any given interval; rather we are interested in the approximation factor over the strip as a whole. For each gap, only disks found in adjacent intervals may form part of the solution. Disk centres are located at least a distance $1/3$ from the end of an interval, and so disk centres in non-adjacent intervals are more than unit distance away from the gap. Therefore, for each interval of the strip, assume that n_ℓ disks are used for covering the gap to the left, n_s disks are used for covering the points in the interval, and n_r disks are used to cover the points in the gap to the right. The minimum number of disks required is $\max\{n_\ell, n_s/2, n_r\}$, since both n_ℓ and n_r are optimal and n_s is a 2-approximation. We conclude that $n_\ell + n_s + n_r \leq 4 \cdot \max\{n_\ell, n_s/2, n_r\}$, and thus it is a 4-approximation algorithm. The worst-case running time is $O(m^4 n + n \log n)$, dominated by the running time of the SSDUDC algorithm.

4.4 3-Approximation Algorithm for $h \leq 4/5$

We now describe a 3-approximate algorithm for narrower strips, specifically when $k \geq 6/5$ ($h \leq 4/5$). In this case⁵, the $\mathcal{P}_{\mathcal{R}}$ sub-problem may be solved optimally using dynamic programming. We define a set of disks D_s as *mutually spanning* if each disk in D_s covers a non-empty set of points which lies to the left of all other disks in D_s , as well as a non-empty set of points lying to the right of all other disks in D_s . We show that no optimal solution to the cover of $\mathcal{Q}_{\mathcal{R}}$ requires a mutually spanning set of more than three disks.

⁵A similar dynamic programming algorithm applies to larger strips, but the running time increases rapidly with h . The running time of the dynamic program is $O(m^{2c}n)$, where c is the cardinality of the largest minimal mutually spanning sets in an optimal solution. As h approaches unit distance, c tends to infinity.

Lemma 4.5. *If $h \leq 4/5$, an optimal solution to $\mathcal{P}_{\mathcal{R}}$ requires mutually spanning sets of cardinality at most three.*

Proof. Suppose that there exists a mutually spanning set of four disks in the optimal solution. Recall that each point in the set $\mathcal{P}_{\mathcal{R}}$ is covered by some rectangle circumscribed by a disk. Using Lemma 4.3, we show that a set of four rectangles exists which covers all of the points covered by the four disks. At least one rectangle is required to cover the left-most point, this may be followed by a pair of rectangles, whose combined width is greater than k , and then there may be a final rectangle spanning an additional width k . The maximum width in a strip for a mutually spanning set of disks of any cardinality is $3 - k/2$; note that $2k > 3 - k/2$ when $k > 6/5$. Any solution which uses four mutually spanning disks to cover a set of points \mathcal{P}' need only use (at most) four rectangles to cover a set of points \mathcal{P}'' , where $\mathcal{P}' \subseteq \mathcal{P}''$. Therefore, four mutually spanning disks are never needed in an optimal solution. \square

By Lemma 4.5, a dynamic program which adds disks to the solution in a left-to-right fashion need only consider up to triples of disks to terminate sub-problems to ensure that the sub-problems are independent and optimal. Such a dynamic program is described in Algorithm 4.5. In the algorithm, \mathcal{D}^2 and \mathcal{D}^3 are the sets of mutually spanning doubles and triples of disks respectively, and \mathfrak{D} is the set of all sets of disks under consideration. Given two sets $\mathcal{D}_i, \mathcal{D}_j \in \mathfrak{D}$, if \mathcal{D}_i covers points left of \mathcal{D}_j , and \mathcal{D}_j does not cover points left of \mathcal{D}_i , we write $\mathcal{D}_i <_c \mathcal{D}_j$ to indicate this relationship. Otherwise, we consider them incomparable under this operator. Hence, we may establish a partially ordered set over all of the sets in \mathfrak{D} with respect to the $<_c$ operator. Note that directed cycles are impossible in this set, since the transitive property holds for the $<_c$ operator. We impose a topological sorting $\mathfrak{D} = \{\mathcal{D}_1, \dots, \mathcal{D}_{|\mathfrak{D}|}\}$ so that for any two sets $\mathcal{D}_i, \mathcal{D}_j$ in this ordering, we have that $i < j \rightarrow \mathcal{D}_j \not<_c \mathcal{D}_i$.

The correctness of $\text{OPTIMAL-}\mathcal{P}_{\mathcal{R}}$ follows from the fact that all points left of a set \mathcal{D}_i are covered in a valid solution to a subproblem terminating with \mathcal{D}_i , and all mutually spanning sets up to size three are considered. $\text{OPTIMAL-}\mathcal{P}_{\mathcal{R}}$ runs in $O(m^6 n)$ time: there are $O(m^3)$ possible combinations of disks that we consider in two nested for loops, and inside the nested loop we check the disks against the point set \mathcal{P} .

We have optimal algorithms for computing the cover of each gap of $\mathcal{P}_{\overline{\mathcal{R}}}$ and each interval of $\mathcal{P}_{\mathcal{R}}$. Further, the disks covering a gap only come from the two adjacent intervals, and the disks covering an interval only come from the interval itself. Since the disks in each interval can contribute to only three problems, each of which is solved optimally, the worst-case is that three times the optimal number of disks is used. The running time of the algorithm (Algorithm 4.6) is dominated by $\text{OPTIMAL-}\mathcal{P}_{\mathcal{R}}$, so the overall running time is $O(m^6 n + n \log n)$.

Theorem 4.4. *Given a strip of height $h \leq 4/5$, a 3-approximate solution may be found to WSDUDC in $O(m^6 n + n \log n)$ time.*

4.5 NP-Completeness of WSDUDC

We prove that WSDUDC is NP-complete by reducing from the minimum vertex cover (VERTEX-COVER) problem on planar graphs of maximum degree three, which is known to be NP-complete (and APX-hard) [80, 5]. In the VERTEX-COVER problem, we are given a graph $G = (V, E)$ (in

Algorithm 4.5 OPTIMAL- $\mathcal{P}_{\mathcal{R}}$ ($\mathcal{D}, \mathcal{P}_{\mathcal{R}}$)

- 1: **Input:** A set $\mathcal{P}_{\mathcal{R}}$ of points in the intervals of a horizontal strip s of height $h \leq 4/5$, and a set \mathcal{D} of unit disks centred in s .
 - 2: **Output:** A set $\mathcal{D}^* \subseteq \mathcal{D}$ of minimum cardinality which covers $\mathcal{P}_{\mathcal{R}}$.
 - 3: $\mathfrak{D} \leftarrow \mathcal{D} \cup \mathcal{D}^2 \cup \mathcal{D}^3$, $m' \leftarrow |\mathfrak{D}|$
 - 4: Topologically sort \mathfrak{D} on the $<_c$ operator
 - 5: $c[0] = 0, c[1 \dots m'] = \infty$
 - 6: **for** $i = 1 \dots m'$ **do**
 - 7: **for** $j = 0 \dots i - 1$ **do**
 - 8: $\text{size} \leftarrow c[j] + |\mathcal{D}_i|$
 - 9: **if** $\text{size} < c[i]$ **and** no points lie between \mathcal{D}_i and \mathcal{D}_j **then**
 - 10: $c[i] \leftarrow \text{size}$
 - 11: **end if**
 - 12: **end for**
 - 13: **end for**
 - 14: Backtrack on c to recover optimal cover \mathcal{D}^*
 - 15: **return** \mathcal{D}^*
-

Algorithm 4.6 WSDUDC-3($\mathcal{P}, \mathcal{D}, s$)

- 1: **Input:** A set \mathcal{P} of points in a horizontal strip s of height $h \leq 4/5$, and a set \mathcal{D} of unit disks centred in s .
 - 2: **Output:** A set $\mathcal{D}^* \subseteq \mathcal{D}$, which covers \mathcal{P} with a factor 3 approximation.
 - 3: Lines 3-13 are identical to Algorithm 4.3
 - 4: $\mathcal{D}' \leftarrow \text{OPTIMAL-}\mathcal{P}_{\mathcal{R}}(\mathcal{D}, \mathcal{P}_{\mathcal{R}})$ (Algorithm 4.5)
 - 5: **return** $\mathcal{D}^* \cup \mathcal{D}'$
-

particular, a planar graph with no vertex of degree greater than three), and we seek a minimum cardinality subset $V^* \subseteq V$ such that for all $e_{(i,j)} = (v_i, v_j) \in E$, either $v_i \in V^*$ or $v_j \in V^*$. In other words, the vertex cover is a minimum cardinality hitting set of all of the edges in the graph. An instance of the decision version of WSDUDC is $\{\mathcal{P}, \mathcal{D}, c\}$ (c is an integer), and we ask whether there exists a solution \mathcal{D}^* such that $\mathcal{P} \cap (\cup \mathcal{D}^*) = \mathcal{P}$ and $|\mathcal{D}^*| < c$.

Theorem 4.5. *WSDUDC is NP-complete on a strip of height h , for any $h > 0$.*

WSDUDC is in NP, since a certificate may be provided as a set of disks that covers all of the points in \mathcal{P} , which is trivial to verify.

In the reduction, we create an instance of WSDUDC from a planar graph so that a solution \mathcal{D}^* to the WSDUDC problem provides a solution V^* to the VERTEX-COVER problem on the graph. For our reduction, it is easier to consider the dual (disk piercing) setting of WSDUDC, but of course once the construction is complete it may be converted back again from the dual so that the problem is in the primal (disk covering) setting. The Within-Strip Discrete Unit Disk Piercing (WSDUDP) problem takes as input a set of points \mathcal{Q} , and a set of unit disks $\mathcal{D}_{\mathcal{P}}$ with centre points \mathcal{P} , where both \mathcal{P} and \mathcal{Q} are contained in a strip of height h . An optimal solution consists of a minimal number of points $\mathcal{Q}^* \subseteq \mathcal{Q}$ such that each disk in $\mathcal{D}_{\mathcal{P}}$ contains at least one point from \mathcal{Q}^* . Let $\text{WS}(G)$ be the WSDUDP instance created from a graph G . Note that a solution \mathcal{Q}^* for WSDUDP is exactly the set of centre points to \mathcal{D}^* , the optimal solution to the WSDUDC problem in the primal setting (see also Lemma 2.1).

Assume that we have a planar embedding of the graph and a horizontal strip so that the terms *left*, *right*, *above* and *below* are all well defined. Let ℓ_{vert}^v be the vertical line through vertex v . For the reduction, we make use of *dummy vertices*, which are simply extra vertices that we may place on an edge of the graph G . A *dummy edge* is an edge which is incident upon at least one dummy vertex. Informally, the steps of the reduction are (the details follow):

1. Obtain a straight line planar embedding of G (e.g. [75]) where each vertex has a distinct x-coordinate.
2. For any vertex v with degree three where all incident edges are left or right of ℓ_{vert}^v , ‘bend’ the lowest edge with a dummy vertex so that the edge becomes incident to v from the opposite side of ℓ_{vert}^v , call this new graph $G' = (V', E')$. We will continue to build on this graph through the reduction.
3. For each vertex $v \in V'$, add a dummy vertex to V' at each point where $\ell_{\text{vert}}^v \cap e \neq \emptyset, \forall e \in E'$.
4. Scan the vertices to identify each vertex v of degree one or two where all edges are incident on the same side of ℓ_{vert}^v , suppose without loss of generality that the edges are incident from the right. Place a vertical line ℓ_{vert} between v and the next vertex to the left in the embedding, and add a dummy vertex at each point where $\ell_{\text{vert}} \cap e \neq \emptyset, \forall e \in E'$. This ensures that consecutive vertical arrays of vertices differ in cardinality by at most one.
5. For any pair of vertices $v_i, v_j \in V$ where there is an edge $e = (v_i, v_j) \in E$, ensure that an even number of vertices have been added onto the edge e in G' by adding an additional dummy vertex if necessary.
6. Create the WSDUDP instance $WS(G)$ from G' so that every edge in E' corresponds to a disk in \mathcal{D} and every vertex in V' corresponds to a point in \mathcal{Q} . We then show that an optimal solution to WSDUDP provides an optimal cover for G' , from which an optimal vertex cover for G may be found, as required.

Lemma 4.6. *Given an edge $e_{(i,j)}$ of the planar graph $G = (V, E)$, one can arbitrarily add a pair of adjacent dummy vertices $V_d = \{v_{i_1}, v_{i_2}\}$ along the edge $e_{(i,j)}$, producing two new edges in the augmented graph $G' = (V \cup V_d, E \cup \{e_{(i,i_1)}, e_{(i_1,i_2)}, e_{(i_2,j)}\} \setminus \{e_{(i,j)}\})$. The graph G' remains planar, and the cardinality of the optimal solution to VERTEX-COVER over G' is $|V^*| + 1$, where V^* is the set of vertices in a minimum vertex cover of G . Given this property, for any optimal solution $V_{G'}^*$ to VERTEX-COVER on G' , we can find an optimal solution V_G^* to VERTEX-COVER on G in polynomial time.*

Proof. By placing the dummy vertices directly on the edge $e_{(i,j)}$ in any embedding, planarity is preserved. There are two possibilities for a minimum vertex cover V^* over an edge $e_{(i,j)}$ in $G = (V, E)$; either one or both of its vertices are in V^* . The addition of a pair of dummy vertices $\{v_{i_1}, v_{i_2}\}$ on $e_{(i,j)}$ creates three edges in place of $e_{(i,j)}$: $\{e_{(i,i_1)}, e_{(i_1,i_2)}, e_{(i_2,j)}\}$, with $e_{(i,i_1)}$ and $e_{(i_1,i_2)}$ sharing the dummy vertex v_{i_1} , and $e_{(i_1,i_2)}$ and $e_{(i_2,j)}$ sharing the other dummy vertex v_{i_2} .

In the first case, $e_{(i,j)}$ is covered by one vertex from V^* , say v_i . $e_{(i,i_1)}$ is covered by V^* , but $e_{(i_1,i_2)}$ and $e_{(i_2,j)}$ are not. The only optimal cover is to add the shared dummy vertex v_{i_2} to the solution. Therefore, the size of the optimal solution increases by one. In the second case, $e_{(i,j)}$ was covered by both v_i and v_j in V^* , and so $e_{(i,i_1)}$ and $e_{(i_2,j)}$ are covered. Therefore, arbitrarily selecting one of the dummy nodes to cover $e_{(i_1,i_2)}$, say v_{i_1} , increases the size of the optimal solution

by one. Although $e_{(i,i_1)}$ is covered by two vertices, the topology of the graph and cover local to v_i remains unchanged.

Suppose there is only one pair of dummy nodes in G' . At least one of the dummy vertices from the pair $\{v_{i_1}, v_{i_2}\}$ must be in the optimal solution $V_{G'}^*$, since they are the endpoints of the edge $e_{(i_1, i_2)}$. If only one is in the optimal solution, say v_{i_1} , we can discard it and return $V_G^* = V_{G'}^* \setminus \{v_{i_1}\}$ as an optimal solution to VERTEX-COVER on G .

If both v_{i_1} and v_{i_2} are in $V_{G'}^*$, we must discard both and add either v_i or v_j to the solution for V_G^* . If v_i or v_j is also in $V_{G'}^*$, then the adjacent dummy vertex is extraneous and $V_{G'}^*$ is a suboptimal cover. Supposing otherwise, without loss of generality, say v_i, v_{i_1} and v_{i_2} are in $V_{G'}^*$ and $V_{G'}^*$ is an optimal cover; this is a contradiction since $V_{G'}^* \setminus \{v_{i_1}\}$ provides a vertex cover of smaller cardinality. Therefore neither v_i nor v_j is in $V_{G'}^*$, and we may arbitrarily set $V_G^* = (V_{G'}^* \setminus \{v_{i_1}\}) \cup \{v_i\}$. This provides another vertex cover for G' which is of equal cardinality, and thus is optimal. Now we are in the first case again.

If there are more pairs of dummy vertices, this argument may be applied iteratively to adjacent pairs in G' until each pair of dummy vertices has only one vertex in $V_{G'}^*$. If there is exactly one dummy vertex from each pair in the optimal vertex cover, these may be removed to provide an optimal vertex cover V_G^* for G . \square

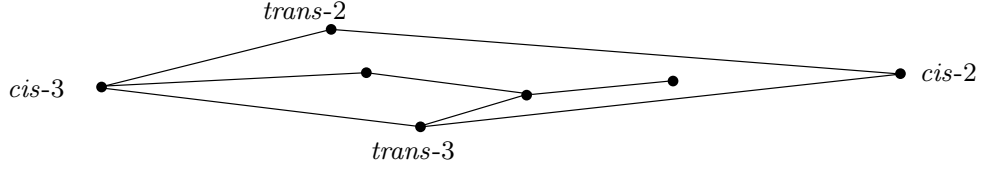
An example WSDUDP construction $WS(G)$ is shown in Figure 4.4, to provide intuition for the gadgets used in the reduction. Each edge of the graph G' (actual or dummy) corresponds to a disk in $WS(G)$, and each vertex (actual or dummy) corresponds to a point in \mathcal{Q} . A point in \mathcal{Q} pierces two disks in $WS(G)$ if the degree of the corresponding vertex in G' is two; points pierce three disks if their corresponding vertices have degree three. This ensures that there is a one-to-one mapping between the solution to the constructed WSDUDP instance, and the minimum vertex cover of the modified graph that it represents.

In the reduction we use *wires*, which are formally defined below. In essence, a wire is a set of disks which are arranged so that any disk intersects at most two others, and for any pair of disks with a non-empty area of intersection, this area contains a point. Conversely, any point in the wire pierces exactly two disks, so that given a wire with m_w disks, the cardinality of the minimum set of points piercing all disks is $\lceil m_w/2 \rceil$. We also use *stacks*, where a stack is a set of disks whose centre points are collinear on a vertical line.

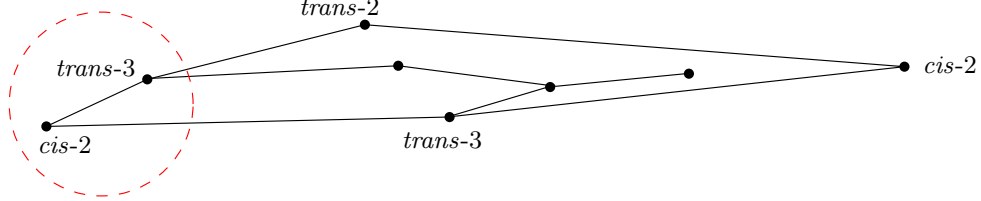
Lemma 4.7. *w horizontal wires may be placed so that they are vertically adjacent in the WSDUDP setting, for any $w > 0$, so that the solution in one wire is independent from that of all other wires.*

Proof. By placing the disk centres of all wires directly above one another to create sets of stacks, the solution for each wire is independent of all others. Suppose that there are w wires, and the strip has height h . If the lenses formed by the area of intersection between consecutive disks on a wire have height $h_\ell < h/w$, then the lenses do not intersect any other disks. However, we require extra room for gadgets, so we restrict each lens to half this height, $h_\ell = h/2w$. The minimum distance between disk centres on a wire is thus $d_{\text{disk}} > 2\sqrt{1 - h_\ell^2}$. \square

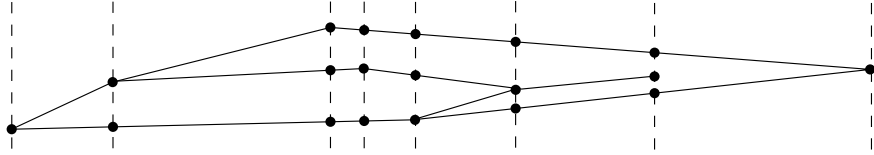
A wire w_i is a sequence of disks positioned so that consecutive centres are spaced d_{disk} units apart, not necessarily collinearly, where $2\sqrt{1 - h_\ell^2} < d_{\text{disk}} \leq \sqrt{2 + 2\sqrt{1 - (3h_\ell/4)^2}}$, so that there



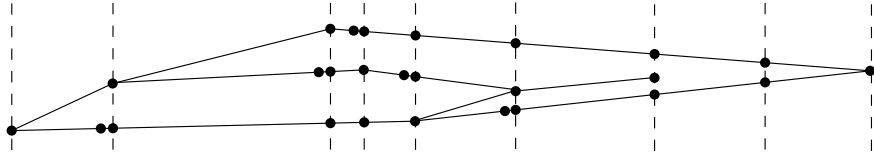
(a) Given a graph G , we compute a planar embedding (see Section 4.5.1 for vertex classes).



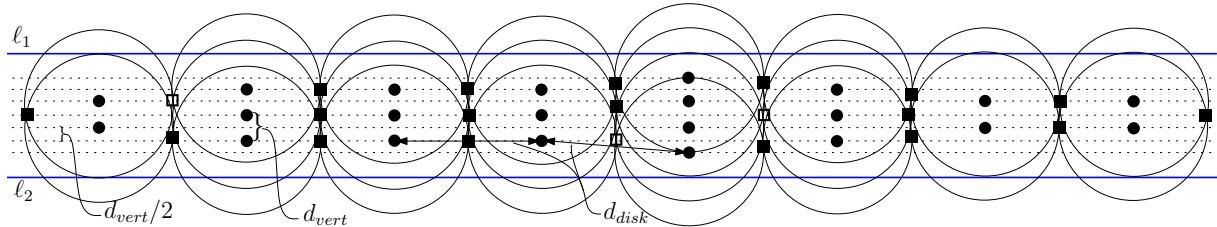
(b) The $cis-3$ vertex has been replaced with a $trans-3$ vertex and a $cis-2$ vertex (i.e. the edge has been ‘bent’ by adding a new vertex on the edge).



(c) Dummy vertices are added at the intersection points of edges of the graph with vertical lines incident upon all vertices.



(d) Dummy vertices have been added onto edges so that adjacent stacks only vary in cardinality by one, and all edges of the input graph have even numbers of dummy vertices incident upon them.



(e) We construct a series of stacks of disks, where disks in adjacent stacks have slight overlap. The disk centres in each stack are aligned vertically and separated by a fixed distance d_{vert} . The number of disks in adjacent stacks may only vary by one. If two consecutive stacks have the same number of disks, the centres are aligned horizontally and separated by d_{disk} . If two consecutive stacks have differing numbers of disks, the centres are staggered vertically by $d_{vert}/2$, so that each disk centre is d_{disk} from two disk centres in the adjacent stack (thus, these stacks are distance $\sqrt{d_{disk}^2 - d_{vert}^2}$ apart). The points of \mathcal{Q} are indicated by squares; those points piercing three disks are drawn as empty squares. The centre points of the disks \mathcal{P} are displayed as filled circles.

Figure 4.4: A sample WSDUDP construction $WS(G)$ for the NP-hardness reduction.

exists a small area of overlap between consecutive disks which contains a point in \mathcal{Q} .⁶ Disk centres on adjacent wires are placed $d_{\text{vert}} = 3h_\ell/2$ units apart vertically, and we define a *stack* as a set of such vertically aligned disks. The centres of the disks in a stack are shifted within the strip by $d_{\text{vert}}/2$ relative to an adjacent stack when the number of disks in the two stacks differs, while the distance between consecutive centres in each wire remains d_{disk} .

Lemma 4.8. *There is a non-empty area of intersection between three disks in two consecutive stacks when the centres of the stacks are shifted vertically by $d_{\text{vert}}/2$ relative to each other, and the distance $d_{\text{disk}} \leq \sqrt{2 + 2\sqrt{1 - (3h_\ell/4)^2}}$.*

Proof. Without loss of generality, assume that disks D_1, D_2 are in the left stack and D_3 is in the right stack, and their centrepoints are p_1, p_2, p_3 respectively (see Figure 4.5). Let \perp_{p_1, p_2} be the perpendicular bisector between p_1 and p_2 ; p_3 lies on \perp_{p_1, p_2} . Let ∂D_i denote the boundary of D_i . Then there is a unique point to the right of p_1 and p_2 at $\perp_{p_1, p_2} \cap \partial D_1 \cap \partial D_2$, call this point p_r . If p_r and p_3 are at most unit distance apart, then D_3 covers p_r and there is a non-empty area of intersection between D_1, D_2 , and D_3 . Suppose $\text{dist}(p_r, p_1) = \text{dist}(p_r, p_2) = \text{dist}(p_r, p_3) = 1$, and we assumed that $d_{\text{vert}} = 3h_\ell/2$, so $\text{dist}(p_1, p_3) = \text{dist}(p_2, p_3) = \sqrt{2 + 2\sqrt{1 - (3h_\ell/4)^2}}$. Therefore, the lemma holds when $d_{\text{disk}} \leq \sqrt{2 + 2\sqrt{1 - (3h_\ell/4)^2}}$, as claimed. \square

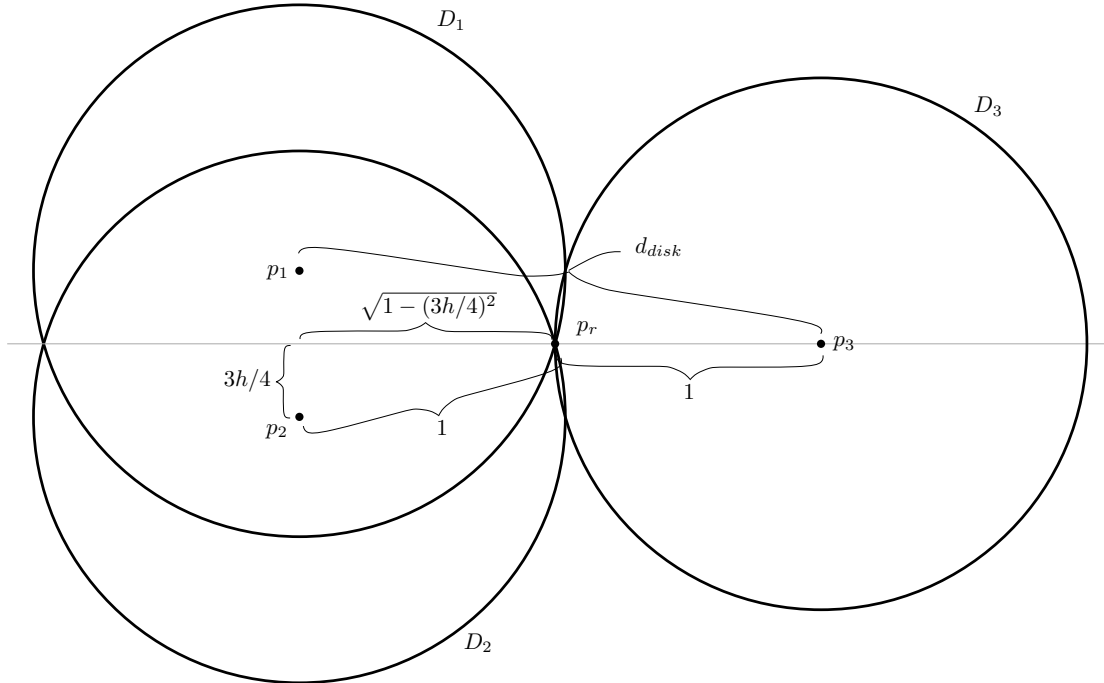


Figure 4.5: The construction demonstrates that there is a non-empty area of intersection between three disks when configured as shown and $d_{\text{disk}} \leq \sqrt{2 + 2\sqrt{1 - (3h_\ell/4)^2}}$.

⁶Note that $2\sqrt{1 - h_\ell^2} < \sqrt{2 + 2\sqrt{1 - (3h_\ell/4)^2}}$ for $h_\ell > 0$.

4.5.1 Gadgets

In the graph, we may encounter vertices of degree one, two, or three. With each vertex in our construction, wires may begin, end, split, merge, or continue unchanged. For vertices of degree one, the incident edge will correspond to a terminal disk on a wire (recall the definition of a wire from page 51). For vertices of degree two, if one edge leaves to the left and the other to the right in the embedding, this is a *trans-2* vertex⁷, and we handle it by continuing all wires. If both edges go in the same direction (left or right), we call this a *cis-2* vertex, and we have a gadget to merge the pair of wires corresponding to the edges. Analogously, we have gadgets for both the *trans-3* and *cis-3* degree three vertices. Finally, we build a gadget to increase the number of vertices on an edge. With each gadget, we apply the analogous modification to G' by adding dummy vertices to the respective edges. This ensures that an optimal solution to $WS(G)$ corresponds exactly to an optimal vertex cover for G' .

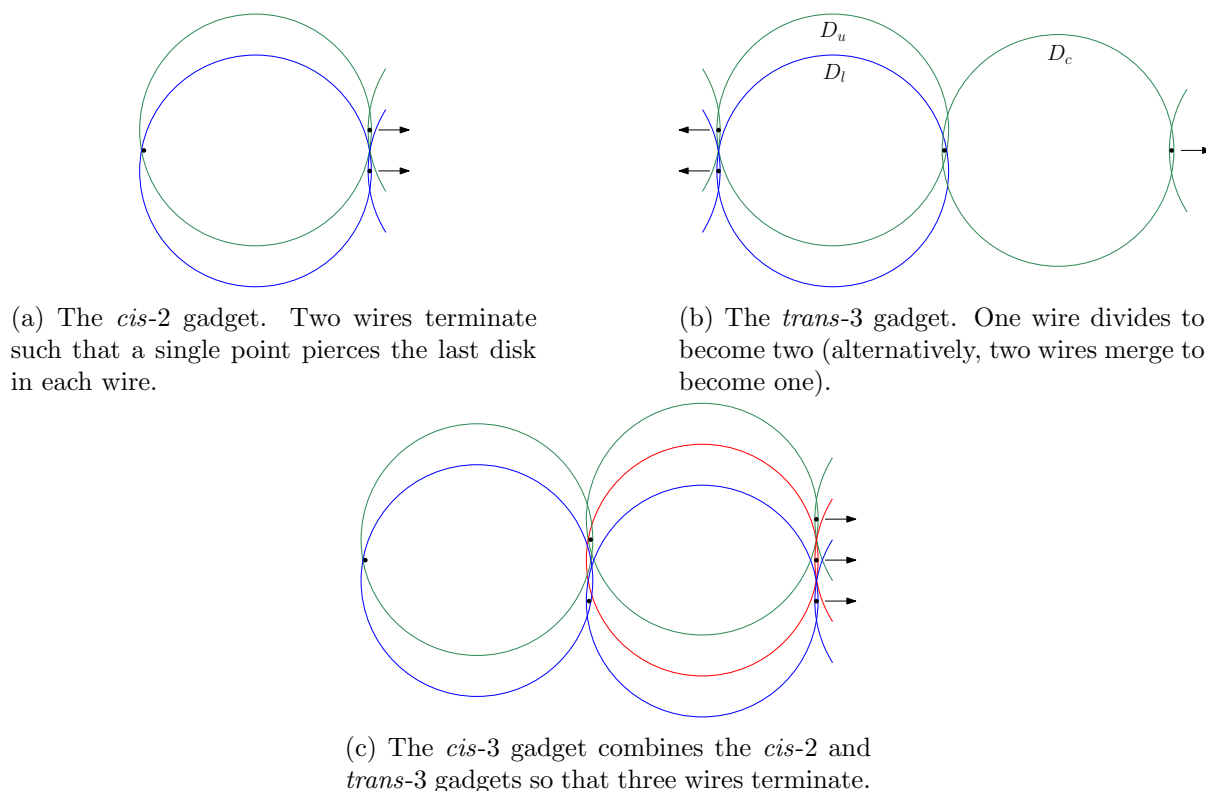


Figure 4.6: The gadgets for the WSDUDC NP-hardness reduction.

cis-2 Gadget

In this case, a pair of wires will terminate, and since the two terminal disks correspond to a pair of edges sharing a vertex, we place a vertex in the area covered by both disks and no others. An

⁷Borrowing from the Latin terms in isomerism, *trans* means “on the other side”, and *cis* means “on the same side”. We label them relative to a vertical line through the vertex, so a *cis* vertex has all incident edges to one side of the vertical line.

extra column of dummy nodes should be used to extend all other wires if the vertex is on an interior face of the planar embedding of the graph, since two wires are terminated simultaneously, and we may only shift wires by $d_{\text{vert}}/2$ with each column (see Figure 4.6a).

trans-3 Gadget

Suppose we have an upper wire ending in disk D_u and a lower wire ending in disk D_l , and they merge into a single wire beginning with disk D_c . We place D_c at a point so that the distance between the centres of both D_c to D_u and D_c to D_l is d_{disk} , as described in Lemma 4.8. By placing a vertex in $D_c \cap D_u \cap D_l$, a single point pierces three disks, which corresponds to a vertex which can cover three edges in the graph (see Figure 4.6b).

cis-3 Gadget

For this gadget, we combine the *trans-3* and *cis-2* gadgets to build a *cis-3* configuration. In the planar graph embedding, this corresponds to introducing a bend in the lowest edge incident to the *cis-3* vertex with a dummy vertex, so that it becomes a *trans-3* vertex (see Figure 4.6c).

CARD+ Gadget

If the total number of dummy vertices added to an edge of G is odd, we require a gadget which increases the number of disks between a pair of points on a wire by one. An extra disk whose centre is very close to the centre point of a disk on the wire (Figure 4.7) allows points to be placed so that the wire remains independent from adjacent wires, while increasing the number of disks on the wire by one.

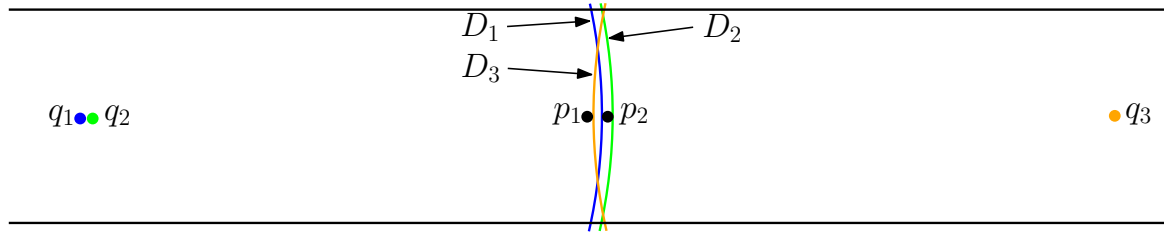


Figure 4.7: The CARD+ gadget. We add a disk D_2 into a wire by inserting the disk very close to one of the others on the wire. Suppose that we originally had D_1 and D_3 on the wire, and point p_1 was in their intersection, i.e. $p_1 \in D_1 \cap D_3$. Now we add a disk D_2 whose centre q_2 is beside q_1 , and p_1 is moved so that $p_1 \in D_1 \cap D_2, p_1 \notin D_3$. A new point p_2 is placed so that $p_2 \in D_2 \cap D_3, p_2 \notin D_1$. Now the wire contains one more disk than previously, and the new disk does not cover any points from adjacent wires.

Now an instance of the WSDUDP problem $\text{WS}(G)$ may be constructed from any planar graph G with no vertex of degree greater than three. A solution \mathcal{Q}^* to $\text{WS}(G)$ is also a solution $V_{G'}^*$ to the VERTEX-COVER problem on $G' = (V', E')$, where $v_i \in V'$ is mapped to $q_i \in \mathcal{Q}$ and $q_i \in D_j \leftrightarrow v_i \in e_j \in E'$. By Lemma 4.6, we can find a minimum vertex cover V_G^* for G from $V_{G'}^*$ in polynomial time. Therefore, there is a hitting set of size $c + (|\mathcal{D}| - |V|) / 2$ for $\text{WS}(G)$ if and only if there exists a vertex cover of size c for G . The optimal hitting set for \mathcal{D} uses all points

corresponding to the set of vertices in the vertex cover, plus exactly half of the extra points added in the construction of $WS(G)$ from G , by Lemma 4.6. The number of disks stacked vertically in any column of $WS(G)$ is in $O(m)$, where m is the number of edges and n is the number of vertices in the graph G . The number of such stacks is in $O(n)$, so the total number of disks and points in the WSDUDP construction is $O(mn)$. In Lemma 4.9, we show that the number of bits required to represent the coordinates of the disk centres and points may be bounded by a polynomial in the size of the input.

Lemma 4.9. *The coordinates of the disk centres and points may be expressed using a number of bits that is polynomial on the size of the input.*

Proof. We will demonstrate that all points are separated from other points by distances that are polynomial on the size of the input in each dimension, and hence their logarithms are linearly related. First we consider the y-coordinate, which is the axis orthogonal to the strip. In the construction, we place all points and disk centres on horizontal grid lines that are spaced by $3h_\ell/4 = 3h/8w$, where h is the height of the strip, and w is the maximum number of edges of the graph that may intersect a vertical line, so $w \in O(m)$. Therefore, the vertical distance between points may always be expressed using $O(\log w + \log(1/h))$ bits, which is polynomial in the size of the input, and so the same number of bits suffices to represent the y-coordinates.

The x-coordinates of points are placed at a unit distance to the right or left of a disk centre (since we consider the disks closed), and so we will show that the bounds hold for the disk centres only. Recall that the disk centres are separated by a value d_{disk} , where $2\sqrt{1-h_\ell^2} < d_{\text{disk}} \leq \sqrt{2+2\sqrt{1-(3h_\ell/4)^2}}$. The upper bound on the difference in the x-coordinate is $1 + \sqrt{1-(3h_\ell/4)^2}$, realized when adjacent disk centres are shifted vertically by $3h_\ell/4$ relative to each other, as shown in Figure 4.5, such as in a *trans-3* gadget. We will show that there exists a coordinate between these bounds that can be expressed in polynomially many bits on the size of the input by showing that the difference between the bounds may be lower bounded by a polynomial in the size the input. Hence a rational number with polynomially many bits fits in the desired region. Let $\delta = 1 + \sqrt{1-(3h_\ell/4)^2} - 2\sqrt{1-h_\ell^2}$, the difference between the upper and lower bounding functions on the horizontal distance between disk centres. We observe that a parabola such as $0.05h^2$ serves as a lower bound on this difference function in the range $h_\ell \in [0 \dots 1]$; a section of the plot of these two functions is illustrated in Figure 4.8.

To formalize this bound, we first take the derivative of $\delta - 0.05h_\ell^2$ to obtain $\delta' = -\frac{9h_\ell}{4\sqrt{16-9h_\ell^2}} + \frac{2h_\ell}{\sqrt{1-h_\ell^2}} - h_\ell/10$. This function has a single real root at $h_\ell = 0$, and so we conclude that $0.05h_\ell^2 \leq \delta$ for all values $h_\ell \in [0 \dots 1]$. Since h_ℓ may be expressed as a polynomial of h and m , the distance between disk centres in the x-coordinate may also be expressed as a polynomial of the input, because there exists such a value for d_{disk} between the upper and lower bounds. Therefore, the upper bound may be computed to a number of bits that is logarithmic in a polynomial of the input, and so one may determine the x-coordinates of the centres of the disks by computing from left to right with these bounds. □

This completes the proof of Theorem 4.5.

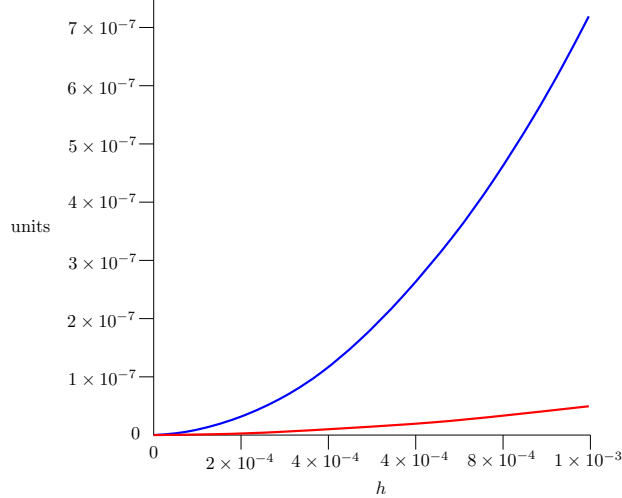


Figure 4.8: A plot suggesting that $0.05h_\ell^2$ (red) serves as a lower bound for $1 + \sqrt{1 - (3h_\ell/4)^2} - 2\sqrt{1 - h_\ell^2}$ (blue).

4.6 Conclusions and Future Work

We have outlined several approximation algorithms for the WSDUDC problem, along with a proof of NP-completeness for the problem. We began with a $O(mn + m \log m + n \log n)$ time 6-approximate algorithm for strips of height $1/\sqrt{2}$. The general $3\lceil 1/\sqrt{1 - h^2} \rceil$ -approximate algorithm and the 4-approximate algorithm for strips of height $\leq 2\sqrt{2}/3$ both run in $O(m^4n + n \log n)$ time. Finally, we presented a 3-approximate algorithm for strips of height $\leq 4/5$ which runs in $O(m^6n)$ time.

There are a few plausible directions for future work. It seems likely that an improved approximation algorithm could be obtained using the shifting strategies discussed in Section 2.1.6. It would also be interesting to see how well the problem could be approximated if the setting were changed to one where the points are confined to the strip, but the disk centres are unconstrained. While the dynamic programming solution for covering $\mathcal{P}_{\mathcal{R}}$ should work with small modifications, it is unclear whether the points in $\mathcal{P}_{\overline{\mathcal{R}}}$ can still be covered optimally (although we conjecture that an optimal solution is possible). Finally, an improvement to the SSDUDC algorithm would improve the running time of the general approximation algorithm and the 4-approximation algorithm, since the running time of SSDUDC is the bottleneck on the running time for these algorithms.

Improved Approximation Algorithms for Discrete Unit Disk Cover

We begin in Section 5.1 by noting that combining the A-LSDUDC algorithm (Algorithm 3.3) of Chapter 3 with techniques from the algorithm of Carmi et al. [29] provides a 22-approximation algorithm to the DUDC problem¹. The remainder of the chapter outlines the best known scheme for a tractable² DUDC approximation algorithm, where the approximation factor and running time are determined by the choice of WSDUDC algorithm that is used. These range from a 15-approximate algorithm which runs in $O(m^6n + n \log n)$ time to a 18-approximate algorithm which runs in $O(mn + m \log m + n \log n)$ time.

5.1 22-Approximate Algorithm for Discrete Unit Disk Cover

As described in Section 2.3 (page 24), Carmi et al. [29] prove that in their framework, any disk can be used in up to eight applications of the A-LSDUDC algorithm, for which they have a 4-approximation. These operations, combined with a 6-approximation algorithm for BOX (Section 2.3.1) results in an $8 \times 4 + 6 = 38$ -approximation algorithm for the general DUDC problem. We have shown improvements that provide a 2-approximation for the A-LSDUDC problem, as stated in Theorem 3.2, and so we now have an approximation ratio of $8 \times 2 + 6 = 22$ for general the DUDC problem using Algorithm 2.5.

There are essentially two main components to Algorithm 2.5. First, the grid of size $3/2 \times 3/2$ is applied to the input data. On each grid line, the A-LSDUDC algorithm (Algorithm 3.3) is run twice (once for each side), and the number of relevant grid lines is bounded by $O(n)$. A-LSDUDC runs in $O(mn + m \log m + n \log n)$ time, and so the running time of the DUDC algorithm is dominated by the BOX approximation algorithm.

Theorem 5.1. *Given sets \mathcal{P} of m points and \mathcal{D} of n disks in the plane, we can compute a 22-approximation of the DUDC problem in $O(m^2n^4)$ time in the worst case.*

¹Elements of this chapter have appeared in [39], [40], [46], [47] and [73].

²Often “tractable” is synonymous with polynomial time. In this context, we use tractable rather loosely and subjectively, but the intent is to state that the algorithm is implementable and it could be expected to terminate in a reasonable amount of time on real world data. The tractability of our 15-approximate algorithm is questionable, but the 18-approximate algorithm is certainly within this definition.

5.2 15-Approximate Algorithm for Discrete Unit Disk Cover

We now improve on the algorithm of the previous section with the use of a different planar decomposition. Let $R^{\{\mathcal{P}, \mathcal{Q}\}}$ be an axis aligned rectangle such that all points in \mathcal{P} and all centres of the disks in \mathcal{D} are inside $R^{\{\mathcal{P}, \mathcal{Q}\}}$. In the DUDC algorithm, we first divide $R^{\{\mathcal{P}, \mathcal{Q}\}}$ into horizontal strips bounded by the line segments ℓ_0, \dots, ℓ_t (indexed from top to bottom) so that $\text{dist}(\ell_i, \ell_{i+1}) = \frac{1}{\sqrt{2}}$ for each $i \in \{0, \dots, t-1\}$. Note that ℓ_0 and ℓ_t are the top and bottom boundaries of the rectangle $R^{\{\mathcal{P}, \mathcal{Q}\}}$ respectively, and $\text{dist}(a, b)$ is the minimum Euclidean distance between two line segments a and b . We denote the horizontal strip bounded by the lines ℓ_i and ℓ_{i+1} as $[\ell_i, \ell_{i+1}]$.

The DUDC algorithm has two fundamental components. In the first step, we describe an algorithm for covering all points in \mathcal{P} which are line-separable from a disk in \mathcal{D} by any line in the set $\{\ell_1, \dots, \ell_{t-1}\}$. By carefully partitioning the set \mathcal{P} , we obtain a 12-approximation algorithm for this setting, which we call the *Outside Strip Discrete Unit Disk Cover* (OSDUDC) problem (not to be confused with the Strip-Separated Discrete Unit Disk Cover (SSDUDC) problem of Section 2.2.3, which is applied to a single strip only). In the second step, we apply one of the WSDUDC algorithms from Chapter 4 to cover the points in $\mathcal{P} \cap [\ell_i, \ell_{i+1}]$, which are covered only by the disks centred inside the strip $[\ell_i, \ell_{i+1}]$, for each $i \in \{0, 1, \dots, t-1\}$ (this is the case for all points remaining after running the OSDUDC algorithm).

5.2.1 Outside Strip Discrete Unit Disk Cover

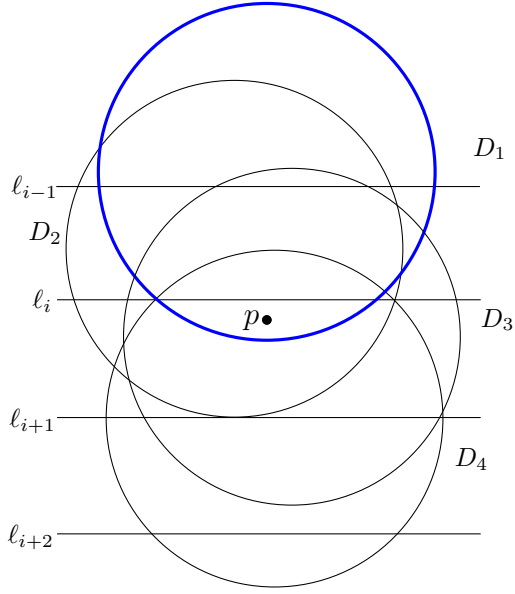
In this section, we present a 12-approximation algorithm for finding a cover for all points $\mathcal{P}^\circ \subseteq \mathcal{P}$, where for each $p \in \mathcal{P}^\circ$, if s is the strip containing p then p may be covered by a disk centred outside of s .³

Definition 5.1. OSDUDC Point Property: *The set of points $\mathcal{P}^\circ \subseteq \mathcal{P}$ to be covered by the OSDUDC problem are the points $p \in \mathcal{P}$ where p may be covered by a disk in \mathcal{D} centred outside of the strip containing p .*

Let \mathcal{D}_i be the set of disks centred in the horizontal strip $[\ell_{i-1}, \ell_i]$. Let $\mathcal{P}_i^L (\subseteq \mathcal{P})$ be the set of points below the line ℓ_i and covered by the disks in \mathcal{D}_i but not by any disk centred above the line ℓ_{i-1} (such points belong to \mathcal{P}_{i-1}^L , see Figure 5.1). Similarly, let $\mathcal{P}_i^U (\subseteq \mathcal{P})$ be the set of points above the line ℓ_i and covered by the disks in \mathcal{D}_{i+1} but not by any disk centred above the line ℓ_{i-1} (these will be in either \mathcal{P}_{i-2}^L or \mathcal{P}_{i-1}^L) or below the line ℓ_{i+1} (if not covered from above, such points belong to \mathcal{P}_{i+1}^U). The sets $\mathcal{P}_1^L, \mathcal{P}_2^L, \dots, \mathcal{P}_t^L, \mathcal{P}_t^U, \mathcal{P}_{t-1}^U, \dots, \mathcal{P}_1^U \subseteq \mathcal{P}$ are determined greedily as follows:

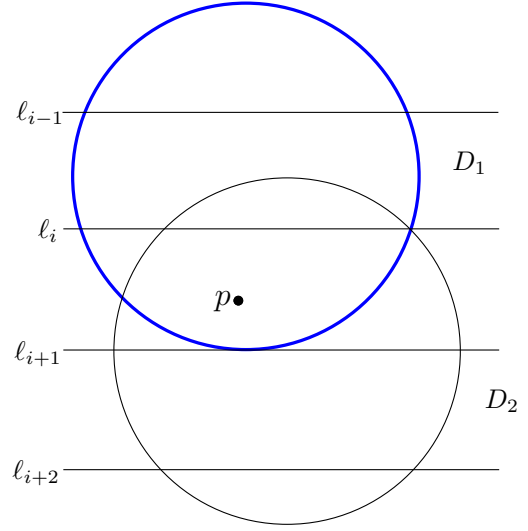
1. Set $\mathcal{P}' = \mathcal{P}$.
2. For each $i \in \{1, 2, \dots, t\}$ compute $\mathcal{P}_i^L = \mathcal{P}' \cap \mathcal{D}_i$ and set $\mathcal{P}' = \mathcal{P}' \setminus \mathcal{P}_i^L$.
3. For each $i \in \{t, t-1, \dots, 1\}$ compute $\mathcal{P}_i^U = \mathcal{P}' \cap \mathcal{D}_{i+1}$ and set $\mathcal{P}' = \mathcal{P}' \setminus \mathcal{P}_i^U$.

We illustrate the possible classifications of points in Figure 5.1. Suppose the point p is contained in strip s . Basically, if the point is covered by a disk centred above s , then the highest such disk



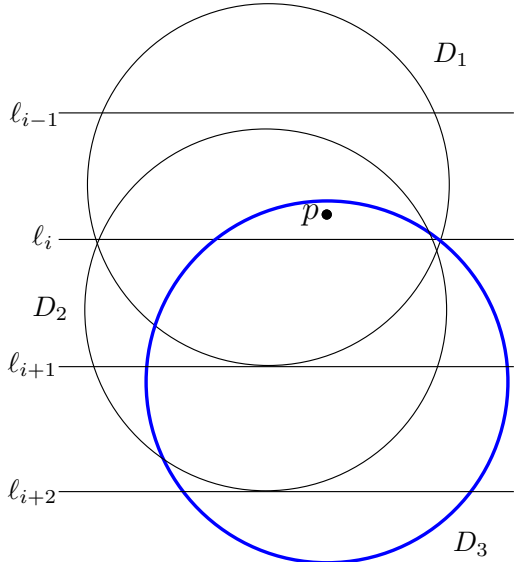
$$p \in \mathcal{P}_{i-1}^L$$

(a) $p \in D_1$, and $D_1 \in \mathcal{D}_{i-1}$.



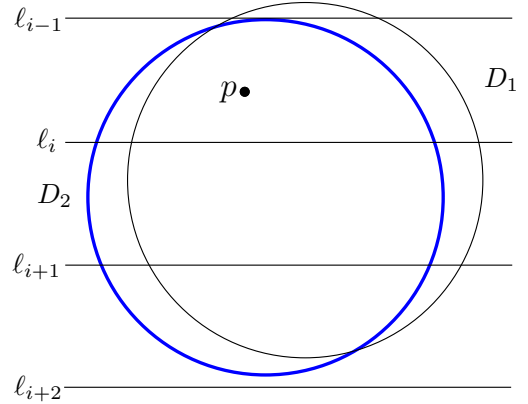
$$p \in \mathcal{P}_i^L$$

(b) $p \in D_1$, and $D_1 \in \mathcal{D}_i$. We check for this condition if (a) does not apply.



$$p \in \mathcal{P}_{i+1}^U$$

(c) $p \in D_3$, and $D_3 \in \mathcal{D}_{i+2}$. We check for this case if neither of the conditions shown in (a) and (b) apply. Notice that $p \in D_1$, where the centre of D_1 and p are in the same strip. Such disks do not affect the classification procedure here.



$$p \in \mathcal{P}_i^U$$

(d) p is covered by disks D_1 and D_2 , both of \mathcal{D}_{i+1} , and none of the conditions shown in (a),(b) and (c) apply. If this last condition also does not apply, then the disks can only be covered by disks centred in the same strip as p . We address such points later.

Figure 5.1: Some examples to illustrate how the points of \mathcal{P} are classified for the OSDUDC algorithm. The disk which determines membership is shown in bold.

determines membership. If this is not the case, then the lowest disk centred below s determines which set p belongs to. Disks centred in s are irrelevant, as far as this classification is concerned.

Note that the sets of points are determined by the disks covering them, and not the strip containing them. All sets \mathcal{P}_i^L are covered by the disks in \mathcal{D}_i , i.e. those disks centred inside the horizontal strip $[\ell_{i-1}, \ell_i]$. Similarly, all sets \mathcal{P}_i^U are covered by the disks in \mathcal{D}_{i+1} , i.e. the disks in \mathcal{D} whose centres are inside the horizontal strip $[\ell_i, \ell_{i+1}]$. No points in \mathcal{P}_i^U are covered by any disk centred above the line ℓ_{i-1} .

Without loss of generality assume that t , the number of strips, is $t = 4s$, for $s \in \mathbb{Z}^+$. Consider six sets of sets related to \mathcal{P}_i for $i \in \{1, 2, \dots, 6\}$ defined below:

$$\begin{aligned}\mathcal{S}_1 &= \{\mathcal{Q}_{1j} (= \mathcal{P}_{1+4j}^L) \mid j = 0, 1, \dots, s-1\} \\ \mathcal{S}_2 &= \{\mathcal{Q}_{2j} (= \mathcal{P}_{2+4j}^L) \mid j = 0, 1, \dots, s-1\} \\ \mathcal{S}_3 &= \{\mathcal{Q}_{3j} (= \mathcal{P}_{3+4j}^L) \mid j = 0, 1, \dots, s-1\} \\ \mathcal{S}_4 &= \{\mathcal{Q}_{4j} (= \mathcal{P}_{4+4j}^L) \mid j = 0, 1, \dots, s-1\} \\ \mathcal{S}_5 &= \{\mathcal{Q}_{5j} (= \mathcal{P}_{1+2j}^U) \mid j = 0, 1, \dots, 2s-2\} \\ \mathcal{S}_6 &= \{\mathcal{Q}_{6j} (= \mathcal{P}_{2+2j}^U) \mid j = 0, 1, \dots, 2s-2\}.\end{aligned}$$

The first four sets include every fourth point set of the type \mathcal{P}_j^L , and the last two sets consist of every other set of the type \mathcal{P}_j^U . We characterize the elements in each of the sets \mathcal{S}_i for $i \in \{1, 2, \dots, 6\}$ using the following lemma:

Lemma 5.1. *If $p \in \mathcal{Q}_{uv}$ and $p' \in \mathcal{Q}_{uw}$ are two points such that for $u \in \{1, 2, 3, 4\}$, $v \in \{1, 2, \dots, s-1\}$, $w \in \{1, 2, \dots, s-1\}$, and $v \neq w$, then a single disk in \mathcal{D} cannot cover both the points p and p' . The same is true for $u \in \{5, 6\}$, $v \in \{1, 2, \dots, 2s-2\}$, $w \in \{1, 2, \dots, 2s-2\}$ and $v \neq w$.*

Proof. **Case $u = 1$:** From the definition of \mathcal{Q}_{1v} and \mathcal{Q}_{1w} , $\mathcal{Q}_{1v} = \mathcal{P}_{1+4v}^L$ and $\mathcal{Q}_{1w} = \mathcal{P}_{1+4w}^L$. Now, from the definition of \mathcal{P}_{1+4v}^L and \mathcal{P}_{1+4w}^L each element of \mathcal{P}_{1+4v}^L and \mathcal{P}_{1+4w}^L is covered by the disks centred inside the horizontal strip $[\ell_{4v}, \ell_{1+4v}]$ and $[\ell_{4w}, \ell_{1+4w}]$, respectively. Since $v \neq w$ and the height of each horizontal strip is $\frac{1}{\sqrt{2}}$, so the distance between a point inside the strip $[\ell_{4v}, \ell_{1+4v}]$ and a point inside the strip $[\ell_{4w}, \ell_{1+4w}]$ is greater than 2. In this case the lemma follows from the fact that the disks are unit radius disks.

Cases $u \in \{2, 3, 4\}$: The argument is analogous to that of the first case.

Case $u = 5$: From the definition of \mathcal{Q}_{5v} and \mathcal{Q}_{5w} ; $\mathcal{Q}_{5v} = \mathcal{P}_{1+2v}^U$ and $\mathcal{Q}_{5w} = \mathcal{P}_{1+2w}^U$. Now, from the definition of \mathcal{P}_{1+2v}^U each element of \mathcal{P}_{1+2v}^U is covered by the disks centred inside the horizontal strip $[\ell_{1+2v}, \ell_{2+2v}]$, but no points in \mathcal{P}_{1+2v}^U are covered by a disk centred above the line ℓ_{2v} . Similarly, from the definition of \mathcal{P}_{1+2w}^U , each element of \mathcal{P}_{1+2w}^U is covered by the disks centred inside the horizontal strip $[\ell_{1+2w}, \ell_{2+2w}]$, but no points in \mathcal{P}_{1+2w}^U are covered by a disk centred above the line ℓ_{2w} . Since $v \neq w$, there does not exist a disk which covers one point in \mathcal{P}_{1+2v}^U and a point in \mathcal{P}_{1+2w}^U .

Case $u = 6$: The argument is analogous to that of the previous case. □

The OSDUDC algorithm for covering the points in $\mathcal{P}_1^L, \mathcal{P}_2^L, \dots, \mathcal{P}_t^L, \mathcal{P}_t^U, \mathcal{P}_{t-1}^U, \dots, \mathcal{P}_1^U$ is described in Algorithm 5.1.

³Other points not having this property (i.e. those points only covered by a disk centred in s) may be covered when we use the OSDUDC algorithm, but this has no effect on the final running time or approximation factor of the DUDC algorithm.

Algorithm 5.1 OSDUDC(\mathcal{P}, \mathcal{D})

```
1: Input: A point set  $\mathcal{P}$  and a unit disk set  $\mathcal{D}$ .
2: Output: A set  $\mathcal{D}_1^* \subseteq \mathcal{D}$  covering  $\mathcal{P}^\circ$  with a 12-approximation (see Definition 5.1, page 60).
3:  $\mathcal{D}_1^* \leftarrow \emptyset$ 
4:  $\mathcal{P}' \leftarrow \mathcal{P}$ .
5: for  $i := 1$  to  $t$  do
6:    $\mathcal{P}_i^L \leftarrow \mathcal{P}' \cap \mathcal{D}_i$ 
7:    $\mathcal{P}' \leftarrow \mathcal{P}' \setminus \mathcal{P}_i^L$ 
8: end for
9: for  $i := t$  to  $1$  do
10:   $\mathcal{P}_i^U \leftarrow \mathcal{P}' \cap \mathcal{D}_{i+1}$ 
11:   $\mathcal{P}' \leftarrow \mathcal{P}' \setminus \mathcal{P}_i^U$ 
12: end for
13: for  $(i = 1, 2, \dots, t)$  do
14:   $\mathcal{D}' \leftarrow \text{A-LSDUDC}(\mathcal{P}_i^L, \mathcal{D}, \ell_i)$ 
15:   $\mathcal{D}_1^* \leftarrow \mathcal{D}_1^* \cup \mathcal{D}'$ 
16: end for
17: for  $(i = t, t-1, \dots, 1)$  do
18:   $\mathcal{D}' \leftarrow \text{A-LSDUDC}(\mathcal{P}_i^U, \mathcal{D}, \ell_i)$ 
19:   $\mathcal{D}_1^* \leftarrow \mathcal{D}_1^* \cup \mathcal{D}'$ 
20: end for
21: return  $\mathcal{D}_1^*$ 
```

Lemma 5.2. *Algorithm 5.1 computes a 12-factor approximation for the OSDUDC problem in $O(mn + m \log m + n \log n)$ time.*

Proof. The sets of points $\mathcal{P}_1^L, \mathcal{P}_2^L, \dots, \mathcal{P}_t^L, \mathcal{P}_t^U, \mathcal{P}_{t-1}^U, \dots, \mathcal{P}_1^U$ can be partitioned into 6 sets of sets such that a disk cannot cover a point from each of $\mathcal{Q}_{uv} \in \mathcal{S}_u$ and $\mathcal{Q}_{vw} \in \mathcal{S}_v$, $v \neq w$, by Lemma 5.1. Thus, the approximation factor of the lemma follows from the fact that the A-LSDUDC algorithm provides a 2-factor approximation (see Section 2.2.1), and it can participate in at most 6 sets. Since we can divide the plane into strips such that each strip contains at least one point in \mathcal{P} , $t = O(n)$. The time complexity of the lemma follows from the fact that each disk in \mathcal{D} can participate at most 6 times in applications of the A-LSDUDC algorithm and the running time of the A-LSDUDC algorithm is $O(mn + m \log m + n \log n)$. \square

5.2.2 The Discrete Unit Disk Cover Algorithm

We describe the algorithm for the DUDC problem in Algorithm 5.2, using Algorithm 5.1 and a WSDUDC solution (Chapter 4) as subroutines.

Theorem 5.2. *Algorithm 5.2 for DUDC is a $12+c$ -approximation algorithm, taking T time in the worst case. The pair (c, T) is determined by the choice of WSDUDC algorithm, and possibilities include $(3, O(m^6n + n \log n))$, $(4, O(m^4n + n \log n))$, and $(6, O(mn + m \log m + n \log n))$.*

Proof. The approximation result follows from Lemma 5.2, which established a 12-approximation for OSDUDC, which is added to the approximation factor of the selected WSDUDC algorithm.

Algorithm 5.2 DUDC(\mathcal{P}, \mathcal{D})

- 1: **Input:** A set of points \mathcal{P} and a set of unit disks \mathcal{D} .
- 2: **Output:** A set $\mathcal{D}^* \subseteq \mathcal{D}$ which covers \mathcal{P} .
- 3: Let $s_i \leftarrow [\ell_i, \ell_{i+1}]$ for $i = 0, 1, \dots, t - 1$ be the set of horizontal strips of height $\frac{1}{\sqrt{2}}$
- 4: $\mathcal{D}_1^* \leftarrow \text{OSDUDC}(\mathcal{P}, \mathcal{D})$ (Algorithm 5.1)
- 5: $\mathcal{P} \leftarrow \mathcal{P} \setminus (\mathcal{D}_1^* \cap \mathcal{P})$
- 6: $\mathcal{D}_2^* \leftarrow \emptyset$
- 7: **for** $i:=0$ **to** $t-1$ **do**
- 8: $\mathcal{D}_i \leftarrow$ the set of disks having centres in s_i
- 9: $\mathcal{P}_i \leftarrow \mathcal{P} \cap s_i$
- 10: $\mathcal{D}_2^* \leftarrow \mathcal{D}_2^* \cup \text{WSDUDC}(\mathcal{P}_i, \mathcal{D}_i)$ (choose any WSDUDC algorithm)
- 11: **end for**
- 12: **return** $\mathcal{D}_1^* \cup \mathcal{D}_2^*$

For the overall time complexity of Algorithm 5.2, observe that in line number 4, the running time of $\text{OSDUDC}(\mathcal{P}, \mathcal{D})$ is $O(mn + m \log m + n \log n)$ (Lemma 5.2). In line number 10, the running time of each call to $\text{WSDUDC}(\mathcal{P}_i, \mathcal{D}_i)$ is determined by that of the particular WSDUDC algorithm chosen. This ranges from $O(|\mathcal{D}_i||\mathcal{P}_i| + |\mathcal{D}_i| \log |\mathcal{D}_i| + |\mathcal{P}_i| \log |\mathcal{P}_i|)$ for the 6-approximate algorithm, to $O(|\mathcal{D}_i|^6|\mathcal{P}_i| + |\mathcal{P}_i| \log |\mathcal{P}_i|)$ for the 3-approximate algorithm. Therefore, the running time of the for loop (lines 7-11), and so the DUDC algorithm overall, is dominated by WSDUDC. \square

5.3 Conclusions and Future Work

We have considered the Discrete Unit Disk Cover (DUDC) problem, where a set \mathcal{P} of n points and a set \mathcal{D} of m unit disks in the 2-dimensional plane are given, and the objective is to find a minimum cardinality subset $\mathcal{D}^* \subseteq \mathcal{D}$ such that unit disks in \mathcal{D}^* cover all the points in \mathcal{P} . We provide a set of approximation algorithms for the DUDC problem, where the approximation factor and running time depend on the choice of WSDUDC algorithm. A summary of our results is presented in Figure 2.2.

Because the results depend on A-LSDUDC and WSDUDC, any improvement to the approximation algorithms for these problems directly provides improvements to these DUDC results. Perhaps the best improvement would be obtained with an improved decomposition (as opposed to the square- or strip-based ones explored in this thesis). An exact algorithm for A-LSDUDC would reduce the approximation factor of Algorithm 5.2 to 9 when combined with the 3-approximate WSDUDC algorithm (Algorithm 4.6). Finally, it would be interesting to see a PTAS result which fills the gap between our algorithm and the existing PTAS for DUDC.

Constrained Polygonal Vertex Cover Problems and the Hausdorff Core

In the next three chapters, we study the Hausdorff Core of a polygon¹. Essentially, given a simple polygon P , a Hausdorff Core of P is a convex polygon Q contained in P that minimizes the Hausdorff distance between P and Q . Given two polygons P and Q , the measure may be expressed formally as²:

$$H(P, Q) = \max\left\{\sup_{p \in P} \inf_{q \in Q} \text{dist}(p, q), \sup_{q \in Q} \inf_{p \in P} \text{dist}(p, q)\right\},$$

where $\text{dist}(p, q)$ is the Euclidean distance from p to q . Additionally, we require that Q , which is considered to be an approximation of P , is convex and fully contained in P . We call this approximation a *Hausdorff Core* of a polygon.

This is equivalent to a Unit Disk Piercing problem as follows: suppose we place unit disks on the vertices of a polygon P , so that disk $D_i \in \mathcal{D}$ has centre point p_i . We wish to pierce all disks in \mathcal{D} with a convex polygon Q . If there exists a point $q_i \in Q \cap D_i$ for each $D_i \in \mathcal{D}$, then we say that Q pierces \mathcal{D} . If we additionally require that $Q \subseteq P$, then Q is a Hausdorff Core of P with unit Hausdorff distance (we establish this relationship in Lemma 6.3).

This work is motivated by the problem of path planning in the context of navigation at sea. In this application, a plotted course must be tested against bathymetric soundings to ensure that the ship will not run aground. We suppose the soundings have been interpolated into contour lines [1] and the plotted course is given as a polygonal line. Although contour lines can be arbitrarily complicated, typical shipping routes run far from potential obstacles for the majority of their trajectories, and only short segments require more careful route planning. As a result, most tests for intersection between a path and a contour line should be easy: we could subdivide the map into polygonal regions so that most intersection tests are against regions with properties that permit fast tests, ideally reserving more expensive tests for the rare cases where the path comes close to intersecting the terrain.

The search for easily-testable areas motivates the study of the simplification of a contour line into a simpler object which is either entirely contained within the contour line or else fully contains it. In this work we consider the case in which the simplified polygon must be convex and contained.

¹Elements of this chapter have appeared in [53] and [74].

²The Hausdorff distance is equivalent to Blaschke's Nachbarschaftsmaß [86].

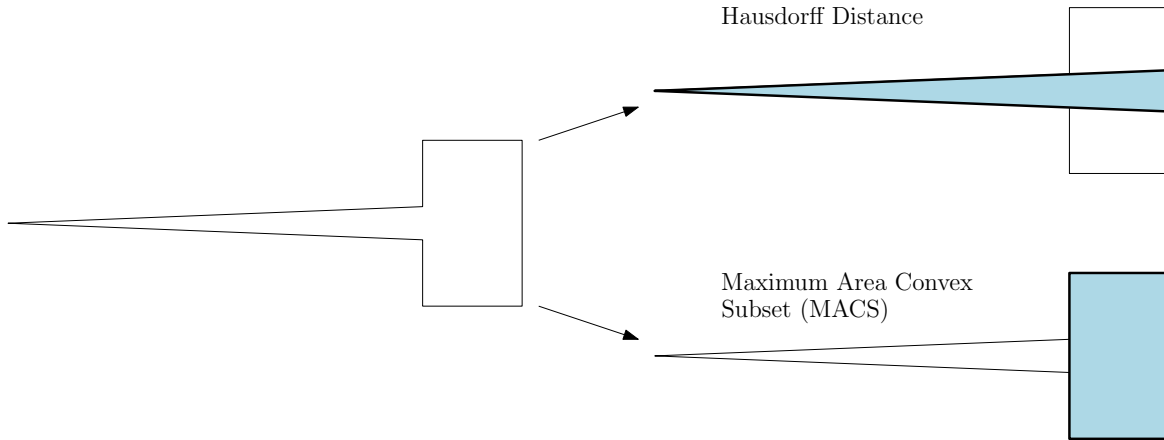


Figure 6.1: The optimal solutions for approximating polygons may vary significantly for different error metrics. In this example, we show the optimal solutions for the Hausdorff distance and the maximum area convex subset (MACS), also known as the ‘potato’ [31]).

Definition 6.1. The Hausdorff Core Problem: *Given a simple polygon P , the Hausdorff Core problem on P is to determine a convex polygon Q , such that $Q \subseteq P$ and $H(P, Q)$ is minimized, where $H(P, Q)$ denotes the Hausdorff distance between P and Q . We call Q a Hausdorff Core of P .*

We restrict the discussion to simple polygons, where we define a *polygon* P as a closed region of the plane, with the boundary of the polygon ∂P represented as a polygonal chain on n distinct vertices $P = \{p_0, \dots, p_{n-1}\}$ and n edges $e_i = (p_i, p_{i'})$, where $i' = \underline{(i + 1)} \bmod n$. Finally, a polygon P is *convex* if for all points p and p' in P , the line segment pp' is contained in P .

The Hausdorff Core problem is intended to capture a sense of “closeness” between the input and approximating polygons. While there are a number of measures that may be used (we discuss a sampling of metrics in Section 6.2.6), the Hausdorff Core is perhaps most intuitive in that it is optimized when the maximum distance between the polygons is minimized. See Figure 6.1 for an illustration of approximating polygons when minimizing the Hausdorff distance between the polygons as well as minimizing the difference in area between the polygons.

6.1 Paradigms for Approximating Polygons and Polygonal Chains

Suppose that we have a contour line represented as a piecewise linear curve or polygon with n points (for our application, this corresponds to some contour of the ocean floor at a depth of x metres). If we wish to represent this same polygon or polygonal chain with an approximation which minimizes the error to the original curve using n/c points for some constant c , this is known as a *min- ϵ problem*. Alternatively, a *min-# problem* is one in which we wish to bound the error of approximation by some maximum value ϵ with respect to the original polygon, while using a minimum number of vertices (we discuss these paradigms in more detail in Section 6.2.6). Our application requires a monotonicity constraint: if a ship has a draft of x' metres, where $x' < x$, then we must be sure that our approximation represents either a depth of $d \geq x$ or $d \leq x$. In other words, an approximating polygon cannot cross the original polygon representing the contour. Our research is a variant of a min- ϵ type problem, where the constraint is that the

approximating polygon is convex and contained by the input, rather than imposing a restriction on the number of vertices.

In general, an approximation will not make exclusive use of the vertices in P . For example, if we are given a convex polygon which we wish to approximate on the outside, vertices must be placed somewhere outside the polygon. Otherwise, you cannot approximate the polygon at all! Some of the approximation schemes used for curves provide insight into other techniques as well as into terrain approximation algorithms, so we begin by reviewing techniques which use subsets of the vertices in the original curve.

6.2 Related Work

Approximation algorithms for curves were initially concerned with searching for discretizations of curves defined by functions [17, 77, 150], but we are interested in approximating piecewise linear (also known as digital) curves and polygons. The problem of approximating piecewise linear curves with simpler piecewise linear curves without an additional monotonicity constraint has been well studied [55, 97, 103, 131, 138], and most of the recent work in the field discusses the optimality of the algorithms with respect to various error metrics. There are three basic approaches to the problem [55]:

1. delete specific features of the terrain represented by the line;
2. approximate the line with a mathematical function;
3. eliminate points along the line until some error threshold is met.

A critical limitation of these approaches with regard to our application is that each approximation technique reviewed here creates an approximation where all points in the approximation were found in the original curve, i.e. given an approximation P' of the curve P , $\forall p \in P', p \in P$. Our application has the condition that the approximation must not cross the original curve, and so we may need to use points in the approximation that were not in the original curve. To see an example of an optimal approximating polygon for the Hausdorff Core problem which uses none of the vertices of P , peek ahead to Figure 6.9 (page 81).

6.2.1 Feature-Based Approximation

The first approach is not often used because it requires the use of domain knowledge. Also, for our application this idea is unsavoury. As an example, if a region of shoreline is characterized by a series of fiords, you might delete all of the fiords but a few [55], and the remaining ones are expected to convey the nature of the shoreline. This is not the sort of approximation that we are interested in, because it is of a subjective nature.

6.2.2 Mathematical Approximation

The second idea is more appealing, and there are numerous functions that may be used to approximate a line. Some techniques that might be used are mean or centroid functions to produce a piecewise linear approximation. For example, using the mean, we begin by selecting a

constant number (say c) of points to approximate by replacing up to c points with their mean. The points are then traversed in the order that they appear on the chain. While consecutive pairs of points are relatively linear, the coordinates of the points are added together. If the number of points summed together reaches c or if the sequence of summed points reaches some threshold of non-linearity, then the mean position of the set of encountered points is computed. The algorithm continues by beginning again with the next point in the sequence [102]. Douglas and Peucker [55] concluded that the use of functions is more appropriate for smoothing applications than for generalizing the curve. They found that the functions mentioned here tended to over-represent the curve in smooth regions, and that extreme points may be removed. The removal of extreme points may be undesirable if these points are critical to the true nature of the curve.

6.2.3 Error Tolerance-Based Approximation

The final approach to approximating a curve is to iteratively remove points until some predefined error threshold is met. Douglas and Peucker³ [55] proposed a technique where the endpoints of the curve are joined, and the vertex p with the highest error is found. Here, the error is the distance from p to the closest point on the curve, and the maximum such value is exactly the Hausdorff distance between the curves. If the error is acceptable, all points but the endpoints are removed, and the algorithm finishes. Otherwise, the curve is split in two using p as a pivot, and the algorithm continues recursively on the two sub-curves [55, “method two”]. The algorithm is illustrated on an example curve in Figure 6.2. Several researchers [159, 90] have concluded that their eponymous Douglas-Peucker simplification algorithm is best, both in terms of subjective and objective measures. Note that it is non-optimal in terms of minimizing the number of points required to obtain a given error bound however (we discuss more rigorous algorithms shortly). An $O(n \log n)$ implementation of the algorithm was provided by Hershberger and Snoeyink [91].

These approaches dictate that all of the points in the approximation are also found in the original curve. Formally, we can define the original curve to be composed of a set of n points $P = \{p_1, \dots, p_n\} \in \mathbb{R}^2$. The approximate curve is defined by m points where $2 \leq m \leq n$, specifically $P' = \{p'_1, \dots, p'_m\}$, where for each point $p'_i \in P'$, we have $p'_i = p_j$ for some i, j [131]. Sato [142] formalizes some useful conditions for this relation:

- Boundary condition - the endpoints of both curves are the same.

$$p_1 = p'_1, p_n = p'_m$$

- Monotonic condition - the relative ordering of the points is the same.

$$p_j > p_{j-1}, p'_i > p'_{i-1}$$

- Range - the range of possible values for an index i and j where $p_i = p'_j$ is well defined.

$$j \leq i \leq j + n - m$$

³The same technique was presented a year earlier in 1972 by Ramer [136] and it was also described as the *iterative end-point fits* algorithm by Duda and Hart [56, p. 338] in 1973, and they in turn credit G.E. Forsen with the idea [56, p. 373]. However, Douglas and Peucker are generally credited with the technique.

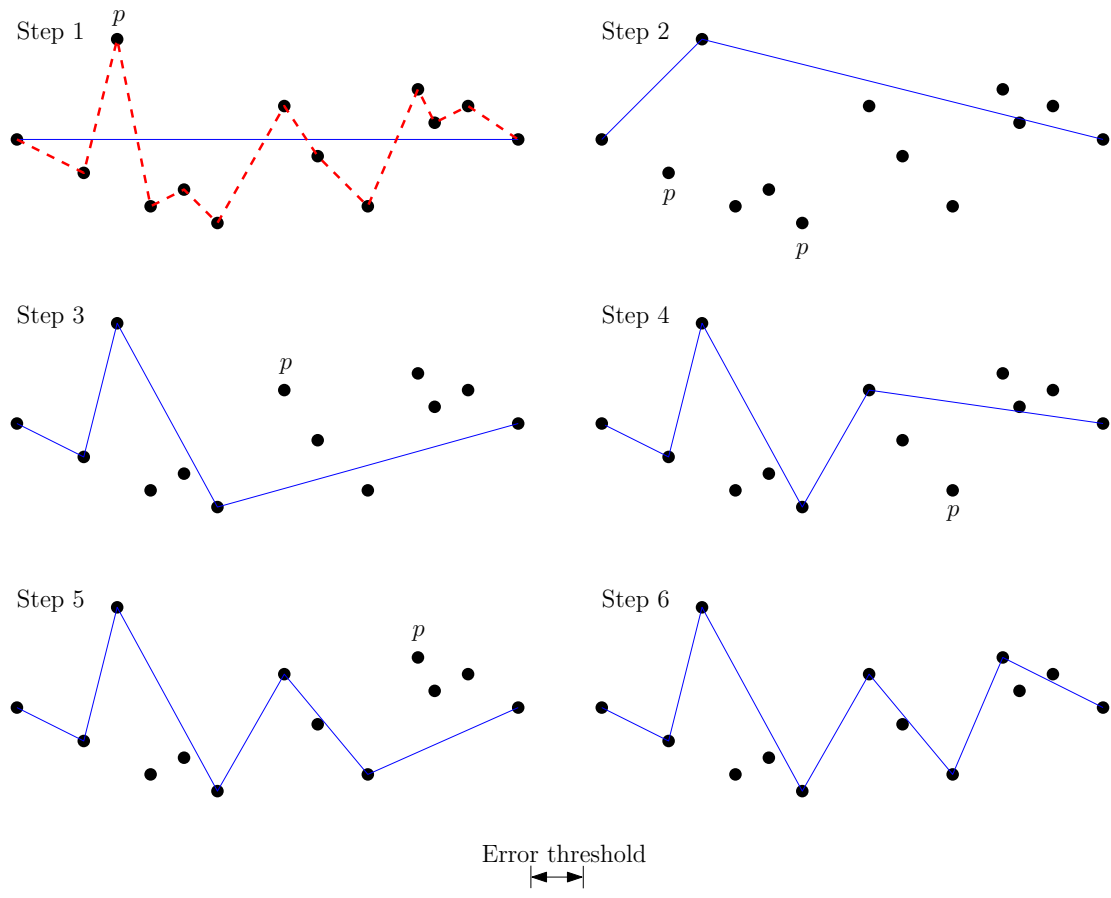


Figure 6.2: An example to illustrate the operation of the Douglas-Peucker technique for curve approximation. The input curve (dashed) is first approximated by a line joining the endpoints. At each step, the point with maximum distance to the approximate curve (indicated as p) is identified over each interval of the approximation, and such points are added to the approximate curve. The procedure iterates until all points are within the maximum acceptable error threshold of the approximate curve.

6.2.4 Chain Coding Scheme

A different approach that might be used to approximate the curve is to have a preset number of directions in which the approximating line may move, such as the four cardinal directions. Thus, moving on a grid, you can define your approximation by providing the sequence of the directional moves defining a path which approximates your curve. This technique is referred to as a chain coding scheme [77, 78]. The problem with a chain coding scheme in our application is that we are not guaranteed to get any reduction in the number of points required to represent our curve.

6.2.5 Two-Strip Solution

The Two-Strip solution is a form of error tolerance-based approximation, but there are enough variants of this algorithm that it warrants further discussion. It was first presented as a method of segmenting continuous curves by Reumann and Witkam [138], although there have been a number of further works with similar concepts (e.g. [44, 92, 106, 107, 132, 139]). The basic technique can be envisioned by having two strips parallel to the original curve at a distance of ε on each side. If the new approximate curve is contained within the two strips, then the approximation is considered an ε -approximation of the original curve (the distance is measured as the Hausdorff distance). This is simplified by stating that each vertex p_i of the original curve has a disk of radius ε around it, $C(p_i, \varepsilon)$, and the new approximating path P' must intersect each of these disks in order.

A hierarchical version of the two-strip solution was presented by Ballard [16], which he coined *Strip Trees*. Essentially, he uses the Douglas-Peucker algorithm to find pivot vertices (refer to Figure 6.2 for an illustration), and a hierarchical decomposition of the line segments is built by progressing through the execution of the algorithm. By proceeding until all vertices are within distance ε of the approximating curve, this algorithm will find a curve which solves the two-strip problem, although there is no approximation guarantee in terms of minimizing the number of vertices used in the approximating curve.

6.2.6 Error Metrics

There are several interesting error metrics that are commonly used that may be practical for our application. Suppose that we wish to optimize the approximating curve. One optimization scenario is that given an error metric and a threshold, one wishes to minimize the number of segments in the approximating curve (the min-# problem). The inverse is to be given the number of desired line segments, and to minimize the error of the approximating curve (the min- ε problem) [33]. There are straightforward techniques for solving each of these problems optimally in $O(n^2 \log n)$ for the L_∞ metric, except the min-# problem with the line-based L_∞ metric, which requires $O(n^2 \log^2 n)$ time in the best-known technique [97].

min-# Metrics

Perez and Vidal [131] favour a sum of squares measure. Given a curve represented by sequence of points p_i, \dots, p_j which is approximated by the straight line $\overline{p_i, p_j}$, they give us the L_2 metric as follows:

$$\Delta(i, j) = \sum_{k=i+1}^{j-1} ((a_{i,j} + b_{i,j}x_k) - y_k)^2,$$

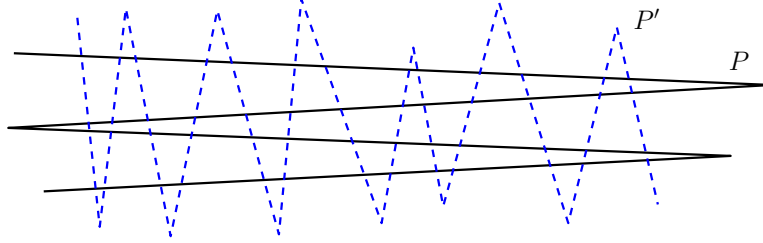


Figure 6.3: The dashed polygonal curve P' is an approximation of the solid polygonal curve P . P' is a poor approximation visually and it has a large distance from P in terms of the Fréchet metric (i.e. $F(P, P')$), but the Hausdorff distance between the curves, $H(P, P')$, is small. This figure is adapted from [7].

where $a_{i,j} = y_j - b_{i,j}x_i$ and $b_{i,j} = (y_j - y_i) / (x_j - x_i)$.

Perez and Vidal [131] presented a dynamic programming algorithm which finds the globally optimal solution with respect to almost any well defined error measure in $O(mn^3)$ time in general, where n is the number of points in the original curve, and m is the number of points in the approximating curve. For error measures that allow incremental computation (such as the square of the Euclidean distance), the problem can be solved in $O(mn^2)$ time.

It is likely that the path with the minimum possible number of line segments for a given approximation ε to a line does not use the original vertices. This problem is solved in a similar fashion to the two-strip solution discussed previously, where we place a disk of radius ε centred at each vertex of the line to be approximated, and then we seek the path with the minimum number of line segments which intersects all of the disks in order. This can be considered a generalization of the min-# problem discussed earlier. Guibas et al. [87] studied this problem for polygonal chains, and their solution entails finding a minimum link path which intersects each disk. They use an ordered stabbing technique [60] to find a path which satisfies the ε approximation of the original curve under the Hausdorff metric, as well as an ε approximation according to the Fréchet metric. This latter measure is colloquially known as the dog on a leash measure, since it can be imagined that a man walks one path and the dog walks the other, while the dog is attached to a leash of length ε . If both the man and dog can walk along their paths with varying but non-negative speeds, then the distance between the paths is less than or equal to ε by the Fréchet measure. Formally, given two polygonal chains $P : [0, p] \rightarrow V$ and $P' : [0, p'] \rightarrow V$, the decision version of the Fréchet distance is [7]:

$$F_\varepsilon(P, P') := \{(i, j) \in [0, p] \times [0, p'] \mid \text{dist}(P(i), P'(j)) \leq \varepsilon\},$$

where $\text{dist}(p, q)$ is the Euclidean distance from p to q . We can also express the general measure formally [59]:

$$F(P, P') = \min_{f:[0,1] \rightarrow P, g:[0,1] \rightarrow P'} \max_t \text{dist}(f(t), g(t)),$$

where f and g are continuous nondecreasing functions which define the curves P and P' over the interval $[0, 1]$.

The Fréchet metric is preferable in some contexts to the Hausdorff distance because it is possible for the Hausdorff distance to be small although the approximation appears poor, as shown in Figure 6.3. However, the Fréchet metric is not intended for closed curves, and a modified version must be used for such applications [7]. The Hausdorff distance is standard for expressing the distance between polygons as a measure of their similarity [6, 32, 71, 86, 114, 141].

Other metrics of possible interest:

- *Geodesic width* [59] - Similar to the Fréchet metric, but uses the shortest link path which does not intersect P or P' as the distance measure rather than standard Euclidean distance:

$$W(P, P') = \min_{f:[0,1] \rightarrow P, g:[0,1] \rightarrow P'} \max_t \text{dist}_E(f(t), g(t)),$$

where $\text{dist}_E(p, q)$ is the length of the shortest path from p to q which avoids P and P' , and f and g are again continuous nondecreasing functions.

- *Link width* [59] - Now we seek the path which does not intersect P or P' with the smallest number of segments:

$$L(P, P') = \min_{f:[0,1] \rightarrow P, g:[0,1] \rightarrow P'} \max_t \text{dist}_L(f(t), g(t)),$$

where $\text{dist}_L(p, q)$ is the minimum number of segments in a path from p to q which avoids P and P' . f and g are continuous functions, but are not required to be non-decreasing for this measure.

- *Eggleston measure* [86] - Similar to the Hausdorff metric, but the sum of the two terms is taken rather than the maximum:

$$d_E(P, P') = \sup_{p \in P} \inf_{p' \in P'} \text{dist}(p, p') + \sup_{p' \in P'} \inf_{p \in P} \text{dist}(p, p').$$

- *Area* - Defined over polygons P and P' , we may measure the area of the region $P \setminus P'$.

6.2.7 Constrained Approximation of Polygonal Curves

We desire an approximation that lies entirely to one side of the original curve. Zhang and Tian [167] provide the only study of this kind on polygonal curves that we are aware of, and their application is closely related to our own. Their solution entails the selective removal of points using the Douglas-Peucker approach, so that points are removed only if the new approximation lies to the deep side of the contour line that is being approximated. While their technique establishes precedent, the amount of data reduction achieved with their technique relative to the original algorithm is unclear from their examples [167, Figures 3-5], and no formal analysis is provided in their work.

There have been studies in the past that have placed constraints on the approximation so that the approximating curve does not violate some other properties of the data. Estkowski and Mitchell [65] studied this problem in the context of GIS data, where adjacent polygonal curves representing features of the terrain are approximated. They wish to ensure that the relative positions of the features remain consistent following approximation, and that the approximated curves do not introduce intersections between adjacent curves. Such concerns did not arise with the techniques discussed previously where curves are studied in isolation. This problem is shown to be in the hardness class *MIN PB-complete* when the output must be composed of a subset of the input points (i.e. no Steiner points are permitted) [65]. Therefore, an approximation technique based on heuristics is proposed to solve the problem, and they provide experimental evidence of the efficacy of their technique. Also of interest is a constrained Douglas-Peucker technique in which it is ensured that there are no self-intersections [161], but this property is implicit in the previous technique. For a more thorough review of constrained approximation techniques, see [109, Section 5.6].

6.2.8 Constrained Approximation of Polygons

We now seek methods for approximating closed polygons. In the context of our bathymetric navigation application, we wish to approximate a polygon P with a simpler polygon Q , where Q either fully contains P or is fully contained by P . A well known solution to the first version of the problem is the convex hull. Finding a polygon that is contained in P is more difficult. If we were unconcerned with the monotonicity constraint, we may apply techniques identical to those discussed for polygonal curves in the previous section, and an optimal solution can be found with a constant amount of extra work [103].

We divide the problem of approximating polygons into two broad classes: inclusion problems seek an approximation contained in the original polygon, while enclosure problems determine an approximation that contains the original polygon. Formally, let \mathcal{P} and \mathcal{Q} be classes of polygons and let μ be a function on polygons such that for polygons P and Q , $Q \subseteq P \Rightarrow \mu(Q) \leq \mu(P)$. Chang and Yap [31] define the inclusion and enclosure problems as:

- $Inc(\mathcal{P}, \mathcal{Q}, \mu)$: Given a polygon $P \in \mathcal{P}$, find a polygon $Q \in \mathcal{Q}$ included in P , maximizing $\mu(Q)$.
- $Enc(\mathcal{P}, \mathcal{Q}, \mu)$: Given a polygon $P \in \mathcal{P}$, find a polygon $Q \in \mathcal{Q}$ enclosing P , minimizing $\mu(Q)$.

The best known enclosure problem is the convex hull, which we may state formally as $Enc(\mathcal{P}_{\text{simple}}, \mathcal{P}_{\text{con}}, \text{area})$, where $\mathcal{P}_{\text{simple}}$ is the family of simple polygons and \mathcal{P}_{con} is the family of convex polygons. Given a convex polygon P as input, many problems are tractable in linear time: $Enc(\mathcal{P}_{\text{con}}, \mathcal{P}_3, \text{area})$ [129], where \mathcal{P}_3 is the set of triangles, $Enc(\mathcal{P}_{\text{con}}, \mathcal{P}_3, \text{perimeter})$ [18], and $Enc(\mathcal{P}_{\text{con}}, \mathcal{P}_{\text{par}}, \text{area})$ [143], where \mathcal{P}_{par} is the family of parallelograms. For general k -gons, $Enc(\mathcal{P}_{\text{con}}, \mathcal{P}_k, \text{area})$ can be solved in $O(kn + n \log n)$ time [4].

Perhaps the best known inclusion problem is the potato-peeling problem of Chang and Yap [31], defined as $Inc(\mathcal{P}_{\text{simple}}, \mathcal{P}_{\text{con}}, \text{area})$. The ‘potato’ of the potato peeling problem is formally known as the maximum area convex subset (MACS) [32], which is the largest area convex polygon contained in P . There is an $O(n^7)$ time algorithm for this problem, where n is the number of vertices of P , and an $O(n^6)$ time algorithm when the measure is the perimeter, i.e. $Inc(\mathcal{P}_{\text{simple}}, \mathcal{P}_{\text{con}}, \text{perimeter})$ [31]. The problem of finding the triangle of maximal area included in a convex polygon, $Inc(\mathcal{P}_{\text{con}}, \mathcal{P}_3, \text{area})$, can be solved in linear time [52]. The generalization of this problem to any k -gon can be solved in time $O(kn + n \log n)$ [3]. If the input polygon is not restricted to be convex, $Inc(\mathcal{P}_{\text{simple}}, \mathcal{P}_3, \text{area})$ can be found in time $O(n^4)$ [120].

The inclusion and enclosure problems can also be formulated as minimizing or maximizing a measure $d(P, Q)$, which we call a d-Core of P .

Definition 6.2. The d-Core Problem: *Given a simple polygon P , the d-Core problem on P is to determine a convex polygon Q , such that $Q \subseteq P$ and $d(P, Q)$ is minimized, where $d(P, Q)$ is a difference metric on P and Q . We call Q a d-Core of P .*

Note that in the case when $\mu(Q)$ is the area, maximizing or minimizing $\mu(Q)$ for the inclusion and enclosure problems, respectively, is equivalent to minimizing the difference in areas ($d(P, Q) = |\mu(P) - \mu(Q)|$). Both the inclusion and enclosure problems using the Hausdorff distance as a measure were studied by Lopez and Reisner [114], who present polynomial-time

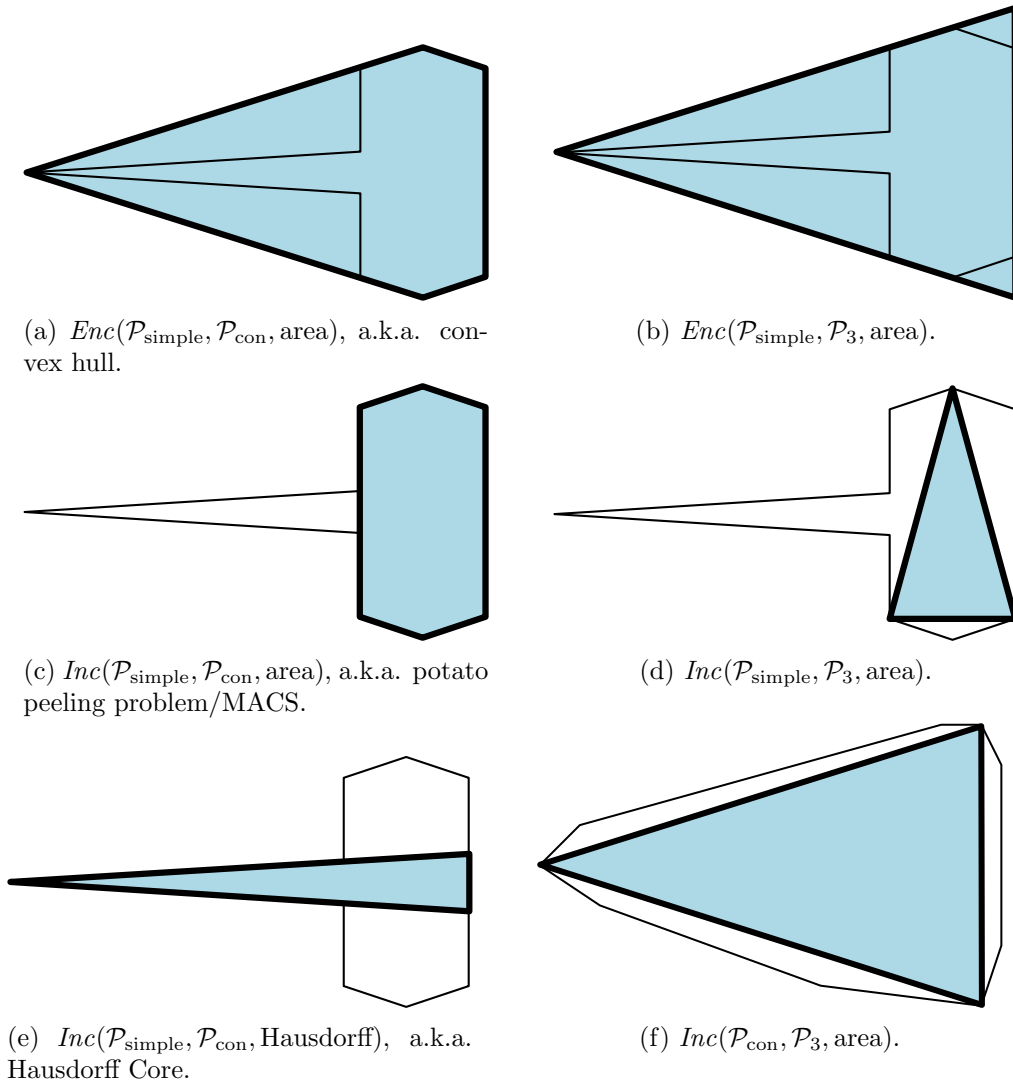


Figure 6.4: We illustrate some of the various inclusion and enclosure problems for approximating polygons.

algorithms to approximate a convex polygon minimizing the Hausdorff distance to within an arbitrary factor of the optimal (i.e. the min- ϵ version of the problem). Since the input polygon is convex, the approximating solution is restricted to contain some maximum number of vertices. In the same work, the authors also studied the min- $\#$ version of the problem, where the goal is to minimize the number of vertices of the approximating polygon, given a maximum allowed error. For this setting, they show that the inclusion and enclosure problems can be approximated to within one vertex of the optimal in $O(n \log n)$ time and $O(n)$ time, respectively.

The work in the following chapters addresses the inclusion problem where the objective is to minimize the Hausdorff distance to a convex approximating polygon given a simple (not necessarily convex) polygon as input, i.e. $Inc(\mathcal{P}_{\text{simple}}, \mathcal{P}_{\text{con}}, \text{Hausdorff})$. Chassery and Coeurjolly [32] addressed this problem first, and they presented an algorithm that returns a Hausdorff Core for the case when the Euclidean 1-centre is contained in the input polygon P . The algorithm shrinks

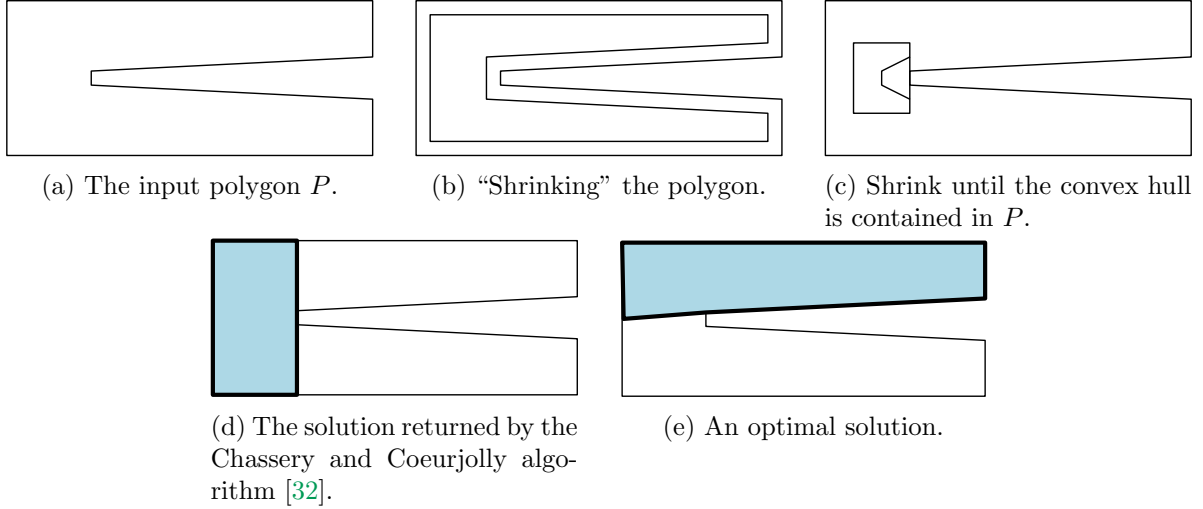


Figure 6.5: Chassery and Coeurjolly algorithm counterexample.

the input polygon P until its convex hull is contained in the original P . If the shrunken polygon P' is not convex, then the convex hull of P' contains a vertex of P which lies on an edge e of P' . e is used as a cutting line upon P to obtain a new polygon P_1 to be shrunk. The procedure is repeated to obtain P'_i from P_i until P'_i is convex. If the Euclidean 1-centre of P is not contained in P , it is possible to construct examples where this algorithm would not return a Hausdorff Core of P , as shown in Figure 6.5.

6.2.9 LP-type Problems

Certain settings for optimization problems using the Hausdorff distance metric belong to the class of problems known as *LP-type problems*. LP-type problems were formalized by Sharir and Welzl [146] to provide a general framework for linear programming problems with n constraints and d variables so that they may be solved in expected $O(d^2 2^d n)$ time, which translates into a linear time algorithm for many of the types of problems that we are interested in. To apply their algorithm to a problem P , it simply must be demonstrated that two properties hold for P . In this section, we follow the discussion as presented in [146].

A linear program with n constraints and d variables may be expressed geometrically as a set H of n half-spaces in \mathbb{R}^d space. The optimal solution to the linear program is a point $s \in \cap_{h \in H} h$ which minimizes the objective function. Given a function $w(H)$ which computes the optimal solution for H , we have that $s = w(H)$. Now the two constraints are as follows:

$$F \subseteq G \subseteq H \Rightarrow w(F) \leq w(G) \tag{6.1}$$

$$w(F) = w(G) \left. \begin{array}{l} F \subseteq G \subseteq H \\ h \in H \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} w(F+h) > w(F) \\ \leftrightarrow \\ w(G+h) > w(G) \end{array} \right. \tag{6.2}$$

This technique is often used for problems that are similar to the Hausdorff Core problem, such as for computing the Euclidean 1-centre of a set of points (which is equivalent to determining

the smallest enclosing ball of a set of points). However, the Hausdorff Core problem is not an LP-type problem, due to the requirement that the solution Q be contained within P . If P is not convex, then the search space in the polygon cannot be described by a set of linear constraints. If P is convex, the constraints are satisfied, but the Hausdorff Core problem is also trivial (since $Q = P$ in this case).

6.2.10 Davenport-Schinzel Sequences

Davenport-Schinzel sequences are used for the analysis of the combinatorial complexity of the upper or lower envelope of a set of functions, usually specifically univariate functions. They are eponymously named for those most commonly credited with their development [48], where the application was calculus. Davenport-Schinzel sequences were shown to have applications for computational geometry problems by Atallah [12], where the dynamic convex hull problem was studied in a setting where the positions of points are described by functions dependent on time. The authoritative treatment of Davenport-Schinzel sequences (especially with respect to computational geometry) is given by Sharir and Agarwal [145], and here we outline relevant details from Chapter 1 of their book.

Definition 6.3. Davenport-Schinzel Sequence [145, Def. 1.1]: *Let n, s be positive integers. A sequence $U = \langle u_1, \dots, u_m \rangle$ of integers is an (n, s) Davenport-Schinzel sequence if it satisfies the following conditions:*

1. $1 \leq u_i \leq n$ for each i ;
2. For each $i < m$ we have $u_i \neq u_{i+1}$;
3. There do not exist $s + 2$ indices $1 \leq i_1 < i_2 < \dots < i_{s+2} \leq m$ such that $u_{i_1} = u_{i_3} = u_{i_5} = \dots = a$, $u_{i_2} = u_{i_4} = u_{i_6} = \dots = b$, and $a \neq b$.

Given the sequence U , they use s to denote the *order* of U , and U is composed of n symbols. The bound that we are interested in, denoted as $\lambda_s(n)$, is defined by the maximum value of m (the cardinality of U), given that U is a (n, s) Davenport-Schinzel sequence. This value may be used to describe the maximum combinatorial complexity of the upper or lower envelope of a set of univariate functions, by first establishing a bound on the maximum number of intersection points between any pair of functions.

Given a family of functions $\mathcal{F} = \{f_1, \dots, f_n\}$ which are defined over a common interval I , the upper envelope of the functions (Figure 6.6) is described by the maximum value in the set of functions when evaluated at each point in the interval I . We call a maximal connected interval of a function on the upper envelope a *portion* of the upper envelope. U is defined by the sequence of the indices of the functions which compose the portions of the upper envelope. Formally, for any $x \in I$, the upper envelope $E_{\mathcal{F}}(x)$ is expressed as

$$E_{\mathcal{F}}(x) = \max_{i \in \{1, \dots, n\}} f_i(x).$$

While there exist tight bounds for $s = 1$ and $s = 2$, we are interested in the upper bound provided by the following lemma:

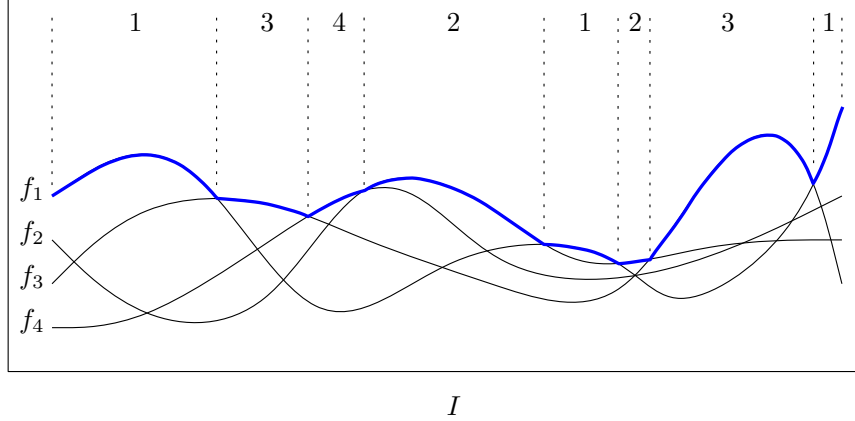


Figure 6.6: Given the set of functions $\{f_1, f_2, f_3, f_4\}$ the upper envelope (drawn in bold) is the maximum value among all functions at each x-coordinate. The sequence $U = \langle 1, 3, 4, 2, 1, 2, 3, 1 \rangle$ is given by the index of the function which is maximum for each such maximal interval over I .

Lemma 6.1. [145, Lemma 1.12]

$$\lambda_s(n) \leq \frac{sn(n-1)}{2} + 1.$$

Furthermore, the value of $\lambda_s(n)$ has a lower bound. The actual lower bounds are more complex, as they are based on the parity of s and n , and so we simply provide a lower bound for the lower bounds:

Lemma 6.2. [140, Equations 3.8, 3.9]

$$\lambda_s(n) \geq \frac{sn(n-1)}{2} - \frac{2n^3 + 9n^2 - 38n + 6}{12}.$$

6.2.11 The Constrained Euclidean 1-Centre

If the Hausdorff Core problem were defined so that Q , the Hausdorff Core solution, must be a single point q , then the problem becomes equivalent to the constrained Euclidean 1-centre problem. In the general unconstrained setting, the *1-centre* of a polygon P (also known as the Euclidean centre) is the point c that minimizes the maximum distance from c to any point in P . This (unconstrained) setting is an LP-type problem, as discussed in Section 6.2.9. In this work, we are interested in the 1-centre inside P , also known as the *constrained Euclidean centre* or *constrained Euclidean 1-centre*, which is a point p_{1c} contained in P that minimizes the maximum distance from p_{1c} to any point in P . Although the unconstrained 1-centre is unique, this is not necessarily true for the constrained version, as shown in Figure 6.7. Throughout the rest of this work, when we refer to a 1-centre, we specifically mean a constrained Euclidean 1-centre. We write p_{1c} to represent a constrained Euclidean 1-centre of P .

The constrained Euclidean centre problem was solved by Bose and Toussaint [22, Algorithm 1], and we provide a sketch of their technique here. The first fundamental observation is that the maximum distance from any point q to a polygon P is realized at a vertex of P , and so the problem is reduced to finding a point in P which minimizes the maximum distance to any vertex

of P . One possibility for the solution is the Euclidean 1-centre, and this is found by determining the centre of the minimum enclosing circle on P . If the 1-centre is in P , then the problem is solved. Otherwise, the farthest point Voronoi diagram of P (call it $\text{VD}(P)$) is computed, and the set V_c of the vertices of $\text{VD}(P)$ contained in P is found. Next, the set of points I_c is computed, where each point in I_c is an intersection point between P and $\text{VD}(P)$. Next, each edge $e_i \in P$ is partitioned into maximal segments such that the farthest point in P from any such segment is unique. For each segment, the point in the segment closest to the farthest point in P is computed, and if this point is not an endpoint of the segment, it is added to a set E_c . Finally, the vertices of P , call them P_c , are merged with these sets of points to produce the set $C = V_c \cup I_c \cup E_c \cup P_c$, and at least one point $c \in C$ is a constrained Euclidean 1-centre, which may be found by brute force. Using this method, a constrained Euclidean 1-centre of a polygon P with n vertices can be computed in $O(n \log n + k)$ time, where k is the number of intersections between P and the furthest point Voronoi diagram of the vertices of P (for simple polygons $k \in O(n^2)$) [22].

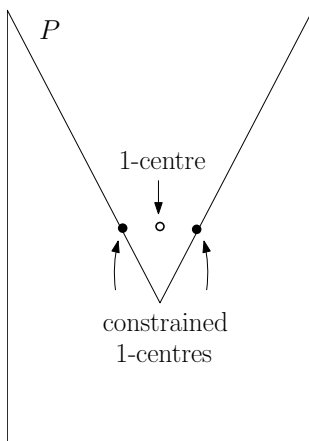


Figure 6.7: An example where the constrained Euclidean 1-centres (the solid points) are not unique, and are distinct from the unconstrained Euclidean 1-centre (the hollow point).

6.3 The Hausdorff Core

We begin our analysis by formalizing the Hausdorff Core problem, and establishing some fundamental properties of the problem. In Chapter 7, we describe an algorithm for solving the Hausdorff Core problem optimally on polygons with a single reflex vertex. In Chapter 8, we provide a parameterized algorithm for the general problem on simple polygons.

6.3.1 Definitions

Recall the definition of Hausdorff distance:

$$H(P, Q) = \max\left\{\sup_{p \in P} \inf_{q \in Q} \text{dist}(p, q), \sup_{q \in Q} \inf_{p \in P} \text{dist}(p, q)\right\},$$

When P and Q are polygons in the plane and $\text{dist}(p, q)$ denotes the Euclidean distance between points p and q , $H(P, Q)$ corresponds to the Hausdorff distance between sets P and Q . We define the corresponding d -core (see Definition 6.2) as the *Hausdorff Core* of P . There is not necessarily

a unique Hausdorff Core solution; our objective is to determine any optimal solution. We consider both the minimization and decision versions of the Hausdorff Core problem for a given simple polygon P . Recall from Definition 6.1 that the minimization version of the Hausdorff Core problem is to determine a Hausdorff Core of P , i.e. a polygon Q covered by P where $H(P, Q)$ is minimized. The decision version is formalized below.

Definition 6.4. Decision Version of the Hausdorff Core Problem: *Given a simple polygon P and a non-negative integer k , determine whether there exists a convex polygon Q contained in P such that $H(P, Q) \leq k$.*

6.3.2 Hausdorff Core Properties

In this section we make observations regarding the properties of polygons, convex polygons, and the Hausdorff distance in the context of the Hausdorff Core problem. These observations will be useful in the following chapters when establishing our main results.

The first property is that given a polygon P and a convex polygon Q inside P , it suffices to minimize the maximum distance from points on the convex hull of P to polygon Q to obtain a Hausdorff Core Q (i.e., the distance from points $q \in Q$ to P need not be considered).

Lemma 6.3. *Given any simple polygon P and any convex polygon Q contained in P ,*

$$\max_{p \in P} \min_{q \in Q} \text{dist}(p, q) \geq \max_{q \in Q} \min_{p \in P} \text{dist}(q, p)$$

Therefore,

$$H(P, Q) = \max_{p \in P} \min_{q \in Q} \text{dist}(p, q).$$

Furthermore, the Hausdorff distance between P and Q is defined by the distance from points on the convex hull of P to Q :

$$H(P, Q) = H(\text{CH}(P)_V, Q).$$

Proof. Suppose $H(P, Q) = k_{\text{OPT}}$, and take the convex hull of P to obtain $\text{CH}(P)$. Now we identify two consecutive vertices of $\text{CH}(P)$, call them p_i and $p_{i'}$ (Figure 6.8). By the definition of a convex hull, we know that all vertices of P lie in one of the two half-planes defined by the line incident upon p_i and $p_{i'}$, call this half-plane h^P . For ease of discussion, we rotate everything so that h^P is equivalent to the $y \leq 0$ half-plane of the Cartesian plane. Since P contains Q , h^P also contains Q .

Now consider two points q_j and $q_{j'}$ on the upper hull of Q , each inside a disk of radius k_{OPT} centred at p_i and $p_{i'}$, respectively. Since Q is convex, the edges of Q must lie on or above the line incident upon q_j and $q_{j'}$, call it $\overleftrightarrow{q_j q_{j'}}$, and also remain below the boundary of P since P contains Q . Therefore, for any point on Q in this range, there is a point in P at most distance k_{OPT} above it. The maximum distance that a point in Q may be from points in P is also k_{OPT} in this range. If $\overleftrightarrow{q_j q_{j'}}$ is parallel to the x-axis, then it is possible that the maximum distances are equal: $\max_{p \in P} \min_{q \in Q} \text{dist}(p, q) = \max_{q \in Q} \min_{p \in P} \text{dist}(p, q)$, and this may be up to k_{OPT} . If $\overleftrightarrow{q_j q_{j'}}$ is not horizontal, then the distance $\max_{p \in P} \min_{q \in Q} \text{dist}(p, q)$ is maximized at one of the vertices of P , and so on this interval $H(P, Q) = \max_{p \in \text{CH}(P)} \min_{q \in Q} \text{dist}(p, q)$.

The lemma follows, because this argument may be applied to any consecutive pair of vertices of the convex hull of P , and points of Q may be chosen so that the entire polygon is considered. \square

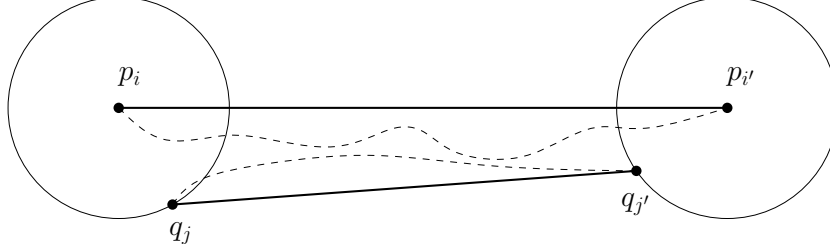


Figure 6.8: Illustration of Lemma 6.3.

$H(P, Q)$ is determined by the vertices of P that lie on the convex hull of P , however all vertices and edges of P must be considered to determine whether Q is contained in P . Therefore, the decision version of the Hausdorff Core problem with parameter k may be redefined as follows: we consider disks of radius k centred at vertices $CH(P)_V$ and ask whether there exists a convex polygon Q such that it intersects all such disks. Let $C(p, k)$ denote a disk of radius k centred at p . We refine the nature of the Hausdorff Core problem based on the following corollary:

Corollary 6.1. *Given a simple polygon P and a convex polygon Q contained in P ,*

$$H(P, Q) \leq k \Leftrightarrow \forall p \in CH(P)_V, C(p, k) \cap Q \neq \emptyset.$$

Finally, we wish to determine a point contained in Q .

Lemma 6.4. *Given a simple polygon P and a convex polygon Q contained in P , at least one point in the set P_V can be contained in Q if Q is a Hausdorff Core of P .*

Proof. In the lemma, we write *can be contained* because it is possible for a solution to be optimal and not contain a point from this set, but we show that any such solution may be expanded to include a vertex of P . An example of such a scenario is shown in Figure 6.9.

Suppose there exists a polygon Q which has $H(P, Q) = k$ where $P_V \cap Q = \emptyset$. We can always expand Q to create a new polygon Q' , where $Q \subset Q'$, and so $H(P, Q') \leq H(P, Q)$. Note that since we assumed Q is an optimal solution, in fact $H(P, Q') = H(P, Q)$. One such algorithm is as follows:

1. Choose an arbitrary edge $e_Q = (v_i, v_j)$ of Q .
2. Choose one of the vertices arbitrarily, say without loss of generality that v_i is chosen. Let $e'_Q = (v_i, v_{j'})$ be the other edge of Q incident upon v_i (it is possible that $e_Q = e'_Q$). Expand e_Q by moving v_i away from v_j along the unique line defined by e_Q until either:
 - (a) one of the edges of Q touches a vertex of P ;
 - (b) moving v_i further would cause Q to lose convexity; or
 - (c) v_i encounters an edge of P , call it e_P .
3. If case 2(a) occurs, we are done. If case 2(b) occurs, we merge e'_Q with the other edge incident upon $v_{j'}$, and then return to step 2. If we arrive at case 2(c), we introduce a new vertex v' on Q at v_i , and edges $e_1 = (v', v_i)$ and $e_2 = (v', v_{j'})$. Note that e_P and e_1 are collinear and e_1 initially has zero length. We move v' along e_P (in a direction which keeps Q simple), again according to the rules in the previous step. When one of the conditions is met, one of the vertices of P must be incident upon Q .

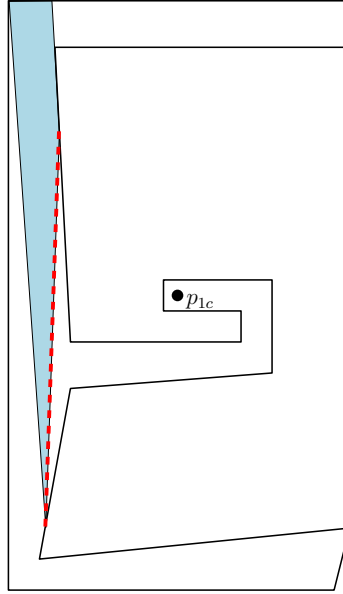


Figure 6.9: An optimal Hausdorff Core solution to this polygon is shown as the thick dashed line in the figure. This line may be grown in a number of ways into a convex polygon which remains contained inside the original polygon, while including a vertex of P on the boundary. The shaded polygon is an example of one such possibility. This is also an example of a polygon in which the Euclidean 1-centre p_{1c} is contained in the polygon, although it is not contained in any optimal Hausdorff Core solution.

□

In the following chapter, we present an algorithm for computing the Hausdorff Core of a simple polygon with a single reflex vertex. We then present a parameterized algorithm for the problem upon general simple polygons in Chapter 6.

An Exact Algorithm for the Hausdorff Core of a Single Reflex Vertex Polygon

Our objective in this chapter is to find an optimal Hausdorff Core for a polygon which has a single reflex vertex¹. The Hausdorff Core of a convex polygon P is P itself, and a polygon with a single reflex vertex is perhaps the simplest non-trivial case. We show that the solution consists of cutting the polygon P with a line which intersects the reflex vertex, and that the difficulty of the problem lies in finding where to place the cutting line, as shown in Figure 7.1.

Theorem 7.1. *There exists an exact algorithm for the Hausdorff Core problem on polygons with a single reflex vertex which runs in $O(n^3)$ time.*

We begin by establishing that the reflex vertex p_r is on ∂Q , i.e. the boundary of the optimal solution, since the cutting line ℓ is incident upon p_r .

Lemma 7.1. *One edge of a Hausdorff Core Q will be a segment of a line ℓ which intersects the reflex vertex p_r .*

¹Elements of this chapter have appeared in [74].

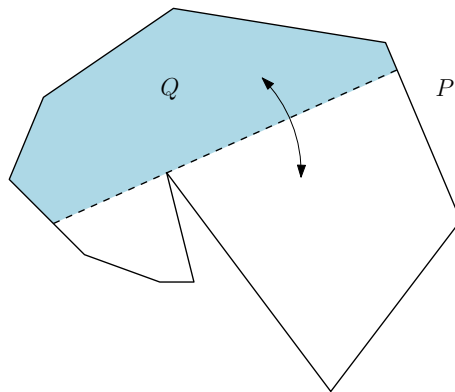


Figure 7.1: To find the Hausdorff Core of a polygon with a single reflex vertex, we simply need to determine where to place a cut line which intersects the reflex vertex.

Proof. Suppose that this is not the case. ℓ , the cutting line on P (which defines an edge of Q), intersects ∂P at two points p_{ℓ_1} and p_{ℓ_2} , so that $\ell = \overleftrightarrow{p_{\ell_1}p_{\ell_2}}$. Choosing point p_{ℓ_1} without loss of generality, rotate ℓ about p_{ℓ_1} until it intersects p_r ; call the resulting polygon Q' . Q' remains convex because ℓ is a straight line defining the cut, and $Q \subseteq Q'$. Therefore, $H(P, Q) \geq H(P, Q')$. \square

7.1 Balancing a Line

Given that p_r is incident upon ℓ , we can regard the resulting partition of P , which is defined by ℓ , as being composed of three convex regions called Q_a, Q_b , and Q_c (see Figure 7.2). Note that one of Q_a or Q_c may be empty. We define these regions in counter-clockwise order around p_r so that the line segment $\overline{p_{\ell_1}p_r}$ defines an edge of Q_a , $\overline{p_{\ell_1}p_{\ell_2}}$ defines an edge of Q_b , and $\overline{p_r p_{\ell_2}}$ defines an edge of Q_c .

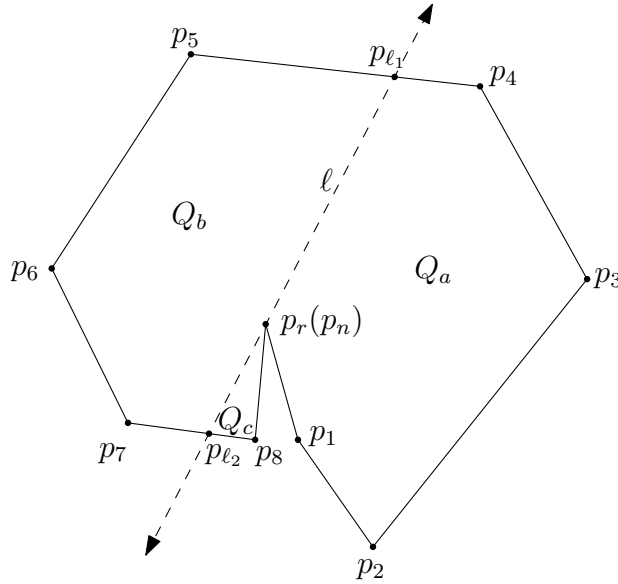


Figure 7.2: The line ℓ defines a cut through polygon P , which defines three convex regions Q_a, Q_b , and Q_c . Note that the closest point to p_2 in Q_b is at p_{ℓ_2} .

Lemma 7.2. *A polygon Q which is an optimal Hausdorff Core solution can always be defined by the region Q_b .*

Proof. Suppose that this is not the case, and instead, without loss of generality suppose Q is defined by Q_a . Choosing ℓ so that $\overline{p_r p_1}$ lies on ℓ results in a configuration where Q'_a is empty and $Q_a \subset Q'_b$ for all possible polygons Q_a . Therefore, $H(P, Q_a) \geq H(P, Q'_b)$. \square

7.2 Computing the Hausdorff Distance

In our problem, we are interested in measuring the distance from the vertices of the polygon P to the approximating polygon Q . We divide the vertices of P into two sets P_Q and $P_{\overline{Q}}$, where P_Q are the vertices of P in $P \cap Q$, and $P_{\overline{Q}} = P \setminus P_Q$.

Since the vertices of P_Q do not need to be considered for the Hausdorff distance measure (as they are contained within Q), the nature of the problem is reduced to finding the distance from the vertices of $P_{\overline{Q}}$ to the closest point in Q_b . The nearest point in Q_b to a vertex $p_i \in P_{\overline{Q}}$ lies either on the boundary of P or on the line segment $\ell \cap P$ (i.e., it is on ∂Q_b).

If the nearest point in ∂Q_b to a vertex $p_i \in P_{\overline{Q}}$ is on ∂P but not on ℓ , we address this case by pre-computing the critical point on P nearest to p_i . We illustrate this setting in Figure 7.3, and we describe the procedure for computing the point in Algorithm 7.1. Let P' be the region $CH(P) \setminus P$. To find the closest point, we look for a point p_{\min} on $\partial P'$ which is nearest to p_i , such that P' covers $\overline{p_i p_{\min}}$, while $\partial P'$ does not. Intuitively, it is a necessary condition that p_{\min} and p_i must lie on opposite sides of $\overleftrightarrow{p_r p_1}$, and $p_{\min} \in P'$. If the algorithm returns a point other than p_r , we add the point to the set considered by the Hausdorff Core algorithm. If the nearest

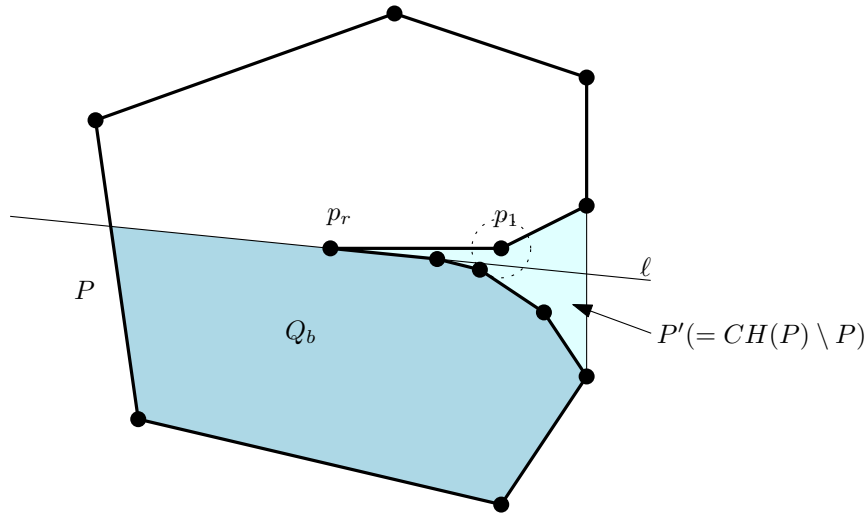


Figure 7.3: An example configuration where the closest point to p_1 in Q_b is not on ℓ . We can precompute the closest point to p_1 in this case. The radius of the dashed circle indicates the distance to the critical point.

point to p_i in Q_b is on $\ell \cap P$, then there are two cases to consider: either the nearest point on $\ell \cap P$ is somewhere on the interior of the line segment, or the nearest point lies at one of the two points in $\ell \cap \partial P$ (i.e. the endpoints of the line segment $\ell \cap P$).

If the first case is true, then the technique for measuring the distance is found using straightforward trigonometry. Suppose we are considering vertex p_i , and the nearest point on ℓ is p_ℓ . In this case the points $\{p_r, p_i, p_\ell\}$ define a right-angled triangle, and the distance between p_i and p_r is fixed since they are vertices of P . Therefore, the distance from p_i to ℓ is defined as $\text{dist}(p_i, p_\ell) = \text{dist}(p_i, p_r) \sin \theta$, where $\theta = \angle(p_i, p_r, p_\ell)$.

In the second case, describing the nearest point is slightly more difficult, as it is either p_{ℓ_1} or p_{ℓ_2} . For our purposes here we will assume without loss of generality that the nearest point is p_{ℓ_1} . We express $\text{dist}(p_i, p_{\ell_1})$ in terms of the intersection between the ray $\overrightarrow{p_r p_{\ell_1}}$ and ∂P . We formalize this case further in the next section.

Algorithm 7.1 FIND-MIN-P(p, p_r, P)

```
1: Input: A polygon  $P$  with reflex vertex  $p_r$ , a vertex  $p \in P$ .
2: Output:  $p_{\min}$ , the nearest point to  $p$  in  $P'$  such that  $\overline{pp_{\min}} \subset P'$  and  $\overline{pp_{\min}} \not\subset \partial P'$ .
3:  $d_{\min} \leftarrow \text{dist}(p, p_r)$ 
4:  $p_{\min} \leftarrow p_r$ 
5:  $P' \leftarrow CH(P) \setminus P$ 
6: for  $e = (p_1, p_2) \subset P'$  (i.e. for each edge in  $P'$ ) do
7:   if  $\overline{pp_1} \cap P' \neq \emptyset$  or  $\overline{pp_2} \cap P' \neq \emptyset$  then
8:      $p' \leftarrow$  nearest point on  $\overline{p_1p_2}$  to  $p$ 
9:     if  $p' \in e \setminus \{p_1, p_2\}$  and  $\text{dist}(p, p') < d_{\min}$  then
10:        $d_{\min} \leftarrow \text{dist}(p, p')$ 
11:        $p_{\min} \leftarrow p'$ 
12:     else
13:       if  $\min\{\text{dist}(p, p_1), \text{dist}(p, p_2)\} < d_{\min}$  then
14:          $d_{\min} \leftarrow \min\{\text{dist}(p, p_1), \text{dist}(p, p_2)\}$ 
15:         if  $\text{dist}(p, p_1) < \text{dist}(p, p_2)$  then
16:            $p_{\min} \leftarrow p_1$ 
17:         else
18:            $p_{\min} \leftarrow p_2$ 
19:         end if
20:       end if
21:     end if
22:   end if
23: end for
24: return  $p_{\min}$ 
```

7.3 Finding the Optimal Solution

Our algorithm consists of sweeping ℓ through P by rotating it about p_r to discover the angle at which the distance between vertices of P_Q and Q_b is minimized. For convenience of exposition, we rotate P about p_r by the angle required to align $\overline{p_r p_1}$ horizontally; i.e. $p_1 = (p_r.x + \text{dist}(p_r, p_1), p_r.y)$, where $p.x$ is the x-coordinate of p . We perform the sweep in a counterclockwise fashion, such that $\theta_1 = 0$ is the initial state when ℓ is incident upon p_1 , and $\theta_{n-1} < \pi$ is the final state when ℓ is incident upon p_{n-1} .

7.3.1 Expressing the Distance to Each Point

First, we use Algorithm 7.1 on each point of P to identify intervals of θ for which the nearest point to the solution is on ∂P . These intervals, if they exist, will comprise an interval at either the upper or lower end of the range of possible values for θ . We construct the arrays Θ_{\min} and Θ_{\max} with Algorithm 7.2, which contain the minimum or maximum values of θ for each point, outside of which the distance is the fixed distance stored in the array D .

Algorithm 7.2 Θ -ARRAYS(p_r, P)

```

1: Input: A polygon  $P$  with reflex vertex  $p_r$ .
2: Output: Arrays  $\Theta_{\min}[i], \Theta_{\max}[i]$  contain the minimum and maximum values of  $\theta$  for  $p_i$ ,
   outside of which  $H(p_i, Q) = D[i]$ .
3:  $\theta_{n-1} \leftarrow \pi - \angle(p_{n-1}, p_r, p_1)$ 
4: for  $i := 1$  to  $|P_V|$  do
5:    $\Theta_{\min}[i] = 0, \Theta_{\max}[i] = \theta_{n-1}$ 
6: end for
7: for each  $p_i \in P$  do
8:    $p_{\min} = \text{FIND-MIN-P}(p_i, p_r, P)$  // Algorithm 7.1
9:    $\theta = \angle(p_1, p_r, p_{\min})$ 
10:   $D[i] = \text{dist}(p_i, p_{\min})$ 
11:  if  $p_r \neq p_{\min}$  and  $p_i.y > 0$  then
12:     $\Theta_{\max}[i] = \theta$ 
13:  end if
14:  if  $p_r \neq p_{\min}$  and  $p_i.y < 0$  then
15:     $\Theta_{\min}[i] = \theta$ 
16:  end if
17: end for
18: return  $\Theta_{\min}, \Theta_{\max}, D$ 

```

Outside of the intervals defined in the arrays Θ_{\min} and Θ_{\max} , we need to compute the distance from each point p_i to the nearest point on $\ell \cap P$. Therefore, for each point p_i , we define a set of intervals I_i using the circle C_i with diameter $\overline{p_r p_i}$ (see Figure 7.4 for an example). Suppose that the circle C_i intersects P at k points, $\{c_1, \dots, c_k\}$, distinct from any p_i . We define the points t_1, \dots, t_{n+k-1} to be $(\cup_{d=1}^k \{c_d\}) \cup (\cup_{e=1}^{n-1} \{p_e\})$, such that $t_1 = p_1$ and each t_j follows t_{j-1} by moving counterclockwise along the perimeter of P . Then $I_i = \{[t_1, t_2], [t_2, t_3], \dots, [t_{n+k-2}, t_{n+k-1}]\}$. Thus, each vertex in P (except p_r) and each intersection point between C_i and P appear as endpoints of the intervals in I_i .

Each interval, $[t_j, t_{j+1})$, is either contained entirely within C_i , or entirely outside C_i . In the former case we refer to the interval as an *inner interval*, and the latter case is an *outer interval*. Let I_i^+ be the set containing all inner intervals from I_i , and let I_i^- be set containing all outer intervals from I_i .

Observation 7.1. *Suppose for some value of θ , p_{ℓ_1} is contained within the interval $[t_j, t_{j+1})$. Then if $[t_j, t_{j+1}) \in I_i^+$, the nearest point on ℓ to p_i is p_{ℓ_1} ; otherwise, if $[t_j, t_{j+1}) \in I_i^-$, the nearest point lies on the interior of the line segment $\overline{p_r p_{\ell_1}}$. An identical claim can be made when the nearest point is p_{ℓ_2} .*

We now define the function $g(p_i, \theta)$ to be the distance between $p_i \in P_{\overline{Q}}$ and ℓ , defined on the interval $\Theta_{\min}[i] \leq \theta \leq \Theta_{\max}[i]$, when ℓ is rotated clockwise by an angle of θ from its starting orientation θ_1 . Since $P_{\overline{Q}}$ changes as ℓ rotates through a vertex, we keep track of all $p_i \in P$, noting that $g(p_i, \theta) = 0$ when $p_i \notin P_{\overline{Q}}$, and $g(p_i, \theta) = D[i]$ if $\theta < \Theta_{\min}[i]$ or $\theta > \Theta_{\max}[i]$. As we discussed earlier, $g(p_i, \theta)$ for fixed p_i is a piecewise continuous function. As we vary θ , the function used to compute the distance changes depending on whether the nearest point from p_i to ℓ is contained in $\overline{p_r p_{\ell_1}}$ or $\overline{p_r p_{\ell_2}}$. For our discussion, without loss of generality, we assume the nearest point p_i is always contained within the segment $\overline{p_r p_{\ell_1}}$. Next, suppose that p_{ℓ_1} lies within the interval (u, v) . If (u, v) is an outer interval, then we can compute the distance to $\ell \cap P$ using a right triangle. If (u, v) is an inner interval, then the minimum distance to $\ell \cap P$ occurs at the endpoint p_{ℓ_1} . Thus we have:

$$g(p_i, \theta) = \begin{cases} \text{dist}(p_i, r) \sin(|\theta_i - \theta|) & \text{if } (u, v) \in I_i^- \\ \text{dist}(p_i, p_{\ell_1}) & \text{otherwise} \end{cases}$$

where

$$p_{\ell_1} = (u_x + z(v_x - u_x), u_y + z(v_y - u_y)),$$

and

$$z = \frac{r_y - u_y + \tan \theta (u_x - r_x)}{v_y - u_y - \tan \theta (v_x - u_x)}.$$

Since C_i can intersect P at most $O(n)$ times, $k \in O(n)$. Thus, for each vertex p_i , $g(p_i, \theta)$ consists of $O(n)$ portions, where a *portion* is a maximal interval over which a single function is used in the piecewise continuous function. We note that $g(p_i, \theta)$ is continuous throughout the domain $[0, \theta_{n-1}]$, and that each portion of $g(p_i, \theta)$ is unimodal for its domain. Thus, a minima of one portion can be found in constant time.

7.3.2 Minimizing the Maximum Distance

We have discussed how to compute the distance between a vertex of P and the line ℓ . However, our objective is to minimize the maximum distance between $P_{\overline{Q}}$ and Q_b . To do this, we need to compute:

$$\min_{0 \leq \theta \leq \theta_{n-1}} G(\theta),$$

where

$$G(\theta) = \max_{1 \leq i \leq n-1} g(p_i, \theta).$$

We now describe how to merge two piecewise continuous functions $g(p_j, \theta)$ and $g(p_{j+1}, \theta)$, to create a new piecewise continuous function:

$$g'(\theta) = \max\{g(p_j, \theta), g(p_{j+1}, \theta)\}.$$

To compute the intersection points between two overlapping pieces of $g(p_j, \theta)$ and $g(p_{j+1}, \theta)$, we observe that there are three cases:

- both pieces represent inner intervals,
- both pieces represent outer intervals, and
- one piece is an inner interval and the other is an outer interval.

In all three cases the intersection points have analytic solutions. The closed form solutions are quite large, and so they are described later in Section 7.5 for completeness. Furthermore, since the domain of each function is $[0, \theta_{n-1}]$ and $\theta_{n-1} < \pi$, the number of intersection points between any two portions is constant. This means that the number of intersection points between $g(p_j, \theta)$ and $g(p_{j+1}, \theta)$ is $O(n)$.

Continuing this process, we can construct $G(\theta)$ by performing $n - 2$ merge steps like the one just described. Since we have at most $n - 1$ continuous functions such that each pair has $O(n)$ intersections, we can apply the upper bound from Lemma 6.1 (which states that by analyzing the upper envelope as a Davenport-Schinzel sequence, the number of distinct portions on the upper envelope of a set of functions is $\lambda_s(n) \leq \frac{sn(n-1)}{2} + 1$). We have shown that for our problem, $s \in O(n)$, and so the upper envelope of the functions, $G(\theta)$, consists of at most $O(n^3)$ portions. Each of these portions can be examined in constant time to find the minimum value. Algorithm 7.2 runs in $O(n^2)$ time, and so the overall running time of the Hausdorff Core algorithm on a polygon with a single reflex vertex is bounded by $O(n^3)$.

We provide an example in Figure 7.4 to illustrate the operation of our algorithm on a simple polygon with one reflex vertex, and a plot of the corresponding distance functions are shown in Figure 7.5.

7.4 Conclusions and Future Work

We presented an exact algorithm for determining a Hausdorff Core of a polygon with one reflex vertex in polynomial time. The algorithm measures the distance from the excluded vertices of the original polygon P to the approximating polygon Q . The Hausdorff distance between the polygons $H(P, Q)$ for a given cut is described by the maximum such distance, and so the problem may be formulated in terms of a set of piecewise functions whose maximum for a given line corresponds to $H(P, Q)$. By determining the minimum maximum value over all possible cuts, we determine the optimal Hausdorff Core for P .

The algorithm is not immediately extendable to polygons containing more than one reflex vertex, since the problem is complicated by the interaction between the cutting lines. It is also interesting to compare our problem to the potato peeling problem [31]. In the potato peeling problem, if P has only one reflex vertex then finding the optimal solution is trivial; only three cases need to be considered. This seems to suggest that finding the exact Hausdorff Core is a more difficult problem in general.

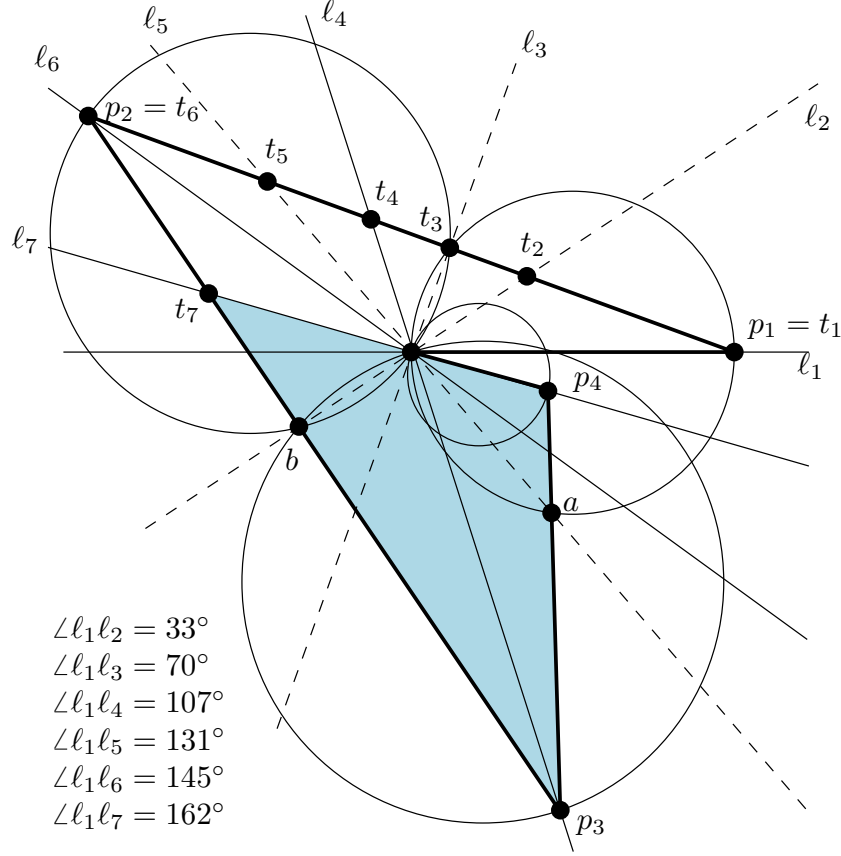


Figure 7.4: A polygon P with reflex vertex p_r is shown. The shaded area indicates the optimal solution Q . Lines indicate boundaries between intervals, where solid lines indicate interval boundaries defined by vertices of P , and dashed lines indicate boundaries defined by the transition between distance functions for a vertex. The latter transitions occur at the intersection points of the boundary of polygon with the diametrical circles defined between p_r and each vertex $p_i \in P \setminus \{p_r\}$. Consider the vertex p_1 . To begin with, we set $\theta = 0$ and the current solution Q_b contains p_1 . As θ is increased, the minimum distance from p_1 to a point on ℓ follows p_{ℓ_1} along $\overline{p_1 p_2}$ until point t_3 (the nearest point on ℓ to p_1 is outside P in this interval). At this point, the minimum distance to ℓ from p_1 follows a point around the circle incident upon $\{p_1, t_3, a\}$ through the interior of P until it meets the boundary at point a . At this point, the minimum distance function changes again to follow the point p_{ℓ_2} along $\overline{p_3 p_4}$ to point p_4 , at which point our sweep concludes. As one would expect, the distance from p_1 to ℓ is maximized at $\theta = \pi/2$ and minimized at $\theta = 0$, as can be seen in Figure 7.5.

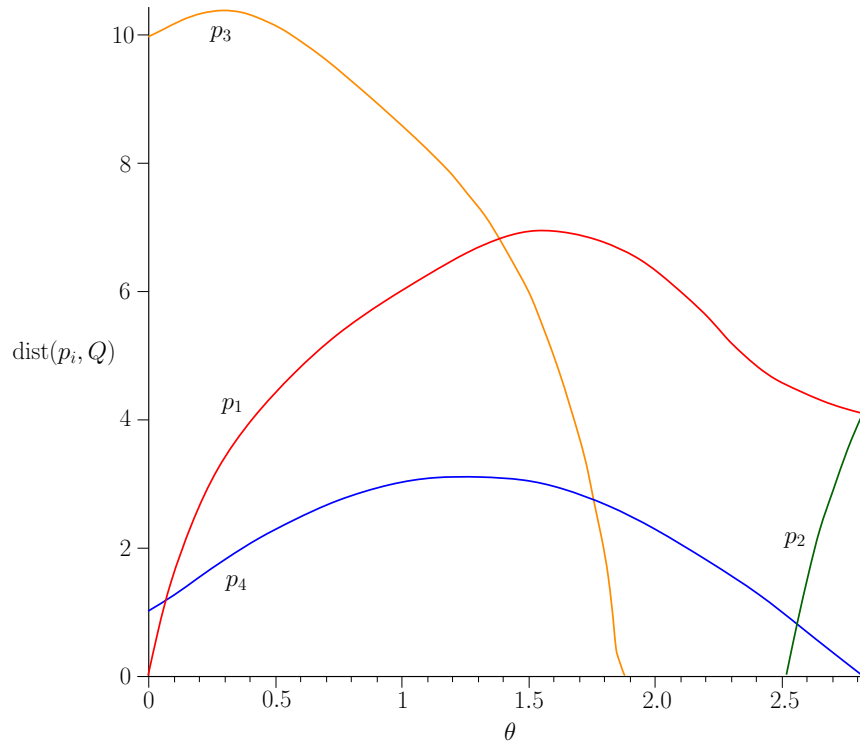


Figure 7.5: A plot for Figure 7.4, showing the curves corresponding to the distances from the vertices of P to the solution polygon Q for all possible values of θ . The point of intersection between the two functions converging on the right of the figure lie just outside of the range of the graph, ($\theta = 2.8201$, while viable solutions lie in the range $\theta = 0 \dots 2.8198$). The minimal value of the upper boundary of the graph is at $\theta = 2.8198$, where the Hausdorff distance is $H(P, Q) = \sqrt{17}$. This corresponds to the distance between points p_1 and p_4 .

7.5 Closed Form Solutions to the Hausdorff Core Problem

In Section 7.3.2, we described the functions for computing the nearest distance to a point on a line intersected with a polygon. Here, we provide the closed-form solutions to these functions. In the worst case, the solutions may be represented as the roots of degree four polynomials, which may be solved exactly when the coefficients are rational numbers (e.g. [66]). We provide these details for completeness. In practice, it would be wiser to find the intersection points using Newton's method or a similar root-finding technique. We simplify the notation by writing d_i rather than $\text{dist}(p_i, p_r)$.

7.5.1 Both Points Inside the Polygon

This is the simplest case, as we are simply seeking the intersection points between the functions

$$f_1(\theta) = d_1 \cdot \sin(|\theta_1 - \theta|) \quad (7.1)$$

and

$$f_2(\theta) = d_2 \cdot \sin(|\theta_2 - \theta|). \quad (7.2)$$

We find the intersection points between these functions by solving $f_1(\theta) - f_2(\theta) = 0$ with Maple. Because of the use of the absolute value functions, the cardinality of the set of intersection points depends on the values of θ_1, θ_2, d_1 , and d_2 . There are two possible intersection points at

$$\theta = \theta_1 - \arctan\left(\frac{d_2 \sin(\theta_1 - \theta_2)}{-d_1 + d_2 \cos(\theta_1 - \theta_2)}\right)$$

and

$$\theta = \theta_1 - \arctan\left(\frac{d_2 \sin(\theta_1 - \theta_2)}{d_1 + d_2 \cos(\theta_1 - \theta_2)}\right).$$

The existence of intersection points at each of the above positions may be checked a priori by testing against a set of conditions, but it is simpler to check by substituting back into the original difference function.

7.5.2 Both Points on the Polygon Boundary

In this case, we have two functions where the distance is measured to a point along an interval of a line. Therefore, we are finding the distance from a point p (resp. q) to a point on the line segment $\overline{u, u'}$ (resp. $\overline{v, v'}$), and so the intersection points are expressed as solutions to the following equation (we use r to denote the reflex vertex, and r_x is the x-coordinate of r):

$$\begin{aligned} & \left(p_x - u_x - (u'_x - u_x) \frac{r_y - u_y - (u_x - r_x) \tan(\theta)}{u'_y - u_y - (u'_x - u_x) \tan(\theta)} \right)^2 \\ & + \left(p_y - u_y - (u'_y - u_y) \frac{r_y - u_y - (u_x - r_x) \tan(\theta)}{u'_y - u_y - (u'_x - u_x) \tan(\theta)} \right)^2 \\ & - \left(q_x - v_x - (v'_x - v_x) \frac{r_y - v_y - (v_x - r_x) \tan(\theta)}{v'_y - v_y - (v'_x - v_x) \tan(\theta)} \right)^2 \\ & - \left(q_y - v_y - (v'_y - v_y) \frac{r_y - v_y - (v_x - r_x) \tan(\theta)}{v'_y - v_y - (v'_x - v_x) \tan(\theta)} \right)^2 = 0 \end{aligned}$$

Solving for the difference between the functions,

$$\theta = \arctan(\text{RootOf}(c_1 \cdot Z^4 + c_2 \cdot Z^3 + c_3 \cdot Z^2 + c_4 \cdot Z + c_5)),$$

where the constants (from the input) are listed below. To simplify the notation, we use the following constants:

$$\begin{aligned} a_1 = u'_x - u_x, a_2 = r_y - u_y, a_3 = u_x - r_x, a_4 = u'_y - u_y, \\ b_1 = v'_x - v_x, b_2 = r_y - v_y, b_3 = v_x - r_x, b_4 = v'_y - v_y. \end{aligned} \quad (7.3)$$

The coefficients are:

$$\begin{aligned} c_1 = & 2 p_y a_1 a_4 a_3 b_1^2 + a_4^2 a_3^2 b_1^2 - 2 p_y a_1^2 u_y b_1^2 - 2 u_y a_1 a_4 a_3 b_1^2 - 2 q_y b_1 b_4 b_3 a_1^2 \\ & - 2 p_x a_1^2 u_x b_1^2 + p_y^2 a_1^2 b_1^2 + 2 q_x b_1^2 v_x a_1^2 + 2 q_y b_1^2 v_y a_1^2 + 2 v_x b_1^2 b_3 a_1^2 \\ & - q_y^2 b_1^2 a_1^2 - 2 u_x a_1^2 a_3 b_1^2 - v_y^2 b_1^2 a_1^2 - q_x^2 b_1^2 a_1^2 - b_4^2 b_3^2 a_1^2 \\ & - v_x^2 b_1^2 a_1^2 - 2 q_x b_1^2 b_3 a_1^2 - b_1^2 b_3^2 a_1^2 + 2 v_y b_1 b_4 b_3 a_1^2 + 2 p_x a_1^2 a_3 b_1^2 \\ & + p_x^2 a_1^2 b_1^2 + u_y^2 a_1^2 b_1^2 + u_x^2 a_1^2 b_1^2 + a_1^2 a_3^2 b_1^2, \end{aligned}$$

$$\begin{aligned} c_2 = & 2 v_y^2 b_1^2 a_4 a_1 + 2 b_4^2 b_3^2 a_4 a_1 - 2 p_x^2 a_1^2 b_4 b_1 - 2 u_x^2 a_1^2 b_4 b_1 - 2 a_1^2 a_3^2 b_4 b_1 \\ & - 2 p_x^2 a_4 a_1 b_1^2 + 2 p_x a_1^2 a_2 b_1^2 - 2 u_x^2 a_4 a_1 b_1^2 - 2 u_x a_1^2 a_2 b_1^2 + 2 a_1^2 a_2 a_3 b_1^2 \\ & - 2 p_y^2 a_1^2 b_4 b_1 - 2 u_y^2 a_1^2 b_4 b_1 - 2 a_4^2 a_3^2 b_4 b_1 - 2 p_y^2 a_4 a_1 b_1^2 - 2 p_y a_4^2 a_3 b_1^2 \\ & + 4 p_y a_4 u_y a_1 b_1^2 - 2 p_x a_4 a_1 a_3 b_1^2 + 2 u_x a_4 a_1 a_3 b_1^2 + 4 p_x a_4 u_x a_1 b_1^2 - 4 p_x a_1^2 a_3 b_4 b_1 \\ & + 4 p_x a_1^2 u_x b_4 b_1 + 4 u_x a_1^2 a_3 b_4 b_1 + 2 p_y a_1 a_4 a_2 b_1^2 - 2 u_y a_1 a_4 a_2 b_1^2 + 4 p_y a_1^2 u_y b_4 b_1 \\ & - 4 q_x b_1^2 v_x a_4 a_1 + 4 q_x b_1^2 b_3 a_4 a_1 - 4 v_x b_1^2 b_3 a_4 a_1 - 4 q_x b_4 v_x b_1 a_1^2 + 2 q_x b_4 b_1 b_3 a_1^2 \\ & - 2 v_x b_4 b_1 b_3 a_1^2 - 4 q_y b_1^2 v_y a_4 a_1 - 4 q_y b_4 v_y b_1 a_1^2 - 2 q_y b_1 b_4 b_2 a_1^2 + 2 v_y b_1 b_4 b_2 a_1^2 \\ & + 4 q_y b_1 b_4 b_3 a_4 a_1 - 4 v_y b_1 b_4 b_3 a_4 a_1 + 4 u_y a_1 a_4 a_3 b_4 b_1 - 2 u_y^2 a_4 a_1 b_1^2 + 2 u_y a_4^2 a_3 b_1^2 \\ & + 2 a_4^2 a_2 a_3 b_1^2 + 2 q_x^2 b_4 b_1 a_1^2 - 2 q_x b_1^2 b_2 a_1^2 + 2 v_x^2 b_4 b_1 a_1^2 + 2 v_x b_1^2 b_2 a_1^2 \\ & - 2 b_1^2 b_2 b_3 a_1^2 + 2 q_x^2 b_1^2 a_4 a_1 + 2 v_x^2 b_1^2 a_4 a_1 + 2 b_1^2 b_3^2 a_4 a_1 + 2 q_y^2 b_4 b_1 a_1^2 \\ & + 2 q_y b_4^2 b_3 a_1^2 + 2 v_y^2 b_4 b_1 a_1^2 - 2 v_y b_4^2 b_3 a_1^2 - 2 b_4^2 b_2 b_3 a_1^2 + 2 q_y^2 b_1^2 a_4 a_1 \\ & - 4 p_y a_1 a_4 a_3 b_4 b_1, \end{aligned}$$

$$\begin{aligned}
c_3 = & -2p_x a_4^2 u_x b_1^2 - 2p_y a_4^2 u_y b_1^2 - 2p_y a_4^2 a_2 b_1^2 + 2u_y a_4^2 a_2 b_1^2 + 2p_y a_1 a_4 a_3 b_4^2 \\
& - 2u_y a_1 a_4 a_3 b_4^2 + u_y^2 a_4^2 b_1^2 + p_y^2 a_4^2 b_1^2 - q_x^2 b_1^2 a_4^2 - v_x^2 b_1^2 a_4^2 \\
& - b_1^2 b_3^2 a_4^2 - b_1^2 b_2^2 a_1^2 - q_y^2 b_1^2 a_4^2 - v_y^2 b_1^2 a_4^2 - b_4^2 b_3^2 a_4^2 \\
& - b_4^2 b_2^2 a_1^2 + a_2^2 a_1^2 b_1^2 + u_x^2 a_4^2 b_1^2 + p_x^2 a_4^2 b_1^2 + a_4^2 a_2^2 b_1^2 \\
& - 2p_x a_4 a_2 a_1 b_1^2 + 4p_x^2 a_4 a_1 b_4 b_1 + 4u_x^2 a_4 a_1 b_4 b_1 - 4p_x a_1^2 a_2 b_4 b_1 - 4a_1^2 a_2 a_3 b_4 b_1 \\
& + 4u_x a_1^2 a_2 b_4 b_1 + 4p_y^2 a_4 a_1 b_4 b_1 + 4p_y a_4^2 a_3 b_4 b_1 + 4u_y^2 a_4 a_1 b_4 b_1 - 4u_y a_4^2 a_3 b_4 b_1 \\
& - 4a_4^2 a_2 a_3 b_4 b_1 - 4q_x^2 b_4 b_1 a_4 a_1 + 2q_x b_4 b_1 b_2 a_1^2 + 4q_x b_1^2 b_2 a_4 a_1 - 4v_x^2 b_4 b_1 a_4 a_1 \\
& - 2v_x b_4 b_1 b_2 a_1^2 - 4v_x b_1^2 b_2 a_4 a_1 + 4b_1^2 b_2 b_3 a_4 a_1 - 4q_y^2 b_4 b_1 a_4 a_1 - 4q_y b_4^2 b_3 a_4 a_1 \\
& - 4v_y^2 b_4 b_1 a_4 a_1 + 4v_y b_4^2 b_3 a_4 a_1 + 4b_4^2 b_2 b_3 a_4 a_1 - 2q_y b_1 b_4 b_3 a_4^2 + 2v_y b_1 b_4 b_3 a_4^2 \\
& + 2u_x a_4 a_2 a_1 b_1^2 + a_1^2 a_3^2 b_4^2 - 2p_x a_1^2 u_x b_4^2 + 2p_x a_1^2 a_3 b_4^2 - 2u_x a_1^2 a_3 b_4^2 \\
& - 2p_y a_1^2 u_y b_4^2 + 2q_x b_4^2 v_x a_1^2 + 2q_y b_4^2 v_y a_1^2 - 4v_y b_1 b_4 b_2 a_4 a_1 + 8q_y b_4 v_y b_1 a_4 a_1 \\
& - 4u_x a_4 a_1 a_3 b_4 b_1 + 4u_y a_1 a_4 a_2 b_4 b_1 - 8p_x a_4 u_x a_1 b_4 b_1 - 8p_y a_4 u_y a_1 b_4 b_1 + 4q_y b_1 b_4 b_2 a_4 a_1 \\
& + 8q_x b_4 v_x b_1 a_4 a_1 - 4p_y a_1 a_4 a_2 b_4 b_1 - v_y^2 b_4^2 a_1^2 + p_x^2 a_1^2 b_4^2 + u_x^2 a_1^2 b_4^2 \\
& + p_y^2 a_1^2 b_4^2 + u_y^2 a_1^2 b_4^2 + a_4^2 a_3^2 b_4^2 - q_x^2 b_4^2 a_1^2 - v_x^2 b_4^2 a_1^2 \\
& - q_y^2 b_4^2 a_1^2 + 2q_x b_1^2 v_x a_4^2 - 2q_x b_1^2 b_3 a_4^2 + 2v_x b_1^2 b_3 a_4^2 + 2q_y b_1^2 v_y a_4^2 \\
& + 2q_y b_4^2 b_2 a_1^2 - 2v_y b_4^2 b_2 a_1^2 + 4p_x a_4 a_1 a_3 b_4 b_1 - 4q_x b_4 b_1 b_3 a_4 a_1 + 4v_x b_4 b_1 b_3 a_4 a_1,
\end{aligned}$$

$$\begin{aligned}
c_4 = & 2b_4^2 b_2^2 a_4 a_1 - 2a_2^2 a_1^2 b_4 b_1 - 2u_x^2 a_4^2 b_4 b_1 - 2p_x^2 a_4^2 b_4 b_1 - 2a_4^2 a_2^2 b_4 b_1 \\
& - 2u_y^2 a_4^2 b_4 b_1 - 2p_y^2 a_4^2 b_4 b_1 + 4p_x a_4^2 u_x b_4 b_1 + 4p_y a_4 u_y a_1 b_4^2 + 2p_y a_1 a_4 a_2 b_4^2 \\
& - 2u_y a_1 a_4 a_2 b_4^2 - 4q_x b_4^2 v_x a_4 a_1 - 4q_y b_4^2 v_y a_4 a_1 + 4p_x a_4 u_x a_1 b_4^2 - 4q_x b_4 v_x b_1 a_4^2 \\
& + 2q_x b_4 b_1 b_3 a_4^2 - 2v_x b_4 b_1 b_3 a_4^2 - 4q_y b_4 v_y b_1 a_4^2 - 2q_y b_1 b_4 b_2 a_4^2 + 2v_y b_1 b_4 b_2 a_4^2 \\
& - 4q_y b_4^2 b_2 a_4 a_1 + 4v_y b_4^2 b_2 a_4 a_1 + 4p_y a_4^2 u_y b_4 b_1 + 4p_y a_4^2 a_2 b_4 b_1 - 4u_y a_4^2 a_2 b_4 b_1 \\
& - 2p_x a_4 a_1 a_3 b_4^2 + 2u_x a_4 a_1 a_3 b_4^2 + 2a_2 a_1^2 a_3 b_4^2 - 2p_x^2 a_4 a_1 b_4^2 + 2p_x a_1^2 a_2 b_4^2 \\
& - 2u_x^2 a_4 a_1 b_4^2 - 2u_x a_1^2 a_2 b_4^2 - 2p_y^2 a_4 a_1 b_4^2 - 2p_y a_4^2 a_3 b_4^2 - 2u_y^2 a_4 a_1 b_4^2 \\
& + 2u_y a_4^2 a_3 b_4^2 + 2a_4^2 a_2 a_3 b_4^2 + 2q_x^2 b_4^2 a_4 a_1 + 2v_x^2 b_4^2 a_4 a_1 + 2q_y^2 b_4^2 a_4 a_1 \\
& + 2v_y^2 b_4^2 a_4 a_1 + 4v_x b_4 b_1 b_2 a_4 a_1 + 4p_x a_4 a_2 a_1 b_4 b_1 + 2q_x^2 b_4 b_1 a_4^2 - 2q_x b_1^2 b_2 a_4^2 \\
& + 2v_x^2 b_4 b_1 a_4^2 + 2v_x b_1^2 b_2 a_4^2 - 2b_1^2 b_2 b_3 a_4^2 + 2b_1^2 b_2^2 a_4 a_1 + 2q_y^2 b_4 b_1 a_4^2 \\
& + 2q_y b_4^2 b_3 a_4^2 + 2v_y^2 b_4 b_1 a_4^2 - 2v_y b_4^2 b_3 a_4^2 - 2b_4^2 b_2 b_3 a_4^2 - 4q_x b_4 b_1 b_2 a_4 a_1 \\
& - 4u_x a_4 a_2 a_1 b_4 b_1,
\end{aligned}$$

and

$$\begin{aligned}
c_5 = & 2q_y b_4^2 v_y a_4^2 - 2p_x a_4^2 u_x b_4^2 - 2p_y a_4^2 u_y b_4^2 + 2q_x b_4^2 v_x a_4^2 + p_x^2 a_4^2 b_4^2 \\
& + u_x^2 a_4^2 b_4^2 + p_y^2 a_4^2 b_4^2 + u_y^2 a_4^2 b_4^2 - q_x^2 b_4^2 a_4^2 - v_x^2 b_4^2 a_4^2 \\
& - q_y^2 b_4^2 a_4^2 - v_y^2 b_4^2 a_4^2 + 2q_y b_4^2 b_2 a_4^2 - 2v_y b_4^2 b_2 a_4^2 + 2q_x b_4 b_1 b_2 a_4^2 \\
& - 2v_x b_4 b_1 b_2 a_4^2 - 2p_x a_4 a_2 a_1 b_4^2 + 2u_x a_4 a_2 a_1 b_4^2 + a_2^2 a_1^2 b_4^2 \\
& - 2p_y a_4^2 a_2 b_4^2 + 2u_y a_4^2 a_2 b_4^2 - b_1^2 b_2^2 a_4^2 - b_4^2 b_2^2 a_4^2 + a_4^2 a_2^2 b_4^2.
\end{aligned}$$

7.5.3 One Point on the Polygon Boundary

In the third setting, we seek the intersection points between two functions, where the first is f_1 (Equation 7.1) from above, and the second is that of a point on the boundary of the polygon. Therefore, to find the intersection points, we solve the following:

$$\sqrt{\left(p_x - u_x - a_1 \frac{a_2 + a_3 \tan(\theta)}{a_4 - a_1 \tan(\theta)}\right)^2 + \left(p_y - u_y - a_4 \frac{a_2 + a_3 \tan(\theta)}{a_4 - a_1 \tan(\theta)}\right)^2} - d_1 \sin(\theta_1 - \theta) = 0, \quad (7.4)$$

where the constants a_1, a_2, a_3 and a_4 are defined in Equation 7.3.

In this case, the solution is much more complex than that of the previous section, and to write out the complete representation of the coefficients would require over 20 pages. The fundamental component of the solution is a root of the following polynomial, whose roots we call the set R :

$$R = \text{RootOf}(c'_1 \cdot Z^4 + c'_2 \cdot Z^3 + c'_3 \cdot Z^2 + c'_4 \cdot Z + c'_5).$$

Now the solutions may be found by evaluating:

$$\arctan\left(\sqrt{R}, \frac{c_6 + c_7 R + c_8 R^2}{c_9 R^{3/2} + c_{10} \sqrt{R}}\right), \arctan\left(\sqrt{R}, \frac{c_6 + c_7 R + c_8 R^2}{-c_9 R^{3/2} - c_{10} \sqrt{R}}\right).$$

The precise values of the constants may be seen by solving Equation 7.4 with Maple. Note that the arctan function used here has two arguments. This means that the solution to $\arctan(y, x)$ is the principal argument of the complex number $x + iy$ [115], where the signs of x and y determine the quadrant of the plane which contains the solution [160]:

$$\arctan(x, y) = \begin{cases} \arctan(y/x) & x > 0 \\ \arctan(y/x) + \pi & y \geq 0, x < 0 \\ \arctan(y/x) - \pi & y < 0, x < 0 \\ \pi/2 & y > 0, x = 0 \\ -\pi/2 & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases}$$

Approximation Algorithms for the Hausdorff Core Problem on Simple Polygons

In this chapter we outline parameterized and approximation algorithms to solve the general Hausdorff Core problem which operate by manipulating disks centred on selected vertices of P (precisely which vertices have disks is discussed shortly)¹. We begin by describing parameterized decision and optimization algorithms, and we describe how the decision algorithm may be modified to obtain a fully polynomial time approximation scheme (FPTAS) for the decision version of the Hausdorff Core problem.

Theorem 8.1. *There exists a parameterized algorithm for the general Hausdorff Core problem on simple polygons with $O(\log(\varepsilon^{-1})(n^3 + n^2\varepsilon^{-6}))$ running time. Given an input polygon P , the algorithm computes a convex polygon Q , where $Q \subseteq P$, and with Hausdorff distance $H(P, Q) < k_{\text{OPT}} + d_{\text{rad}} \cdot \varepsilon$, where k_{OPT} is the value of the optimal solution, and d_{rad} is the radius of the polygon (i.e. the distance from a constrained 1-centre of P to the most distant vertex in P).*

To simplify the discussion, we will scale the problem so that $d_{\text{rad}} = 1$, and so the parameterized algorithm finds a Hausdorff Core where $H(P, Q) < k_{\text{OPT}} + \varepsilon$. By Corollary 6.1 (page 80), Invariant 8.1 implies that there exists a Hausdorff Core solution with $H(P, Q) = k$:

Invariant 8.1. *Given a simple polygon P with convex hull $\text{CH}(P)$, there exists a set of points $\{q_1, \dots, q_{n'}\} \subset Q$, where n' is the number vertices of $\text{CH}(P)$, such that $\forall i, q_i \in C(p_i, k)$ and $\forall i, j, i \neq j, \overline{q_i q_j}$ does not cross outside P (recall that $C(p_i, k)$ is a disk of radius k centred at p_i).*

We sketch a continuous version of the solution in Algorithm 8.1, and we illustrate an example of the operation of the algorithm in Figure 8.1. In this algorithm, we use Lemma 6.4 which states that an optimal Hausdorff Core Q of P may always cover at least one vertex of P . We call this vertex of P covered by Q the point q_p , and we try all possible values of q_p in the algorithm. The algorithm operates by placing disk centres on the vertices of the convex hull of P and shrinking their radii uniformly as long as there exists a Hausdorff Core which pierces all disks. We simplify this test by considering only those disks that do not cover q_p and checking for intersection between Q and the boundary of each disk.

The solution Q is a convex polygon which intersects every disk. If each disk $C(p_i, k)$ touches Q , we know that the distance from each vertex with a disk to Q is at most k , the radius of

¹Elements of this chapter have appeared in [53].

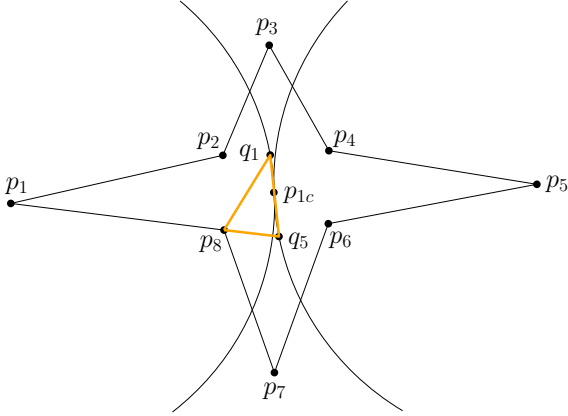
Algorithm 8.1 HCORE(P)

- 1: **Input:** A simple polygon P .
 - 2: **Output:** Q , a Hausdorff Core of P .
 - 3: $Q = \emptyset, k_{\min} = \infty$
 - 4: **for** each $q_p \in P_V$ **do**
 - 5: Begin with disks of radius k_0 centred on the vertices $v \in CH(P)_V$, where $k_0 = 1$ (recall $d_{\text{rad}} = 1$).
 - 6: Any disk centred at a vertex v where $\text{dist}(q_p, v) < k_0$ covers q_p ; such disks are ignored for now.
 - 7: Reduce the radius such that at time $t_i \in [0, 1]$, each disk has radius $k(t_i) = 1 - t_i$. Let $Q(t_i)$ be a Hausdorff Core covering q_p at time t_i , if it exists (we discuss how this may be done approximately in Section 8.2). The radius is reduced until one of three events occurs:
 1. $k(t_i) = \text{dist}(q_p, v_n)$, where v_n is the farthest vertex from q_p that is not the centre of a disk. Add a disk centred at v_n with radius $k(t_i)$.
 2. $Q(t_i)$ cannot cover q_p . In this case, we break and if $k(t_i) < k_{\min}$, then set $Q = Q(t_i)$ and $k_{\min} = k(t_i)$.
 3. A further reduction of $k(t_i)$ would prevent visibility in P between two disks. Again, we break and if $k(t_i) < k_{\min}$, then set $Q = Q(t_i)$ and $k_{\min} = k(t_i)$.
 - 8: **end for**
 - 9: **return** Q
-

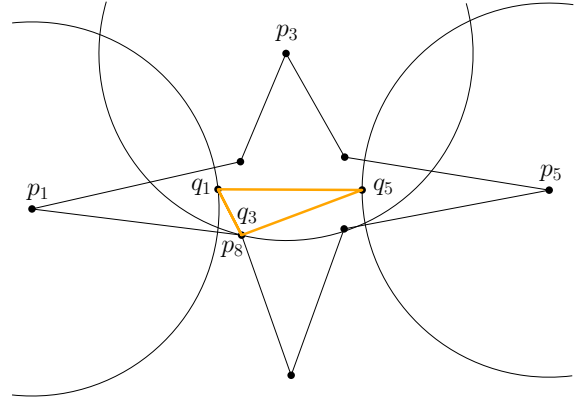
$C(p, k)$. We ensure that if a vertex $v \in CH(P)_V$ (the set of vertices of the convex hull of P) does not have a disk, then $\text{dist}(v, q_p) \leq k$. Therefore, given a simple polygon P , this algorithm finds a convex polygon Q contained in P such that $\forall p \in CH(P)_V, \exists q \in Q$ such that $\text{dist}(p, q) \leq k$. By Lemma 6.3 (page 79), we know that Q is a solution where $H(P, Q) = k$. It remains to be shown that there does not exist a convex polygon Q' such that $\text{dist}(p, Q') \leq k'$, where $k' < k$. This cannot be the case, for if the disks were shrunk any further, no convex polygon could intersect some pair of the disks by Invariant 8.1. Therefore, the polygon would necessarily be of distance $\text{dist}(p, q') > k'$ for some vertex p and any point $q' \in Q$.

Finally, the optimality of the algorithm is guaranteed since we exhaustively explore the different possibilities for the point q_p which is contained in a solution Q . By Lemma 6.4, we know that at least one such point q_p is contained in an optimal solution. By trying all possibilities, we ensure that the globally optimal solution is obtained.

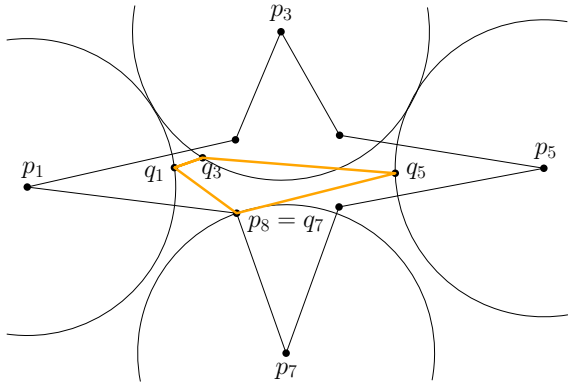
Our formal approximation algorithm, described in Section 8.3, operates in a similar way to Algorithm 8.1 except that we use a parameterized (binary) search on the values of possible Hausdorff Core solutions to obtain the approximation algorithm, rather than reducing the value continuously. We find the Euclidean 1-centre p_{1c} using the technique of [22] described in Section 6.2.11; there may be multiple such vertices, but we can choose one arbitrarily.



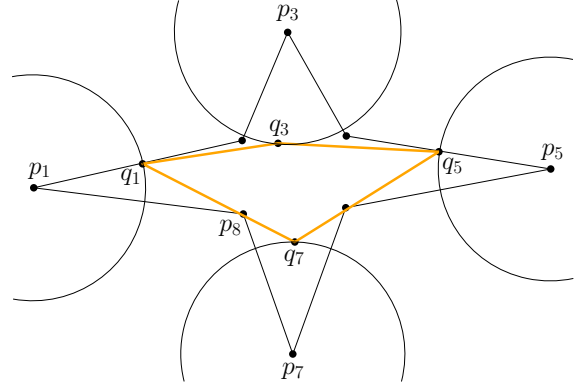
(a) Two disks are centred on p_1 and p_5 , which are the critical points for determining the position of the 1-centre p_{1c} . Let $q_p = p_8$.



(b) We have shrunk the disks to have radius k' , and $\text{dist}(p_3, p_8) = k'$, so we add a new disk $C(p_3, k')$ to the set. The fat lines indicate a set of lines of strong visibility between the disks, and so there exists a Hausdorff Core where $H(P, Q) = k'$.



(c) Another disk is added centred at point p_7 , so now we have four disks $C(p_i, \text{dist}(p_7, p_8))$, for each $i \in \{1, 3, 5, 7\}$.



(d) We cannot shrink the disks any further, otherwise Invariant 8.1 would be violated. Therefore, a solution Q can be composed from the fat line segments such that $H(P, Q) = k$, where k is the radius of the disks. Note that all vertices of P are within distance k of Q , and vice versa.

Figure 8.1: Finding a Hausdorff Core by shrinking disks centred on the vertices of P , as discussed in Algorithm 8.1. We are using $q_p = p_8$ in this example.

8.1 Algorithmic Challenges of the Hausdorff Core Problem

The decision version of the Hausdorff Core problem on P consists of determining whether we can draw a polygon Q with one vertex in or on each disk centred on a vertex of P and so that each successive pair of vertices in Q defines a line segment which does not cross outside P . We may try to solve this by selecting a point on the first disk, and then choosing the clockwise-most point on the next disk, iterating this procedure in a counter-clockwise direction around the interior of P until we return to the first disk.

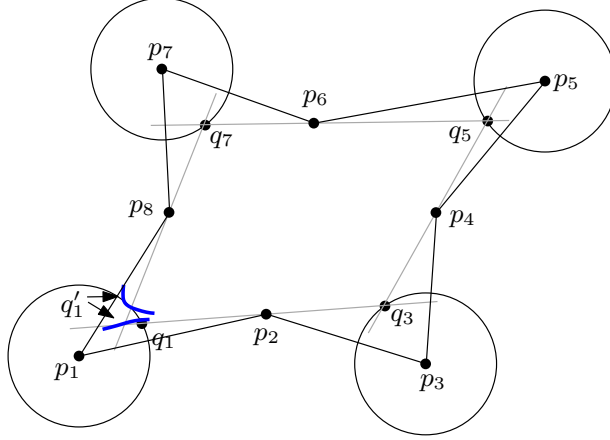


Figure 8.2: There can be two disconnected solution intervals for the Hausdorff Core problem. The points p_i are vertices of the polygon P , and the points q_j are points that are being selected on disks as the vertices of the solution polygon Q .

As shown in Figure 8.2, it is possible for our problem to have two disconnected sets of solutions, even with as few as four disks. For a point q_1 on the first disk, we can trace the polygon through the vertex $p_2 \in P$ to the intersection with the second disk at q_3 , then through the vertex p_4 , and so on, around to p_8 . The lines $\overleftrightarrow{q_1 p_2}$ and $\overleftrightarrow{q_7 p_8}$ intersect at q'_1 , which is a choice for one vertex of the polygon Q (the others being $q_3, q_5,$ and q_7). If q'_1 is inside the same disk as q_1 , then we have a feasible solution. However, the heavy curves show the locus of q'_1 for different choices of q_1 , and the portion of it inside the disk is in two disjoint pieces. The set of solutions to the problem as shown is disjoint, corresponding to a slice (for a constant value of the radius of the disks) through a non-convex optimization region. As a result, neither second-order cone programming (i.e. [112]) nor any other convex optimization technique is immediately applicable.

8.2 Discretization of the Problem

In this section, we discuss the decision version of the approximation algorithm, where we are given a distance k and we wish to determine whether there is an approximate Hausdorff Core solution Q' with $H(P, Q') \leq k + \varepsilon$. This approximation scheme seeks to grow disks by an additive factor ε , and determine whether there exists a solution for these expanded disks. ε is the fraction of d_{rad} that we wish to use as a bound on the approximation, where d_{rad} is the distance from the constrained 1-centre p_{1c} to the most distant vertex in P : $d_{\text{rad}} = \max_{p \in P} \text{dist}(p_{1c}, p)$. We scale the input so that $d_{\text{rad}} = 1$ to simplify the analysis. Notice that this method of approximation maintains a scale invariant approximation factor, and the size of the approximation factor for a given P is constant, regardless of Q and the magnitude of k . We still require that the approximate solution Q' must not cross outside P , and that Invariant 8.1 holds.

The strategy behind this approximation algorithm is to grow the disks by $\varepsilon/3$, so that they may be discretized into arc segments on the boundary of each disk. We grow them by $\varepsilon/3$ rather than ε because there is some additional error that comes into the algorithm, and we wish to have a final additive approximation factor of ε . It is possible to check for strong visibility between discrete intervals², which avoids some of the problems faced by the exact formulation of the

²Strong visibility here means that given a pair of intervals, all points on each interval are visible from any point

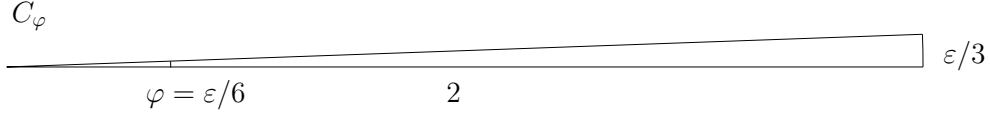


Figure 8.3: Determining whether there exists a straight line segment that would serve as a Hausdorff Core solution. Consider a circular segment C_φ of radius 2, arc length $\varepsilon/3$, and with an interior angle of $\varphi = \varepsilon/6$. If Q' can be covered by C_φ , then a straight line segment Q_ℓ exists such that $H(Q', Q_\ell) < \varepsilon/3$.

problem. One of the challenges of this approach is the selection of the length of the intervals on the new disks of radius $k + \varepsilon/3$. We require that the intervals be small enough so that we will find a solution for the approximation if one existed for the original disk radius. In other words, given an exact solution Q for the original radius k such that $H(P, Q) \leq k$, we are guaranteed that at least one interval on each of the expanded disks will be contained inside Q .

We study two possible cases for the properties of a Hausdorff Core Q : either the minimum angle in Q is $\varepsilon/6$ (we justify this choice shortly), or else Q may be approximated by a line segment. We begin by looking at the latter case.

We first determine whether any Hausdorff Core Q can be approximated by a single line segment. We consider an arc segment of radius 2 (i.e. the maximum diameter of P) and arc length $\varepsilon/3$, as shown in Figure 8.3. The interior angle of the circular segment C_φ formed by this arc is $\varphi = \varepsilon/6$. If an interior angle of Q is less than or equal to φ , then Q may be fully covered by C_φ since Q is convex. In this case, there exists a line segment Q_ℓ which approximates Q' such that $H(Q', Q_\ell) < \varepsilon/3$.

We now describe how to determine such a line segment Q_ℓ , assuming there exists some C_φ that covers Q . First, we grow all existing disks by an additional factor of $\varepsilon/3$, so that they have radius $k^g = k + 2\varepsilon/3$. Since Q is convex, this operation means that any line segment which approximates Q will now intersect at least one arc from each disk if a solution exists where $H(P, Q) \leq k$. By Lemma 6.4, we know that $q_p \in P_V$ is contained in Q . Therefore, we attempt to find a line intersecting a point q_p and a segment of each disk of radius k^g for each q_p . For a selected q_p , we build an interval graph in the range $[0 \dots \pi]$. For each disk $C(p_i, k^g)$, if a line at angle $\theta \bmod \pi$ from an arbitrary reference line intersects a segment of $C(p_i, k^g)$ contained in P before intersecting P itself, then $C(p_i, k^g)$ covers θ in the interval graph. If there is a non-zero intersection between all disks in the interval graph at θ^* , then the solution is a line segment Q_ℓ at angle θ^* to the reference line, intersecting q_p with endpoints on the last disks intersected by Q_ℓ . Therefore, if there exists a solution $H(P, Q) \leq k$ where Q can be approximated by a line segment Q_ℓ with $H(Q, Q_\ell) < 2\varepsilon/3$, then we will find Q_ℓ .

If we have not found a solution Q_ℓ , we know that all interior angles of some polygon Q are greater than φ , and so we wish to determine an approximating polygon Q' . If we divide the expanded disk of radius $k + \varepsilon/3$ into $\frac{36\pi^2(k+\varepsilon/3)}{\varepsilon^2}$ equal intervals, at least one interval is fully contained in Q regardless of where the intervals are placed on the disk.

Lemma 8.1. *If there exists a Hausdorff Core Q with Hausdorff distance k to the polygon P , then by Invariant 8.1, Q pierces all disks of radius k centred on the vertices of the convex hull of P . If disks of radius $k + \varepsilon/3$ are placed on all the same vertices and the minimum angle of Q is at*

on the other interval.

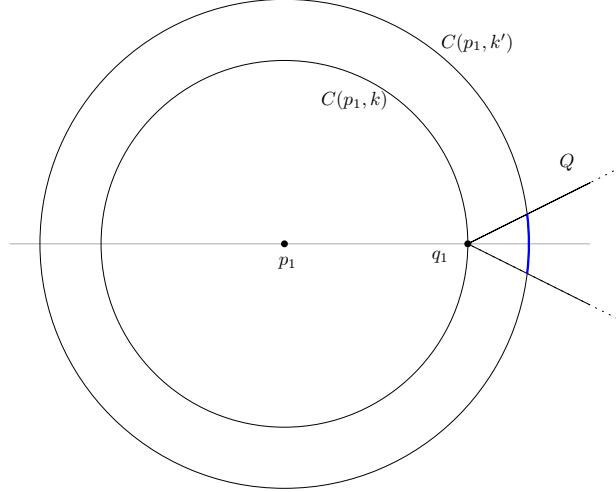


Figure 8.4: To find the minimum length interval for the discretization of the expanded disks, we need to ensure that at least one full interval is always covered by Q . We know the minimum angle at any vertex of Q is at least φ , since there was no single line segment that could approximate Q . The minimum arc length of $C(p_1, k')$ spanned by Q is realized when the bisector of the edges incident upon q_1 is collinear with the line $\overleftrightarrow{p_1 q_1}$.

least φ , then the expanded disks may be divided into $O(\varepsilon^{-2})$ disjoint intervals so that at least one interval on every expanded disk is on the interior of Q .

Proof. The smallest arc length of an expanded disk that may be covered by Q is realized when the bisector of the angle at a vertex q_1 is collinear with the line formed by q_1 and the centre of the disk p_1 (see Figure 8.4). The minimum angle at a vertex in Q is $\varphi = \varepsilon/6$, and so we want to determine the maximum angle φ' for a circular segment C_P of radius $k + \varepsilon/3$ centred at p_1 so that two such circular segments may be covered by Q . A circular segment C_Q of radius $\varepsilon/3$ and angle $\varepsilon/12$ has arc length $\varepsilon^2/36$, and two disjoint such segments may be covered by $Q \cap C(p_1, k + \varepsilon/3)$. Therefore, given points q_{c1} and q_{c2} as the endpoints of the arc on C_Q , we may place the endpoints of the arc on C_P at these points to determine a lower bound on φ' . The arc length of C_P is at least $2/\pi$ times that of C_Q on these points. Since the arc length of C_Q is $\varepsilon^2/36$, the arc length of C_P is at least $\varepsilon^2/18\pi$. The interior angle φ' of C_P is given by $\varphi' = \frac{\varepsilon^2/18\pi}{k + \varepsilon/3} = \frac{\varepsilon^2}{18\pi(k + \varepsilon/3)}$. The number of such circular segments in $C(p_1, k + \varepsilon/3)$ is $\frac{2\pi 18\pi(k + \varepsilon/3)}{\varepsilon^2} = \frac{36\pi^2(k + \varepsilon/3)}{\varepsilon^2}$. Since $k + \varepsilon/3$ is at most 1, the number of segments is in $O(\varepsilon^{-2})$. \square

Now finding Q' is simply a matter of finding a set of intervals such that there exists one interval on each disk which has strong visibility with an interval on all the other disks, and then selecting one point from each interval. A solution has the form $Q' = \{q_1 \dots q_k\}$, where q_i is a point on $C(p_i, k^g)$ in the interval contained in the solution.

We use a dynamic programming algorithm to find a solution given a set of disks in the input polygon. We use a table $A[i, j]$ that stores, for a pair of intervals i and j in different disks, a range of possible solutions that include those intervals (see Figure 8.5). We find the convex polygon that includes intervals i and j by combining two convex polygons, one that includes i and an interval α^* and another that includes j and α^* . In order to compute $A[i, j]$ we lookup the entries for $A[i, \alpha_1] \dots A[i, \alpha_m]$ and $A[\alpha_1, j] \dots A[\alpha_m, j]$, where $\alpha_1, \dots, \alpha_m$ are the intervals of a disk α ,

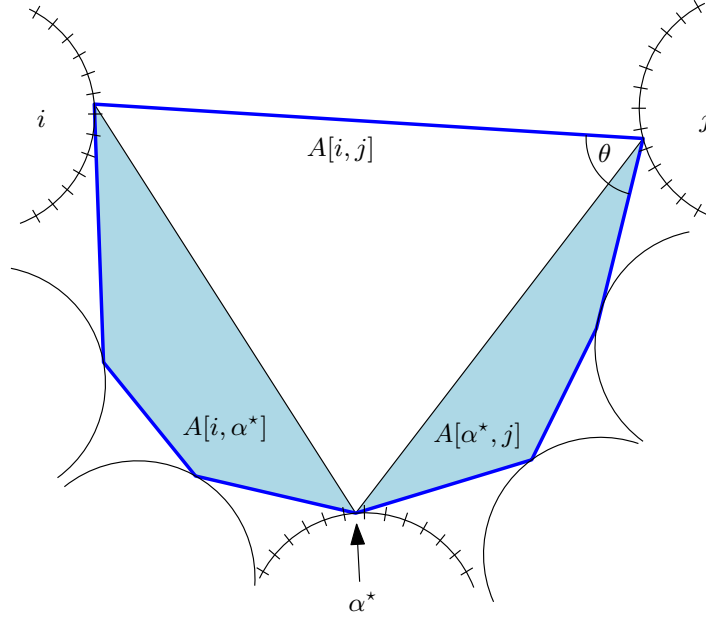


Figure 8.5: The convex polygon that includes intervals i and j is built by combining a polygon that includes i and α^* and one that includes j and α^* (the shaded polygons).

to determine if there exists a α^* for which there are solutions $A[i, \alpha^*]$ and $A[\alpha^*, j]$ that may be combined into a convex polygon. There are many solutions that include a certain pair of intervals, but we store only $O(n)$ solutions for each pair. For example, for the entry $A[i, j]$ we would store the edge coming out of j that minimizes the angle θ for each choice of an edge coming out of interval i , as shown in Figure 8.5. This would be done recursively at each level, which would make partial solutions easier to combine with other solutions while keeping convexity. Note that a particular choice of pairs of disks to form the solution Q' corresponds to a triangulation of Q' , and since there are $O(n)$ pairs of vertices joined in the triangulation, we need to store entries for the intervals of $O(n)$ pairs of disks. Given the clique of strongly visible intervals, we may now freely choose a point from each interval to obtain the solution polygon Q' . We run the dynamic programming algorithm iteratively for each $q_p \in P_V$, using only disks centred on vertices $v \in P_V$ where $\text{dist}(v, q_p) \geq k$. If no solution Q' is found for any q_p , then there is no solution where $H(P, Q) = k$.

We present the following observations pertaining to Q and Q' :

- $\exists Q \Rightarrow \exists Q'$, $\neg \exists Q' \Rightarrow \neg \exists Q$. The intervals are defined such that at least one interval from each disk will be contained in Q' .
- $\exists Q' \not\Rightarrow \exists Q$. The existence of Q' does not imply the existence of Q because the optimal solution may have disks of radius $k + \nu$, where $\nu < \varepsilon/3$.

8.3 The Minimization Problem

If there exists an optimal solution polygon Q where $H(P, Q) = k_{\text{OPT}}$, our algorithm finds an approximate solution Q' such that $H(P, Q') < k_{\text{OPT}} + \varepsilon$. To determine a value of k' such that

$k' \leq k_{\text{OPT}} + \varepsilon$, it suffices to perform a binary search over possible values for k' in the range of $[0 \dots 1]$, executing the decision approximation algorithm at each iteration. At the i^{th} iteration of the algorithm, let the current radius be k_i . If the algorithm finds a solution Q_i such that $H(P, Q_i) = k_i$, we shrink the disks and use $k_{i+1} = k_i - 1/2^i$. If the algorithm fails to find a solution, we use $k_{i+1} = k_i + 1/2^i$. Initially, $k_0 = 1$, and the stopping condition for the parametric search is met when we find an approximate solution for radius k , and the approximate decision algorithm fails for radius $k - \varepsilon/3$. Thus, the minimization version of the approximation algorithm requires $O(\log(\varepsilon^{-1}))$ iterations of the decision algorithm to find a solution. In the decision version, we showed that $H(Q, Q') < 2\varepsilon/3$ (in the case that Q' is a line), if Q exists. In the minimization version, the best solution for a value of k may approach $\varepsilon/3$ less than the optimal value located on one of the radius intervals. Therefore, the minimization algorithm returns a solution Q' where $H(P, Q') < k_{\text{OPT}} + \varepsilon$.

8.4 Running Time and Space Requirements

We begin by analyzing the space requirements and running time of the approximate decision algorithm. We compute the 1-centre using the technique in [22], which takes $O(n^2)$ time (see Section 6.2.11). The single line solution tests a line against $O(n)$ disks, each of which may have $O(n)$ segments. This procedure is repeated $O(n)$ times, so this requires $O(n^3)$ time in total. In the dynamic programming table, there are $O(n)$ pairs of disks. The number of intervals on each disk is bounded by $O(\varepsilon^{-2})$, so we have $O(\varepsilon^{-4})$ possible combinations of intervals between two disks. Therefore there are $O(n\varepsilon^{-4})$ entries in the table, and each of them stores a description of $O(n)$ solutions. Hence the table needs roughly $O(n^2\varepsilon^{-4})$ space. If the number of entries in the table is $O(n\varepsilon^{-4})$, the dynamic programming algorithm should run in time $O(n\varepsilon^{-6})$, since in order to calculate each entry we need to check all the $O(\varepsilon^{-2})$ intervals of one disk. The algorithm may require $O(n)$ iterations to test each value of q_p , so the approximate decision algorithm requires $O(n^3 + n^2\varepsilon^{-6})$ time. Finally, the minimization version of the algorithm performs $O(\log(\varepsilon^{-1}))$ iterations of the approximate decision algorithm, so the complete algorithm requires $O(\log(\varepsilon^{-1})(n^3 + n^2\varepsilon^{-6}))$ time to find a Hausdorff Core with Hausdorff distance $k_{\text{OPT}} + \varepsilon$, where k_{OPT} is the value of an optimal solution.

8.5 An FPTAS for the Hausdorff Core Decision Problem

Definition 8.1. Fully Polynomial Time Approximation Scheme (FPTAS): *An FPTAS is a family of approximation algorithms for a minimization problem A , so that for any constant $\varepsilon > 0$, there exists a $1 + \varepsilon$ -approximation algorithm for A , where the running time of the algorithm is polynomial in both $1/\varepsilon$ and the size of the input instance. Analogously, for a maximization problem, an FPTAS is a family of $1 - \varepsilon$ -approximation algorithms with running times of the same form.*

Modifying the decision algorithm of Section 8.2 to admit an FPTAS rather than an additive approximation factor is straightforward. In this case, we begin by determining whether there exists a circular segment C_φ which covers a Hausdorff Core solution for P , but now we use arc length $k\varepsilon/2$. This way, our algorithm will find a line segment which is a Hausdorff Core solution with Hausdorff distance $(1 + \varepsilon)k_{\text{OPT}}$ if one exists, where k_{OPT} is the value of an optimal solution.

Otherwise, the dynamic program requires $\frac{4\pi^2(1+\varepsilon)}{\varepsilon^2} \in O(\varepsilon^{-2})$ equal intervals so that at least one interval is contained in any solution when the disks are grown to have radius $(1 + \varepsilon/2)k$. The running time is asymptotically the same as the previous approximation algorithm. However, we are not able to perform a parametric search in the same way as before, as this could have an arbitrarily large running time relative to the sizes of the inputs. Since the running time of the decision algorithm remains $O(n^3 + n^2\varepsilon^{-6})$, this is an FPTAS for the decision version of the Hausdorff Core problem.

8.6 Conclusions and Future Work

We have described an algorithm which computes a Hausdorff Core Q of a simple polygon P with Hausdorff distance $H(P, Q) < k_{\text{OPT}} + \varepsilon$, where k_{OPT} is the value of the optimal solution, and ε is a fraction of d_{rad} (the distance from a constrained 1-centre to the most distant vertex in P). The running time of the algorithm is $O(n^3 + n^2\varepsilon^{-6}) \log(\varepsilon^{-1})$ time for the optimization version. We extended this by describing an FPTAS for the decision version of the Hausdorff Core problem. These are the first known algorithms for the Hausdorff Core problem on general simple polygons.

For future work, it would be interesting to explore other metrics. We studied the Hausdorff metric, but any of the other metrics discussed in Section 6.2.6 could be used. In our original application, we envisioned the creation of a hierarchy of simplified polygons, from full-resolution contour lines down to the simplest possible approximations. This would permit testing paths against progressively more accurate (and more expensive) approximate representations of polygons until we found a definitive answer regarding whether the path and polygon intersect. Our definition of d -core (including the Hausdorff Core) requires the solution to be convex. While convexity has many useful consequences, it represents a compromise to the original goal because it only provides one non-adjustable level of approximation. It would be interesting to consider other related problems that might provide more control over the approximation level. Another direction for further work would be to define some other constraint upon the simplified polygon. For instance, we could require that it be star-shaped, i.e. there exists some point $p \in P$ such that every $q \in P$ can see p . A similar but even more general concept might be defined in terms of link distance.

Finally, it would be interesting to study other computational geometry problems in a setting similar to Invariant 8.1. For example, in the following chapter, we study a variant of the Minimum Spanning Tree problem, where each point in the tree is selected from a disk centred on a point in the input set.

Minimum Spanning Trees with Neighborhoods

In the previous chapters, the objective involved piercing a set of disks with a convex polygon. We now turn to the problem of piercing a set of disks with a Minimum Spanning Tree. This problem is best framed in the context of a geometric problem on imprecise data¹. In this setting, for each point of the input we are provided with a *region of uncertainty*, i.e., a geometric object such as a line, disk, set of points, etc., and the exact position of the point may be anywhere in the object. Each object is understood to represent the set of possible positions for the corresponding point. In this chapter, we consider the Euclidean Minimum Spanning Tree (MST) problem. Given a tree T , we define its weight $w(T)$ to be the sum of the weights of the edges in T . For a set of fixed points P in Euclidean space, the weight of an edge is the distance between the endpoints, and we write $mst(P)$ for the weight of the MST on P . Thus, $mst(P) = \min w(T)$, where the minimum is taken over all spanning trees T on P .

Given a set of disjoint disks as input, we wish to determine the minimum and maximum weight MSTs possible when a point is chosen in each disk. The minimum weight MST version of the problem has been studied previously, and is known as the Minimum Spanning Tree with Neighborhoods² problem (MSTN). We introduce the maximum weight MST version of the problem, which we call the MAX-MSTN problem. These problems represent upper and lower bounds respectively on the weight of a Minimum Spanning Tree in the model where one point is chosen in each disk.

Definition 9.1. The MSTN Problem: *Assume we are given a set $D = \{D_1, \dots, D_n\}$ of disjoint disks in the plane, i.e., $D_i \cap D_j = \emptyset$ if $i \neq j$. The MSTN problem on D asks for the selection of a point $p_i \in D_i$ for each $D_i \in D$ such that the weight of the MST of the selected points is minimized.*

Definition 9.2. The MAX-MSTN Problem: *Given a set $D = \{D_1, \dots, D_n\}$ of disjoint disks in the plane, the MAX-MSTN problem is to compute a set of points P , where $p_i \in D_i$ and $|P| = n$, such that the weight of the MST of the selected points is maximized.*

¹Elements of this chapter have appeared in [54].

²While this thesis is written using Canadian spelling, we use the American spelling of ‘neighborhood’ when referring to the MSTN and MAX-MSTN problems, because this is established nomenclature in the literature.

9.1 Related Work

The first known MST algorithm was published over 80 years ago [98], and a number of successful variants have followed (see [85] for the history of the problem). A review of models of uncertainty and data imprecision for computational geometry problems is provided in [113]. Here, we discuss a few results that are directly related to the MST problem and our model of imprecision.

The MSTN problem on disjoint unit disks has been shown to admit a PTAS [166]. An NP-hardness proof for a generalization of MSTN where the neighborhoods are either disks or rectangles appeared in [166]. This proof was faulty however, and one of the authors later conjectured that a reduction from planar 3-SAT might be used to show the NP-hardness of the MSTN problem [165, p.106]. In Section 9.4.3, we prove this conjecture. The hardness of MSTN is of interest, and the previous result has been cited a number of times, e.g. [30, 35].

When regions of uncertainty are modelled as disks or squares, even the problem of maximizing the smallest pairwise distance in a set of n imprecise points is NP-hard [69].

Löffler and van Kreveld [113] discussed two results on computing Minimum Spanning Trees under the model of uncertainty that we are considering. First, they demonstrated that it is algebraically difficult (see Section 9.2) to compute the MST when the regions of uncertainty are continuous regions of the plane, even for very simple inputs such as disks or squares. Secondly, they demonstrated that the problem is also NP-hard if the regions of uncertainty are not pairwise disjoint, through a reduction from the minimum Steiner tree problem. In this thesis we prove the hardness of the special case in which the regions are pairwise disjoint.

Erlebach et al. [62] used a model of uncertainty where information regarding the weight of an edge between a pair of points or the position of a point may be obtained by pinging the edge or vertex, and they sought to minimize the number of pings required while obtaining the optimal solution. The distinction is that in their work, they were interested in reducing the amount of communication that is required to locate points within a region of uncertainty, while in our model, the objective is to optimize the MST given regions of uncertainty.

The minimum k -spanning tree (k -MST) problem asks for the minimum weight MST over a set of $k \leq n$ points, where n is the number of vertices in the input. This was shown to be NP-hard by reduction from the Steiner tree problem [70]. The Euclidean version of k -MST was shown to admit a PTAS by Arora [11], as a corollary of his traveling salesman problem result.

The travelling salesman with neighborhoods (TSPN) problem has been studied extensively. The problem was introduced by Arkin and Hassin [9], in a paper that has been applied, improved, built-upon or otherwise referenced over 150 times. There exists a PTAS for TSPN when the neighborhoods are disjoint unit disks [57]. The most general version of the problem, where regions may overlap and may have varying sizes, is known to be APX-hard [50].

Löffler and van Kreveld [113] studied the convex hull problem on imprecise points in detail, and they presented a number of new results. We provide a sampling of these results in Table 9.1. Their paper includes many results beyond those mentioned in the table; these are simply intended to provide a flavour of the range of problems and their complexities.

We present a variety of results related to the MSTN and MAX-MSTN problems. For both problems we assume the regions of uncertainty are disjoint.

- MAX-MSTN: 1/2-approximation algorithm;

Table 9.1: Algorithms/Hardness of Convex Hulls on Various Models of Imprecision [113]

Setting	Running Time/Hardness
<i>Largest area on:</i>	
parallel line segments	$O(n^3)$
non-intersecting line segments in convex position	$O(n^3)$
line segments	NP-hard
non-intersecting squares	$O(n^7)$
non-intersecting, equal size squares	$O(n^3)$
equal size squares	$O(n^5)$
<i>Largest perimeter on:</i>	
parallel line segments	$O(n^5)$
line segments	NP-hard
non-intersecting squares	$O(n^{10})$
equal size squares	$O(n^{13})$
<i>Smallest area on:</i>	
parallel line segments	$O(n \log n)$
squares	$O(n^2)$
<i>Smallest perimeter on:</i>	
parallel line segments	$O(n \log n)$
squares	$O(n \log n)$

- MAX-MSTN: parameterized $\left(1 - \frac{2}{k+4}\right)$ -approximation algorithm (where k represents the separability of the instance);
- MAX-MSTN: proof of NP-hardness;
- MSTN: parameterized $(1 + 2/k)$ -approximation algorithm (k is the separability of the instance);
- MSTN: proof of NP-hardness.

The approximation algorithm for MAX-MSTN (Section 9.3.1) is based on choosing the centre points of the disks; the interesting aspect in this section lies in the analysis. The parameterized algorithms (Sections 9.3.2 and 9.4.2) for both settings were inspired by the observation that the approximation factor improves rapidly as the distance between disks increases. To address this, we introduce a measure of how much separation exists between the disks, which we call *separability*, and we analyze the approximation factor of the MST on disk centres with respect to separability.

For both hardness results, we establish that not only are the problems NP-hard, but also that there is no FPTAS for the problems unless $P=NP$. Although the hardness proofs both consist of reductions from planar 3-SAT, the gadgets used are quite distinct and either reduction is interesting even given the existence of the other. In both cases, we construct an instance of our problem from the planar 3-SAT instance and then show that the weight of an optimal solution to the given problem on our construction may be determined *a priori*, using the assumption that the

3-SAT instance is satisfiable. If the instance is not satisfiable, we prove that the weight is changed by at least a constant amount (reduced by at least 0.33 units for MAX-MSTN, and increased by at least 0.66 units for MSTN).

9.2 Algebraic Complexity

There are two main types of complexity that one encounters when working with optimization problems in computational geometry: algebraic complexity and combinatorial complexity. In this section, we provide a brief discussion of the former.

Many geometric optimization problems that are simple to formulate and conceptualize can nonetheless have solutions that are difficult to compute. Borrowing from an example in [14] and [113], consider the problem of computing a Minimum Spanning Tree over a set of points, where the tree consists of one root vertex of degree k and k leaf vertices of degree one. If we know the positions of all of the leaves and simply wish to compute an optimal position for the root vertex, then this is an algebraically difficult problem. Formally, suppose we have points $P = \{p_1, \dots, p_k\}$, where $p_i = (x_i, y_i)$, and we wish to place the point $c = (x, y)$ in the plane. Then the optimal weight w of the tree is

$$w = \min_{x,y} \sum_{i=1}^k \sqrt{(x_i - x)^2 + (y_i - y)^2} \quad (9.1)$$

The problem is well known as the Fermat-Weber problem, and Bajaj [14, Table 1] shows that even for $k = 5$, that the solution to this problem requires the determination of the roots of a polynomial of degree 8, for which there is no known solution. The problem was first proposed by Fermat for problems where $k = 3$ [51], while exact solutions to the settings for $k = 3$ and $k = 4$ were found by Cavalieri in 1647 (with a tightening by Heinen in 1834) and Fagnano in 1775, respectively [13].

Durocher and Kirkpatrick [58] studied the Fermat-Weber problem, and they suggest several approximation schemes which maintain a robust approximation while removing the algebraic complexity from the problem. The most basic approach is to compute the centre of mass of the points, which provides a 2-approximation for the position of the Fermat-Weber point (where the value of the optimal solution is the value of w in Equation 9.1).

9.2.1 Euclidean MST Problems are Sum-of-Square-Roots-Hard

We note that Euclidean MST problems are not known to be in NP, since comparing two candidate solutions is an instance of the sum of square roots problem³, which is not known to be in NP. The sum of square roots problem accepts two sequences of integers as input, $A = a_1, \dots, a_m$ and $B = b_1, \dots, b_n$, and is asked to determine whether $\sum_{i=1}^m \sqrt{a_i} > \sum_{i=1}^n \sqrt{b_i}$. While the status of the general version of the sum of square roots problem is unknown, there is a randomized polynomial time algorithm to determine whether $\sum_{i=1}^m \sqrt{a_i} = \sum_{i=1}^n \sqrt{b_i}$ [20]. Every MST problem in the thesis is studied under the Euclidean distance metric, and so are generalizations of the standard Euclidean MST problem and are not known to be in NP. We circumvent this hardness issue in the usual fashion, by working in the real RAM model.

³For a discussion of sum-of-square-roots-hard problems, see [156].

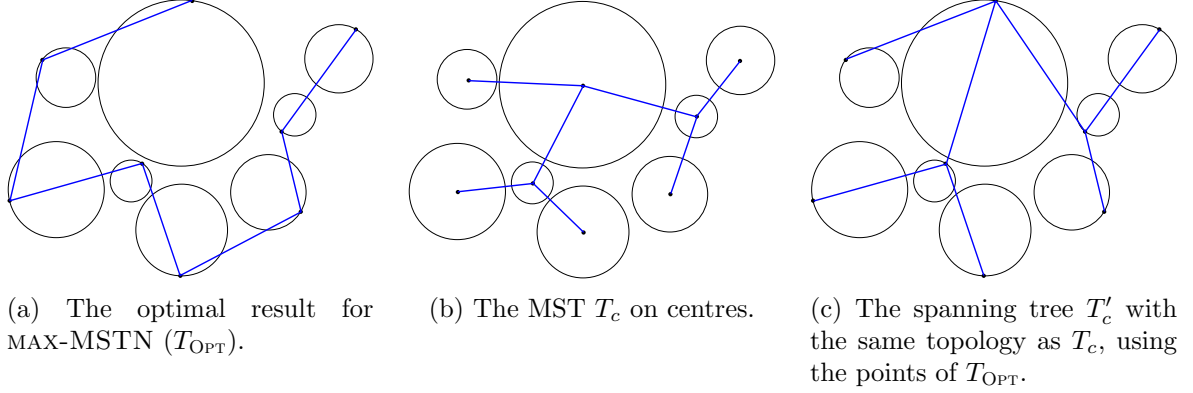


Figure 9.1: To compare $w(T_c)$ with $w(T_{\text{OPT}})$, we use an intermediate tree T'_c .

9.3 MAX-MSTN

In this section we study a few approximation algorithms for the MAX-MSTN problem, and then we present the proof of hardness of approximation. We begin with a $1/2$ -approximation algorithm below, followed by a parameterized algorithm in Section 9.3.2.

9.3.1 $1/2$ -Approximation Algorithm

To approximate the solution to MAX-MSTN, we first consider the algorithm that builds an MST on the centres of the disks. We show this algorithm approximates the optimal solution within a factor of $1/2$, i.e., the weight of the MST built on the centres is not smaller than half of that of the optimal tree.

Theorem 9.1. *Consider the MAX-MSTN problem for a set D of disjoint disks. Let T_c denote the MST on the centres of the disks, and let T_{OPT} be the maximum MST (i.e., the optimal solution to the problem). Then $w(T_c) \geq 1/2 \cdot w(T_{\text{OPT}})$.*

Proof. Let T'_c be the spanning tree (not necessarily an MST) with the same topology (i.e., combinatorial structure of the tree) as T_c but on the points of T_{OPT} (see Figure 9.1). Since T'_c and T_{OPT} span the same set of points, and T_{OPT} is an MST, we have $w(T_{\text{OPT}}) \leq w(T'_c)$. On the other hand, since T'_c and T_c have the same topology, we have $w(T'_c) \leq 2w(T_c)$; this is because when we move the points from the centre to somewhere else in the disks, the weight of each edge increases by at most the sum of the radii of the two involved disks and, since the disks are disjoint, the increase is at most equal to the original weight. To summarize, we have $w(T_{\text{OPT}}) \leq w(T'_c)$ and $w(T'_c) \leq 2w(T_c)$, which completes the proof. □

9.3.2 Parameterized $(1 - \frac{2}{k+4})$ -Approximation Algorithm

Observe that in order to get the approximation algorithm for MAX-MSTN in Section 9.3.1, we require the disks to be disjoint. Intuitively, if we know that the disks are further apart, we can get

better approximation ratios. We formalize this intuition by providing a parameterized analysis, i.e., we express the performance of the algorithm in terms of a *separability parameter*⁴. Let r_{\max} be the maximum radius of any disk in the input set. We say that a given input for our problem satisfies *k-separability* if the minimum distance between any two disks is at least $k \cdot r_{\max}$. The separability of an input instance I is defined as the maximum k such that I satisfies *k-separability*. With this definition, we have the following result:

Theorem 9.2. *For MAX-MSTN when the regions of uncertainty are disjoint disks with separability parameter $k > 0$, the algorithm that builds an MST on the centres of the disks achieves a constant approximation ratio of $\frac{k+2}{k+4} = 1 - \frac{2}{k+4}$.*

Proof. Let T_c be the MST on the centres of the disks. We can extend the analysis in the proof of Theorem 9.1 to show that the approximation factor is $\frac{k+2}{k+4} = 1 - \frac{2}{k+4}$ for any input that satisfies *k-separability*. Define T_{OPT} and T'_c as before. Consider an arbitrary edge e in T'_c and let D_i and D_j be the two disks connected by e . Let r_i and r_j be the radii of D_i and D_j , respectively, and let d be the distance between D_i and D_j . In T_c the disks D_i and D_j are connected by an edge e' whose weight is $d + r_i + r_j$. The weight of e , on the other hand, can be at most $d + 2r_i + 2r_j$. Therefore, the ratio between the weight of an edge in T_c and its corresponding edge in T'_c is at least

$$\frac{d + r_i + r_j}{d + 2r_i + 2r_j} \geq \frac{kr_{\max} + r_i + r_j}{kr_{\max} + 2r_i + 2r_j} \geq \frac{kr_{\max} + r_{\max} + r_{\max}}{kr_{\max} + 2r_{\max} + 2r_{\max}} = \frac{k + 2}{k + 4}.$$

Since this holds for any edge of T'_c , we get $w(T_c) \geq \frac{k+2}{k+4}w(T'_c) \geq \frac{k+2}{k+4}w(T_{\text{OPT}})$, and we get an approximation factor of $\frac{k+2}{k+4} = 1 - \frac{2}{k+4}$. \square

The approximation ratio gets arbitrarily close to 1 as k increases. This confirms our intuition that if the disks are further apart (more separate), we get a better approximation factor.

9.3.3 NP-Hardness of max-MSTN

We present a hardness proof for the MAX-MSTN problem with a reduction from the planar 3-SAT problem. Planar 3-SAT is a variant of 3-SAT in which the graph $G = (V, E)$ associated with the formula is planar.

Theorem 9.3. *MAX-MSTN is NP-hard, and it does not admit an FPTAS unless $P=NP$.*

We show a reduction from any instance of the planar 3-SAT problem to the MAX-MSTN problem. In planar 3-SAT, we have a planar bipartite graph $G = (V, E)$, where $V = V_v \cup V_c$, so that there is a vertex in V_v for each variable and a vertex in V_c for each clause; there is an edge (v_i, v_j) in E between a variable vertex $v_i \in V_v$ and a clause vertex $v_j \in V_c$ if and only if the clause contains a literal of that variable in the 3-SAT instance. In [111] it was shown that the planar 3-SAT problem is NP-hard via a reduction from the standard 3-SAT problem. Further, it was observed that the resulting instance of planar 3-SAT permits the addition of a path through all the vertices V_v while maintaining planarity. We call this path the *spinal path* and denote it by $P = (V_v, E_P)$. We further observe that additional edges can be added to P to get a *spinal tree* T which also covers clause vertices V_c . In this sense T will be a tree that covers all vertices without crossing an edge of G such that all vertices corresponding to clauses are leaves. These

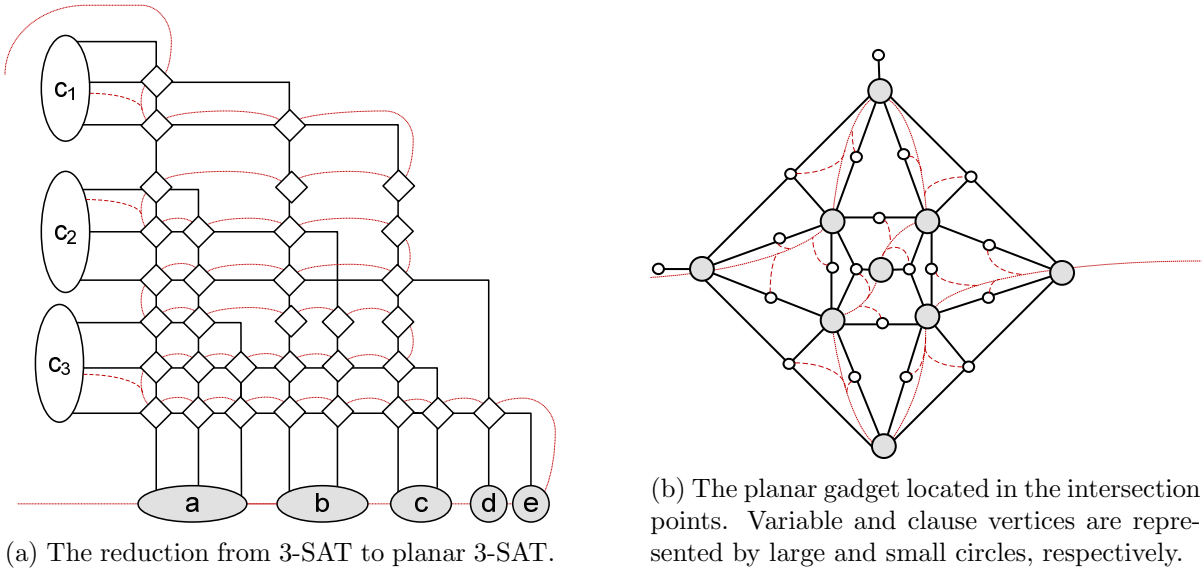


Figure 9.2: The reduction from 3-SAT to planar 3-SAT as presented in [111]. (a) The variable and clause vertices of 3-SAT are located on the x and y axes respectively, and the edges are drawn as orthogonal paths. (b) A planar gadget is placed on each intersection point. Each gadget introduces some new variable and clause vertices. In [111], it is observed that there is a path (we call it the spinal path) that covers all variable vertices of the planar instance without crossing any edge (solid lines). We observe that additional edges can be added to the spinal path to obtain a tree (spinal tree) which spans clause variables as leaves (dashed lines).

observations are illustrated in Figure 9.2. To prove the hardness of MAX-MSTN, we make use of the spinal tree.

We use the following theorem to ensure that a polynomial number of disks are required for the reduction:

Theorem 9.4. [19, Theorem 4] *Let G be a simple graph without nodes of degree ≤ 1 . Then G has an orthogonal drawing in an $\frac{m+n}{2} \times \frac{m+n}{2}$ -grid with one bend per edge. The box size of each node v is at most $\frac{\deg(v)}{2} \times \frac{\deg(v)}{2}$. It can be found in $O(m)$ time.*

We use an orthogonal drawing of the planar 3-SAT instance in our reduction, although we will expand the drawing by a factor of 2 so that all edges of the drawing are separated by at least 2 units. All edges of the drawing will be replaced by *wires* in the reduction, where a wire consists of a set of points (disks of radius 0) placed along an edge of the drawing every unit distance. This ensures that each wire has a unique MST over the points, and that the number of points required for all wires is polynomial in the size of the input. The gadgets used in the reduction may not fit within boxes of size $\frac{\deg(v)}{2} \times \frac{\deg(v)}{2}$, but they are within a constant factor of this bound. The variable gadgets require boxes of size at most $15.75(\deg(v) + 1) \times 11$ (we discuss these gadgets shortly), while clause gadgets require boxes of (constant) size 108×52 units. Therefore, the size of the drawing remains polynomial in the size of the input provided the gadgets also require a polynomial number of disks (and we will show that this is the case).

⁴Separability is similar in spirit to the notion of a well-separated pair; see [25].

Next, we study an important theorem for our reduction which states that an optimal solution for MAX-MSTN on a *chain* of unit disks has a characteristic zigzag pattern. A chain is a set of k unit disks $\mathcal{D}^c = \{D_1^c, \dots, D_k^c\}$ whose centres are collinear, incident upon a horizontal line ℓ_{hz} , and adjacent disk centres are $2d$ (with $d \geq 1$) units distant from each other. Furthermore, there are two terminal points t_l and t_r incident upon ℓ_{hz} , where t_l is located d units to the left of the centre of the leftmost disk, and t_r is located d units to the right of the centre of the rightmost disk of the chain.

Theorem 9.5. *Given a chain of disks, the solution to the MAX-MSTN problem on the chain \mathcal{D}^c and the points t_l and t_r is the set of points $\{t_l, p_1, \dots, p_k, t_r\}$, where p_i is the selected point for disk D_i^c , and these points form one of the two possible zigzag paths that traverse the extreme top and bottom points of the disks (see Figure 9.3).*

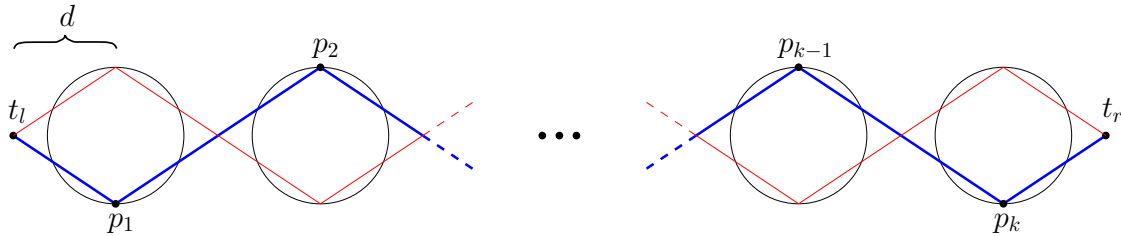


Figure 9.3: The two possibilities for the MAX-MSTN solution for chain of disks. Here we have $d = 1.5$.

We define Z_D as this proposed zigzag path that alternates between the extreme upper and lower points of a set of disks \mathcal{D}^c , where the centres of all disks in \mathcal{D}^c are collinear. Note that the MST for the chain of disks with points in such positions forms a polygonal chain starting from t_l and ending at t_r .

To prove Theorem 9.5, we require a few additional lemmas.

Lemma 9.1. *Given a disk, if the set of edges containing a point in the disk is fixed for any position of the point, any MAX-MSTN solution does not contain a point inside the disk (i.e., the selected point is on the circumference of the disk).*

Proof. Suppose we are given a point p in a disk D , and a set of points Q , $|Q| \geq 1$, where there exists an edge between p and each point in Q . Given any line ℓ , where $p \in \ell$, let p' be a point on $\ell \cap D$. Let w be the sum of the weights of edges between p' and each point in Q . The weight of each edge (as p' runs along ℓ) is a convex function, and therefore so is the sum w [144]. Hence, the sum of the weights of the edges is maximized at one of the two intersection points of ℓ with D .

□

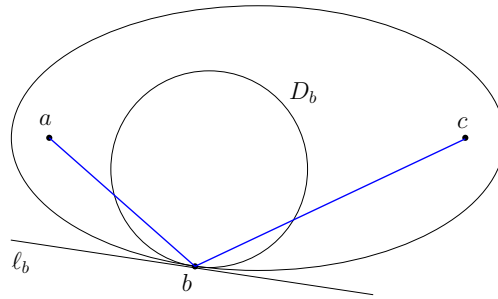
The following lemma states that an optimal MAX-MSTN solution follows the path of a ray initiated at point t_l and reflected at a point of intersection with each disk.

Lemma 9.2. *Let p_1 be the selected point of a MAX-MSTN solution on the leftmost disk in a chain. Consider the ray $t_l p_1$ which is then reflected on the interior face of each disk. Any valid MAX-MSTN solution follows the path traversed by the ray, i.e., the two neighboring segments of the MST are the reflection of each other on the tangent line of the intersection point.*

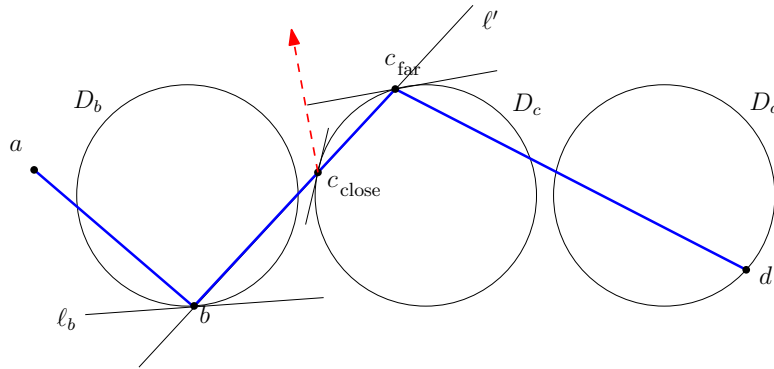
Proof. Let b be the selected point of any disk in a MAX-MSTN solution and a and c be the points of the MST connected to b on the left and right in the tree, respectively (for the leftmost disk we have $b = p_1$ and $a = t_l$). Let D_b be the disk such that $b \in D_b$, and by Lemma 9.1, b is on the circumference of D_b . Let ℓ_b be the line tangent to D_b at point b . For this proof we refer to the diagram in Figure 9.4a. By definition, the ellipse with foci a and c consists of the set of points \mathcal{P} that are equidistant, on the aggregate, to a and c , i.e., $\text{dist}(a, p) + \text{dist}(p, c)$ is a constant for all $p \in \mathcal{P}$. Points inside the ellipse are closer, on the aggregate, to a and c while points outside the ellipse are farther.

If the disk D_b and an ellipse incident upon b are not tangent at b , it follows that there are points in D_b outside the ellipse and hence the length of the MST grows if such a point is chosen instead of b , which contradicts the assumption that the MST through b is maximal. Since a and b are the foci, the projective property of the ellipse implies that \vec{bc} is the reflection of \vec{ab} on the tangent ℓ_b to the ellipse.

Let $c_{\text{close}}, c_{\text{far}}$ be the intersection points of a line ℓ' incident upon b and disk D_c , so that c_{close} is closer to b than c_{far} . If we apply the same argument by replacing a with b and b with c_{close} , we see that the reflected path is located to the left of the line \vec{bc}_{far} , which is not a feasible path (e.g., the red line in Figure 9.4b). This implies that the MAX-MSTN solution cannot contain c_{close} , hence, c can only be located at c_{far} . The proof is complete if we apply this argument by setting $a = t_l$ and $b = p_1$ to set $c = c_{\text{far}}$ and then inductively apply the same reasoning by replacing a with b and b with c . \square



(a) The ellipse with foci a and c and incident upon b should be tangent to D_b if \overline{ab} and \overline{bc} are edges in a MAX-MSTN solution.



(b) If the MAX-MSTN solution selects point b , it has to select point c_{far} .

Figure 9.4: The reflection effect.

Lemma 9.2 implies that selecting the first point p_1 on the leftmost disk D_1 defines all other points that are selected by the MAX-MSTN algorithm. In particular, if p_1 is the extreme top or bottom point of D_1 , then the points selected for all other disks will be on the extreme top or bottom points of the disks as well, which produces the Z_D configuration described in Theorem 9.5. Note that if the reflected line on any disk does not intersect the next disk to the right, or if at the rightmost disk the reflected line is not incident upon the terminal point t_r , then in these cases the initial selection for p_1 does not produce a MAX-MSTN solution (Figure 9.5).

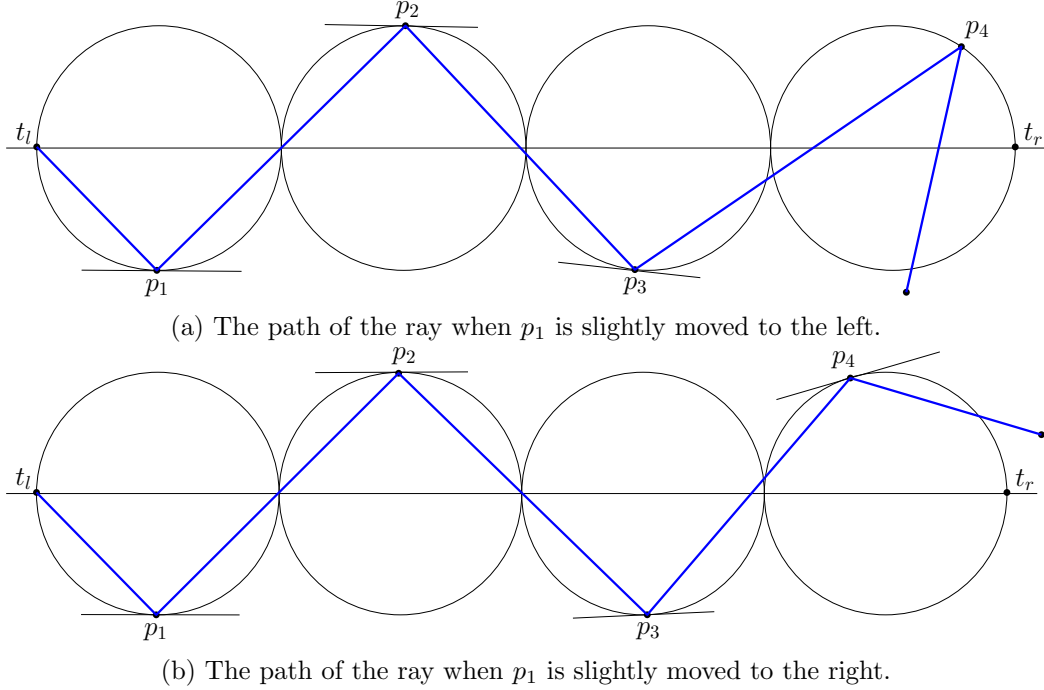


Figure 9.5: If the selected point p_1 on the leftmost disk is not the extreme top or bottom point, then the path of the ray does not pass through t_r .

Lemma 9.3. *Let $\mathcal{P} = \{p_1, p_2, \dots, p_n\}$ be the selected points of a MAX-MSTN solution on the consecutive disks in a horizontal chain, from left to right. Then for any $i \in \{1, \dots, n\}$ we have the following two properties:*

1. *If p_i is on the bottom half of disk D_i , then p_{i+1} is on the top half of disk D_{i+1} and vice versa.*
2. *If p_i is on the right half of disk D_i , then p_{i+1} is on the left half of disk D_{i+1} and vice versa.*

Proof. To establish the first property, let p_i and p_{i+1} both be on the bottom halves of two consecutive disks. A larger MST can be found by replacing all points p_j ($j \geq i + 1$) with their reflection over ℓ_{hz} (the horizontal line passing through the disk centres). Such a transformation increases the weight of the edge (p_i, p_{i+1}) and preserves the weight of other edges of the MST (the same holds if two selected points are on the top halves of consecutive disks). Using mathematical induction, the second property is direct from Lemma 9.2. More precisely, if p_{i-1} and p_i are on the right and left halves of their disks respectively, then Lemma 9.2 dictates that p_{i+1} is on the right

half of D_{i+1} . Similarly, if p_{i-1} and p_i are on the left and right halves of their disks respectively, then p_{i+1} is on the left half of D_{i+1} (this effect may be seen in Figure 9.5). \square

To prove Theorem 9.5, we show that if we select p_1 as any point except the extreme bottom or top points of the leftmost disk D_1 , then the path described in Lemma 9.2 does not pass through the terminal point t_r . For each disk $D_i \in D$, define the *canonical line* as the line passing through the centre of D_i and p_i , and the *canonical angle* α_i as the acute angle between ℓ_{hz} and the canonical line of the disk if the lines are not perpendicular. Note that Lemma 9.3 implies that if the canonical lines have defined slope, then either all canonical lines have positive slope or all have negative slope. To make the explanation simpler, consider another disk D_0 centred at a distance $2d$ to the left of the centre of the leftmost disk, and let p_0 be the intersection of D_0 with the line passing through t_l and p_1 (Figure 9.6). Since t_l is located at the midpoint of the centres of p_0 and p_1 , observe that the canonical angles of D_0 and D_1 are equal regardless of the choice of p_1 , i.e., $\alpha_0 = \alpha_1$.

Lemma 9.4. *If the selected point is any other point than the extreme top or bottom points of disk D_1 , the sizes of the canonical angles form a strictly decreasing sequence, i.e., $\pi/2 > \alpha_0 = \alpha_1 > \alpha_2 > \dots > \alpha_n$.*

Proof. We use mathematical induction. As observed before, we have $\alpha_0 = \alpha_1$. In the base case, we show $\alpha_1 > \alpha_2$. Consider otherwise, i.e., $\alpha_1 \leq \alpha_2$. Note that the slopes of all canonical lines are either positive or negative. Without loss of generality assume all slopes are negative (see Figure 9.6). Let p_x be the intersection point of the line passing through p_1 and p_2 with the horizontal line ℓ_{hz} . $\text{dist}(p_x, q_1)$ is the distance between p_x and the centre point q_1 of D_1 . Since $\alpha_1 \leq \alpha_2$, we have $\text{dist}(p_x, q_1) \leq d$. Let θ be the angle between lines $\overline{p_0 p_1}$ and $\overline{q_1 p_1}$, which is the same as the angle between $\overline{p_1 p_2}$ and $\overline{q_1 p_1}$ (by the reflection law). Applying the Law of Sines for triangle $\triangle q_1 p_1 p_x$ we get $\text{dist}(p_x, q_1) = \sin \theta / \sin(\alpha_1 - \theta)$. Let p'_x be the intersection point of $\overline{p_0 p_1}$ and horizontal line ℓ_{hz} (which is t_l in the base case). Since $\alpha_0 \geq \alpha_1$, we have $\text{dist}(p'_x, q_1) \geq d$ (equivalence occurs in the base case). Applying the Law of Sines on triangle $\triangle q_1 p_1 p'_x$, we get $\text{dist}(p'_x, q_1) = \sin \theta / \sin(\pi - \alpha_1 - \theta)$.

Since $\text{dist}(p_x, q_1) \leq d$ and $\text{dist}(p'_x, q_1) \geq d$, we get $\sin \theta / \sin(\alpha_1 - \theta) \leq \sin \theta / \sin(\pi - \alpha_1 - \theta)$, which implies that $\alpha_1 \geq \pi/2$; this cannot happen as we assumed α_0 is strictly smaller than $\pi/2$. So we conclude $\alpha_1 > \alpha_2$. To complete the proof, we can apply the same argument by replacing p_0, p_1 , and p_2 with p_{i-1}, p_i and p_{i+1} to show $\alpha_i > \alpha_{i+1}$ for $i > 2$. \square

Now we are ready to prove Theorem 9.5.

Proof. Lemma 9.2 implies that the MAX-MSTN solution should follow the path of a ray that is initiated at t_l , reflected at each point p_i of disk D_i , and finally passes through t_r . Note that the two Z_D (zigzag) paths stated in the theorem have this property. We show that no other ray can do so. Assume the selected point of the leftmost disk is not on the extreme top or bottom (otherwise the reflected path would be one of the paths described in the theorem). By Lemma 9.4, the canonical angles of the disks are strictly decreasing. This implies that for any disk D_i , the reflected path does not pass through the mid-point of the segment between q_i and q_{i+1} (the centres of D_i and D_{i+1} , respectively). Otherwise, the canonical angles of the two disks would be equal. In particular, if one assumes the existence of an extra disk on the right, the reflected

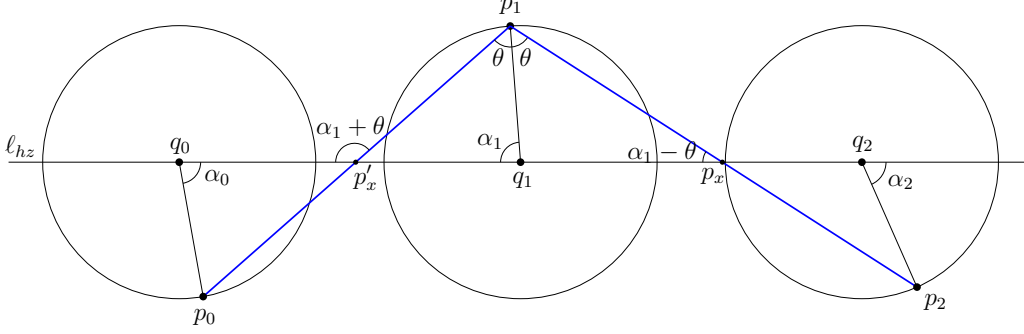


Figure 9.6: If $\alpha_0 < \pi/2$, the canonical angles are strictly decreasing.

path does not pass through t_r . This implies that selecting any point other than an extreme point on the top or bottom of the leftmost disk results in a non-optimal MAX-MSTN solution for the chain of disks. □

Now that we have established Theorem 9.5, we revisit the reduction. Recall that we are reducing to the planar 3-SAT problem, in which an instance of 3-SAT is represented as the planar graph $G = (V, E)$, and that we have a spinal tree $T = (V, E_T)$, where T is connected and $G \cup T = (V, E \cup E_T)$ remains planar while $E \cap E_T = \emptyset$ (Figure 9.2). Furthermore, we make use of wires, where a wire is a set of disks of radius 0 (i.e. points) placed in close succession so that we may interpret them as a fixed line in the MAX-MSTN solution.

For some intuition on the structure of the reduction, consider the graph $G \cup T$. In this graph, we replace all variable and clause vertices with variable and clause gadgets, respectively. Also, we replace the edges in $G \cup T$ with fixed wires so that they will be part of any MST, and these fixed wires are disjoint from the gadgets. The fixed wires correspond to the edges of G (which we call *e-wires*) and also the edges of the spinal tree. The clause and variable gadgets include some disks, and the goal of the MAX-MSTN algorithm is to select points in each disk so that the resulting MST has maximum weight. Note that any MST is composed of the disconnected fixed wires that are each connected to some gadgets. We design the gadgets so that the e-wires (i.e., the edges of G) attach exclusively to clause gadgets. The combinatorial structure of the resulting MST includes the spinal tree as a sub-tree (that is why we call it 'spinal'); hence, an optimal MAX-MSTN algorithm selects the points in gadgets in a way to impose the maximum weight for the edges that connect the e-wires to the spinal tree.

Variable Gadgets

For each variable x_i , we build a set \mathcal{D}^i consisting of $3c + 2$ disks in the configuration described in Theorem 9.5, spaced by $d = 21/8 = 2.625$, where c is the number of clauses containing instances of the variable (note that adjacent disk centres are $2d = 5.25$ units apart). The reasons for the particular choices of distances are explained in the Reduction section below. The terminal points t_l^i and t_r^i of the gadget are joined to the spinal tree of the construction. Specifically, a wire joins t_r^i to t_l^{i+1} , $i \in \{1, \dots, n-1\}$ for each of the n variable gadgets. Assume for ease of discussion that the centres of \mathcal{D}^i are incident upon a horizontal line ℓ_{hz} , so that the terms above, below, left, and right are well defined.

Wires from the clause gadgets may approach the variable gadget from above, below, or both. As mentioned earlier, we call these the *e-wires*, because each wire corresponds to an edge of E in the input planar 3-SAT graph. Each e-wire terminates at a point that is distance 6.5 units from a disk centre, along a line incident upon the disk centre and perpendicular to ℓ_{hz} , i.e., the terminal point of the wire and the disk centre share the same x-coordinate (see Figure 9.7). We provide more details regarding e-wires shortly. Finally, suppose without loss of generality that disk $D_j^i \in \mathcal{D}^i$ has the terminal points of an e-wire above it. All other e-wires are restricted so that no other e-wires may approach D_j^i , and furthermore, no e-wires may approach disks D_{j-1}^i or D_{j+1}^i from above, so that there is at least distance $4d = 10.5$ between adjacent e-wires. In other words, e-wires that approach the variable gadget from the same side have at least one disk between them.

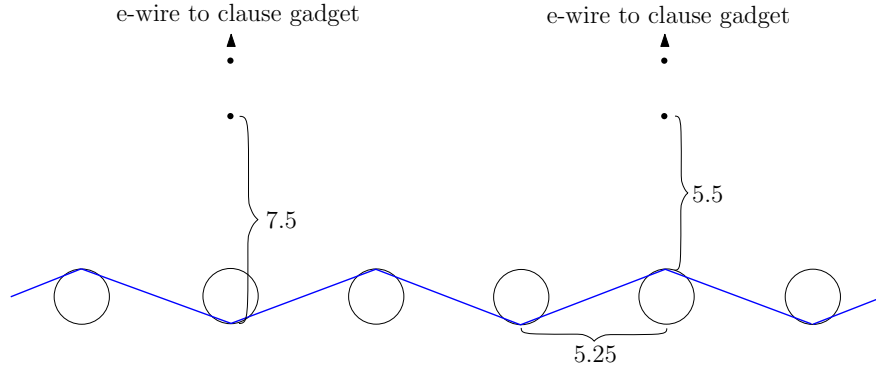


Figure 9.7: The configuration used for the variable gadgets. Here, two clauses include the variable in the opposite truth values. Filled small circles represent disks of radius 0, while larger empty circles represent unit disks. E-wires are placed so that the nearest point on any disk is at distance 5.5, and the terminal point of the e-wire is on a line that is vertical and incident upon the disk centre. This way, if the Z_D configuration for the variable gadget matches the truth value of the e-wire, then the nearest point to the e-wire in the gadget is 7.5 units distant, while a mismatch in truth values means the nearest point in the gadget would only be distance 5.5 away. The disk centres in the gadget are placed at a distance 5.25 apart so that the nearest point to an e-wire can only be from the nearest disk, regardless of the path through the disks.

Lemma 9.5. *Suppose we are given a variable gadget where points are placed in the disks as described in Theorem 9.5. Then a point in a disk is either distance 7.5 or 5.5 from the nearest point in an e-wire, and we may arrange it so that these distances correspond to agreement or disagreement respectively between the truth value of the variable gadget and that of the instance of the variable represented by the e-wire.*

Proof. Given the two possible Z_D configurations shown in Theorem 9.5, we arbitrarily select one of the configurations as the **true** setting, and the opposite configuration as **false**. This way, we can place the terminal points of the e-wires near the disks of the variable gadget so that if the truth value used for the variable gadget matches that of the e-wire, then the minimum distance between the e-wire and the nearest point in the variable gadget is 7.5. However, a mismatched truth value would mean that a point lies only distance 5.5 from the e-wire when the points are in the Z_D configuration.

□

Clause Gadgets

For each clause in the 3-SAT instance, we build a clause gadget by assembling unit disks and wires (composed of disks of zero radius spaced by two units) as shown in Figure 9.8. The unit disks are placed in chain, either horizontally or vertically, using $d = 4$ for spacing (i.e., disks are centred 4 units away from the two terminal points, and disks are placed with centres 8 units apart from each other). From the spinal tree, a wire joins to one of the terminal disks of a chain.

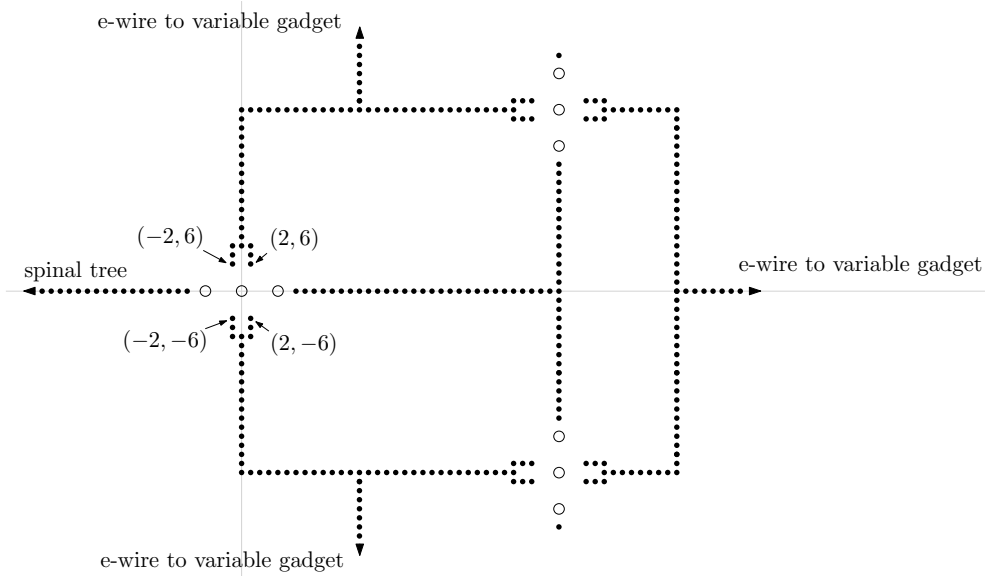


Figure 9.8: The configuration used for the clause gadgets, with the gadget placed in the canonical position. The filled small circles (black dots) are disks of radius 0, while the larger empty circles are unit disks. A wire extends from the gadget as part of the spinal tree, and three e-wires go to three variable gadgets (one for each literal in the corresponding clause of the 3-SAT instance).

Each e-wire terminates with four branches near the clause gadget and a single terminal near the variable gadget which corresponds to the literal. As mentioned in the variable gadgets section, the terminal point p_t of the e-wire near the variable gadget lies at distance 5.5 above a disk D in the gadget. The other terminals of the e-wire are placed so that a pair of points is near the middle disk on two of the rays, as shown in Figure 9.8. To be specific, suppose that the construction is positioned so that the centre point of the middle disk of the horizontal chain is placed at the origin of the plane. Call this the *canonical position* of the gadget. Terminal points from one e-wire are placed at the coordinates $(-2, -6)$ and $(2, -6)$, while terminal points from another e-wire for the clause are placed at positions reflected through the x-axis, i.e., $(-2, 6)$ and $(2, 6)$. This way, the terminal points of each pair of e-wires for a clause are positioned symmetrically about a disk in the clause gadget.

Supposing points were selected in these three disks in a zigzag Z_D configuration (shown to be an optimal configuration in Theorem 9.5), then the edges between points in adjacent disks have weight $2\sqrt{17} \approx 8.25$, and those from the first and last disk to the nearest points along the ray have weight $\sqrt{17} \approx 4.12$. The distance from the point in the middle disk to the nearest point on one e-wire is $\sqrt{29} \approx 5.39$, and to the nearest point on the opposite e-wire is $\sqrt{53} \approx 7.28$.

Lemma 9.6. *Suppose that a clause gadget is placed in the canonical position. Then the weight of the MAX-MSTN solution is optimized when the point for the middle disk is placed at $(0, 1)$ or $(0, -1)$.*

Proof. Observe that the minimum distance from a point on an e-wire to the first or the third disk of the chain is $\sqrt{(8-2)^2 + (0-6)^2} - 1 \approx 7.49$, while the maximum distance from such a point to a point in the middle disk is $\sqrt{53} \approx 7.28$. Therefore, if edges of the MST exist between an e-wire and points in the disks of the clause gadget, such edges join to the point in the middle disk of the chain.

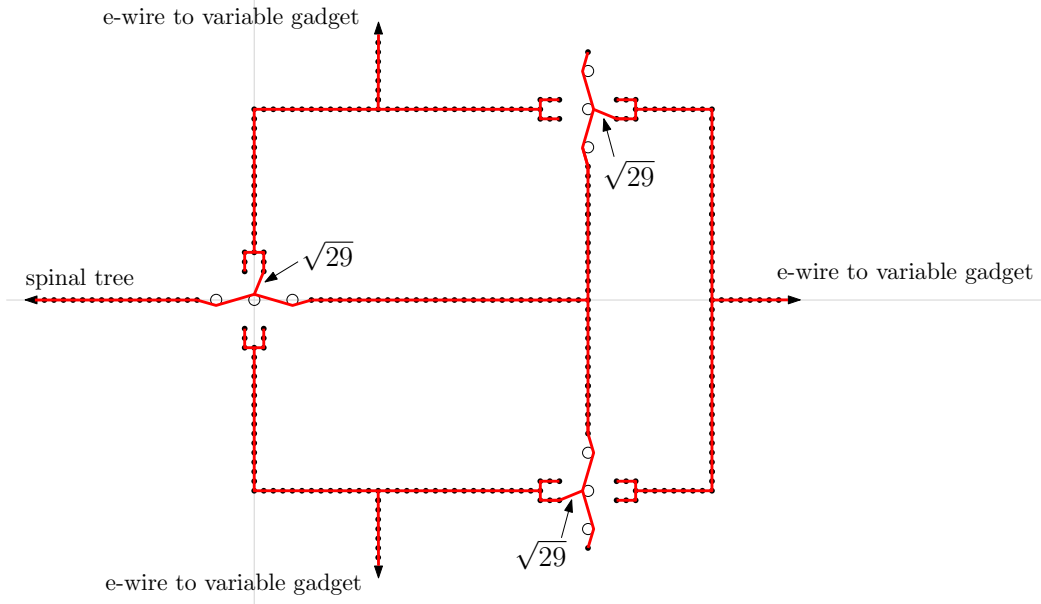
Suppose that no edges exist between the e-wires and the point in the disk; the optimal path through the disks is the Z_D configuration, and the lemma holds. Next, suppose only one of the e-wires is joined with an edge; the weight of this edge is maximized at the farther of the two candidate positions for the point in the middle disk, and so the lemma holds again. Finally suppose that both e-wires have edges joining to the point in the middle disk. We know that the point that maximizes the sum of the weights w_s of all edges incident upon the point is found on the edge of the disk by Lemma 9.1. Suppose without loss of generality that the point p is chosen in the middle disk so that the x-coordinate is ≥ 0 , and so we may assume that the edges join to the right terminals of each e-wire (positioned at $(2, 6)$ and $(2, -6)$). The distance from these two points to a point on the right half of a unit disk centred at the origin is maximized at either $(0, 1)$ or $(0, -1)$, and so the lemma follows. □

Lemma 9.7. *If all three e-wires associated with a clause gadget are joined to the clause gadgets with edges, then the weight of these edges is maximized when two e-wires are joined with edges of weight $\sqrt{29}$, and the other is joined with an edge of weight $\sqrt{53}$.*

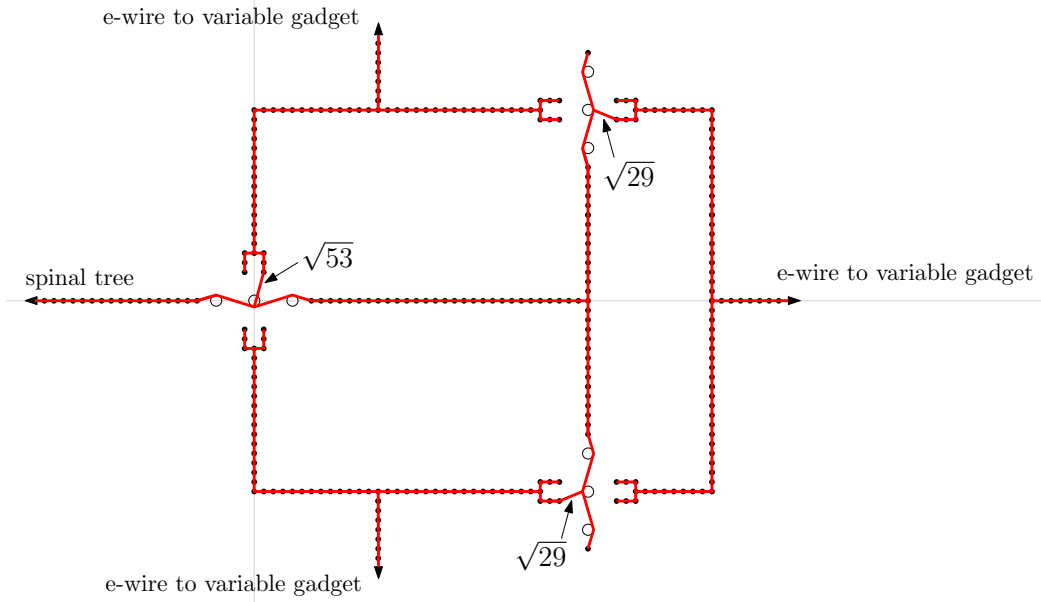
Proof. By Lemma 9.6, we know that all clause gadgets will use the Z_D configuration through their disks. Therefore, the weight of an edge between the clause gadget and an e-wire is either $\sqrt{29}$ or $\sqrt{53}$. Further, because two e-wires approach each set of disks, if one e-wire is distance $\sqrt{29}$ from a point in the clause, then another e-wire is $\sqrt{53}$ from the same point. The e-wires are arranged so that each pair of e-wires associated with a clause gadget has this relationship. Therefore, there are two possible settings:

1. Each e-wire is distance $\sqrt{29}$ from the nearest point in the clause gadget (Figure 9.9a).
2. One e-wire is distance $\sqrt{53}$ from the nearest point in the clause gadget, while the other two e-wires are distance $\sqrt{29}$ from the nearest point in the clause gadget. The configuration of the path through the triple of disks between the latter two e-wires is inconsequential (Figure 9.9b).

Since we are computing a Minimum Spanning Tree with maximum weight, the second configuration is preferable. □



(a) A suboptimal configuration.



(b) An optimal configuration. Note that the top left portion of the gadget requires an edge of weight $\sqrt{53}$ in order to be connected to the remainder of the gadget (one of the four possible such edges is shown).

Figure 9.9: Paths through the MAX-MSTN clause gadget.

Reduction

The key to the reduction is that an optimal MAX-MSTN solution to the construction outlined above will join all of the e-wires to the clause gadgets, leaving the variable gadgets unaffected, if and only if there exists a satisfying assignment to the given planar 3-SAT instance. Therefore, we begin the reduction by determining the weight of an optimal solution. If there is no satisfying assignment, the optimal MAX-MSTN algorithm selects the points in a way that the Z_D path through at least one variable gadget is affected, and the total weight of the optimal MAX-MSTN solution is reduced. We determine a lower bound on this effect, and in so doing, establish the hardness of the problem.

Let us consider the structure of an optimal solution to MAX-MSTN if there exists a satisfying assignment, and in particular we examine the weights of the edges required to join each of the three e-wires associated with a clause gadget (the same reasoning applies to all clause gadgets). Recall that an e-wire may be distance $\sqrt{29} \approx 5.39$ or $\sqrt{53} \approx 7.28$ from the nearest point in the clause gadget, by Lemma 9.6. Assuming that the points in the variable gadget are in the Z_D configuration, the nearest point to an e-wire in the variable gadget may be 5.5 or 7.5, by Lemma 9.5. Since at least one of the literals may be satisfied in the clause (by our assumptions), the corresponding e-wire, call it e_{sat} , is distance 7.39 from the nearest point in the variable gadget when the variable gadget has the Z_D configuration associated with the satisfying truth assignment. To maximize the weight of the MAX-MSTN solution, the paths through the disks in the clause gadget should be set so that e_{sat} is $\sqrt{53}$ from the nearest point in each of the two disks that it approaches. Thus, the weight of the edge to connect to e_{sat} is $\sqrt{53}$. The other two e-wires, whether they correspond to literals that may be satisfied or not, are each connected with an edge of weight $\sqrt{29}$, by Lemma 9.7. Since no point in any variable gadget is closer to an e-wire than the points in the clause gadget, we have shown that the e-wires are all joined to the clause gadget in an optimal MAX-MSTN solution. Therefore, all variable gadgets are connected to the rest of the MST only at their endpoints and, by Theorem 9.5, selecting a setting other than the Z_D configuration in any variable gadget is sub-optimal.

We now compute the optimal weight of a MAX-MSTN solution under the assumption that there exists a satisfying assignment to the planar 3-SAT instance. Let w_{st} be the total weight of the MST over the wires in the spinal tree segments of the construction, and let w_{ew} be that for the e-wires, which are both fixed for any MAX-MSTN solution. The MAX-MSTN solution over the clause gadgets has a fixed weight, consisting of the weight of all wires, plus $2\sqrt{29} + \sqrt{53} \approx 240.05$ for each gadget to join the e-wires to the disks. Finally, assume there is a total of h disks in all of the variable gadgets of the construction. The total weight of the optimal Z_D configuration in these disks is $w_{\text{vg}} = h\sqrt{5.25^2 + 2^2} \approx h \cdot 5.62$. Therefore, the total weight of the optimal MAX-MSTN solution is $w_{\text{tot}} = w_{\text{st}} + w_{\text{ew}} + w_{\text{cg}} + w_{\text{vg}}$, all of which we can compute *a priori* once the MAX-MSTN instance is constructed⁵.

Now consider the case that there is no satisfying assignment for the 3-SAT instance. In particular, consider a truth assignment (defined by the alignment of zigzag paths in variable gadgets) and a clause that is not satisfied by that assignment (such a clause exists since the 3-SAT instance is not satisfiable). Note that a MAX-MSTN algorithm might deviate from selecting zigzag paths; this will be addressed shortly. For the clause that is not satisfied, we may dismiss the setting where each e-wire is $\sqrt{29}$ from a point in the clause gadget as sub-optimal, by Lemma

⁵Note that the number of edges in the construction is a polynomial in the size of the input. Since all we need to determine is whether the weight of the solution is at least 0.34 units less than w_{tot} , we can round the measure of the total weight to a number of bits that is logarithmic in the size of the input.

9.7. Therefore, one of the e-wires, call it e_{nsat} , is $\sqrt{53}$ from the nearest point in the clause gadget, and this affects the positions of the points in the corresponding variable gadget at the other end of the e-wire⁶. Let p_{vg} be the nearest point in the variable gadget to e_{nsat} . Notice that the Fermat point of the two points neighboring p_{vg} in the variable gadget and the nearest point in e_{nsat} lies above the disk containing p_{vg} , so moving p_{vg} down within the disk increases the weight of the total MST if there is an edge between p_{vg} and the nearest point in e_{nsat} . Moving the point at least $\sqrt{53}$ away from e_{nsat} increases the weight of the MAX-MSTN solution as much as possible. Now the configuration of the clause gadget is the same as in the satisfying assignment above, so we need only measure the reduction in weight resulting from the changes to the points in the variable gadget to see the total reduction in weight relative to an optimal MAX-MSTN solution for a satisfying assignment.

To determine a lower bound on this effect, assume that there is only one transition in the zigzag pattern. That is, assume without loss of generality that at some point the Z_D configuration is in the **true** setting, and then the pattern switches to the **false** setting for the remainder of the gadget. All the wires in the construction must maintain a minimum separation of at least $\sqrt{53}$ to preserve the desired structure of the MST, and so e-wires that approach the variable gadget from the same side must have at least two disks between them (since they correspond to mismatched truth values), and those approaching from opposite sides may have one disk between them at minimum. To find an appropriate bound, first we assume that the closest points to each e-wire, call these p_ℓ and p_r , remain in the positions that would be optimal for a Z_D configuration, i.e., at the farthest point possible from the axis of the variable gadget, and then we compute the maximum weight path between p_ℓ and p_r in the gadget. Next, we bound the possible error introduced by placing p_ℓ and p_r in the extreme positions.

Case 1: Two separating disks (Figure 9.10). Without loss of generality, we position points at coordinates $p_\ell = (0, -1)$ and $p_r = (15.75, -1)$, and seek the maximum weight path using points p_1 and p_2 , where p_1 is in a unit disk centred at $(5.25, 0)$, and p_2 is in one centred at $(10.5, 0)$. If we define the points as $p_1 = (5.25 + \sin(\theta_1), \cos(\theta_1))$ and $p_2 = (10.5 + \sin(\theta_2), \cos(\theta_2))$, then the total weight w' is defined by:

$$\begin{aligned} f(\theta_1, \theta_2) = & \sqrt{(-5.25 - \sin(\theta_1))^2 + (-1 - \cos(\theta_1))^2} \\ & + \sqrt{(5.25 + \sin(\theta_2) - \sin(\theta_1))^2 + (\cos(\theta_2) - \cos(\theta_1))^2} \\ & + \sqrt{(5.25 - \sin(\theta_2))^2 + (-1 - \cos(\theta_2))^2}. \end{aligned}$$

Using Maple, we observe that the optimal path uses the values $\theta_1 \approx 0.005$ and $\theta_2 \approx 3.33$ radians, for a total weight of more than 16.49 over the three edges. By moving p_1 to $(5.25, 1)$ and p_2 to $(10.5, -1)$, we reduce the total weight of the edges by less than 0.01 (the weight remains greater than 16.48).

Case 2: One separating disk (Figure 9.11). Now points are placed at $p'_\ell = (0, -1)$ and $p'_r = (10.5, 1)$, and we seek the path using a point p'_1 again found in a unit disk centred at

⁶If the Z_D configuration were maintained, then the e-wire could be joined sub-optimally to the variable gadget with weight 5.5, since the assignment is not satisfying. However, a larger MST can be achieved by deviating from the Z_D configuration in the variable gadget.

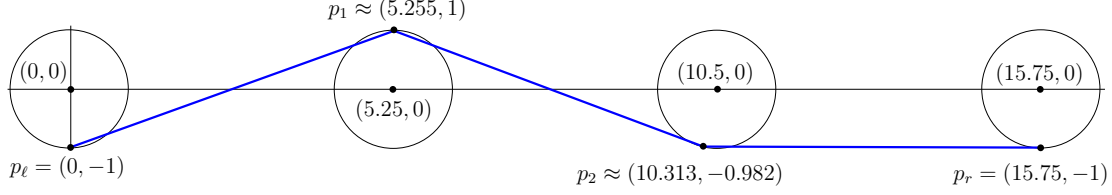


Figure 9.10: The optimal setting for Case 1. The optimal point in the first separating disk is slightly to the right of the extreme upper point in the disk, while that of the second disk is nearly 11° clockwise of the extreme lower point. Despite these differences, the change in path weight between this optimal path and a path using the extreme points is less than 0.01 units.

$(5.25, 0)$. Define $p'_1 = (5.25 + \sin(\theta), \cos(\theta))$, and the total weight w' is now defined by:

$$f(\theta) = \sqrt{(-5.25 - \sin(\theta))^2 + (-1 - \cos(\theta))^2} + \sqrt{(5.25 - \sin(\theta))^2 + (1 - \cos(\theta))^2}.$$

Using Maple, we observe that such a path has maximum weight greater than 10.87, realized when $\theta \approx 2.95$ radians. To create a simplified configuration, we shift this point so that $p'_1 = (5.25, -1)$, and we observe that this change reduces the total weight of the edges by less than 0.01 (so that the weight is greater than 10.86).

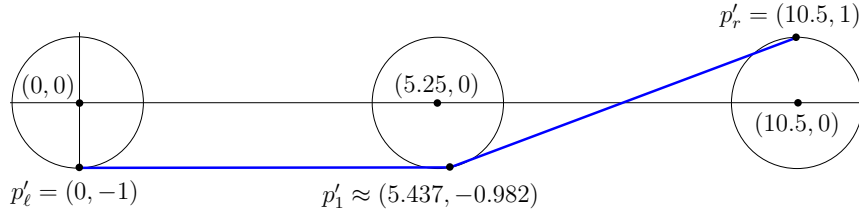


Figure 9.11: The optimal setting for Case 2. The aberration of the point in the separating disk from the extreme lower point is similar to that of the second separating disk in Case 1, and the difference in path weight between the optimal path and that using only the extreme points is less than 0.01 units.

The final remaining possible suboptimality is the assumed positions for p_r in Case 1 and p'_l in Case 2 (p_ℓ and p'_r are in the midst of points in the Z_D configuration). Consider the adjusted path from $p'_1 = (5.25, -1)$ to p'_ℓ to a point in the disk left of D'_ℓ , i.e., the disk centred at $(-5.25, 0)$. This setting is again analogous to Case 2, and so we can conclude that this assumed position reduces the total weight by less than another 0.01 units. We know by Theorem 9.5 that any changes in position from the extreme points in the remainder of the gadget only decrease the total weight of the MST.

Therefore, the overall introduced errors of our assumptions are less than 0.02 for each case. For Case 1, we conclude that the total weight lost as a result of an unsatisfiable clause is greater than $3\sqrt{31.5625} - 16.49 - 0.02 \approx 0.34$. In Case 2, the effect is greater than $2\sqrt{31.5625} - 10.87 - 0.02 \approx 0.34$. Hence, given a planar 3-SAT instance, we know that the total weight of the optimal solution to the MAX-MSTN construction is less than $w_{\text{tot}} - 0.34$ if and only if there is no satisfying assignment for the instance. By Theorem 9.4, we have that the number of disks (including

points) required for the construction is polynomial in the size of the input, and all points are either on the integer grid or else have rational coordinates where the denominator is a constant. This establishes the NP-hardness of the MAX-MSTN problem.

If we choose a value of ε so that $\varepsilon < 0.34/w_{\text{tot}}$, then a $(1 - \varepsilon)$ -approximate solution to the MAX-MSTN problem may be used to determine whether there is a satisfying assignment for the planar 3-SAT instance. Since the latter problem is NP-hard, we conclude that MAX-MSTN does not admit an FPTAS unless $P=NP$. We cannot rule out the possibility of a PTAS, because the gap between solutions on satisfiable and non-satisfiable instances that we have demonstrated is not a factor that is proportional to the size of the optimal solution on the MAX-MSTN construction.

9.4 MSTN

In this section we present an asymptotic 3-approximation algorithm and a parameterized approximation algorithm for the MSTN problem, followed by the proof of NP-hardness. Recall that for MSTN, we seek a minimum weight spanning tree on imprecise input represented as disks.

9.4.1 3-Approximation Algorithm on Disjoint Unit Disks

In this section, we define *disjoint disks* to be disks that are disjoint with respect to their interiors; a pair of disks are allowed intersect at a single point on their boundaries so that the pairwise distance between the disks is zero. This section consists of some refinements of the work of Yang et al. [166]. In that work, an asymptotic 3-approximation was presented for this problem, but the algorithm contained a bug and the lower bound used was false (see Figure 9.12). We conjecture a new lower bound, and we describe an improved approximation algorithm for which the asymptotic 3-approximation holds, based on the new conjecture.

Consider a pair of disjoint unit disks in the plane. The optimal MSTN solution for this problem is trivial: it is simply the minimum length segment between the two disks. In Yang et al. [166], they introduce the *minimum connecting tree* T_c on a set of disks \mathcal{D} in order to generalize this idea and obtain a lower bound on the weight of the optimal MSTN solution.

Definition 9.3. The Minimum Connecting Tree [166]: *Given a set of disks \mathcal{D} , define the graph $G_c = (V, E)$, where $v_i \in V$ corresponds to $D_i \in \mathcal{D}$, and the weight of the edge $e_{ij} = (v_i, v_j)$ is defined by the minimum distance between the disks D_i and D_j . The minimum connecting tree T_c is the Minimum Spanning Tree on G_c , which has weight $L_c \geq 0$.*

Since T_c is the minimum weight tree to connect the disks of the input, any MSTN solution necessarily has greater weight [166, Lemma 4]. Next we examine the general lower bounds on the weight of an MSTN solution on disjoint unit disks, and tighten the conjectured bounds slightly.

Conjecture 9.1. *Given $n \geq 5$ disjoint unit disks, the weight L of an optimal MSTN solution may be bounded as follows:*

- $L \geq n - 6 + 2\sqrt{3} \approx n - 2.54$ for even values of $n \geq 6$;
- $L \geq n - 4 + \sqrt{3} \approx n - 2.27$ for odd values of $n \geq 5$.

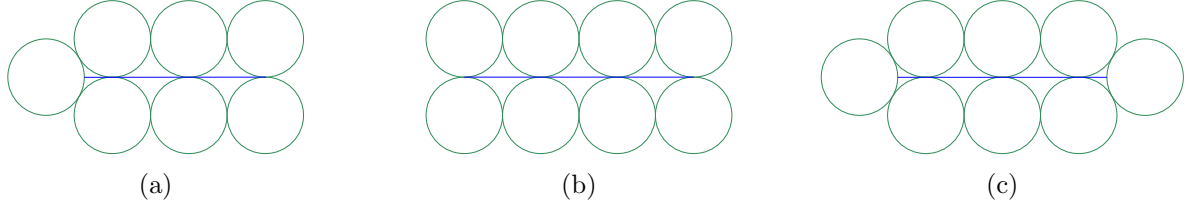


Figure 9.12: The constructions conjectured to have the minimum weight MSTN solutions on disjoint unit disks where the number of disks in the input is at least five. (a) The lower bound for odd numbers of disks from [166, Figure 3(a)]. Each adjacent vertical stack of disks is joined by an edge weight 2, and the odd disk out is connected by an edge of weight $\sqrt{3} - 1$. There are $(n - 1) / 2$ such stacks, and all stacks but one require an edge to an adjacent stack, so the total weight of the construction is $2((n - 1) / 2 - 1) + \sqrt{3} - 1 = n - 4 + \sqrt{3}$. (b) The conjectured lower bound for even numbers of disks from [166, Figure 3(b)]. This construction has weight $n - 2$. (c) A new conjectured lower bound for even numbers of disks. The analysis is similar to Figure 9.12a, except that there are $(n - 2) / 2 - 1$ edges of weight 2, and a pair of edges of weight $\sqrt{3} - 1$. Thus, the total weight of the construction is $2((n - 2) / 2 - 1) + 2(\sqrt{3} - 1) = n - 6 + 2\sqrt{3}$.

Therefore, we conjecture that in general, $L \geq n - 6 + 2\sqrt{3}$ for $n \geq 5$. The constructions supporting the conjecture are shown in Figure 9.12.

We describe the approximation algorithm in Algorithm 9.1, which builds rather simply on the minimum connecting tree T_c . First, the MST of T_c is computed. Then a set of points \mathcal{P} is built using one point from each disk. If a vertex of the MST on T_c has degree one, then we place the point in the disk at the end point of the edge incident upon the disk in T_c ⁷. Otherwise, the centre point of the disk is used. Finally, the MST is computed on the point set \mathcal{P} .

Algorithm 9.1 MSTN-3(\mathcal{D})

- 1: **Input:** A set of disjoint unit disks \mathcal{D} .
 - 2: **Output:** A MSTN solution with an approximation factor of 2 (see Definition 9.1).
 - 3: $\mathcal{P} \leftarrow \emptyset$
 - 4: Compute the minimum connecting tree T_c
 - 5: **for** $i := 1$ **to** n **do**
 - 6: Let \mathcal{P}_i be the set of end points of edges incident upon disk D_i
 - 7: **if** $(|\mathcal{P}_i| > 1)$ **then**
 - 8: Set p_i to the centre point of D_i
 - 9: **else**
 - 10: Set p_i to the single point in \mathcal{P}_i
 - 11: **end if**
 - 12: $\mathcal{P} \leftarrow \mathcal{P} \cup \{p_i\}$
 - 13: **end for**
 - 14: Compute T_a , the Minimum Spanning Tree of \mathcal{P}
 - 15: **return** T_a
-

To analyze the algorithm, we place an upper bound on the increase in weight of T relative to

⁷A better result may be obtained by using the Fermat-Weber point of \mathcal{P}_i (see Section 9.2), although it does not improve the approximation factor, and it is algebraically difficult to determine for more than four points.

T_c . For any disk that is a leaf of T_c , the chosen point will be the endpoint of the edge. For all other disks, there is unit distance from the endpoints of the edges to the approximating point. In any tree, there are $n - 1$ edges, and at least two vertices of any tree must be leaves. We may construct an intermediate tree T_i consisting of the edges T_c , plus edges of at most unit weight to join the endpoints of every edge of T_c to the approximating point in each corresponding disk, and so the weight of T_i is

$$L_i \leq L_c + 2(n - 3) + 2 \leq L_c + 2(n - 2).$$

Since $L_c \leq L$ and $n - 6 + 2\sqrt{3} \approx n - 2.54 \leq L$, we have that

$$L_i \leq L + 2L + 2(6 - 2\sqrt{3} - 2) < 3L + 1.08.$$

If there is an edge between D_i and D_j in T_c , then the edges between vertices p_i and p_j may be replaced by a single edge (p_i, p_j) to create a spanning tree on the points of \mathcal{P} with weight less than L_i (note that L_i is not necessarily an MST). The Minimum Spanning Tree T_a over this same set of points has weight L_a , and $L_a \leq L_i < 3L + 1.08$, since T_a is an MST. Therefore, T_a is solution for the MSTN problem with an asymptotic approximation factor of 3.

Theorem 9.6. *There exists an asymptotic 3-approximate algorithm for MSTN on a set of disjoint unit disks \mathcal{D} for $|\mathcal{D}| \geq 5$, assuming the lower bounds of Conjecture 9.1 hold.*

9.4.2 Parameterized $(1 + 2/k)$ -Approximation Algorithm on Disjoint Disks

Recall that to have k -separability means that the minimum distance between any two disks is at least kr_{\max} , where r_{\max} is the maximum radius of any disk in the instance. The *separability* of an input instance I is defined as the maximum k such that I satisfies k -separability.

Theorem 9.7. *For MSTN when the regions of uncertainty are disjoint disks with separability parameter $k > 0$, the algorithm that builds an MST on the centres of the disks achieves an approximation factor of $\frac{k+2}{k} = 1 + 2/k$.*

Proof. Assume that we have a set D of n disks that satisfies k -separability. Let T_c be the MST on the centres and T_{OPT} be an optimal MST, i.e., an MST that contains one point from each disk and its weight is the minimum possible. Define T_m as the spanning tree (not necessarily an MST) with the same topology as T_{OPT} but on the points of T_c , i.e., on the centres. Since T_c is an MST on centres, we have $w(T_c) \leq w(T_m)$. Consider an arbitrary edge e in T_m and let D_i and D_j be the two disks that are connected by e . Let r_i and r_j be the radii of D_i and D_j , respectively, and let d be the distance between D_i and D_j . In T_{OPT} the disks D_i and D_j are connected by an edge e' whose weight is at least d . The weight of e , on the other hand, is $d + r_i + r_j$. Therefore the ratio between the weight of an edge in T_{OPT} and its corresponding edge in T_m is at least

$$\frac{d}{d + r_i + r_j} \geq \frac{kr_{\max}}{kr_{\max} + r_i + r_j} \geq \frac{kr_{\max}}{kr_{\max} + r_{\max} + r_{\max}} = \frac{k}{k + 2}.$$

Since this holds for any edge of T_m , we get $w(T_c) \leq w(T_m) \leq \frac{k+2}{k}w(T_{\text{OPT}})$. Therefore we get an approximation factor of $\frac{k+2}{k} = 1 + 2/k$ for the algorithm. □

As with the parameterized algorithm for MAX-MSTN, as the disks become further apart (as k grows), the approximation factor approaches 1. The principal difference between this analysis and that of the MAX-MSTN setting (Section 9.3.2) is that the intermediate spanning tree T_m is defined as the tree whose vertices are disk centres and whose topology is the same as T_{OPT} , whereas T'_c is the spanning tree with the same topology as T_c but on the vertices of T_{OPT} .

9.4.3 NP-Hardness of MSTN

To prove the hardness of the MSTN problem, we present a reduction from the planar 3-SAT problem. Planar 3-SAT is a variant of 3-SAT in which the graph $G = (V, E)$ associated with the formula is planar (see Figure 9.2 and the associated text for further details).

Theorem 9.8. *MSTN is NP-hard. Furthermore, MSTN does not admit an FPTAS, unless $P=NP$.*

In the hardness proof of MAX-MSTN (Section 9.3.3), we used a spinal tree in the reduction. In this section, we use the *spinal path*, defined as a path $P = (V_p, E_p)$ with a set of edges E_p such that $E \cap E_p = \emptyset$, where P passes through all variable vertices in G without crossing any edge in E . As mentioned earlier, this restricted version of planar 3-SAT remains NP-hard [111]. To reduce planar 3-SAT to MSTN, we begin by finding a planar embedding of the graph associated with the SAT formula. We force the inclusion of the spinal path as a part of the MST using wires. A wire is a set of disks of radius 0 placed in close succession (usually 2 units apart) so that we may interpret them as a fixed line in the MSTN solution. We replace each variable vertex of V by a *variable gadget* in our construction. These gadgets are composed of a set of disks and some wires, and are defined in such a way that the optimal MSTN solution has a weight less than k (for some value k dependent on the input) if and only if the SAT formula is satisfiable.

As in Section 9.3.3, we use an orthogonal drawing of the planar 3-SAT instance in our reduction, expanded by a factor of 2. All edges of the drawing are replaced by *wires*, where a wire consists of a set of points (disks of radius 0) placed along an edge of the drawing every unit distance so that each wire has a unique MST over the points and the number of points required for all wires is polynomial in the size of the input (see Theorem 9.4). Again, the gadgets used in the reduction may not fit within boxes of size $\frac{\deg(v)}{2} \times \frac{\deg(v)}{2}$, but they are within a constant factor of this bound. The variable gadgets require boxes of size at most $(8 \deg(v) + 5) \times 15$ units (we discuss these gadgets shortly), while clause gadgets consist of a single point (note that these vertices have degree 3, so this does not cause a problem). Therefore, the size of the drawing remains polynomial in the size of the input.

Variable Gadgets

A variable gadget is formed by k disks, where $k \leq 8m_i + 6$ and m_i is the number of clauses in the planar 3-SAT instance that include the variable. Each clause requires 4 disks, each edge of the spinal path requires 3 disks, and we may need to add extra disks as discussed in Figure 9.13. Disks are placed 2 units apart so that each disk is tangent to its two neighboring disks, and each pair of consecutive disks D_i, D_{i+1} around the gadget intersects at a single point $q_{i,i+1} = D_i \cap D_{i+1}$, which we call a *tangent point*⁸. Moreover, there is a set of points in the middle of the gadget,

⁸Using this construction, pairs of disks of the gadget trivially intersect at a single point, which simplifies our analysis. To achieve strict disjointedness, the disks of the gadget may be contracted to have radius $1 - \gamma$ so that

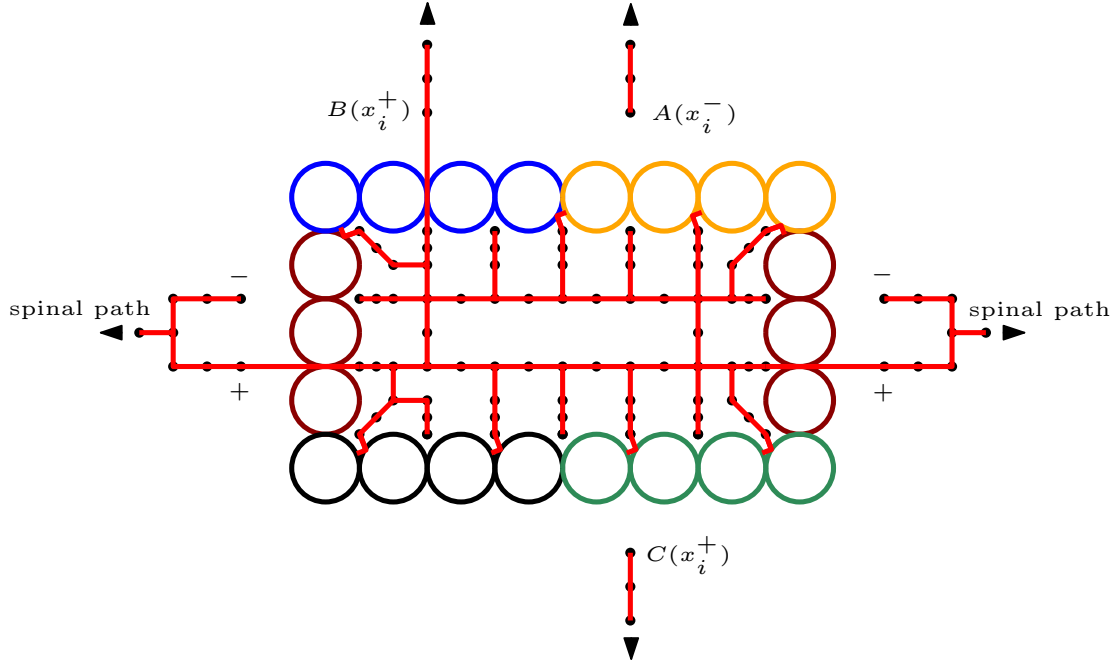


Figure 9.13: A variable gadget contains up to $6 + 8m_i$ disks for a variable x_i , where m_i is the number of clauses containing literals of the variable. In this case, the gadget contains 22 disks. $B(x_i^+)$ and $C(x_i^+)$ are the endpoints of the wires that connect to clauses that include x_i in the positive form, while $A(x_i^-)$ terminates a wire connecting to a clause that includes x_i in negative form. The figure illustrates the case in which the algorithm has used the positive configuration for x_i , and clause B is satisfied by x . Clause C is satisfied via some other variable, as is clause A (assuming that it is satisfiable). Note that disks are connected to the wires inside the gadget in pairs. The gadget contains 4 disks for each wire from a clause gadget, and extra disks are used if there are differing numbers of wires approaching the top and bottom (this is why the upper bound has the factor $8m_i$, since it may happen that all m_i wires approach the top or bottom only).

whose MST has fixed weight (although the topology is not unique). There are k terminal points to this MST, each unit distance from the nearest tangent point, i.e. there exists a point t_i for each $q_{i,i+1}$ so that $\text{dist}(t_i, q_{i,i+1}) = 1$. The spinal path is placed so that it approaches the variable gadget twice, and each of these approaches requires three disks. We split the wires of the spinal path once near the variable gadget as shown in Figure 9.13, and wires terminate at a distance 2.5 units from the nearest tangent point, for reasons discussed in the Clause Gadgets section.

Lemma 9.8. *Suppose we are given two unit disks D_1 and D_2 that intersect exclusively at a single point $q = D_1 \cap D_2$, and a line ℓ such that $q \in \ell$ and ℓ is tangent to both D_1 and D_2 (i.e., ℓ is the perpendicular bisector of the centre points of D_1 and D_2). Now, given a point $p \in \ell$ where p is unit distance from q , the shortest path consisting of points $p, q_1 \in D_1$, and $q_2 \in D_2$ has weight $d \approx 0.755$.*

the tangent point is now distance γ from the nearest point in the adjacent disks. Any path which uses the tangent point in our analysis will have less than 2γ units of additional weight on these shrunk disks, and there are fewer than $n(8m+6)$ disks, where n and m are the number of variables and clauses respectively. Choosing an appropriate value of γ so that $2\gamma n(8m+6) \ll 0.66$ achieves the same result as our simplified analysis.

Proof. If $q_1 = q_2 = q$, then the path has unit length, so a path of length d is shorter. A path with edges $e_1 = (q_1, p)$ and $e_2 = (p, q_2)$ has length at least 0.828, since the nearest point on D_1 or D_2 to p is $\sqrt{2} - 1 > 0.414$ units distant.

Therefore, we may assume without loss of generality that the path consists of the edges $e_1 = (p, q_1)$ and $e_2 = (q_1, q_2)$ and the path has length $d = w(e_1) + w(e_2)$, where $w(e)$ is the length of the edge e . We must choose q_1 and q_2 so that d is minimized. Note that candidate positions for each of q_1 and q_2 may be restricted to the boundaries of their respective disks.

For the purposes of simplifying the proof, assume that p is at the origin of the Cartesian plane, and D_1 and D_2 are centred at $(1, 1)$ and $(1, -1)$, respectively. Then a point q_1 on the boundary of D_1 may be expressed as $(\sin(\alpha) + 1, \cos(\alpha) + 1)$, for some $\alpha \in [0 \dots 2\pi]$, and analogously $q_2 = (\sin(\beta) + 1, \cos(\beta) - 1)$, for some $\beta \in [0 \dots 2\pi]$. Therefore, we simply have to find the minimum of the function

$$f(\alpha, \beta) = \sqrt{(\sin \alpha + 1)^2 + (\cos \alpha + 1)^2} + \sqrt{(\sin \beta - \sin \alpha)^2 + (\cos \beta - \cos \alpha - 2)^2},$$

over the variables $\alpha \in [0 \dots 2\pi], \beta \in [0 \dots 2\pi]$. Using Maple, we see that this minimum has value $d \approx 0.755$, at $\alpha \approx 3.62, \beta \approx 5.89$. The optimal path in this setting is shown in Figure 9.14. Since this path is shorter than all other possible path configurations, we conclude that this is the shortest possible path including p and points $q_1 \in D_1$ and $q_2 \in D_2$.

□

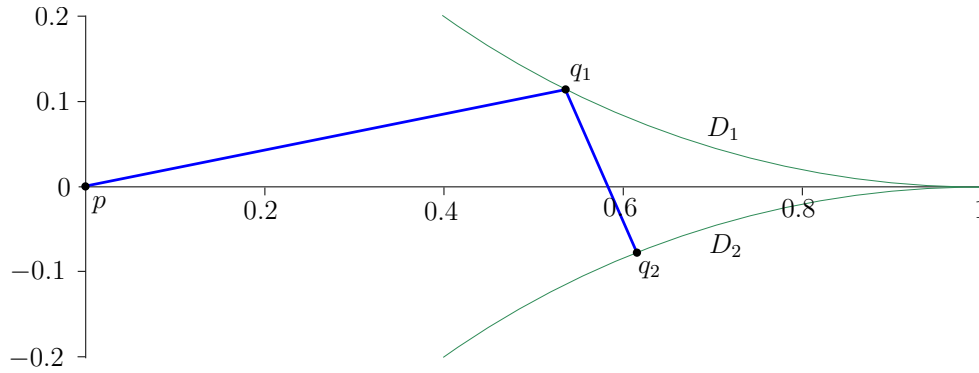


Figure 9.14: The shortest possible path is shown from a point at the origin to some point in each of two unit disks; one of the disks is centred at $(1, 1)$, the other is at $(1, -1)$.

For the remainder of the discussion, we refer to the weight of this shortest path as the constant $d \approx 0.755$. Before going to the details of the reduction, we consider optimal MSTN solutions when the problem instance is a variable gadget as described above (without the wires approaching from clauses). We claim that such an instance has two possible MSTN solutions, and in each of these solutions consecutive pairs of disks are connected to a single wire of the interior of the gadget with a path of length d described in Lemma 9.8. We associate these two possible MSTN solutions with the two assignments for the variable. To prove the claim, we show that in an optimal MSTN solution for the interior of the gadget, there is no path containing points from more than two disks.

Lemma 9.9. *In the optimal MSTN solution for a variable gadget, each consecutive pair of disks is connected to a distinct point on the interior of the gadget via a path of length d .*

Proof. Recall that by Lemma 9.8, connecting a pair of disks in the configuration shown in Figure 9.14 may be done with a path of weight d , while the same point may be connected to a single disk with weight $\sqrt{2} - 1$. Therefore, three consecutive disks in a variable gadget may be connected to two wires of the interior of the gadget using edges with weight $d + \sqrt{2} - 1 \approx 1.169$, while four such disks may be connected with weight $2d \approx 1.51$.

Now consider three consecutive disks that we wish to connect to a single wire of the interior of the gadget. The minimum distance between any two non-adjacent disks is $d_{\min} \geq 2(\sqrt{2} - 1)$, which occurs in the corners. Note that a path spanning this nearest distance passes through the point of the interior of the gadget nearest the three disks in the corner, and so the shortest path to connect to the third disk in the corner requires another edge of $\sqrt{2} - 1$. Two of these adjacent paths may now be replaced by a path of weight d , as shown in Lemma 9.8, and so the weight of the shortest path joining three consecutive disks to a point on the interior of the gadget is $d + \sqrt{2} - 1 \approx 1.169$. Since $d + \sqrt{2} - 1 > 3d/2$, and there are an even number of disks in the gadget, an optimal path containing one point of the interior of a variable gadget in the MST contains points from at most two disks of the variable gadget. \square

Hence, there are two possible solutions for MSTN on a variable gadget. We use this fact to assign a truth value for the variable gadget: one configuration is arbitrarily considered to be **true**, the other **false**. In Figure 9.13, we show an example where the **true** configuration is used, and every other wire of the interior of the gadget has an edge to some point in the disks. The **false** configuration would contain edges between the complementary set of wires of the interior of the gadget and the disks of the variable gadget.

Clause Gadgets

The clause gadgets are composed of three wires that meet at a single point. Each wire associated with the clause gadget is placed so that it terminates at a distance 2.5 from a tangent point of a variable gadget, as shown in Figure 9.13. As a result, a line segment of length 3.5 units can connect the clause gadget to the interior of a variable gadget, while also intersecting the shared point between two disks. If the truth value of the variable gadget matches that of the clause, this means that connecting the clause gadget to the variable gadget requires $3.5 - d$ units of extra weight, since otherwise the two disks are connected to the interior of the gadget with d weight, as outlined in Lemma 9.8. Therefore, given a clause gadget where at least one literal matches the truth value of the corresponding variable gadget, the clause gadget is connected to the MST with $3.5 - d$ units of additional weight.

The spinal path wires terminate in positions exactly analogous to those of the clause gadgets so that the analysis is the same. This raises the possibility that the wires of a clause gadget may be connected to two variable gadgets, leaving a gap in the spinal path, but note that such a configuration does not affect the weight of the optimal tree. The spinal path is necessary however, since some variables may not be used by any clauses in an optimal solution.

Lemma 9.10. *Joining a clause wire to a variable gadget that has a truth value differing from that of the clause requires at least $4.16 - (3.5) \approx 0.66$ units of additional edge weight relative to a configuration with matching truth values.*

Proof. In an optimal MSTN solution on a construction corresponding to a satisfiable 3SAT instance, a pair of disks and a clause wire may be joined to the interior of the gadget with weight 3.5 units, and an additional adjacent pair of disks may be joined to the interior of the gadget with a path of weight d . Therefore, the total weight of the edges incident upon points in four such disks is $3.5 + d$.

Now consider a configuration where the truth value of the literal for each variable in a clause does not match the truth value of the corresponding variable gadgets. Connecting one of the clause gadget wires to the interior of the gadget requires edges of weight of at least 3.5 units, as discussed previously, and the shortest path intersects points from two disks; call them D_i and D_{i+1} . The neighboring two disks in the variable gadget, D_{i-1} and D_{i+2} , are not attached to the interior of the gadget by paths like those found in Lemma 9.8. Rather, each of these adjacent paths may be shortened to $\sqrt{2}$ to cover the two singleton disks. Note that there may be a non-empty sequence of pairs of disks connected as in Lemma 9.8 before the singleton is reached, creating a section of the gadget with an inverted truth value for the variable⁹. Therefore, the net extra weight of such a transition is $3.5 + 2\sqrt{2} - (3.5 + d) = 2\sqrt{2} - d \approx 2.0735$.

A configuration that may require less additional weight is to connect the clause wire to the interior of the gadget using a path with points in disks D_{i-1} and D_i (it is a slightly modified configuration from that of Lemma 9.8). For our analysis, we can place the centre of D_{i-1} at $(1, 1)$, the end of the interior wire between D_{i-1} and D_i at $(0, 0)$, and the end of the clause wire at $(3.5, -2)$ (Figure 9.15). The weight of the path from the interior of the gadget to D_{i-1} to the clause wire may be expressed by the function

$$f(\theta) = \sqrt{(1 + \sin \theta)^2 + (1 + \cos \theta)^2} + \sqrt{(2.5 - \sin \theta)^2 + (-3 - \cos \theta)^2},$$

which has a minimum weight greater than 4.16 units at $\theta \approx 3.43$ radians. Since this path intersects D_i , it is also the shortest path that includes a point $p_i \in D_i$. Therefore, without loss of generality a path connecting a clause wire to a wire in a variable gadget with a mismatched truth value has weight greater than 4.16. Note that such a path does not affect the truth value of the variable gadget, and so D_{i+1} and D_{i+2} may be joined to the interior of the gadget with a path of weight d . Therefore, the extra weight incurred for such a configuration is greater than $4.16 + d - (3.5 + d) \approx 0.66$. \square

As described earlier, the terminal points of the clause wires (and the spinal path) are collinear with wires of the interior of the gadget. Note that the wires approaching the variable gadget need not lie within 4 units of one another, and so there will not be edges directly between different clause wires or between a clause wire and the spinal path.

Reduction

We would like to reduce a given instance of planar 3-SAT to the MSTN problem. Note that the given 3-SAT instance is assumed to be embedded in the plane, and there exists a spinal

⁹ D_{i+2} may be more generally indexed as D_{i+2+4c} , where there is a block of $4c$ disks in the variable gadget joined to the interior of the gadget in a truth configuration opposite of that of the neighboring disks in the variable gadget. This does not affect the analysis, it simply relocates the singleton disk. Recall that by Lemma 9.9, such singletons would exist rather than having three disks connected by a path to a single edge of the interior of the gadget.

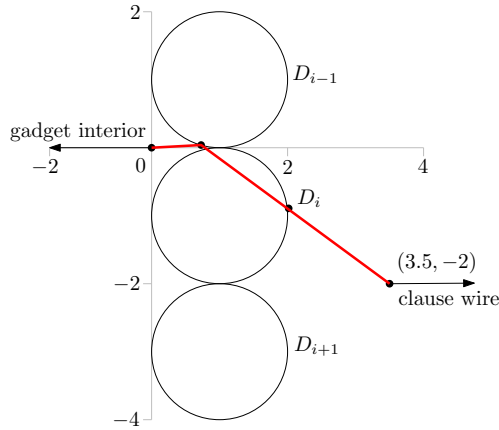


Figure 9.15: The shortest possible path is shown (the dashed line) from the end of the clause wire to points in D_i and D_{i-1} , and finally connecting to the interior of the gadget for D_i and D_{i-1} .

path $P = (V_v, E_P)$ that passes through all variable vertices without crossing any edge of G (as mentioned at the beginning of Section 9.4.3, this restricted version is also NP-hard).

To create the instance of the MSTN problem, we fix the spinal path as a part of the MST, using wires consisting of disks of radius 0. We replace each variable node with a variable gadget as explained. Each clause gadget includes three wires, which we place so that they approach the associated variable gadgets as described.

The wires forming the spinal path, the m clause gadgets, and each of the n variable gadget interiors have a fixed weight, call the total weight of all these wires w_{wires} . The remaining weight of the MST is that of connecting to a point from each disk in the variable gadgets, and that of connecting each clause gadget. Suppose there exists a satisfying assignment for the 3-SAT instance. Each pair of disks in the variable gadgets can be connected with weight d ; this will be the case for all but m pairs. The remaining m pairs will be connected with edges that also join to the clause gadgets with weight 3.5 in the manner described in the Variable Gadget section above. Therefore, assuming that there is a total of i pairs of disks in the variable gadgets of the construction, the remaining weight of the MST is $w_{\text{disks}} = id + (3.5 - d)m$. Thus, if there exists a satisfying assignment to the 3-SAT instance, the total optimal weight of the MST is $w_{\text{tot}} = w_{\text{wires}} + w_{\text{disks}}$ ¹⁰. If there is no satisfying assignment, at least one of the clause gadgets must be connected to the MST in the manner described in Lemma 9.10, which requires an additional weight of at least 0.66 units.

All points in the construction were placed on the integer grid or else have fractional coordinates where the denominator is 2, and the size of the grid required for the construction is polynomially bounded in the size of the input. This establishes the NP-hardness of MSTN.

Now suppose there exists an FPTAS for MSTN. Given an instance of planar 3-SAT, we build the MSTN construction and determine w_{tot} . We choose a value of ε so that $\varepsilon < 0.66/w_{\text{tot}}$, and so a $(1 + \varepsilon)$ -approximate solution to the MSTN problem may be used to determine whether there

¹⁰Note that the number of edges in the construction is a polynomial in the size of the input. Since all we need to determine is whether the weight of the solution is at least 0.66 units greater than w_{tot} , we can round the measure of the total weight to a number of bits logarithmic in the size of the input.

is a satisfying assignment for the planar 3-SAT instance. Since the latter problem is NP-hard, we conclude that MSTN does not admit an FPTAS, unless P=NP.

9.5 Conclusions and Future Work

We considered geometric MST with Neighborhoods problems, and established that computing the MST of minimum or maximum weight is NP-hard in both cases, even when disks are disjoint. We extended the hardness results to show that the problems are hard to approximate by proving that there is no FPTAS for either problem, assuming $P \neq NP$. We conjectured an asymptotic 3-approximation algorithm for the MSTN problem, as well as a parameterized algorithm based upon how well separated the disks are from one another. For MAX-MSTN, we showed that a deterministic algorithm which selects disk centres gives an approximation ratio of $1/2$. Furthermore, we showed that when the instance of the problem satisfies k -separability, the same approach achieves a constant approximation ratio of $1 - \frac{2}{k+4}$.

For further research, it will be interesting to study this problem under different models of imprecision. Depending on the application, the regions of uncertainty may consist of other shapes, e.g., line segments, rectangles, etc., or they may be composed of discrete sets of points, as discussed in Chapter 10.

2-Generalized Minimum Spanning Trees

The discretized version of the imprecise data problem is one where there is a set of points \mathcal{P} in the plane, and a range space \mathcal{S} where each element of \mathcal{S} is a collection of points from \mathcal{P} . The Minimum Spanning Tree problem in this model is known as the Generalized Minimum Spanning Tree (GMST) problem (or also the GMSTP, see below). In this chapter, we consider the setting where the input consists of sets of points, all sets have cardinality 2, and the two points in each set are incident upon a vertical line. We call each set of points in the range space an *imprecise vertex*.

Definition 10.1. The 2-GMST Problem: Assume we are given a set $P = \{p_0^\pm, \dots, p_{n-1}^\pm\}$ of imprecise vertices in which $p_i^\pm = \{p_i^+, p_i^-\}$ is a pair of points in the plane, and p_i^+ and p_i^- have the same x coordinate. The 2-GMST problem is to determine a set of choices $C = (c_0, \dots, c_{n-1}) \in \{+, -\}^n$, inducing the point set $V^c = \{p_j^{c_j} : j = 0, \dots, n-1\}$, which minimizes the weight of the Minimum Spanning Tree on (V^c) .

Definition 10.2. The MAX-2-GMST Problem: Assume we are given a set $P = \{p_0^\pm, \dots, p_{n-1}^\pm\}$ of imprecise vertices in which $p_i^\pm = \{p_i^+, p_i^-\}$ is a pair of points in the plane, and p_i^+ and p_i^- have the same x coordinate. The MAX-2-GMST problem is to determine a set of choices $C = (c_0, \dots, c_{n-1}) \in \{+, -\}^n$, inducing the point set $V^c = \{p_j^{c_j} : j = 0, \dots, n-1\}$, which maximizes the weight of the Minimum Spanning Tree on (V^c) .

We begin by discussing these problems in one dimension, and then we show that the 2-GMST problem is NP-hard in two dimensions. Finally, we show that a simple recursive algorithm solves the problem exactly if the topology of the solution is known.

10.1 Related Work

A generalized version of our problem was first defined as the Generalized Minimum Spanning Tree Problem (GMSTP) by Myung et al. [124]. They showed that the general version is NP-hard, and that there is no constant-factor approximation algorithm for the problem. This generalizes the 2-GMST problem in many senses: the cardinality of each set is unrestricted, the positions of the points in a set are unconstrained, and the problem considers general graphs, rather than being restricted to the Euclidean plane. Due to the hardness results, research on the problem

has primarily been directed toward using heuristics for finding good solutions for the problem or deriving exact solutions which run as quickly as possible, e.g. [82, 94, 124, 134]. A nice review of GMSTP and further generalizations is presented by Feremans et al. [68].

Pop [133] studied several problems related to the GMSTP. While studying the problem on sets of unrestricted size, a dynamic programming solution was described for the case that the number of sets of points is bounded by a constant. Furthermore, it was demonstrated that GMSTP is NP-hard even if the problem is restricted to trees [133, Theorem 2.3] (however, we provide an exact dynamic programming solution for the 2-GMST problem on trees in Section 10.6). Finally, Pop [133, Theorem 4.3] described a 2ρ -approximate algorithm for GMSTP, where ρ is the maximum cardinality of any imprecise vertex. This immediately provides a 4-approximate algorithm for the 2-GMST problem. The algorithm solves the linear programming relaxation of the integer programming formulation of the problem, and then chooses a spanning tree from the points in the solution. This result follows the approximation framework used by Slavík [147, 148] to establish approximation algorithms for the Generalized Travelling Salesman Problem (GTSP) and the Group Steiner Tree problem with approximation factors of $3\rho/2$ and 2ρ , respectively.

Recently, Jørgensen et al. [100] presented results on a number of geometric problems such as determining the radius of the smallest enclosing ball with a model where each point is represented with a set of points in the input, which they call indecisive points, where the set of points is intended to represent the set of candidate positions for the point. A polynomial time algorithm is described for the smallest enclosing ball problem in this setting, and they demonstrate that computing the diameter of the point set is #P-hard.

For the Travelling Salesman Problem, this generalized setting is usually called Group TSP. The solution to Group TSP is the minimum weight tour which visits a point in each element of \mathcal{S} . For Group TSP, there is a $(1 + \varepsilon)(9\alpha + 1)$ approximation algorithm for groups that may be disjointly covered by α -fat objects, and an $O(1)$ approximation algorithm if such objects are allowed to intersect [61].

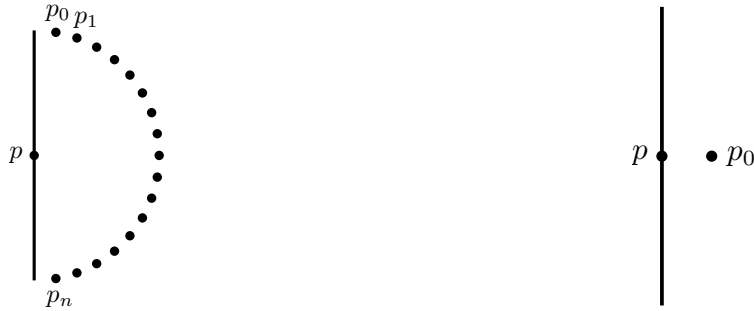
10.2 Minimum Spanning Trees on Intervals

For convex hulls, one of the common settings is to model the areas of uncertainty as parallel line segments [113]. For convex hull algorithms, the vertices that contribute to the solution are invariably one of the endpoints of the segments and so this setting may be reduced to the model of imprecision used for 2-GMST. However, this same property does not hold for the Minimum Spanning Tree problem on line segments, as we show with Lemma 10.1.

Lemma 10.1. *Vertices of the optimal maximum weight Minimum Spanning Tree computed over parallel line segments may be found in the interior of the line segments.*

Proof. We present a proof by example in Figure 10.1a. Suppose point p can be chosen from within the segment $(0, [-10, 10])$, and all other points p_0, \dots, p_n are fixed, with p_0 at $(1, 10)$, p_n at $(1, -10)$, and the other $n - 1$ points have $x > 1$ and lie densely on the circle of radius 10 centred at $(0, 0)$. Then the maximum weight MST is attained when p is at the coordinates $(0, 0)$, the midpoint of the segment. \square

The lemma and proof are easily modified to show that the conditions for the minimum weight MST on intervals are similar, as shown in Figure 10.1b.



(a) The maximum weight Minimum Spanning Tree uses a point p located in the middle of the interval.

(b) The minimum weight Minimum Spanning Tree uses a point p located in the middle of the interval.

Figure 10.1: Constructions which use a point in the middle of an interval when constructing the minimum and maximum weight MSTs.

10.3 One-Dimensional 2-GMST and MAX-2-GMST

Consider the one-dimensional versions of the 2-GMST and MAX-2-GMST problems, where all points are all collinear, and assume the line is vertical. It is not surprising that both problems are easy (each solution consists of an interval over the points, and there are $O(n^2)$ such intervals), but the nature of the solutions illuminates some of the fundamental differences between the problems. The solution to 2-GMST is solved using a linear scan of the vertices, while the solution to MAX-2-GMST has only three possible configurations. To begin, we sort all of the points by descending value of y-coordinate, and label the points so that the set is of the form $\mathcal{P} = \{p_1^+, \dots, p_i^+, \dots, p_i^-, \dots, p_j^-\}$, i.e., p_i^+ appears before p_i^- and p_i^+ appears before p_j^+ if $i < j$. Note that in the set \mathcal{P} , the value of j in the subscript of the last point can take any value in $\{1, \dots, |\mathcal{P}|\}$. The sorting step dominates the running time of both algorithms.

10.3.1 2-GMST Solution

The solution is the minimum weight interval which spans at least one point from each imprecise vertex. The solution may be found by running a scan over the points from top to bottom. We begin with the upper point from each imprecise vertex as the solution (i.e., $T = \cup_{i \in \{1, \dots, |\mathcal{P}|\}} \{p_i^+\}$) and we take note of the weight of the solution (call it w_{\min}). In the iterative step, we remove the top-most point from the solution, which is the point with the smallest value in the subscript, call it p_i^+ . We insert the lower point from the corresponding imprecise vertex, p_i^- , and adjust the value of l_{\min} accordingly. The procedure concludes when the upper-most point in the solution is the lower point of an imprecise vertex, i.e. some p_j^- , since removal of p_j^- would mean that T no longer spans the set of input vertices. We test all of the relevant possible solutions in linear time, and can output the structure and size of the solution to 2-GMST.

10.3.2 max-2-GMST Solution

The solution is trivial. It consists of the interval spanned by p_1^+ and p_j^- , where p_j^- is the last point in the sorted set, unless $j = 1$ (i.e. the points come from the same imprecise vertex). If $j = 1$,

we compare the weight of the interval spanning p_2^+ to p_1^- (which we now know is the last point) to the weight of the interval spanning p_1^+ to p_i^- , where p_i^- is the second-to-last point in the set \mathcal{P} . All intermediate points in the solution can be chosen from the imprecise vertices arbitrarily.

10.4 NP-Hardness of 2-GMST

In this section, we establish the hardness of the 2-GMST problem.

Theorem 10.1. *2-GMST is NP-hard in \mathbb{R}^2 . Furthermore, there is no FPTAS for the 2-GMST problem in \mathbb{R}^2 , unless $P=NP$.*

We show the hardness of 2-GMST by reduction from MAX2-SAT, a known APX-hard problem [130]¹. In the MAX2-SAT problem, we are given a set of m clauses, each of which contains two literals from a set of n variables. The objective is to select truth values for the variables so that a maximal number of clauses are satisfied.

Definition 10.3. The MAX2-SAT Problem: *The input to MAX2-SAT consists of a set $X = \{x_1, \dots, x_n\}$ of boolean variables and a set of clauses $C = \{c_1, \dots, c_m\}$, where a clause $c_i \in C$ is of the form $([\neg]x_j, [\neg]x_k)$, where $j, k \in \{1, \dots, n\}$ and $[\neg]$ indicates that the literal may be the negation of the truth value assigned to the corresponding variable. The solution consists of a truth assignment $\phi = (p_1, \dots, p_n) \in \{+, -\}^n$ for X which maximizes the number of satisfied clauses in C .*

10.4.1 Gadgets

We outline the structure of the construction used for the reduction before delving into the details. The hardness construction consists of a number of gadgets on imprecise vertices which we use to assemble a configuration like that shown in Figure 10.2. All vertices corresponding to variables will be placed in order vertically from top to bottom in the construction, and wires (defined below) extend from these vertices to the right. The solution is a tree which is rooted at the left side of the construction, and all leaves are to the right. Vertices corresponding to clauses will only be leaves of the tree in any optimal 2-GMST solution, although not all leaves correspond to a clause. Each clause that may be satisfied in the MAX2-SAT instance reduces the weight of the optimal 2-GMST solution by some fixed amount. Using this, we show that an exact 2-GMST algorithm may be used to determine an optimal solution for any MAX2-SAT instance.

Root

The root of the tree is constructed by aligning many imprecise vertices vertically as shown on the left in Figure 10.2. Generally, the root is composed of imprecise vertices, aligned vertically, the lower point of one placed two units above the upper point of the imprecise vertex below. The pair of points in each imprecise vertex are placed two units apart with two exceptions: one imprecise vertex in the root is aligned horizontally with each imprecise variable vertex (discussed below),

¹In [130], it is shown that MAX2-SAT is MAX SNP-hard, and in [101] it is demonstrated that the closure of MAX SNP is in APX, ruling out the possibility of the existence of a PTAS for all MAX SNP-hard problems if $P \neq NP$. MAX2-SAT was first shown to be NP-complete by Garey et al. [81].

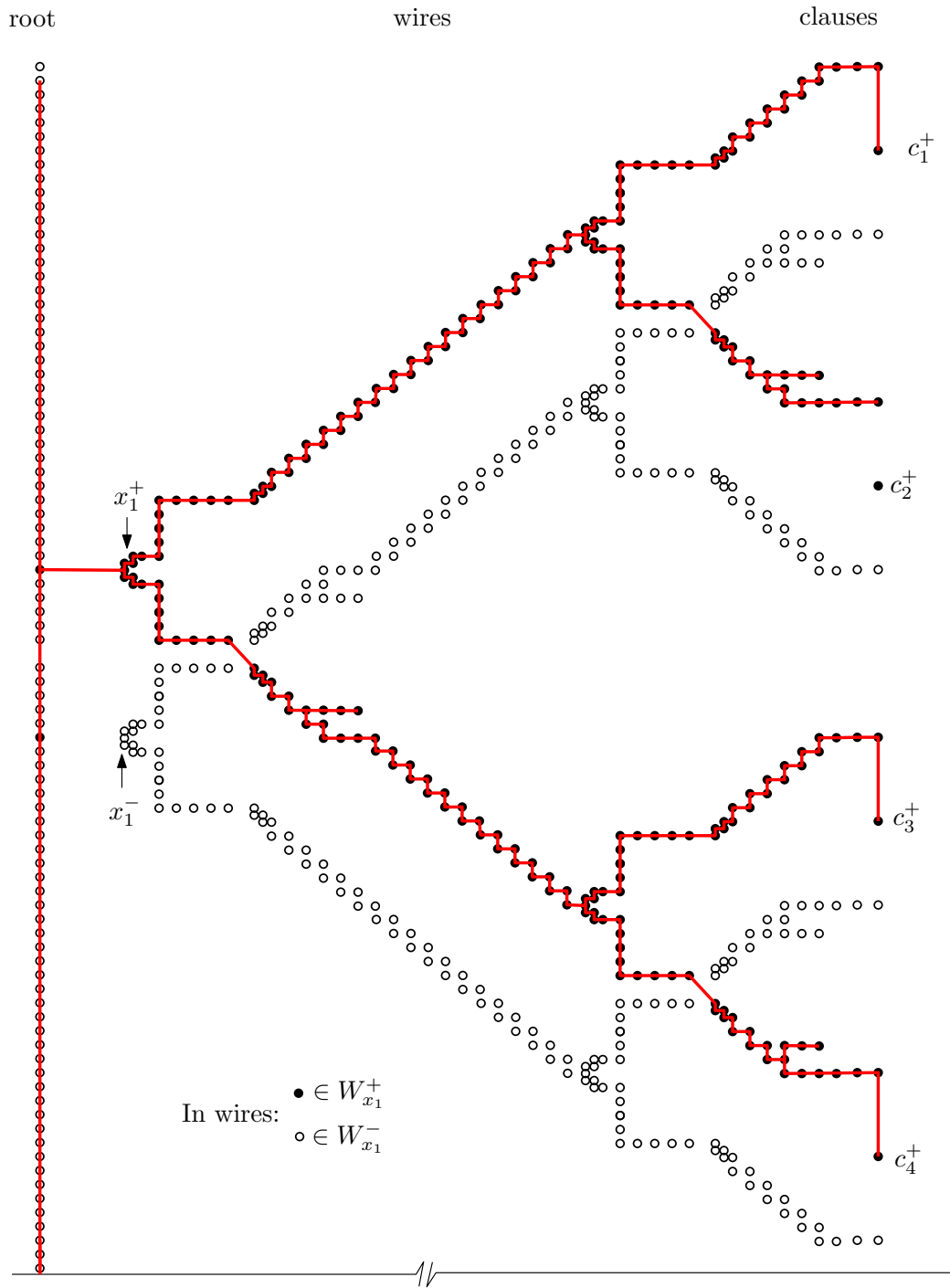


Figure 10.2: An example section (the top half) of a construction for the reduction from MAX2-SAT to 2-GMST. Only the section corresponding to the variable x_1 is shown, and so there is only one imprecise variable vertex: $x_1^\pm = \{x_1^+, x_1^-\}$. The clauses in the MAX2-SAT instance are $C_1 = (x_1 \vee x_2)$, $C_2 = (\bar{x}_1 \vee x_2)$, $C_3 = (x_1 \vee \bar{x}_2)$, $C_4 = (\bar{x}_1 \vee \bar{x}_2)$. Both x_1 and x_2 have been set to **true**, and c_1^+ , c_2^- , c_3^+ and c_4^+ are the imprecise clause vertices used. This is one of several optimal solutions for this configuration. The bottom half of the figure, which is very similar to the top half shown here, is omitted for clarity. The filled circles are points associated with x_1^+ , and the hollow circles are points associated with x_1^- . You may wish to peek ahead to Figure 10.4 to see how the wires are branched.

and so these points are separated by 24 units, and one imprecise vertex with points separated by 4 units is nested in between each of the previous vertices so that we may use integer coordinates. The optimal 2-GMST solution over the root consists of many collinear vertical edges and n horizontal edges, one for each variable of the MAX2-SAT instance, as shown in Figure 10.3(a).

Wires

Each variable x_i of the MAX2-SAT instance is represented by an imprecise vertex x_i^\pm , which we call an *imprecise variable vertex*. The imprecise variable vertices are stacked vertically at the left of the wires section of the construction (see Figure 10.2). Choosing the point x_i^+ of the imprecise variable vertex x_i^\pm is equivalent to setting the value of x_i to **true**, while choosing the point x_i^- in the MAX2-SAT instance is equivalent to setting x_i to **false**. The initial distance between x_i^+ and x_i^- is 24 (all distances are relative to some unit distance).

Each imprecise variable vertex is the first of a sequence of imprecise vertices, which we call a *wire*. For a variable x_i , the wire W_{x_i} consists of a set of imprecise vertices configured so that if the top point w_j^+ of any imprecise vertex $w_j^\pm \in W_{x_i}$ is in our MST, then the top points of all of the imprecise vertices forming the wire will be used in the MST (we define $W_{x_i}^+$ as this set of all top points of the imprecise vertices in the wire W_{x_i} ; analogously $W_{x_i}^-$ is the set of all bottom points of W_{x_i}). In other words, over the set W_{x_i} , given any w_j^\pm where $c_j = +$ (inducing the selection of w_j^+ , recall Definition 10.1), then $\forall k \in \{1, \dots, |W_{x_i}|\}, w_k^\pm \in W_{x_i} \implies c_k = +$. To realize this, the imprecise vertices are placed so that the nearest points to the top point of an imprecise vertex are the top points of the adjacent imprecise vertices in the wire. This argument is analogous for $W_{x_i}^-$.

For each variable x_i in the MAX2-SAT instance, we use the branching gadget (Figure 10.3) to duplicate the wire W_{x_i} extending from the imprecise variable vertex x_i^\pm until we have a number of branches equivalent to the number of occurrences of the variable x_i in the clauses of the input. That is, given an instance of MAX2-SAT, if the variable x_i appears in clauses k times, we need k branches of the wire corresponding to x_i . To ensure that the truth value for the variable is consistent across all branches of the wire, we build the branching gadget shown in Figure 10.3(b). The area where the actual crossing of wires takes place is between $W_{x_i}^-$ of the upper branch and $W_{x_i}^+$ of the lower branch. The points at the crossing form the corners of a 3×4 rectangle. After the crossing, the points of $W_{x_i}^-$ on the upper branch and those of $W_{x_i}^+$ on the lower branch have an extra set of edges (the “extra tail”) so that the path connecting all of the points on the wires has weight 30 (see Figure 10.3(b)). This configuration ensures that the MST over the wires consists only of either the set $W_{x_i}^+$ or $W_{x_i}^-$; the proof of correctness is presented on page 147. The distance between the points in the imprecise vertices of the wires is consistent at 24 prior to each branching as well as before the clause gadgets.

Furthermore, wires are shifted up or down as needed, with imprecise vertices on the wire placed every two units, and the edges of the resulting MST alternate between horizontal and vertical orientations. These are sufficiently dense to ensure that the truth value from the imprecise variable vertex is preserved, since the minimum distance between W^+ and W^- of any wire at any point is always greater than 12.

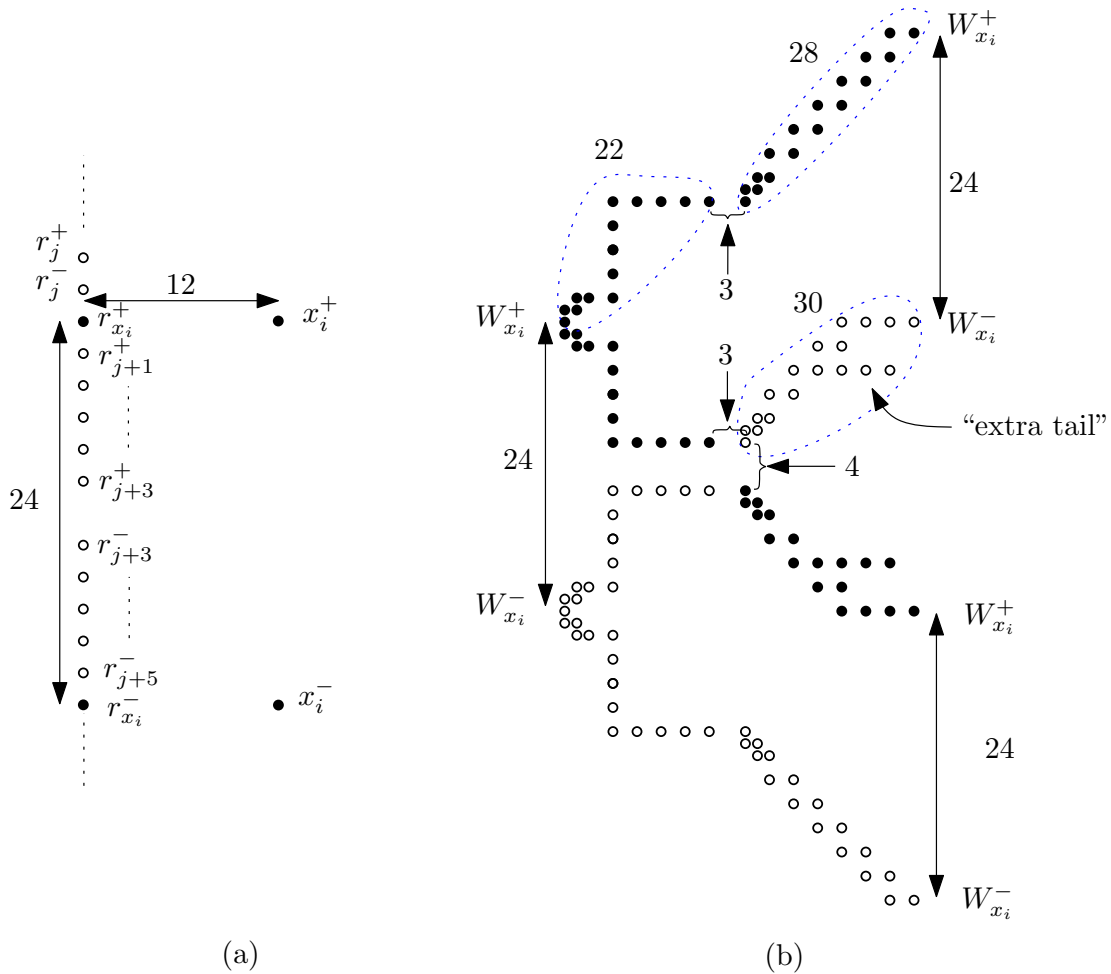


Figure 10.3: The numbers in the figure indicate the weight of the MST over each portion of the gadgets. (a) The root consists of a set of imprecise vertices $R = \{r_1^\pm, \dots, r_{|R|}^\pm\}$ stacked vertically (drawn as pairs of empty circles), as well as an imprecise vertex (pair of filled disks) $r_{x_i}^\pm$ for each imprecise variable vertex x_i^\pm . Using two units of spacing between the imprecise vertices as shown in the figure ensures that there is a vertical edge which joins all of the points along the root, and horizontal edges reach out to the points in the imprecise variable vertices (see Figure 10.2). Note the special imprecise vertex r_{j+3} whose points are separated by 4 units. (b) The branching gadget is used to provide wires from each imprecise variable vertex to all of the clauses where they are required. The spacing between top and bottom points in the imprecise vertices of the wires is consistent at 24 before and after the branching gadget.

Clauses

For each clause C_k in the instance MAX2-SAT, we have an *imprecise clause vertex* c_k^\pm in the clauses portion of the construction. The mapping is simple: if a clause has the form $C_k = (x_i \vee \bar{x}_j)$, then wires extend from each of the imprecise variable vertices x_i^\pm and x_j^\pm to the vicinity of the imprecise clause vertex c_k^\pm . The points in c_k^\pm are set so that they are slightly closer to the truth value on the wire matching that in the clause. For example, given the clause $C_k = (x_i \vee \bar{x}_j)$, the point c_k^+ of c_k^\pm is distance $12 - \beta$ from a point in $W_{x_i}^+$ and $12 + \beta$ from a point in $W_{x_j}^-$ on the same wire (we discuss the precise value of β on page 145). Correspondingly, the point c_k^- is $12 - \beta$ from a point in $W_{x_j}^-$ and $12 + \beta$ from a point in $W_{x_i}^+$. This way, the MST has an extra weight of 2β if and only if the truth values carried by the wires are $x_i = -$ and $x_j = +$, and so the weight of an edge to join c_k^\pm to the MST is $12 + \beta$ regardless of which point is used.

10.4.2 Weight of the MST

We begin by determining the weight of the MST on the wires section of the construction, followed by the root and finally the clauses.

Wires

To simplify this analysis, we increase the weight of the structure by building redundant wires and branching gadgets. Determine $\alpha \in \{1, \dots, n\}$ such that the variable x_α appears in the clauses of the MAX2-SAT instance a maximal number of times over all variables x_1, \dots, x_n ; we call this number of occurrences n' . Clearly, we require n' wires for variable x_α . If n' is not a power of 2, we set n' to the next largest power of 2, so that $n' = 2^\gamma$, $\gamma \in \mathbb{N}$. Note that $\gamma < \log m + 2$, where m is the number of clauses, since n' can be at most $2m$. We create n' wires for each variable x_1, \dots, x_n , so that each set $W_{x_1}^+, \dots, W_{x_n}^+, W_{x_1}^-, \dots, W_{x_n}^-$ has the same cardinality.

The MST over the set of wires has a fixed weight regardless of the truth values of the imprecise variable vertices. When all wires are strictly horizontal (if $n' = 1$) this is trivially true, so we show that the weight remains fixed with branching gadgets by case analysis. Suppose that the MST on the wire for x_i is using the points $W_{x_i}^+$ prior to branching² as W_{u,x_i} and the lower wire as W_{ℓ,x_i} , as shown in Figure 10.4. The possible outcomes after branching are (this analysis is analogous for $W_{x_i}^-$):

1. $W_{x_i}^+ \rightarrow W_{u,x_i}^+, W_{\ell,x_i}^+$ (Figure 10.4(a)) - This route has weight $22 + 3 + 28$ for $W_{x_i}^+ \rightarrow W_{u,x_i}^+$, and $22 + 5 + 30$ for $W_{x_i}^+ \rightarrow W_{\ell,x_i}^+$, for a total of 110.
2. $W_{x_i}^+ \rightarrow W_{u,x_i}^-, W_{\ell,x_i}^+$ (Figure 10.4(b)) - This route has weight $22 + 3 + 30$ for $W_{x_i}^+ \rightarrow W_{u,x_i}^-$, and $22 + 4 + 30$ for $W_{x_i}^+ \rightarrow W_{\ell,x_i}^+$, for a total of 111.
3. $W_{x_i}^+ \rightarrow W_{u,x_i}^+, W_{\ell,x_i}^-$ (Figure 10.4(c)) - This route has weight $22 + 3 + 28$ for $W_{x_i}^+ \rightarrow W_{u,x_i}^+$, plus at least $22 + 28$ for $W_{x_i}^+ \rightarrow W_{\ell,x_i}^-$, and at least another 4 units for the “extra tail”, for a total of 107, but the W_{ℓ,x_i}^- wire is disconnected from the tree. There is no way to

²In general however, these are considered the same wire W_{x_i} , as shown in Figure 10.3(b).

connect the W_{ℓ, x_i}^- wire to the MST built so far so that the total weight of the branching plus the connection is 131 or less, since the branches are set so that the wires are never within distance 24 of another connected part of the tree.

4. $W_{x_i}^+ \rightarrow W_{u, x_i}^-, W_{\ell, x_i}^-$ (Figure 10.4(d))- This route has weight $22 + 3 + 30$ for $W_{x_i}^+ \rightarrow W_{u, x_i}^-$, and $22 + 28$ for $W_{x_i}^+ \rightarrow W_{\ell, x_i}^-$, for a total of 105, but again the W_{ℓ, x_i}^- wire is disconnected from the tree. The total cost is therefore at least 129.

Therefore, the minimum weight feasible path through the gadget is that which preserves the truth values of the variables, and the weight is fixed at 110 regardless of the truth value carried by the wires.

As we mentioned earlier, each variable is split into n' wires from a single initial wire, which requires $n' - 1$ branching gadgets in general, each of which has a weight of 110. Between each level of branching gadgets, we need to shift the wires to create room for the gadgets to follow (see Figure 10.2). Recall that there are $n' = 2^\gamma$ wires created for each variable, so $\log n'$ levels of branching gadgets are required, and thus $\log n' - 1$ levels of shifting wires. Let $h = 1$ be the level of the wires at the clause gadgets, and let $h = \log n'$ be the level of the wires at the imprecise variable vertices. At each level $h \in \{1, \dots, \log n'\}$, the wires are shifted by a vertical distance of $12 \cdot (2^h - 1)$, and $2^{\gamma-h+1}$ wires are needed. Given that the shifting wires climb at 45° using orthogonal edges, the length of the wires is 2 times the vertical shift required. Therefore, the total weight of all the wires needed for shifting each variable is $12 \cdot 2 \cdot 2^{\gamma+1} \log n' = 48 \cdot n' \log n'$. Call the total weight of the MST in the wires section of the construction w_{wires} ; it follows that $w_{\text{wires}} = n \cdot (48 \cdot n' \log n' + 110(n' - 1))$.

Root

The total weight of the root structure for any MST over the structure is simply the distance from the top point of the bottom-most imprecise vertex to the bottom point of the top-most imprecise vertex. Further, we consider the distance to connect the root to the wires. The imprecise vertices of the root are placed to span all of the wires at the clauses section of the construction, as shown in Figure 10.2, so that the total weight of the vertical edges is $48nn'$ (there are nn' wires in total, and there is a vertical distance of 24 both between wires and between points in the imprecise vertices of the wires). By having imprecise vertices in the root aligned horizontally with the imprecise variable vertices, the MST will join the root to these vertices with horizontal edges, since any other option would have greater weight. The weight of these edges is fixed by setting this horizontal distance to 12, so that the total weight over all imprecise variable vertices is $12n$. Let the weight of the root and these horizontal edges be $w_{\text{root}} = 12n + 48nn'$.

Clauses

The only remaining weight to the tree is that of the edges to connect the wires to the imprecise clause vertices. If either one of the terms in clause C_i matches the value carried on the relevant wires, then the edge to connect c_i^\pm has weight $12 - \frac{1}{2n'}$, otherwise it has weight $12 + \frac{1}{2n'}$ (we have now defined $\beta = \frac{1}{2n'}$). If none of the clauses are satisfied, the weight of the clauses section is $w_{\text{clauses_max}} = m \cdot (12 + \frac{1}{2n'})$, where m is the number of clauses in the MAX2-SAT instance.

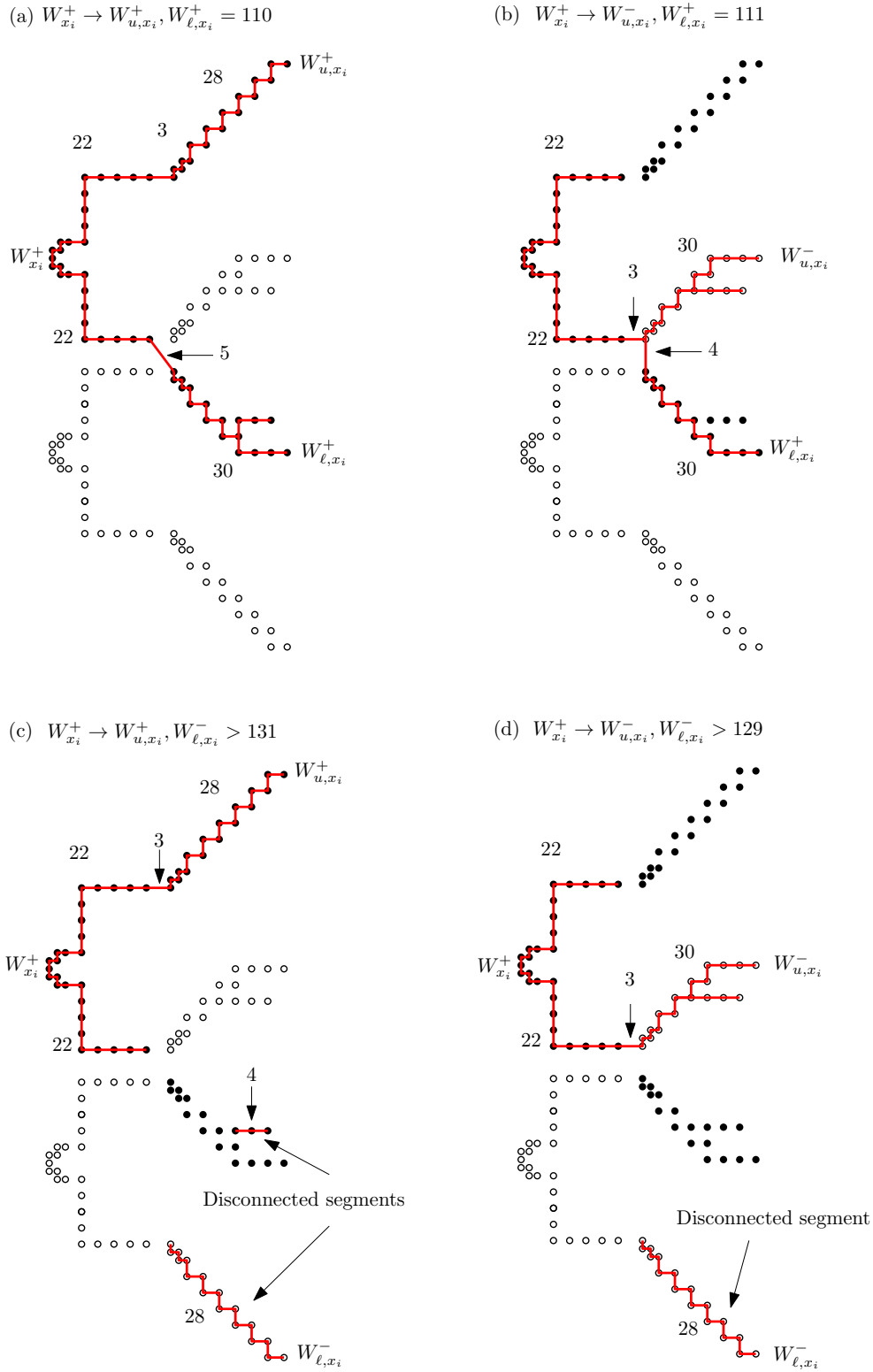


Figure 10.4: The four possible routes through the crossing gadgets are shown. The optimal configuration, shown in (a), is that which preserves the truth value of the incoming wire.

Total Weight

Let w_{\max} be the total weight of the 2-GMST solution, supposing that no clauses of the MAX2-SAT instance are satisfied:

$$\begin{aligned}
 w_{\max} &= w_{\text{root}} + w_{\text{wires}} + w_{\text{clauses.max}} \\
 &= 12n + 48nn' + n \cdot (48 \cdot n' \log n' + 110(n' - 1)) + m \cdot \left(12 + \frac{1}{2n'}\right) \\
 &= 2n \cdot (24 \cdot n' \log n' + 79n' - 49) + m \cdot \left(12 + \frac{1}{2n'}\right)
 \end{aligned} \tag{10.1}$$

Notice that all points except those of the imprecise clause vertices have integer coordinates, and that there are a polynomial number of vertices in the construction. The imprecise clause vertices coordinates, although not integers, have coordinates that are polynomial in the size of the input.

For each satisfied clause, the total weight of the tree reduces by $w_{\text{inc}} = \frac{1}{n'}$. If we subtract the total weight computed for an optimal 2-GMST instance from w_{\max} , the remaining weight is of the form δ/n' , for some $\delta \in \mathbb{N}$. δ is equivalent to the number of unsatisfied clauses, so $m - \delta = k$ is the number of clauses that are satisfied in the MAX2-SAT instance. Therefore, a solution to the reduced 2-GMST instance has a solution with weight at most $w_{\max} - \delta/n'$ if and only if at least k clauses may be satisfied in the MAX2-SAT instance. In the optimization version of 2-GMST, we ask for the minimum weight MST over the input set of imprecise vertices V , and with our reduction an optimal solution to 2-GMST provides an optimal solution to the MAX2-SAT instance. This establishes the NP-hardness of the 2-GMST problem.

Now suppose there exists a FPTAS for the 2-GMST problem. We have demonstrated that the smallest increment in weight for the construction is $w_{\text{inc}} = 1/n'$, which is the increase in weight for each unsatisfied clause of the MAX2-SAT instance. Now if there exists a FPTAS for 2-GMST, we may select a value ε so that it is smaller than the fraction of this increment over the weight of the MST:

$$\varepsilon < \frac{w_{\text{inc}}}{w_{\max}} = \frac{1}{2nn' \cdot (24 \cdot n' \log n' + 79n' - 49) + m \cdot (12n' + \frac{1}{2})},$$

which is polynomial in the size of the input, since $n' \leq 2m$. This choice of ε allows us to solve this instance of 2-GMST optimally, which in turn provides an optimal solution for the MAX2-SAT instance. This completes the proof of the theorem.

For an illustration of the reduction, consider once again Figure 10.2. We see that there are 4 clauses over 2 variables, and so $m = 4$, $n = 2$ and $n' = 4$ (each variable appears in a clause 4 times). Given these values, $w_{\max} = 2 \cdot 2 \cdot (24 \cdot 4 \log 4 + 79 \cdot 2 - 49) + 4 \cdot (12 + \frac{1}{2 \cdot 4}) = 1252.5$ and we know that each satisfied clause reduces this weight by $1/n' = 1/4$. Therefore, by choosing $\varepsilon = \frac{1}{4 \cdot 1252.5} = \frac{1}{5010}$, we may find the optimal solution to the MAX2-SAT instance given the existence of an FPTAS for 2-GMST.

10.5 Approximation Algorithms for 2-GMST

We mentioned that there is a 4-approximation algorithm for 2-GMST which uses an LP-based technique [133]. The 2-GMST problem is resistant to most typical strategies for developing

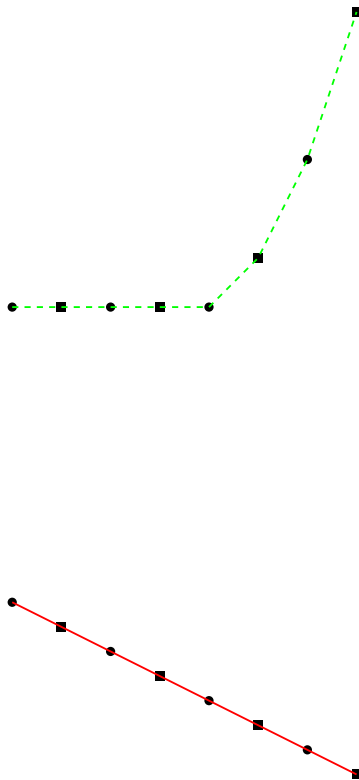


Figure 10.5: A difficult problem for 2-GMST. We place points as shown, where imprecise vertices are alternately drawn with circles and squares from left to right to visually simplify the figure. The optimal 2-GMST solution for this problem (in red) uses each bottom point. If points from the imprecise vertices are added to the solution greedily (e.g., something like Kruskal’s or Prim’s MST algorithms), then all top points would be added to the solution (the green dashed lines), since the top left points have the smallest pairwise distance of any in the input. By increasing the slope between points at the top right of the figure, an arbitrarily poor approximation factor may be realized.

a combinatorial constant factor approximation algorithm. A first attempt usually uses greedy strategies of some sort. In Figure 10.5, we show an example where many greedy strategies fail.

Another line of attack would be to compute the MST of all points in the input set (and so eliminating the imprecision notion for the time being), in the hopes that the optimal 2-GMST solution would be a subset of the edges of this tree. In Figure 10.6 we provide a counter-example to this approach with a construction in which the optimal 2-GMST solution uses edges that are not part of the MST on all points.

10.6 Problems with Known Topology

If an oracle provides us with the topology of the Minimum Spanning Tree, the 2-GMST problem becomes much easier to solve. In this binary setting the imprecision of each vertex v is modelled by two choices v^+ and v^- . Here, we consider the case when the topology of the tree T forming the MST is given.

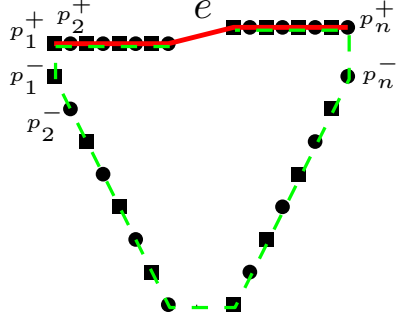


Figure 10.6: Another difficult problem for 2-GMST. We place points as shown, where imprecise vertices are alternately drawn with circles and squares from left to right to visually simplify the figure. The optimal 2-GMST solution for this problem (in red) uses each top point. If all points of the input (all individual points, not just one from each imprecise vertex) are added to the solution greedily, then all points would be added to the solution (the green dashed lines) before the edge e would be considered, yet this edge is part of the optimal solution. In other words, the optimal 2-GMST solution does not use of subset of the edges of the MST over all points. A generalized version of this construction may be used to realize an arbitrarily poor approximation factor for any solution which does not include e .

Theorem 10.2. *An optimal solution to 2-GMST where the topology (combinatorial structure) of the MST is known may be found in $O(n)$ time using a recursive algorithm.*

To find the minimum weight of the MST, given a free tree T that defines its topology, we first root the tree at an arbitrary vertex, say r . For any vertex v , let T_v be the subtree rooted at v . Then we define the following values:

- $mst(T_{v^+})$ = weight of the MST of T_v in which v^+ is chosen.
- $mst(T_{v^-})$ = weight of the MST of T_v in which v^- is chosen.
- $mst(T_v)$ = weight of the MST of T_v = $\min\{mst(T_{v^+}), mst(T_{v^-})\}$.

The weight of the MST in T is $mst(T_r)$, where r is the root of the tree, and it can be computed in constant time once we have $mst^+(T_r)$ and $mst^-(T_r)$.

Base Case

We can express $mst(T_{v^+})$ and $mst(T_{v^-})$ recursively. For the base case, assume that l is a leaf, so T_l contains only one imprecise vertex, l . A tree without any edges has weight 0. So $mst(T_{v^+}) = 0$ and $mst(T_{v^-}) = 0$.

Recursive Case

Lemma 10.2. *Let v be a vertex in T , let w_1, \dots, w_k be the children of v in T , and let $\text{dist}(v^-, w_i^-)$ (resp. $\text{dist}(v^-, w_i^+)$) be the weight of the edge (v, w_i) when we imprecise vertices*

v^- and w_i^- (resp. w_i^+) are chosen. Then

$$mst(T_{v^-}) = \sum_{i=1}^k \min\{\text{dist}(v^-, w_i^-) + mst^-(T_{w_i}), \text{dist}(v^-, w_i^+) + mst^+(T_{w_i})\}. \quad (10.2)$$

The case for v^+ is analogous.

Proof. As the vertex v^- has been fixed, each subtree T_{w_i} rooted at a child node w_i contributes to the weight of the MST rooted at T_{v^-} , independently of its siblings. The minimization considers the two single options for the MST rooted at T_{w_i} and deeper nodes in this subtree can be assumed fixed because they are far enough from v^- (i.e. there is no edge in the MST solution connecting them with v). Combining all individual contributions gives Equation 10.2. \square

10.6.1 2-GMST Algorithm

The algorithm is simple given the above definitions. While doing a post-order traversal of the tree T , compute the values of $mst(T_{v^-})$ and $mst(T_{v^+})$ for all vertices v . Since we use a post-order traversal, upon reaching any node in the traversal the values for the children have been previously computed and can simply be looked up. This can be done in $O(\text{deg}(v))$ time per vertex v and thus the running time of the algorithm is $O(n)$.

10.7 Conclusions and Future Work

We have studied a restricted version of the Generalized Minimum Spanning Tree (GMST) problem, in which each imprecise vertex contains exactly two points, and the points in a set are incident upon a vertical line. We call this version the 2-GMST problem. We have shown that even this setting, perhaps the simplest non-trivial form of the Generalized Minimum Spanning Tree problem, is NP-hard and that no FPTAS exists for the problem unless $\text{P} = \text{NP}$. The hardness of 2-GMST separates from that of GMST when the input graph is restricted to a tree (or if the topology of the solution is known), and we provide an exact recursive solution for this setting on 2-GMST. Since the previous best solution to 2-GMST (a 4-approximation) is geared toward a much broader class of problems, it is tempting to believe that a better approximation algorithm may exist if it were tailored to our setting. However, the problem does not yield to several traditional approaches.

For future work, it will be interesting to see if an approximation algorithm for 2-GMST exists with an approximation factor of less than 4, or alternatively to determine a lower bound on the approximation factor.

In this thesis, we have examined an array of challenging problems in computational geometry, and several of the solutions outlined in this work have advanced the state of the art in these areas. While some of the problems yielded to polynomial time algorithms, the majority of the results presented here are approximation algorithms for hard problems.

In Chapters 2-5, we described the Discrete Unit Disk Cover (DUDC) problem, along with some variants. We showed a simple algorithm for Line-Separated DUDC, and we demonstrated that, rather surprisingly, Within-Strip DUDC is NP-complete. Based on these results, we presented some of the best known approximation algorithms for the DUDC problem.

In Chapters 6-8, we examined a new problem, which we call the Hausdorff Core of a polygon. We demonstrated that the solution may be found in polynomial time on polygons with a single reflex vertex. Contrary to our intuition, demonstrating this result was not a straightforward exercise. Predictably however, the problem on general polygons is more convoluted, and we described parameterized algorithms and an FPTAS for this setting.

Finally, we studied Generalized Minimum Spanning Tree problems in Chapters 9 and 10. We looked at minimization and maximization versions of the Minimum Spanning Trees with Neighborhoods problem, and we demonstrated that each setting is NP-hard. We augment this by bounding the effectiveness of approximation algorithms which simply use the centre point of each disk. The minimization of the 2-GMST problem was also shown to be NP-hard, and we described simple algorithms for the problem in one dimension, or in any number of dimensions if the topology (combinatorial structure) of the solution is known.

There are a number of open problems described in this thesis, and we conclude by providing several high-level research directions that are worth pursuing. For the DUDC problem, there remains a gap in terms of a smooth trade-off between running time and approximation factors for the known approximation algorithms, lying between the PTAS and the remainder of the techniques. There are also many variants and generalizations of DUDC that would be worthy of further study, such as on different types of regions or weighted ranges. For the Hausdorff Core problem, we described a number of other useful metrics for measuring distance between polygons, and the d-Core problem remains unexplored on most of these metrics. Finally, computational geometry on imprecise data is a burgeoning field of study with broad applications. Maximization versions of problems on imprecise input are analogous to the determination of upper bounds, which are a fundamental component of any analysis. Such bounds have not been determined for most problems on any model of imprecision, and so this is a ripe area for further study.

Bibliography

- [1] P. K. Agarwal and C. M. Procopiuc. Exact and approximation algorithms for clustering. In *Proceedings of the Symposium on Discrete Algorithms (SODA)*. ACM Press, 1998. [18](#), [65](#)
- [2] P. K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Computing Surveys*, 30:412–458, 1998. [18](#)
- [3] A. Aggarwal, M. Klawe, S. Moran, P. Shor, and R. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(1):195–208, 1987. [73](#)
- [4] A. Aggarwal and J. Park. Notes on searching in multidimensional monotone arrays. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 497–512. IEEE, 1988. [73](#)
- [5] P. Alimonti and V. Kann. Some APX-completeness results for cubic graphs. *Theoretical Computer Science*, 237(1-2):123 – 134, 2000. [48](#)
- [6] H. Alt, B. Behrends, and J. Blömer. Approximate matching of polygonal shapes. In *Proceedings of the ACM Symposium on Computational Geometry (SoCG)*, pages 186–193, 1991. Full version in: *Annals of Mathematics and Artificial Intelligence*, Vol. 15, pp. 251–265, 1995. [71](#)
- [7] H. Alt and M. Godau. Measuring the resemblance of polygonal curves. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, pages 102–109, New York, NY, USA, 1992. ACM. [71](#)
- [8] C. Ambühl, T. Erlebach, M. Mihal’ák, and M. Nunkesser. Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs. In *Proceedings of the Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 3–14, 2006. [9](#), [10](#), [23](#), [38](#)
- [9] E. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55(3):197 – 218, 1994. [108](#)
- [10] B. Aronov, E. Ezra, and M. Shair. Small-size epsilon-nets for axis-parallel rectangles and boxes. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 639–648. ACM, 2009. [14](#)

- [11] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998. [108](#)
- [12] M. J. Atallah. Some dynamic computational geometry problems. *Computers & Mathematics with Applications*, 11(12):1171 – 1181, 1985. [76](#)
- [13] C. Bajaj. *Geometric Optimization and Computational Complexity*. PhD thesis, Cornell University, 1984. [110](#)
- [14] C. Bajaj. The algebraic degree of geometric optimization problems. *Discrete and Computational Geometry*, 3(1):177 – 191, 1988. [110](#)
- [15] B. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, 1994. [17](#)
- [16] D. H. Ballard. Strip trees: a hierarchical representation for curves. *Communications of the ACM*, 24(5):310–321, 1981. [70](#)
- [17] R. Bellman. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM*, 4(6):284, 1961. [67](#)
- [18] B. Bhattacharya and A. Mukhopadhyay. On the minimum perimeter triangle enclosing a convex polygon. In *Proceedings of the Japan Conference on Discrete and Computational Geometry*, volume 2866 of *LNCS*, pages 84–96. Springer, 2002. [73](#)
- [19] T. Biedl and M. Kaufmann. Area-efficient static and incremental graph drawings. In *Proceedings of the European Symposium on Algorithms*, volume 1284 of *Lecture Notes in Computer Science*, pages 37–52. Springer Berlin / Heidelberg, 1997. [113](#)
- [20] J. Blömer. Computing sums of radicals in polynomial time. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 670–677. IEEE Computer Society, 1991. [110](#)
- [21] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 273–282. ACM, 1986. [14](#)
- [22] P. Bose and G. Toussaint. Computing the constrained Euclidean, geodesic and link centre of a simple polygon with applications. In *Proceedings of the Pacific Graphics International*, pages 102–111. IEEE, 1996. [77](#), [78](#), [98](#), [104](#)
- [23] H. Brönnimann and M. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete and Computational Geometry*, 14(1):463–479, 1995. [9](#), [10](#), [11](#), [14](#)
- [24] B. R. Calder and L. A. Mayer. Automatic processing of high-rate, high-density multibeam echosounder data. *Geochemistry Geophysics Geosystems*, 4(6), 2003. [1](#)
- [25] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, 1995. [113](#)
- [26] S. Cameron. Approximation hierarchies and s-bounds. In *Proceedings of the ACM Symposium on Solid Modeling Foundations and CAD/CAM Applications*, SMA '91, pages 129–137, New York, NY, USA, 1991. ACM. [2](#)

- [27] S. Cameron and C.-K. Yap. Refinement methods for geometric bounds in constructive solid geometry. *ACM Transactions on Graphics*, 11(1):12–39, January 1992. [2](#)
- [28] Caris. Lots browser. <http://www.caris.com/>, 2009. [1](#)
- [29] P. Carmi, M. Katz, and N. Lev-Tov. Covering points by unit disks of fixed location. In *Proceedings of the International Symposium on Algorithms and Computation (ISAAC)*, volume 4835 of *Lecture Notes in Computer Science*, pages 644–655, 2007. [vii](#), [xvii](#), [10](#), [19](#), [21](#), [24](#), [25](#), [33](#), [34](#), [59](#)
- [30] E. Chambers, A. Erickson, S. Fekete, J. Lenchner, J. Sember, S. Venkatesh, U. Stege, S. Stolpner, C. Weibel, and S. Whitesides. Connectivity graphs of uncertainty regions. In *Proceedings of the International Symposium on Algorithms and Computation (ISAAC)*, volume 6507 of *Lecture Notes in Computer Science*, pages 434–445. Springer Berlin / Heidelberg, 2010. [108](#)
- [31] J. S. Chang and C. K. Yap. A polynomial solution for the potato-peeling problem. *Discrete and Computational Geometry*, 1(1):155–182, 1986. [66](#), [73](#), [89](#)
- [32] J.-M. Chassery and D. Coeurjolly. Optimal shape and inclusion. In *Mathematical Morphology: 40 Years On*, volume 30, pages 229–248. Springer, 2005. [71](#), [73](#), [74](#), [75](#)
- [33] D. Chen and O. Daescu. Space-efficient algorithms for approximating polygonal curves in two dimensional space. *Proceedings of Computing and Combinatorics (COCOON)*, LNCS 1449:169–183, 1998. [70](#)
- [34] X. Cheng, X. Huang, D. Li, W. Wu, and D.-Z. Du. A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. *Networks*, 42(4):202–208, 2003. [16](#)
- [35] C.-C. Chou. An efficient algorithm for relay placement in a ring sensor networks. *Expert Systems with Applications*, 37(7):4830 – 4841, 2010. [108](#)
- [36] V. Chvátal. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, 4(3):233–235, 1979. [9](#), [12](#)
- [37] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(13):165 – 177, 1990. [15](#)
- [38] K. L. Clarkson and K. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete and Computational Geometry*, 37(1):43–58, 2007. [17](#)
- [39] F. Claude, G. Das, R. Dorrigiv, S. Durocher, R. Fraser, A. López-Ortiz, B. Nickerson, and A. Salinger. An improved line-separable algorithm for discrete unit disk cover. *Discrete Mathematics, Algorithms and Applications (DMAA)*, 2(1):77–87, 2010. [7](#), [10](#), [19](#), [20](#), [27](#), [59](#)
- [40] F. Claude, R. Dorrigiv, S. Durocher, R. Fraser, A. López-Ortiz, and A. Salinger. Practical discrete unit disk cover using an exact line-separable algorithm. In *Proceedings of the International Symposium on Algorithms and Computation (ISAAC)*, volume 5878 of *LNCS*, pages 45–54. 2009. [7](#), [10](#), [27](#), [59](#)
- [41] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. McGraw-Hill Science/Engineering/Math, July 2001. [11](#), [12](#)

- [42] G. Călinescu, I. I. Măndoiu, P.-J. Wan, and A. Z. Zelikovsky. Selecting forwarding neighbours in wireless ad hoc networks. *Mobile Networks and Applications*, 9(2):101–111, 2004. [9](#), [19](#)
- [43] G. D. Da Fonseca, C. M. H. de Figueiredo, V. G. P. de Sá, and R. Machado. Linear time approximation for dominating sets and independent dominating sets in unit disk graphs. In *Proceedings of the Workshop on Approximation and Online Algorithms (WAOA)*, 2012. [16](#)
- [44] O. Daescu. New results on path approximation. *Algorithmica*, 38(1):131–143, 2003. [70](#)
- [45] G. K. Das, S. Das, and S. C. Nandy. Homogeneous 2-hop broadcast in 2D. *Computational Geometry: Theory and Applications*, 43:182–190, 2010. [19](#)
- [46] G. K. Das, R. Fraser, A. López-Ortiz, and B. G. Nickerson. On the discrete unit disk cover problem. In *Proceedings of the International Workshop on Algorithms and Computation (WALCOM)*, volume 6552 of *Lecture Notes in Computer Science*, pages 146–157. Springer Berlin / Heidelberg, 2011. [10](#), [37](#), [59](#)
- [47] G. K. Das, R. Fraser, A. López-Ortiz, and B. G. Nickerson. On the discrete unit disk cover problem. *International Journal of Computational Geometry and Applications*, to appear, 2012. [10](#), [37](#), [59](#)
- [48] H. Davenport and A. Schinzel. A combinatorial problem connected with differential equations. *American Journal of Mathematics*, 87(3):pp. 684–694, 1965. [76](#)
- [49] M. De, G. Das, and S. Nandy. Approximation algorithms for the discrete piercing set problem for unit disks. In *Proceedings of the Canadian Conference on Computational Geometry (CCCG)*, pages 375–380, 2011. [16](#)
- [50] M. de Berg, J. Gudmundsson, M. Katz, C. Levcopoulos, M. Overmars, and A. van der Stappen. TSP with neighborhoods of varying size. *Journal of Algorithms*, 57(1):22 – 36, 2005. [108](#)
- [51] P. de Fermat. *Oeuvres de Fermat (Tome 1)*. Gauthier-Villars et Fils, Paris, 1891. [110](#)
- [52] D. Dobkin and L. Snyder. On a general method for maximizing and minimizing among certain geometric problems. In *Proceedings of Foundations of Computer Science (FoCS)*, pages 9–17, 1979. [73](#)
- [53] R. Dorrigiv, S. Durocher, A. Farzan, R. Fraser, A. López-Ortiz, J. Munro, A. Salinger, and M. Skala. Finding a Hausdorff core of a polygon: On convex polygon containment with bounded Hausdorff distance. In *Proceedings of the Workshop on Algorithms and Data Structures (WADS)*, volume 5664 of *LNCS*, pages 218–229. Springer Berlin / Heidelberg, 2009. [65](#), [97](#)
- [54] R. Dorrigiv, R. Fraser, M. He, S. Kamali, A. Kawamura, A. López-Ortiz, and D. Seco. On minimum- and maximum-weight minimum spanning trees with neighborhoods. In *Proceedings of the Workshop on Approximation and Online Algorithms (WAOA)*, 2012. [107](#)
- [55] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973. [67](#), [68](#)

- [56] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons Inc, 1973. [68](#)
- [57] A. Dumitrescu and J. S. Mitchell. Approximation algorithms for TSP with neighborhoods in the plane. *Journal of Algorithms*, 48(1):135 – 159, 2003. [108](#)
- [58] S. Durocher and D. Kirkpatrick. The projection median of a set of points. *Computational Geometry*, 42(5):364 – 375, 2009. Special Issue on the Canadian Conference on Computational Geometry (CCCG 2005 and CCCG 2006). [110](#)
- [59] A. Efrat, L. Guibas, S. Har-Peled, J. Mitchell, and T. Murali. New similarity measures between polylines with applications to morphing and polygon sweeping. *Discrete and Computational Geometry*, 28(4):535–569, 2001. [71](#), [72](#)
- [60] P. Egyed and R. Wenger. Ordered stabbing of pairwise disjoint convex in linear time. *Discrete Applied Mathematics*, 31(2):133–140, 1991. [71](#)
- [61] K. Elbassioni, A. Fishkin, N. Mustafa, and R. Sitters. Approximation algorithms for Euclidean group TSP. In *Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 105–105. Springer Berlin / Heidelberg, 2005. [138](#)
- [62] T. Erlebach, M. Hoffmann, D. Krizanc, M. Mihalák, and R. Raman. Computing minimum spanning trees with uncertainty. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 277–288, 2008. [108](#)
- [63] T. Erlebach and M. Mihalák. A $(4 + \varepsilon)$ -approximation for the minimum-weight dominating set problem in unit disk graphs. In *Proceedings of the Workshop on Approximation and Online Algorithms (WAOA)*, volume 5893 of *Lecture Notes in Computer Science*, pages 135–146. Springer Berlin / Heidelberg, 2010. [15](#)
- [64] T. Erlebach and E. van Leeuwen. PTAS for weighted set cover on unit squares. In *Proceedings of Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 6302 of *Lecture Notes in Computer Science*, pages 166–177. Springer Berlin / Heidelberg, 2010. [10](#), [18](#), [38](#)
- [65] R. Estkowski and J. S. B. Mitchell. Simplifying a polygonal subdivision while keeping it simple. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, pages 40–49. ACM, 2001. [72](#)
- [66] W. M. Faucette. A geometric interpretation of the solution of the general quartic polynomial. *The American Mathematical Monthly*, 103(1):pp. 51–57, 1996. [92](#)
- [67] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998. [9](#), [12](#), [14](#)
- [68] C. Feremans, M. Labbé, and G. Laporte. Generalized network design problems. *European Journal of Operational Research*, 148(1):1 – 13, 2003. [138](#)
- [69] J. Fiala, J. Kratochvíl, and A. Proskurowski. Systems of distant representatives. *Discrete Applied Mathematics*, 145(2):306–316, 2005. [108](#)
- [70] M. Fischetti, H. W. Hamacher, K. Jørnsten, and F. Maffioli. Weighted k-cardinality trees: Complexity and polyhedral structure. *Networks*, 24(1):11–21, 1994. [108](#)

- [71] R. Fleischer, K. Mehlhorn, G. Rote, E. Welzl, and C. Yap. Simultaneous inner and outer approximation of shapes. *Algorithmica*, 8(1):365–389, 1992. [71](#)
- [72] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, 12(3):133–137, 1981. [8](#), [17](#)
- [73] R. Fraser and A. López-Ortiz. The within-strip discrete unit disk cover problem. In *Proceedings of the Canadian Conference on Computational Geometry (CCCG)*, 2012. [7](#), [10](#), [37](#), [59](#)
- [74] R. Fraser and P. K. Nicholson. Hausdorff core of a one reflex vertex polygon. In *Proceedings of the Canadian Conference on Computational Geometry (CCCG)*, pages 183–186, 2010. [65](#), [83](#)
- [75] H. Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10:41–51, 1990. [50](#)
- [76] G. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, 1987. [13](#)
- [77] H. Freeman. On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers*, EC-10(2):260–268, June 1961. [67](#), [70](#)
- [78] H. Freeman. Computer processing of line-drawing images. *ACM Computing Surveys*, 6(1):57–97, 1974. [70](#)
- [79] S. Funke, A. Kesselman, U. Meyer, and M. Segal. A simple improved distributed algorithm for minimum CDS in unit disk graphs. *ACM Transactions on Sensor Networks*, 2(3):444–453, 2006. [16](#)
- [80] M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32(4):pp. 826–834, 1977. [48](#)
- [81] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237 – 267, 1976. [140](#)
- [82] B. Golden, S. Raghavan, and D. Stanojević. Heuristic search for the generalized minimum spanning tree problem. *INFORMS Journal on Computing*, 17(3):290–304, July 2005. [138](#)
- [83] T. Gonzalez. Covering a set of points in multidimensional space. *Information Processing Letters*, 40:181–188, 1991. [17](#), [18](#)
- [84] Google. Google maps. <http://maps.google.com/>, 2009. [1](#)
- [85] R. L. Graham and P. Hell. On the history of the minimum spanning tree problem. *IEEE Annals of the History of Computing*, 7(1):43–57, 1985. [108](#)
- [86] P. M. Gruber. Approximation of convex bodies. In *Convexity and its Applications*, pages 131–162. Birkhauser Verlag, 1983. [65](#), [71](#), [72](#)
- [87] L. Guibas, J. Hershberger, J. Mitchell, and J. Snoeyink. Approximating polygons and subdivisions with minimum-link paths. *International Journal of Computational Geometry & Applications*, 3(4):383–415, 1993. [71](#)

- [88] S. Har-Peled. *Geometric Approximation Algorithms (Mathematical Surveys and Monographs)*. American Mathematical Society, 2011. [14](#)
- [89] D. Haussler and E. Welzl. epsilon-nets and simplex range queries. *Discrete and Computational Geometry*, 2(1):127–151, 1987. [14](#)
- [90] P. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms: Multiresolution surface modeling course. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, 1997. [68](#)
- [91] J. Hershberger and J. Snoeyink. Speeding up the Douglas-Peucker line-simplification algorithm. In *Proceedings of the International Symposium on Spatial Data Handling*, pages 134–143, 1992. [68](#)
- [92] J. D. Hobby. Polygonal approximations that minimize the number of inflections. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 93–102. SIAM, 1993. [70](#)
- [93] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32:130–136, 1985. [8](#), [17](#), [18](#), [38](#)
- [94] B. Hu, M. Leitner, and G. Raidl. Combining variable neighborhood search with integer linear programming for the generalized minimum spanning tree problem. *Journal of Heuristics*, 14:473–499, 2008. [138](#)
- [95] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *Journal of Algorithms*, 26(2):238 – 274, 1998. [16](#)
- [96] R. Z. Hwang, R. C.-T. Lee, and R. C. Chang. The generalized searching over separators strategy to solve some NP-hard problems in subexponential time. *Algorithmica*, 9:398–423, 1993. [18](#)
- [97] H. Imai and M. Iri. Computational-geometric methods for polygonal approximations of a curve. *Computer Vision, Graphics, and Image Processing*, 36(1):31–41, 1986. [67](#), [70](#)
- [98] V. Jarník. O jistém problému minimálním (About a certain minimal problem). *Práce Moravské Přírodovědecké Společnosti*, 6:57–63, 1930. (in Czech, German summary). [108](#)
- [99] D. Johnson. The NP-completeness column: An ongoing guide. *Journal of Algorithms*, 3(2):182–195, 1982. [8](#), [11](#)
- [100] A. Jørgensen, M. Löffler, and J. M. Phillips. Geometric computations on indecisive points. In *Proceedings of the Workshop on Algorithms and Data Structures (WADS)*, volume 6844 of *Lecture Notes in Computer Science*, pages 536–547. Springer Berlin / Heidelberg, 2011. [138](#)
- [101] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. *SIAM Journal on Computing*, 28(1):164–191, 1999. [140](#)
- [102] C. Koeman and F. van der Weiden. The application of computation and automatic drawing instruments to structural generalisation. *Cartographic Journal*, 7(1):47–49, 1970. [68](#)

- [103] A. Kolesnikov and P. Fr anti. Polygonal approximation of closed contours. In *Proceedings of the Scandinavian Conference on Image Analysis (SCIA)*, pages 409–417, 2003. [67](#), [73](#)
- [104] J. Komlós, J. Pach, and G. Woeginger. Almost tight bounds for epsilon-nets. *Discrete and Computational Geometry*, 7(2):163–173, 1992. [14](#)
- [105] I. Koutis and G. Miller. A linear work, $o(n^{1/6})$ time, parallel algorithm for solving planar Laplacians. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1002–1011, 2007. [13](#)
- [106] M. K. Leung and Y.-H. Yang. Dynamic two-strip algorithm in curve fitting. *Pattern Recognition*, 23(1-2):69–79, 1990. [70](#)
- [107] M. Leung and Y.-H. Yang. Dynamic strip algorithm in curve fitting. *Computer Vision, Graphics, and Image Processing*, 51(2):146 – 165, 1990. [70](#)
- [108] N. Lev-Tov. *Algorithms for Geometric Optimization Problems in Wireless Networks*. PhD thesis, Weizmann Institute of Science, 2005. [25](#)
- [109] Z. Li. *Algorithmic Foundation of Multi-Scale Spatial Representation*. CRC Press, Boca Raton, 2007. [72](#)
- [110] C. Liao and S. Hu. Polynomial time approximation schemes for minimum disk cover problems. *Journal of Combinatorial Optimization*, pages 1–14, 2009. [10](#)
- [111] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982. [15](#), [112](#), [113](#), [129](#)
- [112] M. Lobo, L. Vandenberghe, S. Boyd, and H. Lebet. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284(1–3):193–228, 1998. [100](#)
- [113] M. Löffler and M. van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56(2):235–269, 2010. [108](#), [109](#), [110](#), [138](#)
- [114] M. Lopez and S. Reisner. Hausdorff approximation of convex polygons. *Computational Geometry: Theory and Applications*, 32(2):139–158, 2005. [71](#), [73](#)
- [115] Maplesoft, a division of Waterloo Maple Inc. invtrig - maple help. <http://www.maplesoft.com/support/help/Maple/view.aspx?path=invtrig>, 2012. [95](#)
- [116] M. V. Marathe, H. Brey, H. B. Hunt, S. S. Ravi, and D. J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25(2):59–68, 1995. [15](#), [16](#)
- [117] S. Masuyama, T. Ibaraki, and T. Hasegawa. The computational complexity of the m-center problems on the plane. *IEICE Transactions (1976-1990)*, E64-E(2):57–64, 1981. [15](#)
- [118] J. Matoušek, R. Seidel, and E. Welzl. How to net a lot with little: small epsilon-nets for disks and halfspaces. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, pages 16–22, 1990. [13](#), [14](#)
- [119] T. Matsui. Approximation algorithms for maximum independent set problems and fractional coloring problems on unit disk graphs. In *Proceedings of the 1998 Japanese Conference on Discrete and Computational Geometry (JCDCG)*, volume 1763 of *Lecture Notes in Computer Science*, pages 194–200. Springer Berlin / Heidelberg, 2000. [38](#)

- [120] E. Melissaratos and D. Souvaine. On solving geometric optimization problems using shortest paths. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, pages 350–359. ACM, 1990. [73](#)
- [121] N. Mustafa and S. Ray. Improved results on geometric hitting set problems. *Discrete and Computational Geometry*, 44:883–895, 2010. [9](#), [10](#), [11](#), [12](#), [13](#), [18](#), [38](#)
- [122] N. Mustafa and S. Ray. Improved results on geometric hitting set problems. <http://www.mpi-inf.mpg.de/~saurabh/Papers/Hitting-Sets.pdf>, 2009. [10](#)
- [123] N. Mustafa and S. Ray. PTAS for geometric hitting set problems via local search. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, 2009. [12](#), [15](#)
- [124] Y.-S. Myung, C.-H. Lee, and D.-W. Tcha. On the generalized minimum spanning tree problem. *Networks*, 26(4):231–241, 1995. [137](#), [138](#)
- [125] S. Narayanappa and P. Vojtěchovský. An improved approximation factor for the unit disk covering problem. In *Proceedings of the Canadian Conference on Computational Geometry (CCCG)*, volume 18, pages 15–18, 2006. [9](#), [10](#)
- [126] National Oceanic and Atmospheric Administration. A hull-mounted multibeam sonar while towing a sidescan sonar (JPEG image). http://oceanexplorer.noaa.gov/explorations/05fire/background/mapping/media/multi_sonar.html, 2012. [2](#)
- [127] Natural Resources Canada, the Geological Survey of Canada, and Her Majesty the Queen in Right of Canada. Canadian marine multibeam bathymetric data. http://gdr.nrcan.gc.ca/multibath/index_e.php, 2009. [1](#)
- [128] T. Nieberg and J. Hurink. A PTAS for the minimum dominating set problem in unit disk graphs. In *Proceedings of the Workshop on Approximation and Online Algorithms (WAOA)*, volume 3879 of *Lecture Notes in Computer Science*, pages 296–306. Springer Berlin / Heidelberg, 2006. [16](#)
- [129] J. O’Rourke, A. Aggarwal, S. Maddila, and M. Baldwin. An optimal algorithm for finding minimal enclosing triangles. *Journal of Algorithms*, 7:258–269, 1986. [73](#)
- [130] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425 – 440, 1991. [140](#)
- [131] J.-C. Perez and E. Vidal. Optimum polygonal approximation of digitized curves. *Pattern Recognition Letters*, 15(8):743–750, 1994. [67](#), [68](#), [70](#), [71](#)
- [132] T. K. Peucker. A theory of the cartographic line. *International Cartographic Yearbook*, 16:134–143, 1976. [70](#)
- [133] P. C. Pop. *The Generalized Minimum Spanning Tree Problem*. PhD thesis, University of Twente, 2002. [138](#), [147](#)
- [134] P. C. Pop, W. Kern, and G. Still. A new relaxation method for the generalized minimum spanning tree problem. *European Journal of Operational Research*, 170(3):900 – 908, 2006. [138](#)
- [135] E. Pyrga and S. Ray. New existence proofs for epsilon-nets. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, pages 199–207. ACM, 2008. [10](#), [14](#)

- [136] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1:244–256, 1972. 68
- [137] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 475–484, 1997. 8
- [138] K. Reumann and A. Witkam. Optimizing curve segmentation in computer graphics. In *Proceedings of the 1973 International Computing Symposium*, pages 467–472, 1974. 67, 70
- [139] J. Robergé. A data reduction algorithm for planar curves. *Computer Vision, Graphics, and Image Processing*, 29:168–195, 1985. 70
- [140] D. P. Roselle and R. G. Stanton. Some properties of Davenport-Schinzel sequences. *Acta Arithmetica*, 17(4), 1970-1971. 77
- [141] G. Rote. The convergence rate of the sandwich algorithm for approximating convex functions. *Computing*, 48(3-4):337–361, 1992. 71
- [142] Y. Sato. Piecewise linear approximation of plane curves by perimeter optimization. *Pattern Recognition*, 25(12):1535–1543, 1992. 68
- [143] C. Schwarz, J. Teich, A. Vainshtein, E. Welzl, and B. Evans. Minimal enclosing parallelogram with application. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, pages 434–435. ACM, 1995. 73
- [144] J. Sekino. n-ellipses and the minimum distance sum problem. *The American Mathematical Monthly*, 106(3):pp. 193–202, 1999. 114
- [145] M. Sharir and P. Agarwal. *Davenport-Schinzel sequences and their geometric applications*. Cambridge University Press, 1995. 76, 77
- [146] M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 577 of *Lecture Notes in Computer Science*, pages 567–579. Springer Berlin / Heidelberg, 1992. 75
- [147] P. Slavík. The errand scheduling problem. Computer Science Technical Report 97-2, State University of New York at Buffalo, 1997. 138
- [148] P. Slavík. *Approximation Algorithms For Set Cover And Related Problems*. PhD thesis, State University of New York at Buffalo, 1998. 138
- [149] W. H. Smith and D. T. Sandwell. Global sea floor topography from satellite altimetry and ship depth soundings. *Science*, 277(5334):1956–1962, 1997. 1
- [150] H. Stone. Approximation of curves by line segments. *Mathematics of Computation*, 15(73):40–47, 1961. 67
- [151] J. Strong, H. Mitchell, and T. Watters (NASA/Goddard Space Flight Center Scientific Visualization Studio). Hologlobe: Topography and bathymetry on a globe. <http://svs.gsfc.nasa.gov/vis/a000000/a001300/a001305/index.html>, 1996. 1

- [152] K. J. Supowit. *Topics in Computational Geometry*. PhD thesis, University of Illinois at Urbana-Champaign, 1981. [17](#)
- [153] C. Toregas, R. Swain, C. ReVelle, and L. Bergman. The location of emergency service facilities. *Operations Research*, 19(6):pp. 1363–1373, 1971. [7](#)
- [154] G. Toussaint. Solving geometric problems with the rotating calipers. In *In Proceedings of IEEE Mediterranean Electrotechnical Conference (MELECON)*, pages 1–8, 1983. [37](#)
- [155] P. Unwin. *The Wolf's Head: Writing Lake Superior*. Viking Canada, 2003. [1](#)
- [156] J. E. (user JεffE). Sum-of-square-roots-hard problems? *Theoretical Computer Science on StackExchange*, URL: <http://cstheory.stackexchange.com/questions/4053/sum-of-square-roots-hard-problems>, 2011. [110](#)
- [157] V. Vazirani. *Approximation Algorithms*. Springer, 2nd edition, 2004. [12](#)
- [158] P.-J. Wan, K. Alzoubi, and O. Frieder. Distributed construction of connected dominating set in wireless ad hoc networks. In *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 3, pages 1597 – 1604, 2002. [16](#)
- [159] E. White. Assessment of line-generalization algorithms using characteristic points. *Cartography and Geographic Information Science*, 12:17–28(12), April 1985. [68](#)
- [160] Wikipedia contributors. Atan2. <http://en.wikipedia.org/w/index.php?title=Atan2>, 2012. Page Version ID: 480067366. [95](#)
- [161] S.-T. Wu and M. R. G. Márquez. A non-self-intersection Douglas-Peucker algorithm. *Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing*, 2003. [72](#)
- [162] W. Wu, H. Du, X. Jia, Y. Li, and S. C.-H. Huang. Minimum connected dominating sets and maximal independent sets in unit disk graphs. *Theoretical Computer Science*, 352(13):1 – 7, 2006. [16](#)
- [163] X. Xu and Z. Wang. Wireless coverage via dynamic programming. In *Proceedings of Wireless Algorithms, Systems, and Applications (WASA)*, volume 6843, pages 108–118. LNCS, 2011. [24](#)
- [164] D. Yang, S. Misra, X. Fang, G. Xue, and J. Zhang. Two-tiered constrained relay node placement in wireless sensor networks: Efficient approximations. In *Proceedings of IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 1 –9, 2010. [7](#)
- [165] Y. Yang. *On several geometric network design problems*. PhD thesis, State University of New York at Buffalo, 2008. [108](#)
- [166] Y. Yang, M. Lin, J. Xu, and Y. Xie. Minimum spanning tree with neighborhoods. In *Proceedings of Algorithmic Aspects in Information and Management (AAIM)*, volume 4508 of *Lecture Notes in Computer Science*, pages 306–316. Springer Berlin / Heidelberg, 2007. [108](#), [126](#), [127](#)

- [167] L. Zhang and Z. Tian. Refinement of Douglas-Peucker algorithm to move the segments toward only one side. In *Proceedings of the International Cartographic Conference*, pages 831–835, 1997. [72](#)

- k -separability, 128
- 1-Centre, 77
- Approximation Algorithm, 9
- Approximation Factor, 9
- Assisted Cover, 21
- Assisted Line-Separated Discrete Unit Disk Cover (A-LSDUDC) Problem, 20
- Constrained Euclidean Centre, 77
- d -Core Problem, 73
- Davenport-Schinzel sequence, 76
- Discrete Unit Disk Cover (DUDC) Problem, 7
 - Approximation Algorithms, 9
 - PTAS, 12
- ε -net, 14
- e-wire, 118
- Fermat-Weber Problem, 110
- Geometric Disk Cover, 16
- Hausdorff Core Problem, 66
 - Decision Version, 79
 - Hausdorff Core, 66
- Hausdorff Distance, 65
- Hitting Set, 11
- Imprecise Vertex, 137
- Line-Separable Discrete Unit Disk Cover (LSDUDC) Problem, 18
- LP-Type Problems, 75
- MAX-2-GMST Problem, 137
- MAX-MSTN Problem, 107
- 2-GMST Problem, 137
- MSTN Problem, 107
- MAX2-SAT Problem, 140
- Minimum Connecting Tree, 126
- Minimum Dominating Set, 15
- Outside Strip Discrete Unit Disk Cover, 60
- Piercing, 11, 19
- Polygon, 66
 - Convex, 66
- Polynomial Time Approximation Scheme (PTAS), 9
- Portion, 88
- Range Space, 10
- Semi-chain, 21
- Set Cover, 11
- Shifting Lemma, 17
- Shifting Parameter, 17
- Spinal Path, 129
- Strip-Separable Discrete Unit Disk Cover (SSDUDC) Problem, 21
- Unit Disk Graph, 15
- Within-Strip Discrete Unit Disk Cover (WS-DUDC) Problem, 37