On Quadtrees, Voronoi Diagrams, and Lattices: Results in Geometric Algorithms

by

Huxley David Bennett

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

September 2017

Chee Yap

Daniel Dadush

Acknowledgements

I can't imagine a better academic environment than Courant, and I feel truly privileged to have been a part of it for the past five years.

First and foremost I would like to thank my advisors, Chee Yap and Daniel Dadush, for their guidance and support. Chee introduced me to the world of computational geometry, and guided me through my first research. I thank him for his patience, kindness, and reliability. Starting when he was a postdoc at NYU, Daniel taught me about lattices and many other things. His enthusiasm is infectious, and his vast knowledge and knack for picking good research questions are inspiring. I thank him for his patience, dedication, and generally excellent mentorship.

Alongside Chee and Daniel, I would like to thank Igor Shinkar as another important mentor. Igor was in every way an unofficial advisor, and was also one of my good friends at NYU. I thank him especially for the tremendous amount of time he dedicated to mentoring and working on problems with me and other students.

I would also like to thank the other members of my thesis committee: Richard Cole, Subhash Khot, and Oded Regev. They continue to inspire me as both amazing researchers and teachers. Subhash's class on "Computational Complexity" and Oded's class on "Cryptography" are among my favorites of all time. It really stuck with me that Subhash, after learning in the spring of 2014 that he had won the Nevanlinna Prize, continued to deliver the same enthusiastic, flawless lectures to our tiny six-person class as before.

Next, I would like to thank several key teachers and mentors from before my time at NYU. To begin with, I would like to thank Shawn Tank at Longmont High

School for introducing me to computer science.

I would like to thank Shuchi Chawla and Jin-Yi Cai at Wisconsin. Shuchi introduced me to the world of research, and Jin-Yi taught a fantastic "Introduction to Theoretical Computer Science" class which led me to pursue the subject further as much as anything.

I would like to thank Sriram Sankaranarayanan and Evan Chang at Colorado. Sriram was my advisor during my time as a master's student at CU, and bent over backwards to help me in every way possible. I can't thank him enough. Evan taught a fantastic "Compilers" class, which single-handedly made me a proper programmer and directly led to all of my industry jobs since.

I also thank my two bosses during summer internships at Google, Saurabh Nasa and Dave Detlefs, for being great mentors, and my not-elsewhere-mentioned co-authors, Evanthia Papadopoulou and Daniel Reichman, for collaborating with me.

I thank Esther Ezra and John Iacono for their early work serving on my depth qualifying exam committee. I thank Joe Mitchell, Jon Lenchner, and Alexander Kulikov for making it possible for me to attend various interesting workshops. I thank Rosemary Amico for being the world's most competent administrator and for holding the NYU computer science department together.

Perhaps most of all, I would like to thank my wonderful and inspiring friends at Courant: Azam Asl, Jop Briët, Sandro Coretti, Yair Daon, Sylvester Eriksson-Bique, Laura Florescu, Chaya Ganesh, Alessandro Giacomini, Sasha Golovnev, Siyao Guo, Tom Hsu, Siddharth Krishna, Matt Morse, Shravas Rao, Igor Shinkar, Noah Stephens-Davidowitz, Devanathan Thiruvenkatachari, Sara Vagnarelli, Ashwin Venkataraman, Aravindan Vijayaraghavan, and Omri Weinstein.

During the fall of 2017 I spent a wonderful semester at CWI in Amsterdam, and I would like to thank everyone in the Networks and Optimization group there at once: a big “dank je!”

Before NYU I spent two great years as a master’s student at the University of Colorado, and I would like to thank everyone in the CUPLV group. I especially want to recognize my perennial academic collaborator, Aleks Chakarov, who became and remains one of my best friends. I also thank Devin Coughlin whose great competence and charming grouchiness I looked up to.

Finally, I would like to thank my family. I thank Linda and David Kahn, Ellen and Hy Portnoy, and all of their offspring. I particularly thank Ellen who despite having a large direct family tree of her own always made sure to include me. I thank my Aunt Mary Ann Koslov, Uncle Josh Koslov, Cousin Liz Koslov, Cousin Stephen King, and Aunt Julie Parker who are all awesome to hang out with.

I also want to thank a few people who are essentially family: Anna, Augustine, and Mary Louise Barón; John Lyon Paul and Katy Gottschalk; and Erica Gruen. Special thanks to Erica for taking me in during Hurricane Sandy. I won’t even try to name them here, but many thanks also go to my friends outside of NYU.

I thank my stepmother Kara Edin and stepfather Béla Scheiber for their support and lots of fun times. I give huge thanks to my girlfriend for making my world bright. Lastly, I thank my father Stith Bennett and mother Judy Koslov for all of their love and for always being there for me.

Preface

The unifying theme of this thesis is *geometric algorithms*. It contains two parts, and is based on content from four papers.

Part I is on *subdivision algorithms*, and has content based on the papers "Amortized Analysis of Smooth Quadtrees in All Dimensions", joint with Chee Yap [BY17]; and "Planar Minimization Diagrams via Subdivision with Applications to Anisotropic Voronoi Diagrams", joint with Evanthia Papadopoulou and Chee Yap [BPY16].

Part II is on *lattice algorithms*, and has content based on the papers "On the Lattice Distortion Problem", joint with Daniel Dadush and Noah Stephens-Davidowitz [BDS16]; and "Algorithms for Computing Nearly Orthogonal and Well-Conditioned Lattice Bases" [Ben17].

A note on co-authorship

As with anyone writing a thesis based on already-published, joint work, I was faced with the question, "To what extent should I rewrite what we've already done?" Throughout, I have tried to make the content cohesive and more detailed when possible. Moreover, I have made modifications which reflect my own view on the work.

Nevertheless, the original papers often described our work as best as I know how, and therefore parts of this thesis are very similar or identical to the writing in its constituent papers. Therefore, not only the results but also the writing in chapters corresponding to joint papers should be considered joint work with my respective co-authors. Of course, all remaining and introduced errors are my own.

I thank all of my co-authors for allowing me to include our joint work in this thesis.

A note on open problems

A goal of mine for this thesis is to describe not only what I have done so far, but what open questions remain. Throughout the thesis I have used the theorem environment “**Open Problem**” to indicate open problems explicitly. I hope that they will serve as a guide both for me and for others wishing to extend the work described in this thesis in the future.

Abstract

The unifying theme of this thesis is *geometric algorithms*, and somewhat more specifically algorithmic aspects of geometric structures including quadtrees, Voronoi diagrams, and lattices. It contains two parts, the first of which is on *subdivision algorithms*, and the second of which is on *lattice algorithms*.

Part I of this thesis is on subdivision algorithms. In Chapter 1, we study the amortized cost of smooth splits in quadtrees and their higher-dimensional analogs. A quadtree is *smooth* if any two adjacent leaf boxes differ by at most one in depth. A basic operation on a quadtree is to expand it by splitting any given leaf. We analyze quadtrees that restore smoothness after each split operation and also maintain neighbor pointers. Our main result shows that the *smooth split* operation in such quadtrees has an amortized cost of at most $2^D \cdot (D+1)!$ auxiliary split operations, which corresponds to amortized constant time in quadtrees of any fixed dimension D.

In Chapter 2, we present a subdivision-based algorithm for computing isotopic ε-approximatations of planar minimization diagrams. Given a family $\mathcal{F} = \{f_1, \ldots, f_n\}$ of continuous functions with $f_i : \mathbb{R}^2 \to \mathbb{R}$, the minimization diagram of $\mathcal{F}$ partitions the plane into regions on which f_i is minimal. Minimization diagrams generalize many natural Voronoi diagrams, and we show how to use our framework to compute an anisotropic Voronoi diagram on polygonal sites. We have implemented a prototype of our algorithm for anisotropic Voronoi diagrams, and provide experimental results. Our algorithm uses the smooth quadtree studied in Chapter 1 as its primary underlying data structure. We note that the focus of Chapter 2 is both more conceptual and more applied than other chapters.

Part II of this thesis is on lattice algorithms. In Chapter 3, we provide back-

ground material about linear algebra and lattices for the following chapters. In Section 3.3 we also give a high-level overview of the connections between fundamental domains, algorithms for the closest vector problem, and basis reduction which builds context for notions of basis reduction studied in the remaining two chapters.

In Chapter 4 we introduce and study the *Lattice Distortion Problem* (LDP). LDP asks how "similar" two lattices are, i.e., what the minimum distortion of a linear bijection between two lattices is. We first show that the distortion between any two lattices is approximated up to a $n^{O(\log n)}$ factor by a simple function of their successive minima. Our methods are constructive, allowing us to compute low-distortion mappings with a tradeoff between approximation quality and running time. Our algorithms rely on a notion of basis reduction introduced by Seysen (Combinatorica 1993), which we show is intimately related to lattice distortion. Lastly, we show that LDP is NP-hard to approximate to within any constant factor (under randomized reductions).

Finally, in Chapter 5 we study how to compute lattices bases that (approximately) minimize two basis quality measures. Namely, we study the problem of finding bases B with low orthogonality defect $\delta(B)$ and with low Seysen condition number $S(B)$ (the quality measure used to bound the distortion between two lattices in Chapter 4). Our main results are algorithms for computing bases B of a lattice which minimize $\delta(B)$ and $(1+\varepsilon)$-approximately minimize $S(B)$, while running in time only depending only on the rank of the lattice times a polynomial in the input length. Both algorithms are enumeration-based, and work by breaking a lattice into pieces according to gaps in its successive minima, a technique which may be of independent interest.

Contents

List of Figures

List of Tables

Part I

Subdivision Algorithms

Chapter 1

Amortized Analysis of Quadtree Smoothing

This chapter is based on the publication [BY17] and its preliminary version [BY14], both of which were joint work with Chee Yap.

1.1 Introduction

Quadtrees [dBCvKO08, FB74, Sam90] are a well-known data structure for representing geometric data in two dimensions. In this case there exists a natural one-to-one correspondence between quadtree nodes v and boxes B in an underlying subdivision of a square; see Figure 1.1. (We therefore abuse notation slightly, and refer to boxes and nodes interchangeably throughout this chapter.) Here we consider the extension to a subdivision of a D-dimensional box in which an internal node is a box containing 2^D congruent sub-boxes.[1] We refer the reader to Chapter

[1]We continue to use the term *quadtrees* for such higher-dimensional extensions, which are also frequently called *octrees* for $D \geq 3$.

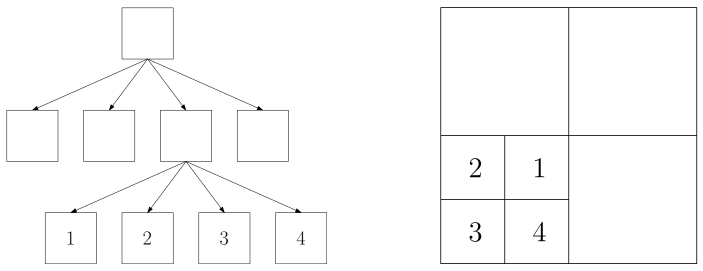


Figure 1.1: A quadtree (left) and its corresponding subdivision (right).

14 in [dBCvKO08] whose nomenclature we largely follow.

Two boxes (or nodes in a quadtree) are *adjacent* if the boxes share a $(D-1)$-dimensional facet, but have disjoint interiors. The *neighbors* of a box B are those boxes adjacent to B. We call a quadtree *smooth* if any two adjacent leaf boxes differ by at most one in height. Other sources use the term *balanced* to refer to this condition, which we avoid in order to avoid conflation with the standard meaning of balanced trees in computer science.

We study three operations on quadtrees: `split`, `smooth`, and `neighbor_query` as well as the hybrid operation `ssplit` which combines a `split` and a `smooth`.

A basic operation is a *split* of a leaf box B, written `split`(B). This divides B into 2^D congruent sub-boxes which become its children (B is no longer a leaf). A `split` operation is a useful abstraction of many common operations performed on quadtrees including point insertion and mesh refinement. A `smooth` operation performs the unique minimum sequence of splits necessary to restore smoothness. A *smooth split* operation `ssplit`(B) consists of performing a split `split`(B) followed by a `smooth` of the resulting tree; see Figure 1.2.

Let $d \in \{\pm e_1, \pm e_2, \ldots, \pm e_D\}$ identify one of the $2D$ semi-axis directions (here e_i denotes the ith standard normal vector). If box B' is a neighbor of B, and the depth of B' is maximal subject to $\text{depth}(B') \leq \text{depth}(B)$ over all neighbors of B

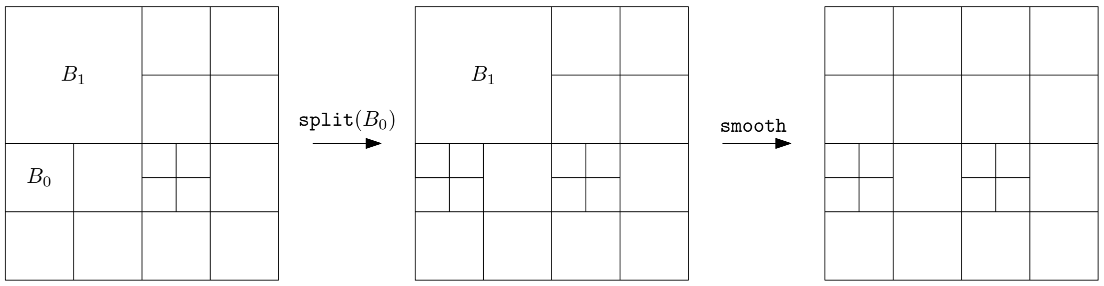


Figure 1.2: A smooth split operation `ssplit`(B_0). After performing `split`(B_0), the width of the leaf box B_1 is four times the width of the children of B_0, which are its neighbor leaf boxes in the subdivision. (Equivalently, the depth of the children of B_0 is two more than the depth of B_1 in the quadtree.) Therefore, the `smooth` operation splits B_1 as well to restore smoothness to the quadtree.

in direction d, then we call B' the *principal d-neighbor* of B. We note that the principal d neighbor of a box B is unique if it exists (it may not if B is on the boundary of the subdivision). A *neighbor query* operation `neighbor_query`(B, d) returns the principal d-neighbor of B, or `NULL` if B is on the subdivision boundary and has no d-neighbors.

In many quadtree applications, such as [WCY13] and Chapter 2, one is interested in the set of leaf neighbors of a box. The goal is to enumerate these in $O(1)$ time per leaf neighbor. We can achieve $O(1)$ time neighbor queries by giving each box a constant number of pointers to its principal neighbors. We can then enumerate all leaf neighbors of a box by performing a neighbor query in each direction, and enumerating the appropriate children of each neighbor. Without such pointers neighbor queries require $\Theta(h)$ time in order to traverse to the nearest common ancestor in a tree of height h. We also motivate our work by showing that a tree with neighbor pointers must maintain smoothness to ensure $O(1)$ time splits.

This neighbor enumeration functionality makes smooth quadtrees useful in motion planning [WCY13]. They are also useful in other domains including good mesh generation [dBCvKO08, BEG94].

1.1.1 The Smooth Quadtree Model

In this chapter we present and analyze a quadtree model that we call the *smooth quadtree*, which maintains smoothness as an invariant between splits via the smooth split operation, and maintains principal neighbor pointers. This model has been proposed before such as in Exercise 14.8 in [dBCvKO08], but to the best of our knowledge the complexity of smooth splits has never been studied rigorously. To provide context for our smooth quadtree model, we discuss two options in designing quadtrees:

1. A quadtree can either maintain or not maintain neighbor pointers. We use the letters `P` (Pointer) and `N` (No Pointer) to denote this.

2. A quadtree can either maintain or not maintain smoothness as an invariant. It maintains smoothness by replacing the `split` operation with `ssplit`. We use the letters `S` (Smooth) and `U` (Unsmooth) to denote this.

If a quadtree maintains neighbor pointers, we assume that the pointers are to its $2D$ principal neighbors. Then the `neighbor_query` operation requires worst case $O(1)$ time. These considerations give rise to four models of quadtrees: `PS`, `PU`, `NS`, `NU`, where our smooth quadtree corresponds to the `PS` quadtree model because it maintains both pointers and smoothness. We also refer to the `NU` quadtree model as the *simple* quadtree model, which is frequently used as the primary definition of a quadtree (see, e.g., [dBCvKO08]). Intermediate between these two extreme models are the `PU` and `NS` models. `NS` quadtrees are similar to `PS` quadtrees, but

	Smooth (`PS`)	`NS`	`PU`	Simple (`NU`)
`neighbor_query`	$\Theta(1)$	$\Theta(h)$	$\Theta(1)$	$\Theta(h)$
`ssplit/split`	Amortized $\Theta(1)$	Amortized $O(h)$	Amortized $\Omega(\log n)$	$\Theta(1)$
`smooth`	(Invariant)	(Invariant)	$\Omega(n \log n)$	$O((h+1)n)$

Table 1.1: A comparison of the time complexity of operations in four two-dimensional quadtree models. Here h denotes the height of the tree, and n denotes the number of nodes in the tree. Costs are worst-case unless otherwise noted. All four models have $\Theta(n)$ space complexity. The `PU` lower bounds for smoothing and smooth splits follow from Lemma 1.2.8, and the `NU` upper bound for smoothing follows from Fact 1.1.3. The `PS` and `NS` models maintain smoothness as an invariant.

lose a factor of h in the cost of `neighbor_query` and `ssplit` because traversing to the nearest common ancestor requires $\Omega(h)$ time in the worst case.

Table 1.1 compares the cost of our three main operations on these quadtree models. We use n to denote the number of nodes in a quadtree, and use h to denote its height.

The smooth (`PS`) quadtree achieves improvements to the `neighbor_query` and `smooth` operations at the cost of `split` operations requiring amortized rather than worst-case $O(1)$ time. The $O(1)$ time bounds for the `ssplit` and `split` operations are for the "local operations", i.e., when the algorithm already has a pointer to the box it wishes to split and does not need to traverse from the root. This is common in meshing applications which maintain a collection of boxes to be refined, such as in [WCY13] and the subdivision-based approach to Voronoi diagrams described in Chapter 2.

Algorithm 1 shows the simplicity of the algorithm for performing smooth splits: simply recursively check whether any neighbors of a node need to be split to regain smoothness. The correctness of this algorithm is also straightforward. Indeed, because we maintain smoothness as an invariant, the only boxes potentially violating smoothness after `ssplit`(B) are the neighbors of B. Nevertheless, the analysis of

Algorithm 1: Smooth Split (`ssplit`)

```
Input: A smooth quadtree $T$ and a leaf $v \in T$ to split.
Output: The minimal smooth refinement $T'$ of $T$ such that $v$ is split.
split($v$)
foreach $v' \in$ principal_neighbors($v$) \ siblings($v$) do
    if depth($v'$) < depth($v$) then
        ssplit($v'$)
    end
end
```

the amortized time complexity of smooth splitting is subtle.

1.1.2 Our Results

Let

$$m(n, D) := \max_{\sigma \text{ of length } n} \# \text{ of } \texttt{split} \text{ operations in } \sigma,$$

where σ ranges over all sequences of smooth splits of length n in an initially trivial D-dimensional smooth quadtree, and define the *asymptotic amortized cost* of a smooth split as

$$\mathrm{ss}(D) := \limsup_{n\to\infty} \frac{m(n, D)}{n}.$$

The primary contribution of this chapter is to show that $\mathrm{ss}(D)$ is upper bounded by a constant for any fixed dimension D, and in particular does not depend on n. Because each `split` operation requires at most $O(D \cdot 2^D)$-time (to initialize each of the $2D$ principal neighbor pointers of a node's 2^D children), this also implies a constant upper bound on the time complexity of a smooth split operation for any fixed D. We give a self-contained, simple proof of the 2-dimensional case in Section 1.2, and prove the result for arbitrary dimensions in Section 1.3. More formally, we show the following.

Theorem 1.1.1. *Starting from an initially trivial subdivision consisting of one D-dimensional box B_1 the total number of* **`split`** *operations performed in any sequence of smooth splits* $\texttt{ssplit}(B_1), \ldots, \texttt{ssplit}(B_n)$ *is at most* $2^D \cdot (D+1)! \cdot n$. *Therefore,* $ss(D) \le 2^D \cdot (D+1)!$.

Additionally, we give lower bounds motivating our data structure and analysis. In Section 1.2.4, we show that without smoothing we cannot achieve an amortized constant cost for both splits and neighbor queries simultaneously even in two dimensions. In Section 1.4 we prove a lower bound on $\mathrm{ss}(D)$, showing that the exponential dependence on D in Theorem 1.1.1 is unavoidable. More formally, we show the following.

Theorem 1.1.2. *Starting from an initially trivial subdivision consisting of one D-dimensional box B_1 there exists a sequence of $n + O_D(1)$ smooth splits*[2] *that requires $n \cdot (D+1) \cdot 2^D$ splits. Therefore, $ss(D) \ge (D+1) \cdot 2^D$.*

Combining the bounds in Theorem 1.1.1 and Theorem 1.1.2, we get that

$$2^D \cdot (D+1) \le \mathrm{ss}(D) \le 2^D \cdot (D+1)! \ . \qquad (1.1)$$

Besides being of theoretical interest, our smooth quadtree data structure is useful in applications, such as the work described in Chapter 2 on computing a general class of Voronoi diagrams using subdivision. We have implemented it as part of the Core Library [Cor].[3]

[2]The notation $O_D(1)$ means that $O(1)$ holds for any fixed D.

[3]The code is also available as a stand-alone package at `https://github.com/hbennett/SmoothQuadtree`.

1.1.3 Monolithic Smoothing

The following theorem is a well-known result, saying that a simple quadtree can be smoothed using $O(n)$ splits:

Fact 1.1.3 (Theorem 14.4 in [dBCvKO08], Theorem 3 in [Moo95])**.** *Let T be a simple quadtree with n nodes and of height h. Then the smooth version of T has $O(n)$ nodes and can be constructed in $O((h+1) \cdot n)$ time.*

Fact 1.1.3 gives a bound for "monolithic" tree smoothing, the operation that we call `smooth` in Table 1.1. It says that given an arbitrary quadtree we can smooth it all at once in $O((h+1) \cdot n)$ time using $O(n)$ splits. In this chapter we study "dynamic" tree smoothing in which we smooth the tree after each split, therefore maintaining smoothness as an invariant.

Intuitively any single `split` operation should not "unsmooth" a quadtree much, so only a few additional splits should be required to "resmooth" a tree afterward. To capture this intuition, we define a potential function which measures how smooth a quadtree, and prove that no splitting operation increases it by more than a small amount. This leads to Theorem 1.1.1.

Note that the worst-case linear bound in Fact 1.1.3 on the number of additional smoothing splits required after each split does not suffice to prove Theorem 1.1.1.

1.1.4 Related Work

In recent work Löffler et al. [LSS13] recognize that maintaining smoothness "could cause a linear 'cascade' of cells needing to be split." This cascading behavior – what we define formally in terms of *forcing chains* – is the focus of our analysis and main result.

A natural question asks whether there exists a *worst-case* $O(1)$ time algorithm for smooth splitting a box B. The most natural such algorithm would recursively check whether neighbors of a split box must themselves be split, as in Algorithm 1, but would only recurse to some fixed depth. However, a forcing chain may be arbitrarily long in general meaning that this approach does not work in our model.

We may generalize the notion of smoothness as follows: call two neighbors k-*smooth* if the boxes differ in height by at most k in the quadtree. In two dimensions this is equivalent to having at most 2^k neighbors in a given direction. We have used the term "smoothness" to denote 1-smoothness. A natural question asks whether the relaxed smoothness constraint induced by increasing k would lead to a worst-case $O(1)$ algorithm. In general, this does not help because a forcing chain may still be arbitrarily long.

However, Löffler et al. [LSS13] sketch an $O(1)$ worst-case algorithm for performing smooth splits in a related quadtree model. The most important distinction in their model comes from defining two types of quadtree nodes – *true* cells which would be present in any unsmoothed quadtree, and B-cells which are only present to ensure smoothness. Different smoothness invariants hold for these two types of cells – true cells are required to be 1-smooth with respect to their neighbors while B-cells are only required to be 2-smooth. The splitting operation is defined on true cells whose children are not true cells. If a true cell A has B-cells as children then `ssplit`(A) promotes the children of A to true cells.

The algorithm sketched in the paper omits details and a proof of correctness for several key points, such as the promotion of B-cells to true cells, however it appears to be correct. The model differs from ours in that it only allows splits on "true" nodes, maintains a weaker balance invariant, and requires more complicated algorithms. Our result, although requiring involved analysis, shows that smoothing is efficient using a simple algorithm and quadtree model.

Moore [Moo92, Moo95] proves that "monolithic" smoothing of arbitrary quadtrees requires $O(n)$ splits as given in Fact 1.1.3. Although this result seems to have been known earlier, Moore reproves this result in [Moo95] for basic quadtrees using a gadget called a "barrier", and then extends the result to generalizations of quadtrees including triangular quadtrees, higher degree quadtrees, and higher dimensional quadtrees.

In [dBRS12], de Berg et al. study *refinement* of compressed quadtrees. They consider a refinement T_1 of a quadtree T_0 to be an extension of T_0 in which all boxes that were in T_0 have $O(1)$ neighbors in T_1. This is a relaxation of the notion of smoothing both in terms of the precise number of neighbors that a box may have (which is simply assumed to be bounded, but not by a particular constant) and in the sense that boxes in T_1 need not be smooth with respect to each other. The authors prove that a refinement of a compressed quadtree may be performed in $O(n)$ time, where n is the size of the quadtree. This result has a similar flavor to the "monolithic" smoothing result described in Fact 1.1.3.

Amortized analysis of quadtree operations has appeared in previous work. Park and Mount [PM12] introduce the *splay quadtree*, in which they use amortized analysis to analyze the cost of a sequence of data accesses in a quadtree whose balance is dynamically updated using rotations in a similar manner to standard splay trees. Overmars and van Leeuwen [OvL82] analyze dynamic quadtrees, studying the amortized (what they call average-case) cost of insertions into quadtrees.

Recently Sheehy [She] proposed extending results in his previous work on optimal mesh sizes [She12] to prove the efficient smoothing results presented in this chapter. A reviewer of [BY14] proposed a similar proof strategy based on Ruppert's work on local feature size [Rup93]. Future work involves studying these continuous techniques, and determining whether the approach is both viable and leads to better bounds than those given by the combinatorial approach used in this chapter.

1.1.5 Open Questions

The most natural open question related to our work is whether one can improve our amortized $O_D(1)$-time bound for smooth splitting to a *worst-case* $O_D(1)$-time bound.

Because forcing chains can have length $\Omega(n)$ after n smooth splits, our algorithm requires $\Omega(n)$ splits in the worst case. (See the lower bound construction in Section 1.4). However, one can imagine making "preemptive splits" to avoid this problem.

Open Problem 1.1.4. *Is there a worst-case $O_D(1)$-time algorithm for smooth splitting in smooth quadtrees?*

Another basic question is whether our bounds on $\mathrm{ss}(D)$ can be improved.

Open Problem 1.1.5. *Improve the bounds on $ss(D)$ given in Equation* (1.1).

Sheehy [She] proposed extending results in his previous work on optimal mesh sizes [She12] to prove the efficient smoothing results presented in this chapter. A reviewer of [BY14] proposed a similar proof strategy based on Ruppert's work on local feature size [Rup93]. Future work involves studying these continuous techniques, and determining whether the approach is both viable and leads to better bounds than those given by the combinatorial approach that we use.

Open Problem 1.1.6. *Does the use of continuous techniques lead to a better upper bound on $ss(D)$?*

Finally, we ask whether our techniques work for proving amortized smoothing bounds for the refinement of other types of subdivisions. Moore [Moo95] considers triangular subdivisions, and Atalay and Mount [AM06] consider the cost of refining a simplicial mesh.

Open Problem 1.1.7. *Can we extend our techniques to prove amortized bounds on the cost of refining other types of subdivisions?*

Acknowledgments

We would like to thank Don Sheehy for helpful conversations at the Fall Workshop on Computational Geometry in 2013 and his subsequent outline of a strategy for attacking our problem using continuous techniques. We would also like to thank Joe Simons for answering questions about his co-authored paper [LSS13].

Finally, we would like to thank the anonymous reviewers of [BY14] and [BY17] for helpful references and comments. One reviewer was so thorough that we asked for permission to identify them by name. We especially thank the now non-anonymous reviewer Betul Atalay for her exceptionally careful review.

1.2 The 2-Dimensional Case

We start by giving a self-contained proof of Theorem 1.1.1 for the special case of 2-dimensional quadtrees that develops most of the essential ideas for the D-dimensional case. Namely, we prove the following:

Theorem 1.2.1 (2-dimensional case of Theorem 1.1.1). *Starting from an initially trivial subdivision consisting of one* 2*-dimensional box* B_1*, the total cost of any sequence of smooth splits* $\texttt{ssplit}(B_1), \ldots, \texttt{ssplit}(B_n)$ *is* $O(n)$*. Therefore the amortized cost of a smooth split is* $O(1)$*.*

1.2.1 Definitions

Suppose that a box B is adjacent to a box B' and $\texttt{depth}(B) > \texttt{depth}(B')$. In that case, we say that B *forces* B' or $B \Longrightarrow B'$. The forcing terminology comes from our main application, the analysis of smoothing: suppose B, B' belongs to a subdivision S. If we split B, then we are forced to split B' and possibly other boxes in order to smooth the resulting subdivision. More precisely, let $\texttt{depth}(B) - \texttt{depth}(B') = k \geq 1$. Then we must split B' and recursively split exactly $k - 1$ proper descendants of B' in order to maintain smoothness in S. Of course if S was originally smooth, then no child of B' needs to be further split. We will mostly deal with the case where S is originally smooth and in this case we always have $k = 1$.

A *forcing chain* $B_1 \Longrightarrow B_2 \Longrightarrow \cdots \Longrightarrow B_m$ is a sequence of boxes $B_1, \ldots, B_m$ such that $B_i \Longrightarrow B_{i+1}$ for every $i \in [m-1]$. [4] Call B_1 the *head* of this chain. A forcing chain is *maximal* if it cannot be extended to a longer chain. Let the *forcing graph* $F(B)$ be the directed acyclic graph rooted at B, whose maximal paths are all the maximal chains beginning at B. In other words, the boxes in $F(B)$ are exactly those that would be split as part of the operation $\texttt{ssplit}(B)$.

We write $B \overset{d}{\Longrightarrow} B'$ (resp. $B' \overset{d}{\Longrightarrow} B'$) and say that B' is *d-forced* (resp. *d-forcing*) if $B \Longrightarrow B'$ and B' is a d-thern neighbor of B.[5] Here a direction d is specified by a

[4]Recall that the notation $[n]$ denotes the set of integers $\{1, \ldots, n\}$.

[5]This last notation derives from the cardinal directions such as "northern".

standard normal unit vector e_i or its negation $-e_i$.

We write $*\Longrightarrow B$ if there exists B' such that $B'\Longrightarrow B$, and similarly write $B\Longrightarrow *$ if there exists B' such that $B\Longrightarrow B'$. Lastly, we denote the parent of a box B as $p(B)$, and the kth ancestor of a box as $p^k(B)$.

1.2.2 Reasoning about Forcing Chains

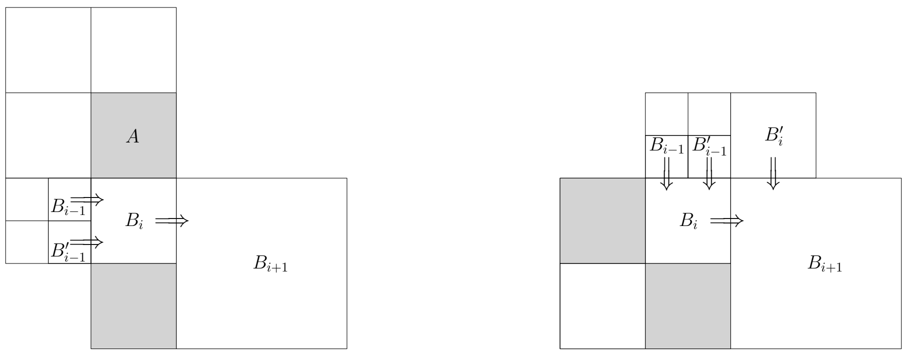
The following sequence of lemmas reasoning about forcing chains leads to the proof of Theorem 1.2.1.

Lemma 1.2.2. *A box B_1 heads at most two non-trivial maximal chains.*

Proof. We get an immediate upper bound of 2 on the number of chains that can be headed by a box B_1 since a box will never force in the direction of an adjacent sibling of which every box has two. Furthermore, we show that $*\Longrightarrow B_i$ implies that there exists at most one box B_{i+1} such that $B_i\Longrightarrow B_{i+1}$. Since the head B_1 of a splitting chain B_i is the only box in a splitting chain which may not be forced itself, this will imply that there are at most two splitting chains caused by splitting a box B_1.

Clearly, if $*\overset{d}{\Longrightarrow}B_i$ then $B_i \overset{-d}{\not\Longrightarrow} *$. There are then 3 other directions B_i may force in. We consider two cases, as shown in Figure 1.3:

- Case I, $p^2(B_{i-1}) = p(B_i)$: A box in one of the remaining three directions is a sibling of B_i. A box in another direction, A, must exist and be split to at least the level of B_i because $p(A)$ is adjacent to B_{i-1} (or a sibling of B_{i-1} of the same size). These must both be split to at least the level of B_i, leaving a single possibility for B_{i+1}.



Case I Case II

Figure 1.3: Two cases for the forcing relationships between quadtree boxes: Case I, $p^2(B_{i-1}) = p(B_i)$, and Case II, $p^2(B_{i-1}) \neq p(B_i)$. The arrows denote forcing relationships between boxes. Principal neighbors of B_i other than $p(B_{i-1})$ which must be split to at least the level of B_i are colored gray.

- Case II, $p^2(B_{i-1}) \neq p(B_i)$: Boxes in two of the possible three remaining directions are siblings of B_i. These must both be split to at least the level of B_i, leaving a single possibility for B_{i+1}.

□

Lemma 1.2.3. *Assume B_1, B_2 are boxes in a smooth quadtree, and that $* \overset{d}{\Longrightarrow} B_1 \Longrightarrow B_2$ for some d. Then $* \overset{d}{\Longrightarrow} B_2$.*

Proof. We again refer to Figure 1.3, and evaluate each case separately:

- Case I, $p^2(B_{i-1}) = p(B_i)$: Here $B_{i-1} \overset{d}{\Longrightarrow} B_i \overset{d}{\Longrightarrow} B_{i+1}$ so the claim trivially holds.

- Case II, $p^2(B_{i-1}) \neq p(B_i)$: We have assumed that $B_{i-1} \overset{d}{\Longrightarrow} B_i \overset{d'}{\Longrightarrow} B_{i+1}$ where $d \neq d'$. In this case, either B_{i-1} or its d'-thern sibling must have B'_i as its d'-thern neighbor. However B'_i must be a $(-d)$-thern neighbor of B_{i+1}, but of greater depth. Therefore $B'_i \overset{d}{\Longrightarrow} B_{i+1}$ and the claim holds.

□

By transitivity we conclude:

Corollary 1.2.4. *If $B_1 \overset{d}{\Longrightarrow} B_2 \Longrightarrow \cdots \Longrightarrow B_n$ then B_i is d-forced for $i \geq 2$.*

The following additional corollary says that a forcing chain may go in at most two directions:

Corollary 1.2.5. *Given a forcing chain $B_1 \overset{d_1}{\Longrightarrow} B_2 \overset{d_2}{\Longrightarrow} \cdots \overset{d_{n-1}}{\Longrightarrow} B_n$, we have that $|\{d_i : i \in [n-1]\}| \leq 2$.*

Proof. Clearly, if $* \overset{d}{\Longrightarrow} B$ then $B \overset{-d}{\not\Longrightarrow} B$, and it follows that a box may force in at most two directions. However, Lemma 1.2.3 shows that $* \overset{d}{\Longrightarrow} B_i \Longrightarrow B_{i+1}$ implies that $* \overset{d}{\Longrightarrow} B_{i+1}$, meaning that a box in a forcing chain is always forced in all of the same directions as its predecessors. Therefore, if B_i is forced in two directions then for all $j > i$, B_j is also forced in the same two directions, and cannot force in any additional directions.

□

Lemma 1.2.6. *If for some boxes B_1, B_2, B_3 we have $B_1 \overset{d}{\Longrightarrow} B_2 \overset{d}{\Longrightarrow} B_3$ then B_2 has a split sibling.*

Proof. Figure 1.4 shows the idea behind Lemma 1.2.6. Because $B_2 \overset{d}{\Longrightarrow} B_3$ we have that B_2 is a d-thern child of its parent, meaning that its $(-d)$-thern neighbor of the same size is also its sibling.

Furthermore, because $B_1 \overset{d}{\Longrightarrow} B_2$, we have that B_1 is a $(-d)$-thern neighbor of B_2. Because B_1 has side length exactly half that of B_2, it follows that $p(B_1)$ and B_2 are siblings. Finally, because $p(B_1)$ has B_1 as a child it is split. □

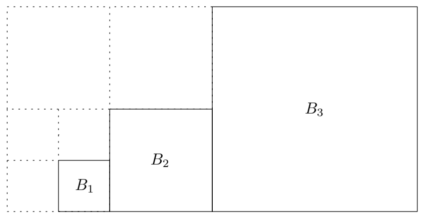


Figure 1.4: A two-link forcing chain $B_1 \overset{d}{\Longrightarrow} B_2 \overset{d}{\Longrightarrow} B_3$ implies that B_2 has a split sibling. In particular, the dotted boxes must exist, and therefore the parent of B_1 must be split and a sibling of B_2.

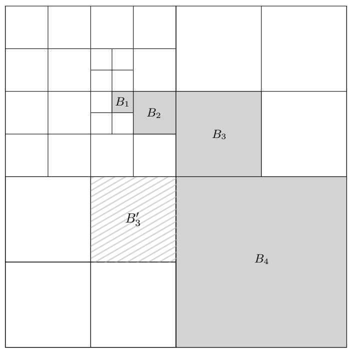


Figure 1.5: A forcing chain $B_1 \overset{d}{\Longrightarrow} B_2 \overset{d}{\Longrightarrow} B_3 \overset{d'}{\Longrightarrow} B_4$ of four nodes illustrating Lemma 1.2.7. Note that B_1 and B_3 have no split siblings, and B_4 may also be the northwest child of its parent, and therefore also may not have any split siblings. Box B_2, on the other hand, satisfies Lemma 1.2.6. Furthermore, B_4 is d-forced although not by B_3.

Lemma 1.2.7 (Main Lemma)**.** *At most three nodes in a forcing chain* $B_1 \overset{d_1}{\Longrightarrow} B_2 \overset{d_2}{\Longrightarrow} \cdots \overset{d_{m-1}}{\Longrightarrow} B_m$ *have no split siblings.*

Proof. We combine Corollaries 1.2.4 and 1.2.5 with Lemma 1.2.6 to prove the Main Lemma. Assume without loss of generality that there exists a minimum index i such that $d_i \neq d_1$. We show that each of the boxes B_1, B_i, and B_m may not have a split sibling and that all other boxes in the forcing chain do. (If $d_i = d_1$ for all $i \in [m-1]$, then we show that only B_1 and B_m may not have a split sibling)

If $B_{j-1} \overset{d}{\Longrightarrow} B_j \overset{d}{\Longrightarrow} B_{j+1}$ then B_j has a split sibling by Lemma 1.2.6. Box B_1 need not be forced from any direction, and B_m need not force in any direction, so Lemma 1.2.6 does not apply. Furthermore, $* \overset{d_1}{\Longrightarrow} B_i$, but $B_i \overset{d_1}{\not\Longrightarrow} *$, so again Lemma 1.2.6 does not apply.

To see that all other boxes must have split siblings we consider two cases:

(i) Case $1 < j < i$: We have that $B_{j-1} \overset{d_1}{\Longrightarrow} B_j \overset{d_1}{\Longrightarrow} B_{j+1}$ by assumption that $d_j = d_1$ for all $j < i$. Therefore Lemma 1.2.6 applies to B_j.

(ii) Case $i < j < m$: We have that $B_j \overset{d_j}{\Longrightarrow} B_{j+1}$ where $d_j \in \{d_1, d_i\}$ since by Corollary 1.2.5 a forcing chain may go in at most two directions. Furthermore, by Corollary 1.2.4, $* \overset{d_1}{\Longrightarrow} B_j$ and $* \overset{d_i}{\Longrightarrow} B_j$ meaning that either $* \overset{d_1}{\Longrightarrow} B_j \overset{d_1}{\Longrightarrow} B_{j+1}$ or $* \overset{d_i}{\Longrightarrow} B_j \overset{d_i}{\Longrightarrow} B_{j+1}$. In either case Lemma 1.2.6 applies to B_j.

□

1.2.2.1 Potential Function

Using the characterization of boxes in a forcing chain given in Lemma 1.2.7, we define the following potential function for a node $v \in T$:

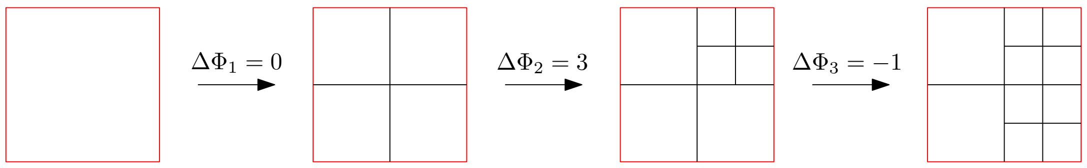


Figure 1.6: Example of the three cases presented in Equation 1.4. We consider the change each split has on $\Phi(v)$, where v corresponds to the outer red box in each case.

$$\Phi(v) := \begin{cases} 0 & \text{if no children of } v \text{ have been split,} \\ \#\text{ of unsplit children of } v & \text{otherwise.} \end{cases} \tag{1.2}$$

We also extend this definition to give a potential function for the quadtree:

$$\Phi(T) := \sum_{v \in T} \Phi(v). \tag{1.3}$$

We note that $\Phi(v) = 0$ if either all or none of the children of v are split. Furthermore, if v is itself a leaf then $\Phi(v) = 0$ vacuously. It follows that only parents of leaf nodes have non-zero contribution to the potential $\Phi(T)$. Furthermore, splitting a node changes the potential of at most one node (its parent).

Let T be a quadtree, and T' be the quadtree resulting from splitting a leaf v. Splitting v does not change the potential of v, but changes the potential of the parent $p(v)$ of v by either 3 if $p(v)$ had no split children or -1 if $p(v)$ had other split children. A leaf v always has a parent except when v is the root of the tree. We then get the following:

$$\Delta\Phi = \Phi(T') - \Phi(T) = \begin{cases} 0 & \text{If } v \text{ is the root of } T, \\ 3 & \text{If } v \text{ has no split siblings,} \\ -1 & \text{If } v \text{ has a split sibling.} \end{cases} \tag{1.4}$$

Because the first case only occurs on the first split, in which case only a single box splits and $\Delta\Phi = 0$, it suffices to consider the last two cases for our analysis.

1.2.3 Upper Bound

We now give the proof of Theorem 1.2.1 using the Main Lemma.

Proof of Theorem 1.2.1. We set the cost of a single split operation $\texttt{split}(B_j)$ to be $\text{cost}_j = 1$. To prove a constant amortization bound, we need to show that for each smooth split operation $\texttt{ssplit}(B_i)$ there exists $\text{charge}_i = O(1)$ such that

$$\text{charge}_i \geq \sum_{j:B_j \in F(B_i)} (\text{cost}_j + \Delta\Phi_j),$$

where $\Delta\Phi_j$ denotes the quadtree's change in potential from executing $\texttt{split}(B_j)$. By Equation (1.4) we have

$$\text{cost}_j + \Delta\Phi_j = \begin{cases} 4 & \text{if } B_j \text{ has no split siblings,} \\ 0 & \text{if } B_j \text{ has a split sibling.} \end{cases} \tag{1.5}$$

By Lemma 1.2.7 at most three boxes per forcing chain have no split siblings. Furthermore, by Lemma 1.2.2 a box B_0 heads at most two forcing chains. Combining these observations with Equation (1.5) shows that it suffices to set $\text{charge}_i = 4 \cdot 3 \cdot 2 = 24$. □

We are interested in precise upper and lower bounds on $\text{ss}(D)$, especially for small D (say, $D \leq 3$). We first remark that it suffices to set $\text{charge}_i = 20$ rather than 24 in the preceding proof. This is because we charged separately for the head of each of the two possible chains, but actually B_1 is the head of both. In Theorem 1.1.2, we give general bounds which imply an asymptotic amortized cost of at least 12 in the 2-dimensional case. Putting these two bounds together, we get that $12 \leq \text{ss}(2) \leq 20$. As perhaps the most interesting special case of Open Problem 1.1.5, we ask what the right value of ss(2) is.

1.2.4 A Lower Bound for `PU`-Quadtrees

The motivation for studying the quadtree model presented in this chapter comes from the ineffectiveness of other natural models to support both efficient `neighbor_query` and `split` operations. We next analyze what happens if we use our model but without smoothing.

Suppose that we maintain principal neighbor pointers in an unsmoothed subdivision, i.e., the `PU` quadtree model in Table 1.1. The following lemma gives an amortized $\Omega(\log n)$ lower bound on the time complexity of a split in this model, based on the high number of neighbor pointer updates required:

Lemma 1.2.8. *Let B_1 denote the root box of a 2-dimensional* ***PU*** *quadtree. Then, in the worst case, a sequence of n splits* ***split***$(B_1), \ldots,$ ***split***(B_n) *followed by a* ***smooth*** *operation requires $\Omega(n \log n)$ time.*

Proof. We refer to the setup shown in Figure 1.7, where the boxes are subdivided on the left in the first stage, and then subdivided on the right in the second stage. The boxes on the boundary of the halves are split to depth $k+1$ on the left, and depth k on the right. The splits performed in the second stage are exactly those

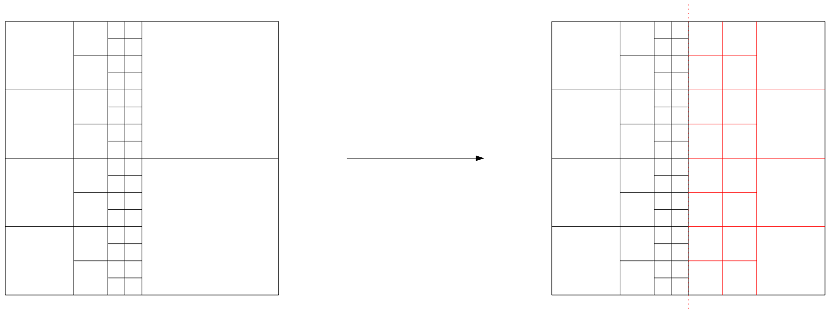


Figure 1.7: A sequence of splits leading to an unsmooth `PU` subdivision (left) and a sequence of smoothing splits (right) that requires amortized $\log n$ pointer updates between boxes on opposite sides of the dotted center line per split.

needed to smooth the quadtree after the splits in the first stage.

After an initial split of the rootbox, the first stage requires $\sum_{i=1}^{k} 2^i = 2^{k+1} - 2$ additional splits and the second stage requires $2^k - 2$. The total number of splits is therefore $n = 1 + (2^{k+1} - 2) + (2^k - 2) = \Theta(2^k)$.

For the lower bound we consider only updates to the principal neighbor pointers of boxes on the left half which point to boxes on the right half (across the vertical center line) in the second splitting phase. We must update 2^{k-i} such pointers for each of the 2^i boxes of depth i that we split in the second phase. We therefore must update $\sum_{i=1}^{k-1} 2^i 2^{k-i-1} = (k-1) \cdot 2^{k-1} = \Theta(n \log n)$ pointers.

Because the splits performed in the second stage were exactly those required to smooth the quadtree after the first stage, this proves both the amortized bound for `split` operations and the worst-case bound for the `smooth` operation. □

1.3 The Higher Dimensional Case

We next prove Theorem 1.1.1 in higher dimensions. To do this we will need to develop some additional notation and concepts. As in the 2-dimensional case, the idea behind the proof is to analyze what conditions lead to smooth splits propagating through the data structure, and to show that a suitably defined cost-potential invariant is only violated a bounded number of times per smooth split.

In Section 1.3.1 we introduce terminology related to our proofs. Next, in Section 1.3.2 we prove results reasoning about forcing chains of length two. These are very similar to those given in Section 1.2.2 for the 2-dimensional case, but formalized differently. After that, in Section 1.3.3 we introduce the key new idea for the higher dimensional case. Namely, we show that the number of direction in which a box is forced increases along any path in $F(B)$, which allows us to conclude that the number of directions in which it forces decreases. Finally, in Section 1.3.4 we use the tools we have developed and the same potential function as in the 2-dimensional case to prove Theorem 1.1.1.

1.3.1 Notation for the higher-dimensional case

We consider a (higher-dimensional) quadtree which forms a subdivision of the D-dimensional hypercube $[-1,1]^D$ for $D \geq 1$. If boxes B and B' are neighbors, there is a unique direction such that B' is *adjacent to B in direction d*, which we denote by $B \xrightarrow{d} B'$. Clearly, $B \xrightarrow{d} B'$ if and only if $B' \xrightarrow{-d} B$. We simply write $B \longrightarrow B'$ to indicate that there exists some d such that $B \xrightarrow{d} B'$.

Let $p(B)$ denote the *parent* of box B (this is well-defined except when B is the root), and let $p^n(B)$ denote the nth ancestor of B for any $n \geq 0$. Additionally, we write $B \prec B'$ if B is a child of B'.

We define the *(co-)projection* of a box $B = I_1 \times \cdots \times I_D$ with respect to index $i \in [D]$ as follows.

- (Projection) $\mathtt{Proj}_i(B) := \prod_{j=1, j\neq i}^{D} I_j$.
- (Co-Projection) $\mathtt{Coproj}_i(B) := I_i$.

Note that $\mathtt{Proj}_i(B)$ is $(D-1)$-dimensional, while $\mathtt{Coproj}_i(B)$ is 1-dimensional. We define the *indexed Cartesian product* $\otimes_i$ so that any box B can be recovered from its corresponding projection and co-projection:

$$B = \underset{i}{\mathtt{Coproj}}(B) \otimes_i \underset{i}{\mathtt{Proj}}(B). \tag{1.6}$$

As a convention, if d is a direction then we may write $\mathtt{Proj}_d(B)$ (resp. $\mathtt{Coproj}_d(B)$) instead of $\mathtt{Proj}_i(B)$ (resp. $\mathtt{Coproj}_i(B)$). Note that projecting (resp. co-projecting) the set of boxes in an aligned subdivision induces a new subdivision of dimension $D-1$ (resp. dimension 1).

1.3.1.1 Forcing Chains

Recall that a sequence of forcing relations

$$C : B_0 \overset{d_1}{\Longrightarrow} B_1 \overset{d_2}{\Longrightarrow} \cdots \overset{d_k}{\Longrightarrow} B_k \tag{1.7}$$

is called a *forcing chain.* The set $\{d_1, \ldots, d_k\}$ are the directions of C; we say C is *monotone* if its direction set does not contain any pair of opposite directions.

The following lemma follows from the definition of forcing.

Lemma 1.3.1. *The forcing relationship* $B \overset{d}{\Longrightarrow} B'$ *is equivalent to the following two conditions:*

(i) $\texttt{Proj}_d(B) \prec \texttt{Proj}_d(B')$,

(ii) $\texttt{Coproj}_d(B) \Longrightarrow \texttt{Coproj}_d(B')$.

Note that conditions (i) and (ii) refer to child and forcing relationships in dimensions $D-1$ and 1, respectively.

1.3.2 Analysis of Two Link Chains

In this part, we consider chains with two links, i.e., chains of the form $B \overset{d}{\Longrightarrow} B' \overset{d'}{\Longrightarrow} B''$. Our analysis consists of analyzing the cases $d = d'$ and $d \neq d'$. The first case already arises in one dimension.

Lemma 1.3.2 (One Direction)**.** *Suppose* $I \Longrightarrow I' \Longrightarrow I''$ *holds for intervals in a smooth subdivision. Then* $p^2(I) = p(I')$.

We omit the easy proof, which is shown in Figure 1.3, Case I. Note that $p^2(B) = p(B')$ means that $p(B)$ and B' are siblings.

We show that this works in higher dimensions as well, but we now need an additional condition. When $D = 1$, the fact that $I \Longrightarrow I' \Longrightarrow I''$ implies that there is a direction d such that $I \overset{d}{\Longrightarrow} I' \overset{d}{\Longrightarrow} I''$. In higher dimensions, we must explicitly specify this requirement. Figure 1.3, Case I illustrates two cases in $D = 2$.

Theorem 1.3.3 (One Direction)**.** *Suppose* $B \overset{d}{\Longrightarrow} B' \overset{d}{\Longrightarrow} B''$ *holds for boxes in a smooth subdivision. Then* $p^2(B) = p(B')$.

Proof. Without loss of generality, assume that $d = e_1$. Then

$$B = I \times E, \qquad B' = I' \times E', \qquad B'' = I'' \times E'',$$

where $I \Longrightarrow I' \Longrightarrow I''$ and $E \prec E' \prec E''$ by Lemma 1.3.1. This implies that $p(E) = E'$ or

$$p^2(E) = p(E') = E''. \tag{1.8}$$

By Lemma 1.3.2, we conclude that

$$p^2(I) = p(I'). \tag{1.9}$$

But Equations (1.8) and (1.9) together imply that $p^2(I \times E) = p(I' \times E')$, which is what our theorem claims. $\square$

The second phenomenon arises for $D \geq 2$ for forcing chains of the form $B \overset{d}{\Longrightarrow} B' \overset{d'}{\Longrightarrow} B''$ where $d \neq d'$.

Lemma 1.3.4 (Two Directions)**.** *Let B, B', B'' be boxes in a smooth subdivision of $[-1,1]^2$, and suppose that $B \overset{d}{\Longrightarrow} B' \overset{d'}{\Longrightarrow} B''$ for some $d \neq d'$. Then $p^2(B) \neq p(B')$.*

We omit the elementary proof, which is illustrated in Figure 1.3, Case II. We next extend this result to higher dimensions.

Theorem 1.3.5 (Two Directions)**.** *Consider boxes in a smooth subdivision of $[-1,1]^D$ ($D \geq 2$). Suppose $B \overset{d}{\Longrightarrow} B' \overset{d'}{\Longrightarrow} B''$ holds where $d \neq d'$. Then $p^2(B) \neq p(B')$.*

Proof. We must have that $d \neq \pm d'$, so without loss of generality assume that $d = e_1$ and $d' = e_2$. We can then write

$$\begin{aligned} B &= I \times J \times E, \\ B' &= I' \times J' \times E', \\ B'' &= I'' \times J'' \times E'', \end{aligned}$$

for some intervals I, I', I'', J, J', J'' and $(D-2)$-dimensional boxes E, E', E''. From the premise $B \overset{d}{\Longrightarrow} B' \overset{d'}{\Longrightarrow} B''$, we conclude that

$$\begin{array}{ccccc} I & \overset{d}{\Longrightarrow} & I' & \prec & I'', \\ J & \prec & J' & \overset{d'}{\Longrightarrow} & J'', \\ E & \prec & E' & \prec & E''. \end{array}$$

Therefore

$$(I \times J) \quad \overset{d}{\Longrightarrow} \quad (I' \times J') \quad \overset{d'}{\Longrightarrow} \quad (I'' \times J''),$$

and therefore Lemma 1.3.4 implies that $p^2(I \times J) \neq p(I' \times J')$. This implies that $p^2(B) \neq p(B')$. $\square$

The next result is a kind of commutative diagram argument whose proof depends on Theorem 1.3.5. We first give the result in two dimensions (see Figure 1.8).

Lemma 1.3.6 (Commutative Diagram). *Let B, B', and B'' be boxes in a smooth subdivision of $[-1,1]^2$. Suppose $B \overset{d}{\Longrightarrow} B' \overset{d'}{\Longrightarrow} B''$ holds for some $d \neq d'$. Then there exists a box A' such that $A' \overset{d}{\Longrightarrow} B''$.*

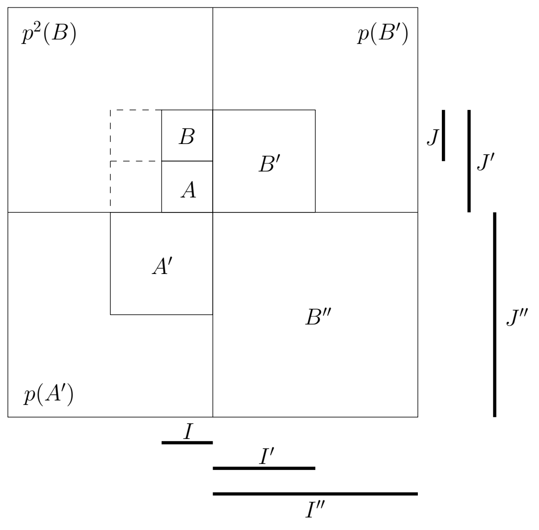


Figure 1.8: A commutative diagram for forcing.

Proof. Let

$$B = I \times J$$
$$B' = I' \times J'$$
$$B'' = I'' \times J'',$$

as illustrated by Figure 1.8. Without loss of generality, let $d = (1,0)$ and $d' = (0,1)$ so that

$$\begin{aligned} I &\implies I' \prec I'', \\ J &\prec J' \implies J''. \end{aligned}$$

By Lemma 1.3.4, $p^2(B) \neq p(B')$. And since $B \overset{d}{\Longrightarrow} B'$, $B \subseteq p^2(B)$ and $B' \subseteq p(B')$, we conclude $p^2(B) \xrightarrow{d} p(B')$. Likewise, $B' \overset{d'}{\Longrightarrow} B''$ implies $p(B') \xrightarrow{d'} B''$. Summarizing, we have shown that

$$p^2(B) \xrightarrow{d} p(B') \xrightarrow{d'} B''. \tag{1.10}$$

Since $p^2(B)$, $p(B')$ and B'' are all at the same depth, Equation (1.10) implies

$$\begin{array}{ccccc} p^2(I) & \longrightarrow & p(I') & = & I'', \\ p^2(J) & = & p(J') & \longrightarrow & J''. \end{array}$$

By an application of Equation (1.6), there is an aligned box $p(A') = p^2(I) \times J''$ at the depth of B'' that completes Equation (1.10) into the following commutative diagram:

$$\begin{tikzcd} p^2(B) \arrow[r, "d"] \arrow[d, "d'"'] & p(B') \arrow[d, "d'"] \\ p(A') \arrow[r, "d"'] & B'' \end{tikzcd} \tag{1.11}$$

As illustrated in Figure 1.8, the commutative diagram involves four adjacent boxes at the same depth. From Equation (1.11), we see that there is a box A in the subdivision with $p(A) = p(B)$ and $A \overset{d}{\Longrightarrow} B'$, $\qquad A \xrightarrow{d'} p(A')$. This last relationship would violate smoothness if $p(A')$ belongs to our subdivision, since $\text{depth}(p(A')) - \text{depth}(A) = 2$. Hence there is a child A' of A' such that $A \overset{d'}{\Longrightarrow} A' \overset{d}{\Longrightarrow} B''$. Moreover, A' must belong to the subdivision because otherwise, if it split, it would have a child $C \overset{d}{\Longrightarrow} B''$, which would violate smoothness. We thus have the following commutative (forcing) diagram which establishes our lemma:

$$\begin{tikzcd} A \arrow[r, Rightarrow, "d"] \arrow[d, Rightarrow, "d'"'] & B' \arrow[d, Rightarrow, "d'"] \\ A' \arrow[r, Rightarrow, "d"'] & B'' \end{tikzcd} \tag{1.12}$$

□

The previous lemma is best understood in terms of a commutative diagram as shown in Figure 1.8. It says that there exists some A where $p(A) = p(B)$ and some A' such that $A \overset{d}{\Longrightarrow} B' \overset{d'}{\Longrightarrow} B''$ and $A \overset{d'}{\Longrightarrow} A' \overset{d}{\Longrightarrow} B''$. The lemma also holds in higher dimensions, as stated in the following theorem. Intuitively this is because we can project the higher dimensional subdivision into the plane spanned by directions d, d' and then apply the lemma.

Theorem 1.3.7 (Commutative Diagram)**.** *Consider boxes in a smooth subdivision of $[-1,1]^D$ for $D \geq 2$. Suppose $B \overset{d}{\Longrightarrow} B' \overset{d'}{\Longrightarrow} B''$ holds for some $d \neq d'$. Then there exists a box A' in the subdivision such that $A' \overset{d}{\Longrightarrow} B''$.*

Proof. To construct A', let us assume without loss of generality that $d = e_1$ and $d' = e_2$. We can thus write

$$\begin{aligned} B =& I \times J \times E, \\ B' =& I' \times J' \times E', \\ B'' =& I'' \times J'' \times E'', \end{aligned}$$

where the I's and J's are intervals. From the premise $B \overset{1}{\Longrightarrow} B' \overset{2}{\Longrightarrow} B''$, we conclude that

$$\begin{array}{ccccc} I & \Longrightarrow & I' & \prec & I'', \\ J & \prec & J' & \Longrightarrow & J'', \\ E & \prec & E' & \prec & E''. \end{array}$$

Therefore,

$$I \times J \quad \overset{d}{\Longrightarrow} \quad I' \times J' \quad \overset{d'}{\Longrightarrow} \quad I'' \times J'',$$

and by Lemma 1.3.6, there exists $\widehat{A}$ such that $\widehat{A} \overset{d}{\Longrightarrow} I'' \times J''$. Therefore, $\widehat{A} \times$

$E' \overset{d}{\Longrightarrow} I'' \times J'' \times E''$. Our theorem follows by choosing $A' = \widehat{A} \times E'$. $\square$

1.3.3 Monotonicity of Forcing Chains

Theorem 1.3.7 motivates the following notions about forcing. Recall that if there exists A such that $A \overset{d}{\Longrightarrow} B$ then we say B is *d-forced*, and if there exists A such that $B \overset{d}{\Longrightarrow} A$ then we say that B is *d-forcing*.

Let $R(B)$ denote the set of directions d such that B is d-forced, and let $r(B) = |R(B)|$ be its cardinality. Note that $0 \le r(B) \le 2D$. Similarly, let $S(B)$ denote the set of directions d in which B is d-forcing, and let $s(B) = |S(B)|$. Note that $0 \le s(B) \le D$. Furthermore, note that $R(B) \cap -S(B) = \emptyset$ holds because $B \overset{d}{\Longrightarrow} B'$ implies that $B \overset{-d}{\not\Longrightarrow} B'$.

The following result is a direct consequence of Theorem 1.3.7.

Corollary 1.3.8. *For boxes in a smooth subdivision, $B \Longrightarrow B'$ implies $R(B) \subseteq R(B')$ and hence $r(B) \le r(B')$.*

In a general subdivision, we could have non-monotone chains (i.e., a chain whose directions include both d and $-d$ for some d). However, we show next that smoothness implies monotone chains.

Lemma 1.3.9. *Chains in a smooth subdivision are monotone.*

Proof. Consider any chain as in Equation (1.7). It follows from Corollary 1.3.8 that $\{d_1, \ldots, d_i\} \subseteq R(B_i)$ for each i. It suffices to note that $-d_{i+1} \notin R(B_i)$ and $d_{i+1} \in S(B_i)$. Indeed, because $R(B) \cap -S(B) = \emptyset$, this shows that $-d_{i+1} \notin R(B_i)$. $\square$

If $A \Longrightarrow B$ and $p^2(A) = p(B)$, then we call $p(A)$ a *split adjacent sibling* of B. The next lemma upper bounds $s(B)$ when B has split adjacent siblings.

Lemma 1.3.10. *Let B be a box in a smooth subdivision. Then:*

(i) If B has exactly one split adjacent sibling, then $s(B) \leq 1$.

(ii) If B has at least two split adjacent siblings, then $s(B) = 0$.

Proof. We prove each case. Case (i): by assumption there is a direction d and box A such that $A \overset{d}{\Longrightarrow} B$ and $p^2(A) = p(B)$. Assume for contradiction that $s(B) \geq 2$. Then there is some $d' \neq d$ and B' such that $A \overset{d}{\Longrightarrow} B \overset{d'}{\Longrightarrow} B'$. But then by Theorem 1.3.5, $p^2(A) \neq p(B)$, which is a contradiction.
Case (ii): By assumption, there are two directions $d \neq d'$ and boxes A, A' such that $A \overset{d}{\Longrightarrow} B$ and $A' \overset{d'}{\Longrightarrow} B$, and $p^2(A) = p^2(A') = p(B)$. Assume for contradiction that $s(B) > 0$. Then there exists B' such that $B \overset{d''}{\Longrightarrow} B'$ for some d''. So $d'' \neq d$ or $d'' \neq d'$. Without loss of generality, suppose $d'' \neq d$. Since $A \overset{d}{\Longrightarrow} B \overset{d''}{\Longrightarrow} B'$, Theorem 1.3.5 implies that $p^2(A) \neq p(B)$, contradiction. ☐

The next result shows that $r(B)$ must increase whenever B can force in more than one direction.

Lemma 1.3.11. *Let $B \Longrightarrow B'$ in a smooth subdivision. If $s(B) > 1$ then $r(B') > r(B)$.*

Proof. Since $s(B) > 1$, there are two directions d, d' such that $B \overset{d}{\Longrightarrow} *$ and $B \overset{d'}{\Longrightarrow} *$. Without loss of generality, let $B \overset{d}{\Longrightarrow} B'$ and $B \overset{d'}{\Longrightarrow} A'$ for some A' in the subdivision. We already know that $r(B) \leq r(B')$. Clearly, $d \in R(B')$. So the inequality $r(B) < r(B')$ follows if we show that $d \notin R(B)$. By way of contradiction, assume $d \in R(B)$. So there exists a box A in the subdivision such that $A \overset{d}{\Longrightarrow} B \overset{d}{\Longrightarrow} B'$. By Theorem 1.3.3, $p^2(A) = p(B)$. However, we also have $A \overset{d}{\Longrightarrow} B \overset{d'}{\Longrightarrow} A'$. By Theorem 1.3.5, $p^2(A) \neq p(B)$. This is our contradiction. ☐

The next lemma shows that high $r(B)$ implies low $s(B)$.

Lemma 1.3.12. *For any non-root box B,*

$$s(B) \leq \begin{cases} 0 & \text{if } r(B) > D, \quad \text{(Case (i))} \\ 1 & \text{if } r(B) = D, \quad \text{(Case (ii))} \\ D - r(B) & \text{if } r(B) < D. \quad \text{(Case (iii))} \end{cases} \tag{1.13}$$

Proof. Since B is not the root it has D siblings $A_1, \ldots, A_D$ with corresponding, distinct directions $d_1, \ldots, d_D$ such that $A_i \xrightarrow{d_i} B$. Let $N(B) = \{d_1, \ldots, d_D\}$ and let $-N(B) = \{-d_1, \ldots, -d_D\}$. Note that $S(B) \subseteq N(B)$ and recall that $R(B) \cap -S(B) = \emptyset$. Note that $|R(B) \cap N(B)|$ indicates the number of split adjacent siblings of B. We consider each case in Equation (1.13).

(i) $r(B) > D$. There are two possibilities: if $|R(B) \cap N(B)| > 1$ then Lemma 1.3.10 implies that $s(B) = 0$, as desired. Otherwise by Lemma 1.3.10, $|R(B) \cap N(B)| = 1$. This means that $r(B) = D + 1$ and $-N(B) \subseteq R(B)$. In other words, B is forced by D non-sibling-neighbors. This implies that $s(B) = 0$.

(ii) $r(B) = D$. If $|R(B) \cap N(B)| \geq 1$, then Lemma 1.3.10 implies that $s(B) \leq 1$, as desired. Otherwise $R(B) = -N(B)$ and $s(B) = 0$ as in case (i).

(iii) $r(B) < D$. If $|R(B) \cap N(B)| \geq 1$, then Lemma 1.3.10 implies that $s(B) \leq 1$ as in case (ii). Otherwise $R(B) \cap N(B) = \emptyset$ so $R(B) \subseteq -N(B)$. Since $S(B) \subseteq (-N(B)) \setminus R(B)$, we conclude that $s(B) \leq D - r(B)$, as desired.

□

Let $B \in T$. Recall that the forcing graph $F(B)$ of B is the directed acyclic graph rooted at B, whose maximal paths are all the maximal chains beginning at B. The smooth split of B amounts to splitting every node in $F(B)$. Each node B' in $F(B)$ has $s(B')$ children, so B' is a leaf (or sink) if and only if $s(B') = 0$. If $s(B') > 1$, we call B' a *branching node.* Note that $F(B)$ would be a tree rooted at B if all the maximal chains are disjoint except at B. However, in general, maximal chains can merge.

Using Lemmas 1.3.11 and 1.3.12 we get the following about $F(B)$.

Theorem 1.3.13. *Let B be a box in a smooth subdivision. There are at most $(D - r(B))!$ maximal paths in the forcing graph $F(B)$, where we define $x! = 1$ for $x \le 0$.*

Proof. Write r for $r(B)$. The result holds if there are no branching nodes, which in particular is true if $r \ge D - 1$ by Lemma 1.3.12. In these cases, $F(B)$ consists of a single path, and $(D - r)! = 1$.

So assume that $r \le D - 2$ and that there are branching nodes. Then there is a unique branching node $B' \in F(B)$ of minimum depth, so that B' has children $A_1, \ldots, A_s$ in $F(B)$, where $s = s(B')$.

By Lemma 1.3.11, $r(A_i) \ge r(B') + 1 \ge r + 1$, and therefore by Lemma 1.3.12, $s(A_i) \le D - r(A_i) \le D - r - 1$ for every $i \le i \le s$. Therefore by induction we conclude that there are at most $(D - r(B))!$ maximal paths in $F(B)$. □

1.3.4 Amortized Bounds for Smooth Splits

As in the 2-dimensional case, we now show how to use the analysis of forcing chains to obtain an upper bound on the amortized complexity of smooth splits.

Let T be a smooth quadtree. Define the potential $\Phi(T)$ of a quadtree T to be the sum of the potential $\Phi(B)$ of all the nodes $B \in T$, which we define to as

$$\Phi(B) := \begin{cases} 0 & \text{if } B \text{ has no split children,} \\ \#\text{ of unsplit children of } B & \text{otherwise.} \end{cases} \tag{1.14}$$

Note that $\Phi(B) = 0$ if and only if it has no split children or all its children are split. Otherwise, $1 \leq \Phi(B) \leq 2^D - 1$. Intuitively, each unit of potential pays for the cost of a single split. This naturally generalizes the potential function given in the 2-dimensional case.

For a leaf $B \in T$ let $c(B)$ denote the number of nodes B' in $F(B)$ such that $\Phi(p(B')) = 0$. $\Phi(p(B')) = 0$ if and only if $p(B')$ has no split children or all of its children is split. Since such a B' is a leaf in T, $\Phi(p(B')) = 0$ implies that B' has no split siblings. Thus, $c(B)$ is counting the number of nodes in $F(B)$ with no split siblings.

We are now ready to prove our main result.

Proof of Theorem 1.1.1. A smooth split of B amounts to splitting each node in its forcing graph $F(B)$. Recall that $c(B)$ is the number of nodes $B' \in F(B)$ with $\Phi(p(B')) = 0$. We will show that $c(B) \leq (D+1)!$.

By Theorem 1.3.13 we know that there are at most $D!$ maximal paths in $F(B)$. We then need to show that each maximal chain $B = B_0 \overset{d_1}{\Longrightarrow} B_1 \overset{d_2}{\Longrightarrow} \cdots \overset{d_k}{\Longrightarrow} B_k$ has at most $D+1$ indices $i \in [k]$ such that $\Phi(p(B_i)) = 0$. For such an i, we claim that $d_{i+1} \notin R(B_i)$ and $d_{i+1} \in R(B_{i+1})$, and therefore $r(B_i) < r(B_{i+1})$.

Suppose for contradiction that $d_{i+1} \in R(B_i)$. Because $B_i \overset{d_{i+1}}{\Longrightarrow} B_{i+1}$, there is an adjacent sibling A of B_i such that $A \overset{d_{i+1}}{\longrightarrow} B_i$. Therefore we must have $A' \overset{d_{i+1}}{\Longrightarrow} B_i$ for some child A' of A. But because $\Phi(p(B_i)) = 0$, A has not been split and so A' cannot exist. Therefore $r(B_{i+1}) > r(B_i)$.

It follows that if there are $\geq D+1$ such indices, the $(D+1)$-st index i has the property that $r(B_{i+1}) \geq D+1$. Then $s(B_{i+1}) = 0$ by Lemma 1.3.12. Hence B_{i+1} must be the last node B_k in the chain. It follows that $c(B) \leq (D+1)!$.

The smooth split of B amounts to splitting each box $B' \in F(B)$. There are two cases to consider for each such B':

(i) $\Phi(p(B')) > 0$. Then splitting B' can be charged to the corresponding unit decrease in potential $\Phi(T)$, since $\Phi(p(B'))$ decreases by one when B' is split.

(ii) $\Phi(p(B')) = 0$. Then splitting of B' will be charged 2^D, corresponding to one unit for splitting B' and $2^D - 1$ units for the increase in $\Phi(p(B'))$.

It follows that the total charge for the smooth split of B is at most $2^D \cdot c(B) \leq 2^D \cdot (D+1)!$, as claimed. $\square$

1.4 A Lower Bound Construction

In this section we show that the exponential dependence on D in Theorem 1.1.1 is unavoidable. Namely, we show the 2-dimensional case of Theorem 1.1.2, which says that $\mathrm{ss}(D) \geq (D+1) \cdot 2^D$, and sketch its straightforward extension to higher dimensions. (We refer the reader to Section 4 of [BY17] for further details on the higher dimensional case.)

To present our lower bound, we introduce notation for *child indicators* $c \in \{-1, 1\}^D$. Namely, $B.c$ identifies the child in the (higher-dimensional) quadrant c of a non-leaf box B. For notational convenience, we define $B.c^n := (B.c^{n-1}).c$ for $n \geq 1$, and $B.c^0 := B$.

1.4.1 The 2-dimensional case

We now present and analyze the 2-dimensional lower-bound construction.

Lemma 1.4.1 (2-dimensional case of Theorem 1.1.2)**.** *There is a sequence of* $n + O(1)$ ***ssplit*** *operations that causes* $12n$ ***split*** *operations in a smooth subdivision of* $[-1, 1]^2$.

Proof. Let $B := [-1, 1]^2$ be the initial box in a 2-dimensional subdivision. We describe our lower bound construction in three stages.

Let $c^* := (1, 1)$. The first stage of our construction performs three smooth splits on the following sequence of boxes.

$$B, B.(-c^*), B.(-c^*).c^*. \tag{1.15}$$

None of the smooth splits in the first stage triggers additional splits. See Figure 1.9. Let $B' := B.(-c^*).c^*$.

The second stage performs n smooth splits on the following sequence of boxes.

$$B'.(c^*)^1, B'.(c^*)^2, \ldots, B'.(c^*)^n. \tag{1.16}$$

Each such smooth split triggers four splits, as shown in Figure 1.10.

For each $c \in \{-1, 1\}^D$, the third stage performs a smooth split on the box $B.c.(-c)^{n-1}.c$. Unlike the first two stages, the order in which these four boxes are split is irrelevant. Each of the four smooth split operations in the third stage triggers $2n - 1$ splits. See Figure 1.11.

In total, our construction performed $3 + n + 4 = n + O(1)$ smooth splits, and triggered $3 + 4n + 8n - 4 = 12n - O(1)$ splits. Letting n go to infinity, we get that

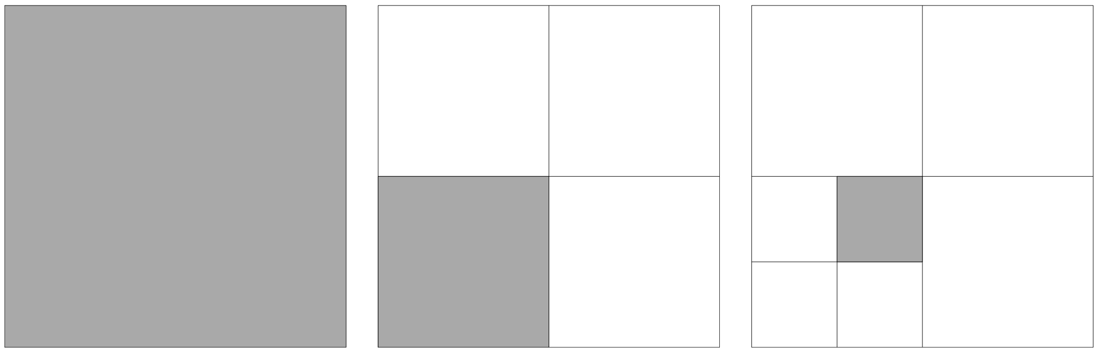


Figure 1.9: The first stage in the 2-dimensional smooth quadtree lower bound construction. The boxes B, $B.(-c^*)$, $B.(-c^*).c^*$ appear in dark gray in the first, second, and third subdivisions from the left, respectively.

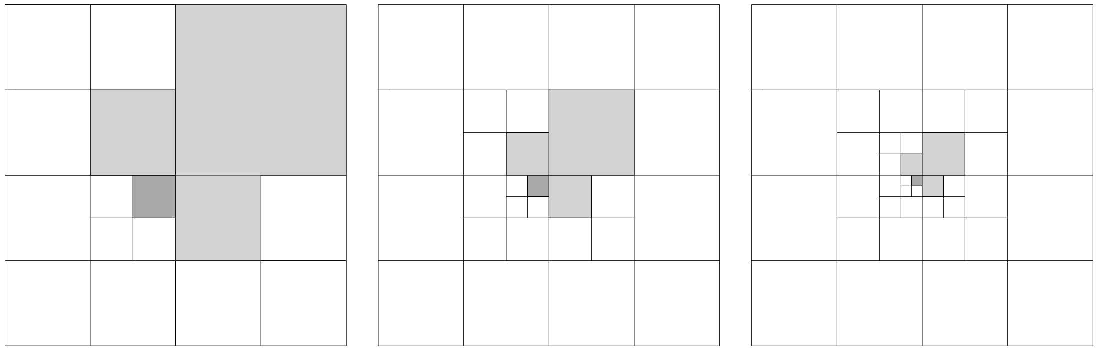


Figure 1.10: The second stage in the 2-dimensional smooth quadtree lower bound construction. The boxes $B'.(c^*)^i$ for $i = 1, 2, 3$ appear in dark gray, and the boxes which must be split to restore smoothness appear in light gray.

$\mathrm{ss}(2) \geq 12$. □

We next sketch how to extend our 2-dimensional lower bound construction to higher dimensions.

Proof sketch of Theorem 1.1.2. We state our construction in three stages, which are similar to those in the 2-dimensional case. Let $D \geq 1$, let $B := [-1, 1]^D$, and let $c^* := (1, 1, \ldots, 1)$ denote the all ones child indicator.

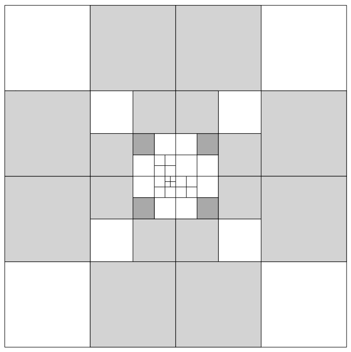


Figure 1.11: The third stage in the 2-dimensional smooth quadtree lower bound construction. The four boxes $B.c.(-c)^{n-1}.c$ with $c \in \{-1,1\}^2$ appear in dark gray, and the boxes which must be split to restore smoothness appear in light gray.

First, we perform smooth splits on the following $D+1$ boxes.

$$B, B.(-c^*), B.(-c^*).c^*, B.(-c^*).(c^*)^2, \ldots, B.(-c^*).(c^*)^{D-1}. \tag{1.17}$$

Let $B' := B.(-c^*).(c^*)^{D-1}$. Next, we perform smooth splits on the following n boxes.

$$B'.(c^*)^1, B'.(c^*)^2, \ldots, B'.(c^*)^n. \tag{1.18}$$

Like in the 2-dimensional case shown in Figure 1.10, each smooth split in the second stage causes a split in each quadrant of $[-1,1]^D$. Therefore, we perform n smooth splits and $2^D n$ splits in the second stage.

For each $c \in \{-1,1\}^D$, the third stage performs a smooth split on the box $B.c.(-c)^{n-1}.c$. Each of the 2^D smooth split operations in the third stage triggers $Dn-(D-1)$ splits, for a total of $2^D \cdot (Dn-(D-1))$ splits total.

In total, our construction performed $(D+1)+n+2^D = n+O_D(1)$ smooth splits, and triggered at least $(D+1)+2^D \cdot n+2^D \cdot (Dn-(D-1)) = (D+1)\cdot 2^D \cdot n - O_D(1)$ splits. Letting n go to infinity, we get that $\mathrm{ss}(D) \geq (D+1)\cdot 2^D$. □

Chapter 2

Planar Minimization Diagrams via Subdivision with Applications to Anisotropic Voronoi Diagrams

This chapter is based on the publication [BPY16], which was joint work with Evanthia Papadopoulou and Chee Yap.

2.1 Introduction

Voronoi diagrams are one of the most important and extensively studied objects in computational geometry [OBSC00, AKL13]. They appear in a tremendous number of applications, including nearest neighbor search [Lee82, KS04], motion planning [ÓY85, TS89], and meshing [LS03] within geometry, as well as in many areas of computer science and science more broadly.

In the simplest setting, given a set of input points (called *sites*) in the plane, a

Voronoi diagram partitions the plane into a collection of polygonal regions each of which consists of the points closest in Euclidean distance to some input site. These regions are called *Voronoi cells*, and the common boundary of two Voronoi cells is called a *Voronoi bisector*. The intersection of three or more Voronoi bisectors is a *Voronoi vertex*.

Every aspect of this simple setting generalizes: our input may consist of more complicated sites than points (say, line segments, polygons, or circles), we may measure distance in a non-Euclidean metric, and the ambient space that we partition may be $\mathbb{R}^d$ for some $d > 2$ or some other manifold. In the most general setting, Voronoi diagrams specify a scheme for partitioning an ambient space into a collection of disjoint subsets, where each subset is labeled with a collection of input sites.

Edelsbrunner and Seidel [ES86] introduced a general way to define many types of Voronoi diagrams as *minimization diagrams*. Given a family $\mathcal{F} = \{f_1, \ldots, f_n\}$ of continuous scalar functions $f_i : \mathbb{R}^2 \to \mathbb{R}$, the minimization diagram $M(\mathcal{F})$ partitions the plane into interior-disjoint sets of points X_i on which function f_i is minimal. There exists a simple representation of nearest-site Voronoi diagrams as minimization diagrams: simply set the functions $f_i(\boldsymbol{x})$ to be the distance of $\boldsymbol{x}$ to site S_i. In particular, an algorithm for computing minimization diagrams also works for computing nearest-site Voronoi diagrams.

One issue with arbitrary minimization diagrams is that they may not have the nice geometric properties that many standard Voronoi diagrams have, and therefore in general it is not clear in general how to compute (or approximate) $M(\mathcal{F})$. Klein [Kle89] gave one solution to this in terms of *abstract Voronoi diagrams*, which defines a Voronoi diagram in terms of how its Voronoi bisectors interact. He gives several conditions for how bisectors interact, including that they should intersect in finitely many connected components, and that the Voronoi regions in the underlying diagram should be connected. Unfortunately, the latter condition rules out a number of interesting diagrams, including the weighted Voronoi diagram (see Section 2.1.2).

Additionally, the abstract Voronoi diagram framework and many algorithms for specific, concrete Voronoi diagrams assume a Real RAM model of computation in which one can perform certain operations on arbitrary real numbers at unit cost. For example, the abstract Voronoi diagram framework assumes the ability to determine the exact intersection points of two bisectors. This is possible if the bisectors are algebraic curves, but is expensive. For non-algebraic curves it is not even clear that these intersections are computable (see [CCK$^+$06]).

The aforementioned issues with frameworks and computational models lead to the major motivating question for our work, which also arose in the predecessor paper [YSL12]:

What does it mean to "compute" a Voronoi Diagram?

In [YSL12], Yap et al. present "Three Views of a Voronoi Diagram," which include the "geometric" view of a Voronoi diagram as the set of points closest to two or more input sites, and the "topological" view of a Voronoi diagram as a cell complex.

2.1.1 Our Contribution

In this chapter, as in [YSL12], we take a hybrid view and consider the task of computing Voronoi diagrams that have correct topology as well as high geometric accuracy. Namely, as our main contribution, we present a practical framework for computing an *isotopic ε-approximation* of the minimization diagram of a set of scalar functions which satisfy certain niceness properties. By an isotopic ε-approximation, we mean that the output is both topologically correct (up to isotopy), and approximately geometrically correct (off by at most ε in Hausdorff distance). We do this by using a subdivision-based algorithm and tools from numerical computation related to interval arithmetic, root isolation, and meshing.

Our other main contribution is to introduce the class of anisotropic Voronoi diagrams on polygonal sites. I.e., we consider a diagram on polygonal input sites, each of which is equipped with a (possibly different) anisotropic norm. One can characterize any norm in terms of its unit ball, which must be a centrally symmetric convex body. Anisotropic norms are those whose unit balls are ellipses. Our diagrams generalize the anisotropic Voronoi diagrams on point sites introduced by Labelle and Shewchuk [LS03], which in turn generalize weighted Voronoi diagrams on point sites.

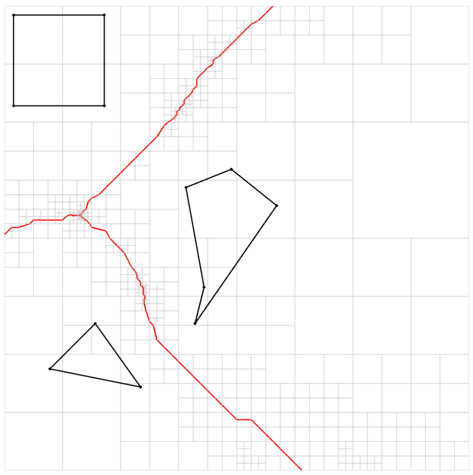
Finally, we show how to use our framework to compute anisotropic Voronoi diagrams on polygonal sites, and report on experimental results from our prototype implementation of our algorithm, SubVor, which is available as a stand-alone package on GitHub [BLPY16] and as part of the Core Library [Cor].

2.1.2 A First Example

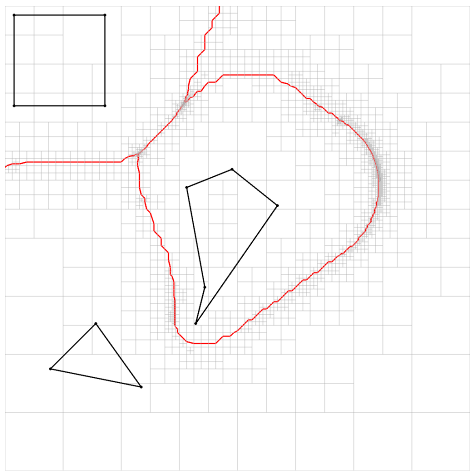
A *weighted Voronoi diagram* on input sites $S_1, \ldots, S_n \subseteq \mathbb{R}^2$ is one in which each site S_i is assigned a weight $w_i > 0$, and the *separation* of a point $\boldsymbol{x}$ from S_i is given by the scaled Euclidean distance $\operatorname{Sep}_{S_i}(\boldsymbol{x}) := \min\{\|\boldsymbol{x} - \boldsymbol{y}\|/w_i : \boldsymbol{y} \in S_i\}$.[3] In other

[2]If viewed on a computer, these images look much better zoomed in to at least 200%.

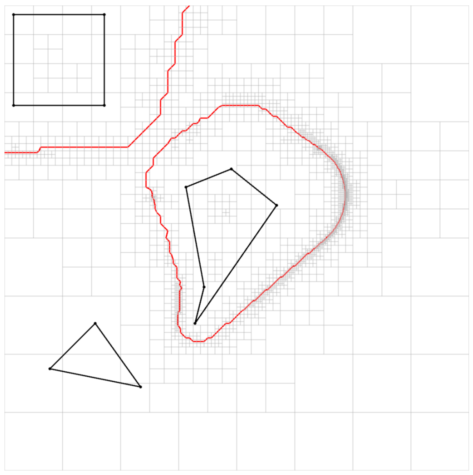
[3]In general we use the term *separation* instead of *distance* throughout this chapter to emphasize that our "distance" functions need not correspond to metrics. For example, our algorithm



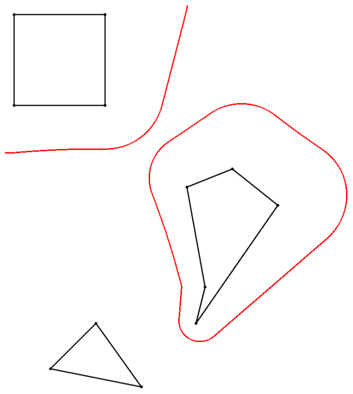
Weights $(1,1,1)$.



Weights $(3,1,1)$.



Weights $(4,1,1)$.



Weights $(4,1,1)$ with high accuracy.

Figure 2.1: Four weighted Voronoi diagrams on polygons, produced by our program, SubVor. Input sites are shown in black, the subdivision grid in gray, and the computed (approximate) Voronoi diagram in red. The triples of numbers (i,j,k) denote the weights given to the triangle, square, and pentagon, respectively. As the weight of the triangle increases from 1 to 3 to 4, the topology of the Voronoi diagram changes. The diagram in the lower right is computed to higher accuracy (ε is smaller), and shown without the underlying subdivision grid.[2]

words, a point is "closer" by a factor of w_i to S_i than its Euclidean distance, and

handles additively weighted Voronoi diagrams, in which separation functions consist of distances plus a scalar weight.

sites with larger weights have more close points (i.e. have larger Voronoi regions).

As a first example, we present several weighted Voronoi diagrams on polygons produced by our prototype program, SubVor, shown in Figure 2.1. The input to the diagrams is a collection of polygonal input sites (a triangle, a square, and a pentagon) inside a bounding box B_0, each of which is assigned a weight.

The figures show how the topology of the underlying Voronoi diagram changes as the weight of the triangle increases. In the first figure, the weights are all the same, and there is a single Voronoi vertex. In the second figure, the triangle has a weight 3 times higher than the other sites, and there are two Voronoi vertices within B_0. In the third and fourth figures, the triangle has a weight 4 times higher than the other sites, and there are no Voronoi vertices. The Voronoi diagrams in the third and fourth figures are *isotopic*, i.e., one may be smoothly deformed to the other, but the fourth figure is computed to higher geometric accuracy. That is, the Hausdorff distance of the diagram shown in the fourth figure to the actual underlying Voronoi diagram is lower.

2.1.3 Related Work

Two closely related papers to our present work are the predecessor paper [YSL12] by Yap et al. and [LSVY14] by Lien et al. In [YSL12], Yap et al. discuss issues related to what it means to “compute” a Voronoi diagram, and give a subdivision-based algorithm for computing an isotopic ε-approximation of a Euclidean Voronoi diagram with polygonal input sites. In the present work we extend [YSL12] largely by using the more powerful numerical techniques described in [LSVY14] for computing an isotopic ε-approximation of an arrangement of two curves. In particular, we follow their high-level approach of (1) detecting and isolating all of the roots

(Voronoi vertices), and (2) using the Plantinga-Vegter algorithm to connect the roots.

In additional closely related work, Emiris et al. [EMM13] present an algorithm to compute isotopic ε-approximations of minimization diagrams via subdivision, and present anisotropic Voronoi diagrams on point sites as an example of their technique. Although the goal of their work and ours is very similar, our work differs in terms of the techniques we use and is more general. In particular their work assumes that the underlying curves are algebraic, and uses techniques for finding roots of polynomial equations described in [MP09].

Much work has gone into finding exact algorithms for Voronoi diagrams (working in the real RAM model), as well as computing geometric approximations of Voronoi diagrams. Work on the latter topic has come from both the computational geometry and geometry processing communities.

In [Har01], Har-Peled studies computing approximate Voronoi diagrams which have near-linear combinatorial complexity, in contrast to exact Voronoi diagrams which have combinatorial complexity which is exponential in the dimension. In follow-up work, Har-Peled and Kumar [HK15] study the problem of computing an approximate minimization diagram. As one motivating example, they consider a Voronoi diagram on point sites each of which has an "ellipse norm" (which are our anisotropic norms). Both papers use compressed quadtrees as the underlying data structure. Furthermore, a paper by Labelle and Shewchuk [LS03] introduced anisotropic Voronoi diagrams on point sites in the context of generating high-quality anisotropic meshes.

However, besides [MP09, YSL12], there has been little focus on numerical algorithms which ensure correct topology. This is a key part of Yap's research program for finding practical (implementable), certifiably correct algorithms using numerical techniques [Yap09].

2.1.4 Summary and Open Problems

The overall focus of this chapter is both more conceptual and more applied than other chapters. The idea from a conceptual standpoint is to showcase the power of subdivision and numerical algorithms. We give a framework, that unlike most algorithms which either work in the (often unrealistic) Real RAM model or only focus on geometric accuracy, is both mostly numerical and focuses on correct topology. Our framework includes powerful and interesting techniques which are underused in computational geometry, and should have further applications.

On the other hand, this chapter is also more applied than other chapters. We presented a new type of Voronoi diagram (an anisotropic diagram on polygonal sites), and one of the main contributions of this chapter is to validate our framework by reporting on experimental results from our prototype implementation.

There are several downsides to our algorithm from a theoretical standpoint. The first is that, although in principal our algorithm outputs a diagram which is both topologically and geometrically accurate in a precise sense, we do not prove this rigorously. We do prove the correctness of parts of our framework (and use existing proofs of correctness for other parts), but the full algorithm described in Section 2.3 has many moving parts and lacks a full proof of correctness.

Open Problem 2.1.1. *Simplify and provide a full proof of correctness for the algorithm described in Section 2.3.*

The second downside to our algorithm is that it lacks time complexity analysis. This is largely because of the components in our predicate Root (given in Equation (2.8)) for testing whether a box contains an isolated Voronoi vertex.

Open Problem 2.1.2. *Analyze how many splits are required to ensure that for every Voronoi vertex in B_0 there exists a subdivision box B containing the vertex on which* Root(B) *holds.*

One approach to addressing this problem is through the "continuous amortization" framework introduced by Burr et al. [BKY09, Bur16] for analyzing the time complexity of subdivision algorithms.

A final downside is that our algorithm as described does not fully handle input which is not in general position. (It does handle such input in a more limited sense; see Section 2.3.6). This is in part because degenerate input is hard for subdivision algorithms to handle in general. For example, they have difficulty distinguishing a single root from a pair of very close roots. Nevertheless, an important problem is to handle such input better.

Open Problem 2.1.3. *Modify our algorithm to provide better guarantees for input that is not in general position.*

On the positive side, our framework is very general, and should work for computing minimization diagrams in higher dimensional space and other settings. In particular, all of the box predicates described in Section 2.2.3 work in higher dimensions.

Open Problem 2.1.4. *Extend our algorithm to computing minimization diagrams in $\mathbb{R}^3$ and other spaces.*

2.2 Preliminaries

We next present background material about relevant math, box predicates, minimization diagrams, and Voronoi diagrams.

2.2.1 Mathematical Preliminaries

We start by giving definitions related to sets and functions. An *implicit curve* T is the zero set of a continuous scalar function $f : \mathbb{R}^2 \to \mathbb{R}$. I.e., $T = f^{-1}(0)$. A *square system* of equations $F = (f_1, \ldots, f_n) : \mathbb{R}^n \to \mathbb{R}^n$ is one in which each f_i takes a vector $\boldsymbol{x} \in \mathbb{R}^n$ as input.

We define a *box* as $B = I_1 \times \cdots \times I_n$, where each $I_j = [a_j, b_j]$ is an interval. The volume of B is $\mu(B) := \prod_{i=1}^{n}(b_j - a_j)$. We define the evaluation of a function $f : \mathbb{R}^n \to \mathbb{R}^m$ on a set $S \subseteq \mathbb{R}^n$ to be $f(S) := \{f(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^n\}$. Following [PV04], we define a *convergent inclusion* interval form $\Box f$ of a function $f : \mathbb{R}^n \to \mathbb{R}^m$ as a function that satisfies the following two properties:

1. $\boldsymbol{x} \in B$ implies that $f(\boldsymbol{x}) \in \Box f(B)$ (Inclusion),

2. Given a sequence of boxes $B_1 \supset B_2 \supset \cdots$ with $\mu(B_i) \to 0$ as $i \to \infty$, it holds that $\mu(f(B_i)) \to 0$ as $i \to \infty$ (Convergence).

The idea is to use convergent inclusion interval forms of functions to avoid exact computation as much as possible while still ensuring correctness. For example, given a system of equations $F = (f_1, \ldots, f_n)$, if we know that $\mathbf{0} \notin \Box F(B)$ for some i then we know that F cannot have a root in B. In fact, to ensure that F does not have a root in B it suffices to show that $\mathbf{0} \notin \Box f_i(B)$ for some i.

Define the *gradient* ∇f of a function $f : \mathbb{R}^n \to \mathbb{R}$ to be the row vector of partial derivatives of f. Namely,

$$\nabla f := (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}).$$

Define the *Jacobian* J_F of a square system $F : \mathbb{R}^n \to \mathbb{R}^n$ to be the matrix

$$J_F := \begin{pmatrix} \nabla f_1 \\ \vdots \\ \nabla f_n \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \tag{2.1}$$

where $(J_F)_{i,j} = \frac{\partial f_i}{\partial x_j}$ denotes the partial derivative of f_i with respect to x_j.

Following [LSVY14], our definitions of topological and geometric correctness will be based on isotopy and Hausdorff distance, respectively. Given closed sets $S, T \subseteq \mathbb{R}^2$, we say that S is *isotopic* to T if there exists a continuous mapping $\gamma : [0,1] \times \mathbb{R}^2 \to \mathbb{R}^2$ such that for every $t \in [0,1]$, the function $\gamma_t : \mathbb{R}^2 \to \mathbb{R}$ (with $\gamma_t(\boldsymbol{x}) = \gamma(t, \boldsymbol{x})$) is a homeomorphism, γ_0 is the identity map, and $\gamma_1(S) = T$.

The *Euclidean Hausdorff distance* between a pair of sets $X, Y \subseteq \mathbb{R}^n$ is

$$d_H(X,Y) := \max\{\sup_{\boldsymbol{x}\in X} \inf_{\boldsymbol{y}\in Y} \|\boldsymbol{x} - \boldsymbol{y}\|, \sup_{\boldsymbol{y}\in Y} \inf_{\boldsymbol{x}\in X} \|\boldsymbol{y} - \boldsymbol{x}\|\}.$$

2.2.2 Minimization Diagrams and Voronoi Diagrams

In this section we formally define minimization diagrams, Voronoi diagrams, and related terminology. We slightly abuse notation and extend terminology for Voronoi diagrams (including Voronoi regions, bisectors, and vertices) to the more general setting of minimization diagrams.

Given a collection of continuous functions $\mathcal{F} = \{f_1, \ldots, f_n\}$ with $f_i : \mathbb{R}^2 \to \mathbb{R}$, we define the *clearance* of a point $\boldsymbol{x}$ with respect to $\mathcal{F}$ as $\mathrm{Clr}(\boldsymbol{x}) = \mathrm{Clr}_{\mathcal{F}}(\boldsymbol{x}) := \min_{i\in[n]} f_i(\boldsymbol{x})$. Given a collection $\mathcal{F}' \subseteq \mathcal{F}$ of functions, we define the *Voronoi variety* of $\mathcal{F}'$ as

$$\mathrm{Vvar}(\mathcal{F}') := \{\boldsymbol{x} \in \mathbb{R}^2 : \forall f \in \mathcal{F}', f(\boldsymbol{x}) = \mathrm{Clr}(\boldsymbol{x})\}. \tag{2.2}$$

Using this definition, we formalize minimization diagrams as the set of points $\boldsymbol{x}$ on which (at least) two distinct functions f_i, f_j achieve the clearance of $\boldsymbol{x}$.

Definition 2.2.1. The *minimization diagram* of a collection of continuous functions $\mathcal{F} = \{f_1, \ldots, f_n\}$ with $f_i : \mathbb{R}^2 \to \mathbb{R}$ is $M(\mathcal{F}) := \bigcup\limits_{i\neq j} \mathrm{Vvar}(\{f_i, f_j\})$.

We call sets $\mathrm{Vvar}(\{f_i\})$ the *Voronoi regions* corresponding to f_i. We call each connected component of $\mathrm{Vvar}(\{f_i\})$ a *Voronoi cell.* Similarly, given distinct f_i, f_j, f_k, we call sets of the form $\mathrm{Vvar}(\{f_i, f_j\})$ and $\mathrm{Vvar}(\{f_i, f_j, f_k\})$ the *Voronoi bisectors* (or simply bisectors) and *Voronoi vertices* of $M(\mathcal{F})$, respectively. We note that Voronoi bisectors are intersections of two Voronoi regions, and that Voronoi vertices are the intersections of three or more Voronoi bisectors. Additionally, we note that a Voronoi bisector $\mathrm{Vvar}(\{f_i, f_j\})$ with $f_i \neq f_j$ is a restriction of the implicit curve $(f_i - f_j)^{-1}(0)$, which will allow us to use the machinery described in Section 2.2.3.

We next define Voronoi diagrams as a special case of minimization diagrams. Consider a collection of sites $\mathcal{S} = \{S_1, \ldots, S_n\}$ with $S_1, \ldots, S_n \subseteq \mathbb{R}^2$, where each S_i is equipped with a norm $\|\cdot\|_{S_i}$. Let the *separation* of a point $\boldsymbol{x} \in \mathbb{R}^2$ from a site S_i be $\mathrm{Sep}_{S_i}(\boldsymbol{x}) := \inf_{\boldsymbol{y}\in S_i} \|\boldsymbol{x} - \boldsymbol{y}\|_{S_i}$. We then define the Voronoi diagram of $\mathcal{S}$ as the minimization diagram of the separation functions.

Definition 2.2.2. The *Voronoi diagram* of a collection of sites $S_1, \ldots, S_n \subseteq \mathbb{R}^2$ is the minimization diagram of $\{\mathrm{Sep}_{S_1}, \ldots, \mathrm{Sep}_{S_n}\}$.

In other words, a Voronoi diagram is the set of points that are "closest" to two or more input sites.

Finally, we give a formal definition of what it means to be a topologically and geometrically accuracy approximation of a minimization (Voronoi) diagram.

Definition 2.2.3. Given a family of continuous functions $\mathcal{F} = \{f_1, \ldots, f_n\}$ with $f_i : \mathbb{R}^2 \to \mathbb{R}$, a set $\widetilde{M}(\mathcal{F}) \subseteq \mathbb{R}^2$ is an *isotopic ε-approximation* of a minimization diagram $M(\mathcal{F})$ if $\widetilde{M}(\mathcal{F})$ is isotopic to $M(\mathcal{F})$, and $d_H(\widetilde{M}(\mathcal{F}), M(\mathcal{F})) \leq \varepsilon$.

We define the special case of isotopic ε-approximate Voronoi diagrams analogously. Our goal will be to give an algorithm for computing isotopic ε-approximations of minimization diagrams over functions that satisfy some natural properties. In particular, we will primarily consider families of functions $\mathcal{F}$ that are in *general position.* I.e., $\mathcal{F}$ is in general position if $\mathrm{Vvar}(\mathcal{F}')$ is empty when $\mathcal{F}' \subseteq \mathcal{F}, |\mathcal{F}'| > 3$, when all Voronoi bisectors are 1-dimensional, and when all Voronoi vertices are 0-dimensional.

2.2.3 Box Predicates

One of the key ideas in subdivision algorithms is the use of *box predicates* as primitives. These allow us to verify properties of an equation or system of equations in a local region (often a box B in the subdivision, or the union of several such adjacent boxes) using interval arithmetic.

Our algorithm for computing minimization diagrams uses several predicates, three of which we describe below. Namely, we describe the Moore-Kioustelidis Test [MK80], the Jacobian Test, and the Plantinga-Vegter Test [PV04]. The

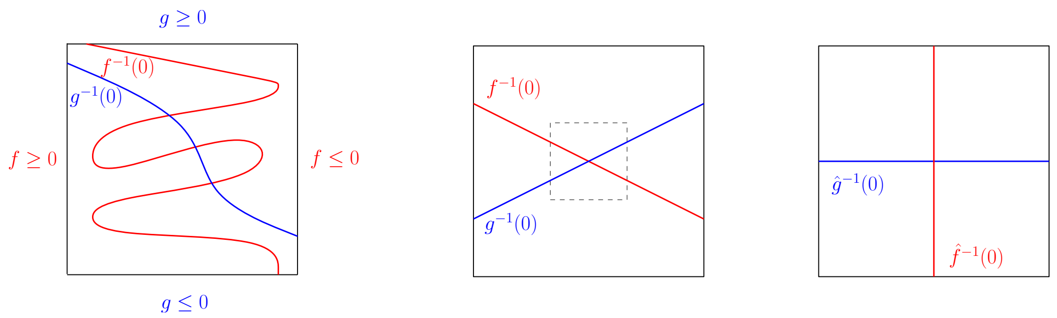


Figure 2.2: A successful Poincaré-Miranda Theorem on implicit curves $f^{-1}(0)$ and $g^{-1}(0)$ induced by functions f, g (left). Even when a system of equations $F = (f, g)$ is linear, an arbitrarily small box containing a root of F may not satisfy the Poincaré-Miranda Theorem, as shown by the dotted inner box (center). The Moore-Kioustelidis Test remedies this by preconditioning F in a way that "locally orthoganilizes" it (right).

Moore-Kioustelidis Test and Jacobian Test put together allow us to isolate a single root of a system of equations in a box. They have been used together in previous work by Mantzaflaris et al. [MMT11] and Lien et al. [LSVY14] for root isolation. The Plantinga-Vegter Test ensures that the curvature of a Voronoi bisector is not too high in a given box, a property which is essential for our construction algorithm.

We state these tests with respect to a square system of equations $F = (f_1, \ldots, f_n)$ where $f_i : \mathbb{R}^n \to \mathbb{R}$ is continuous and has continuous first derivatives (f_i is C^1) for every i.

The Moore-Kioustelidis Test

The Moore-Kioustelidis Test $\mathrm{MK}_F(B)$ [MK80] asserts that F has at least one root in B. The Moore-Kioustelidis Test amounts to a preconditioned version of the Poincaré-Miranda Theorem, which we describe next following the exposition in [Kul97]. For a box $B = [x_1^-, x_1^+] \times \cdots \times [x_n^-, x_n^+]$ we denote the ith opposite faces as

$$B_i^- = \{\boldsymbol{x} \in B : x_i = x_i^-\}, \qquad B_i^+ = \{\boldsymbol{x} \in B : x_i = x_i^+\},$$

where x_i denotes the ith coordinate of $\boldsymbol{x}$.

Lemma 2.2.4 (Poincaré-Miranda Theorem)**.** *Let $F : \mathbb{R}^n \to \mathbb{R}^n$, $F = (f_1, \dots, f_n)$ be a continuous system of equations for which there exists a permutation $\pi : [n] \to [n]$ such that for each $i \in [n]$, $\square f_i(B_{\pi(i)}^-) \subseteq (-\infty, 0]$ and $\square f_i(B_{\pi(i)}^+) \subseteq [0, \infty)$, or vice-versa. Then F has at least one root in B.*

The Poincaré-Miranda Theorem says that if each f_i is non-negative and non-positive on a different pair of opposite faces of a box B then F has a root in B (see the left diagram in Figure 2.2). Unfortunately, the Poincaré-Miranda Theorem is not complete in the sense that there may be systems of equations with roots which it fails to detect even when evaluated on arbitrarily small boxes which contain the root (see the center diagram in Figure 2.2). To fix this, the Moore-Kioustelidis Test preconditions the system F by multiplying by the inverse of its Jacobian evaluated at the midpoint of B. This "locally orthogonalizes" F in B, and forms a complete test [MK80] (see the right diagram in Figure 2.2).

Definition 2.2.5 (The Moore-Kioustelidis Test)**.** The Moore-Kioustelidis Test evaluated on a box B, $\mathrm{MK}_F(B)$, holds if and only if the Poincaré-Miranda Theorem (given in Lemma 2.2.4) holds on $\hat{F} := J_F^{-1}(\boldsymbol{m}_B) \cdot F$.

Here $J_F^{-1}(m_B)$ denotes the inverse of the Jacobian of F evaluated at the midpoint point m_B of B. Note that $\hat{F}$ has a root in B if and only if F has a root in B.

The Jacobian Test

The MK-test gave a condition in which a system of equations has at least one root in a box. We next give a condition in which such a system has at most one root. Define the Jacobian Test applied to a system of equations F on a box B as

$$\mathrm{JC}_F(B) := 0 \notin \det(\Box J_F(B)). \tag{2.3}$$

Lemma 2.2.6. *If the Jacobian Test $JC_F(B)$ is true then F has at most one root in B.*

The Jacobian Test is folklore. The main idea behind its correctness (Lemma 2.2.6) is to show the contrapositive using the mean value theorem. See, e.g., Theorem 12.1 and its corollary in [Abe07].

The Plantinga-Vegter Test

Last, we introduce the Plantinga-Vegter Test $\mathrm{PV}_f(B)$ [PV04] which restricts the amount of curvature of a single function $f : \mathbb{R}^2 \to \mathbb{R}$ in a box B.

$$\mathrm{PV}_f(B) := \langle \Box\nabla f(B), \Box\nabla f(B) \rangle > 0. \tag{2.4}$$

As Plantinga and Vegter observe, the success of this test ensures that the direction of the gradient of f (and hence the direction of f itself) does not change by more than $\pi/2$ radians in B. Moreover, the success of the PV test ensures that at least one of $\Box\frac{\partial f}{\partial x}(B) \cdot \Box\frac{\partial f}{\partial x}(B)$ and $\Box\frac{\partial f}{\partial y}(B) \cdot \Box\frac{\partial f}{\partial y}(B)$ is strictly positive. This implies that f is strictly increasing or decreasing in either the x or y direction, and hence that it is parameterizable in that direction.

The Plantinga-Vegter Test is *hereditary* in the sense that if $B' \subseteq B$ and PV(B) holds then PV(B') holds as well. The Jacobian Test is hereditary as well. The Moore-Kioustelidis Test is not hereditary in the same sense as the other two tests since B' may not contain any of the roots of B, but Lemma 6 in [LSVY14] shows that MK(B') holds for all sufficiently small boxes B' containing a root of F. We will make use of these hereditary properties to continue splitting until multiple box predicates hold simultaneously.

2.2.4 Tracking Active Functions

Usually only a small subset of all n functions in $\mathcal{F}$ affect the minimization diagram in any subdivision box B. Therefore, one of the crucial things to keep track of during subdivision is the set of active sites $\phi(B)$ for each box B in the subdivision, which we represent with a quadtree. A function f is *active* if its separation from some point in B achieves the clearance of that point:

$$\phi(B) := \{f \in \mathcal{F} : \exists \boldsymbol{p} \in B,\ \mathrm{Clr}(\boldsymbol{p}) = f(\boldsymbol{p})\}. \tag{2.5}$$

Using this definition we also define *active bisectors* for B to be those corresponding to a pair of distinct active functions $f, g \in \phi(B)$. Because $\phi(B)$ is difficult to compute exactly, we use a convergent over-approximation $\tilde{\phi}(B)$ of $\phi(B)$ for which we need to introduce the concept of Lipschitz constants.

Given a function $f : \mathbb{R}^2 \to \mathbb{R}$, we define the *Lipschitz constant* of f to be the minimum constant K_f such that for all $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^2$,

$$|f(\boldsymbol{p}) - f(\boldsymbol{q})| \leq K_f \cdot \|\boldsymbol{p} - \boldsymbol{q}\|. \tag{2.6}$$

We define the radius of a box B to be the Euclidean distance from its midpoint to one of its corners.

Lemma 2.2.7. *Let B be a box with midpoint $\boldsymbol{m}_B$ and radius r_B. Let $\boldsymbol{p} \in B$ and $f, g \in \mathcal{F}$ such that Clr$(\boldsymbol{p}) = f(\boldsymbol{p})$ and Clr$(\boldsymbol{m}_B) = g(\boldsymbol{m}_B)$. Then $f(\boldsymbol{m}_B) \leq$ Clr$(\boldsymbol{m}_B) + (K_f + K_g) r_B$.*

Proof. We have that

$$\begin{aligned} f(\boldsymbol{m}_B) &\leq f(\boldsymbol{p}) + K_f \cdot \|\boldsymbol{p} - \boldsymbol{m}_B\| \\ &\leq g(\boldsymbol{p}) + K_f \cdot \|\boldsymbol{p} - \boldsymbol{m}_B\| \\ &\leq g(\boldsymbol{m}_B) + (K_f + K_g) \cdot \|\boldsymbol{p} - \boldsymbol{m}_B\| \\ &\leq \mathrm{Clr}(\boldsymbol{m}_B) + (K_f + K_g) r_B. \end{aligned}$$

The first and third inequalities follow by Equation (2.6), while the second and fourth inequalities follow by the assumptions that $\mathrm{Clr}(\boldsymbol{p}) = f(\boldsymbol{p})$ and $\mathrm{Clr}(\boldsymbol{m}_B) = g(\boldsymbol{m}_B)$, respectively. $\square$

Given a collection $\mathcal{F}' \subseteq \mathcal{F}$ of two or more functions, let

$$K_2(\mathcal{F}') := \max\{K_f + K_g : f, g \in \mathcal{F}', f \neq g\}.$$

We now define and justify the definition of $\tilde{\phi}(B)$ as follows. Let $\tilde{\phi}(B_0) = \mathcal{F}$, and let

$$\tilde{\phi}(B) := \{f \in \tilde{\phi}(p(B)) : f(\boldsymbol{m}_B) \leq \mathrm{Clr}(\boldsymbol{m}_B) + K_2(\tilde{\phi}(p(B))) \cdot r_B\}. \tag{2.7}$$

for $B \subsetneq B_0$, where $p(B)$ denotes the parent box of B.

We show that $\tilde{\phi}(B)$ is a convergent over-approximation version of $\phi(B)$.

Lemma 2.2.8. *For all subdivision boxes B, $\phi(B) \subseteq \tilde{\phi}(B)$. Furthermore, for all sufficiently small boxes B, $\tilde{\phi}(B) = \phi(B)$.*

Proof. Suppose that $f \in \phi(B)$. Then by definition there exists $\boldsymbol{p} \in B$ such that $f(\boldsymbol{p}) = \mathrm{Clr}(\boldsymbol{p})$. If $f(\boldsymbol{m}_B) = \mathrm{Clr}(\boldsymbol{m}_B)$ then clearly $f \in \tilde{\phi}(B)$ by Equation (2.7). Otherwise, there exists some $g \neq f$ such that $g(\boldsymbol{m}_B) = \mathrm{Clr}(\boldsymbol{m}_B)$, in which case we again have that $f \in \tilde{\phi}(B)$ by Lemma 2.2.7. It follows that $\phi(B) \subseteq \tilde{\phi}(B)$.

Furthermore, given a sequence of boxes $B_1 \supset B_2 \supset \cdots$ with $r_{B_i} \to 0$ we have that $f(\boldsymbol{m})_{B_i} \to f(\boldsymbol{p})$, $\mathrm{Clr}(\boldsymbol{m}_{B_i}) \to \mathrm{Clr}(\boldsymbol{p})$, and $r_{B_i} \to 0$. Therefore, $f(\boldsymbol{m}_B) \leq \mathrm{Clr}(\boldsymbol{m}_B) + K_2(\tilde{\phi}(p(B))) \cdot r_B$ eventually holds only if $f \in \phi(B)$. □

2.3 Algorithm

In this section we present our algorithm for computing an isotopic ε-approximation $\widetilde{M}(\mathcal{F})$ of the minimization diagram of a family of functions $\mathcal{F}$. The idea is based on the subdivision paradigm: we repeatedly subdivide an initial bounding box B_0 into smaller boxes until certain box predicates hold. For this, we use a smooth quadtree (as defined in Chapter 1) as the primary underlying data structure to store our subdivision. We then use the guarantees made by the box predicates in the initial splitting stage to construct our approximate diagram in each subdivision box "locally."

2.3.1 The Main Algorithm

In this section we present our main algorithm, whose outline is below. We assume that the input is a family $\mathcal{F}$ of C^1 scalar functions in general position with convergent interval forms. To simplify the description of our algorithm, we also make the (somewhat unrealistic) assumptions that no Voronoi bisector intersects the corner of a subdivision box, and that no Voronoi vertex lies on the boundary of a subdivision box. In Section 2.3.6 we discuss these last assumptions and how to remove them.

- Input: A family $\mathcal{F}$ of C^1 scalar functions in general position, a geometric accuracy parameter $\varepsilon > 0$, and a bounding box B_0.
- Output: A piece-wise linear isotopic ε-approximation $\widetilde{M}(\mathcal{F})$ of the minimization diagram of $\mathcal{F}$ in B_0.

1. Subdivide B_0 (taken as the root of a smooth quadtree) until $|\tilde{\phi}(B)| \leq 3$ for all leaf boxes B in the subdivision.
2. Compute a set of well-isolated root boxes Q_{root} (Section 2.3.2).
3. Perform the Plantinga-Vegter curve tracing construction on $B_0 \setminus (\cup_{B \in Q_{\text{root}}} 5B)$ (Section 2.3.3).
4. Perform construction on the root box B for every $B \in Q_{\text{root}}$ (Section 2.3.4).
5. Perform construction on each box B' in the annulus $5B \setminus B$ of an extended root box for every $B \in Q_{\text{root}}$ (Section 2.3.5).

2.3.2 Isolating Root Boxes

We next describe how to compute a set Q_{root} of subdivision boxes each of which contains exactly one Voronoi vertex. For a box B with center $\boldsymbol{m}_B$, we define its c-scaling to be $cB = \{c(\boldsymbol{p} - \boldsymbol{m}_B) + \boldsymbol{m}_B : \boldsymbol{p} \in B\}$.

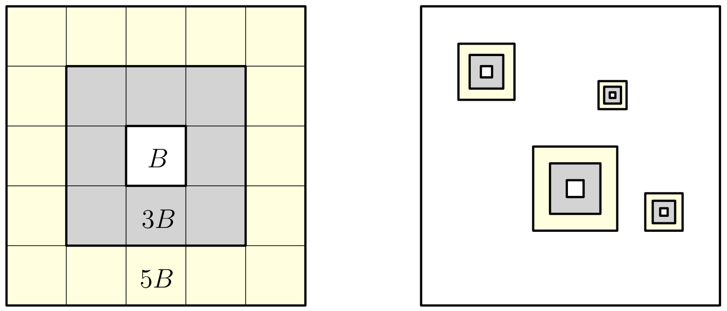


Figure 2.3: A root box B and its extended root box $5B$ (left), and a collection of isolated extended root boxes (right).

Recall that the Voronoi bisectors in $M(\mathcal{F})$ are implicit curves of the form $(f-g)^{-1}(0)$ for some distinct functions $f, g \in \mathcal{F}$. Our root isolation technique depends on the following predicate, which guarantees the existence of a well-isolated root. Namely, we add a box B to Q_{root} if the following predicate holds.

$$\begin{aligned}
\textsc{Root}(B) := {} & 7B \subseteq B_0 \\
& \wedge (\forall B' \in Q_{\text{root}}, 7B \cap 7B' = \emptyset) \\
& \wedge |\tilde{\phi}(B)| = 3 \\
& \wedge \tilde{\phi}(B) = \tilde{\phi}(5B) \\
& \wedge \mathrm{PV}_{\tilde{\phi}(B)}(5B) \\
& \wedge \mathrm{JC}_{\tilde{\phi}(B)}(5B) \\
& \wedge \mathrm{MK}_{\tilde{\phi}(B)}(B).
\end{aligned} \tag{2.8}$$

At a high level $\textsc{Root}(B)$ ensures that there is a unique, well-isolated root in B (see Figure 2.3). We call a box B in the subdivision a *root box* if $\textsc{Root}(B)$ holds, and we call $5B$ its *extended root box*. We now explain each clause in Equation (2.8).

The first clause, $7B \subseteq B_0$, ensures that B is well-separated from the boundary of the bounding box B_0. The second clause, $\forall B' \in Q_{\text{root}}, 7B \cap 7B' = \emptyset$, ensures that B is well-isolated from the set of root boxes already in Q_{root}. The third clause, $|\tilde{\phi}(B)| = 3$, ensures that B has at most 3 active functions. The fourth clause, $\tilde{\phi}(B) = \tilde{\phi}(5B)$, ensures that there are no additional active functions (and hence no additional active bisectors) in the extended root box of B.

For a family of functions $\mathcal{F}' \subseteq \mathcal{F}$, $\mathrm{PV}_{\mathcal{F}'}(B)$ means that PV_f holds for every $f \in \mathcal{F}'$. The fifth clause, $\mathrm{PV}_{\tilde{\phi}(B)}(5B)$, bounds the curvature of each Voronoi bisector in the extended root box.

For a family of functions $\mathcal{F}' \subseteq \mathcal{F}$ with at least three elements, $\mathrm{JC}_{\mathcal{F}'}(B)$ and $\mathrm{MK}_{\mathcal{F}'}(B)$ mean that $\mathrm{JC}_F(B)$ and $\mathrm{MK}_F(B)$ hold for every system $F(\boldsymbol{x}) = ((f - g)(\boldsymbol{x}), (f - h)(\boldsymbol{x}))$ comprised of distinct functions $f, g, h \in \mathcal{F}'$. The sixth clause, $\mathrm{JC}_{\tilde{\phi}(B)}(5B)$, combined with the fact that $|\tilde{\phi}(B)| = 3$, ensures that there is at most one Voronoi vertex in the extended root box $5B$. Finally and most importantly, the seventh clause, $\mathrm{MK}(B)_{\tilde{\phi}(B)}$, ensures that there is at least one Voronoi vertex in B.

The clauses in $\textsc{Root}(B)$ appear in heuristic order of the amount of computation needed to evaluate them, from lowest to highest. Importantly, the predicate $\textsc{Root}$ is hereditary in the sense that if $\textsc{Root}(B)$ holds and $B' \subseteq B$ contains a Voronoi vertex then $\textsc{Root}(B')$ holds as well.

2.3.3 Curve Tracing

Outside of extended root boxes, we use the Plantinga-Vegter curve tracing algorithm [PV04] on a single curve at a time. The Plantinga-Vegter algorithm outputs a piece-wise linear, isotopic approximation of an underlying implicit curve. It amounts to using the well-known marching cubes algorithm [LC87] on a smooth

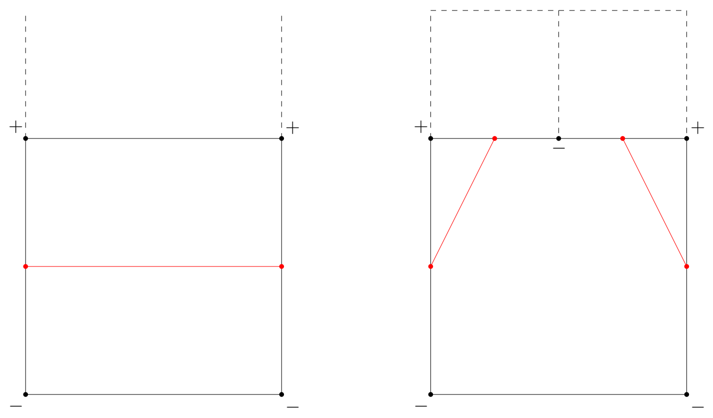


Figure 2.4: Two examples of the marching cubes curve-tracing algorithm in a smooth subdivision applied to an implicit curve $(f-g)^{-1}(0)$. We evaluate the sign of $(f-g)$ at each corner of the box, and, if the neighboring box along a side is split, at the midpoint of the side as well. We then place a bisector node at the midpoint of each side segment whose endpoints have different signs, and attach the bisector nodes in the unique way that is consistent with the corner signs (shown in red).

subdivision, together with the PV predicate given in Equation (2.4).

More precisely, for every subdivision box B outside of an extended root box, we split until $|\tilde{\phi}(B)| \leq 2$. Then, for each such box, the Plantinga-Vegter algorithm ensures that the following predicate holds:

$$(|\tilde{\phi}(B)| = 1) \vee \mathrm{PV}_{\tilde{\phi}(B)}(B).$$

If $|\tilde{\phi}(B)| = 1$, then B lies in the interior of a Voronoi region, and no construction is necessary. Otherwise, $|\tilde{\phi}(B)| = 2$ and we keep splitting until $\mathrm{PV}_{\tilde{\phi}(B)}(B)$ holds, i.e., until $\mathrm{PV}_{(f-g)}(B)$ holds where $\tilde{\phi}(B) = \{f, g\}$.

For each such box B satisfying $|\tilde{\phi}(B)| = 2 \wedge \mathrm{PV}_{(f-g)}(B)$, we then perform the marching cubes construction on B (see Figure 2.4). This consists of (exactly) computing the sign of $(f-g)$ at each corner of B, and placing a *bisector node* on each side of B that has corners with different signs. If B has two neighbors on a

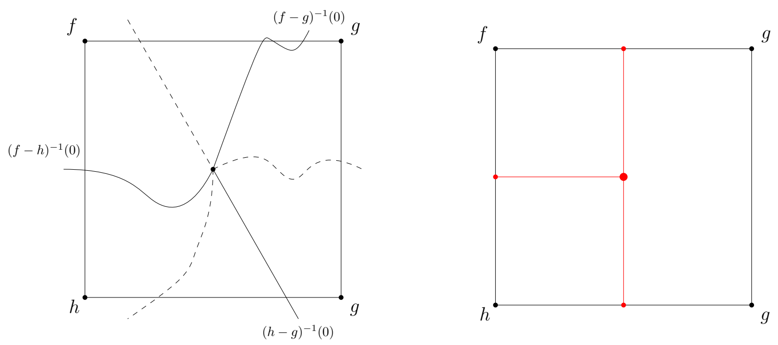


Figure 2.5: A root box containing a Voronoi vertex of the functions f, g, and h with inactive bisector halves shown as dashed (left). If such a box B satisfies $\textsc{Root}(B)$, we label each corner of B with the function with minimal value at each corner of B, and apply a "multi-label marching cubes" type construction (right). A full case analysis appears in Figure 2.6.

side, then we additionally compute the sign at the midpoint of the side, and place a bisector node on each half-side whose endpoints differ. Because we assume that the underlying subdivision is smooth, each leaf box in the subdivision has at most two neighbors.

As Plantinga and Vegter showed, the fact that the PV predicate holds ensures that there is an unambiguous way to connect the bisector nodes on the sides of B that gives correct topology. As our final step, we connect bisector nodes with line segments in this way. Our final construction within B then consists of a line segment or a pair of line segments which form $\widetilde{M}(\mathcal{F}) \cap B$.

2.3.4 Construction Within Root Boxes

We now describe our construction for root boxes $B \in Q_{\text{root}}$. Because $\textsc{Root}(B)$ holds for every $B \in Q_{\text{root}}$, we in particular know that there are three distinct functions $f, g, h, \in \mathcal{F}$ such that $\text{MK}_F(B)$ holds, where $F(\boldsymbol{x}) = (f(\boldsymbol{x}) - g(\boldsymbol{x}), f(\boldsymbol{x}) - h(\boldsymbol{x}))$. This guarantees that there is a Voronoi vertex in B. Moreover, because

$\mathrm{JC}_F(5B)$ holds, we know that this vertex is unique within $5B$.

Our root box construction amounts to a multi-label version of the marching cubes algorithm; see Figure 2.5. A key difference between our setting and the standard root isolation setting is how Voronoi bisectors intersect. There are two differences.

First, because $f(\boldsymbol{x}) - g(\boldsymbol{x}) = f(\boldsymbol{x}) - h(\boldsymbol{x}) = 0$ implies that $g(\boldsymbol{x}) - h(\boldsymbol{x}) = 0$, we have that every root $\boldsymbol{x}$ of two Voronoi bisectors is in fact a root of at least three bisectors (exactly three when $\mathcal{F}$ is in general position). I.e., Voronoi bisectors are dependent.

Second, only half of each bisector going into a Voronoi vertex is active (see the left diagram in Figure 2.5). More formally, given a parameterization $S(t)$ of a curve $S = (f - g)^{-1}(0)$ with $S(0) = \boldsymbol{x}$ for a Voronoi vertex $\boldsymbol{x} \in B$, only one of $\{S(t) : t \geq 0\} \cap B \subseteq M(\mathcal{F})$ and $\{S(t) : t \leq 0\} \cap B \subseteq M(\mathcal{F})$ holds.

Our construction must therefore determine which bisector halves are active. To do this, we first label each corner $\boldsymbol{c}$ of B with $\arg\min_{f \in \tilde{\phi}(B)} f(\boldsymbol{c})$. Then, we place bisector nodes for each of the three bisectors on the boundary of B according to the case analysis in Figure 2.6 to ensure correct topology. Finally, we place a Voronoi vertex in the center of B, and connect each bisector node to the Voronoi vertex.

2.3.5 Construction Within Extended Root Boxes

We now describe our construction of $\widetilde{M}(\mathcal{F}) \cap B'$ for boxes B' in an extended root box $5B \setminus B$ where B is a root box.

Because $\textsc{Root}(B)$ holds, $\mathrm{MK}(B)$ and $\mathrm{JC}(5B)$ hold, and therefore we are guaranteed that there are no roots in $5B \setminus B$. Therefore, the main idea for constructing

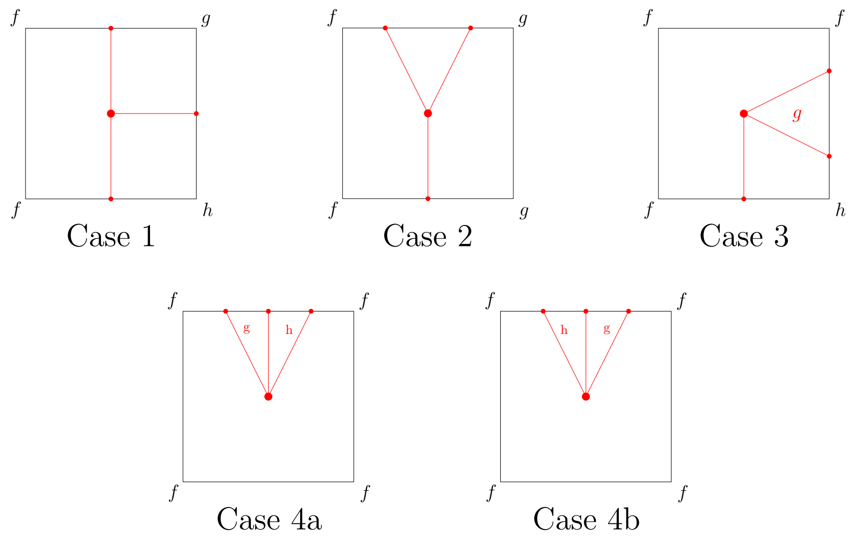


Figure 2.6: Cases for constructing inside a root box B which contains a Voronoi vertex up to rotation and relabeling of the functions. Here the vertex is the intersection of three Voronoi bisectors $(f-g)^{-1}(0)$, $(f-h)^{-1}(0)$, and $(g-h)^{-1}(0)$. In Cases 1, 2, and 3, the corner labels determine the topology of the underlying diagram. In the (hypothetical) Cases 4a and 4b – when all of the corner labels are the same and thus all three bisector nodes lie on one side of the box – the underlying combinatorial pattern is ambiguous. However, Lemma 8 in [LSVY14] shows that these cases are impossible in a box B in which MK(B) holds.

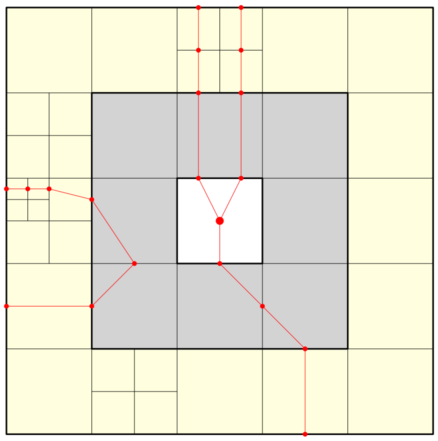


Figure 2.7: Construction within an extended root box $5B$. Boxes in the outer annulus $5B \setminus 3B$ may be split to conform with the outer subdivision $B_0 \setminus 5B$, but the boxes in the inner annulus $3B \setminus B$ remain unsplit and are all congruent with B. Two Voronoi bisectors connected to the Voronoi vertex in B may also intersect the same boxes in $5B/B$, as shown in the gray box above the root box. $5B$ may contain multiple connected components (as shown by the curve at left), but only one principal component.

$\widetilde{M}(\mathcal{F})$ within $5B \setminus B$ is again to use the Plantinga-Vegter curve tracing algorithm (as described in Section 2.3.3) on each Voronoi bisector separately.

However, there are two differences. First, two Voronoi bisectors emanating from the Voronoi vertex in B may intersect the same subdivision box B' in $5B \setminus B$ (as demonstrated by the gray box above the root box in Figure 2.7). However, we are guaranteed that they do not intersect. We therefore perform the marching cubes algorithm on each such Voronoi bisector separately, but place separate nodes on each side segment of B' in order to ensure that the connected segments do not cross.

The second issue only affects boxes B' in $3B \setminus B$ that are neighbors of B. For such boxes, the Plantinga-Vegter algorithm may not "detect" a bisector node on the boundary between B and B' placed during the construction in B described in Section 2.3.4. However, because $3B$ consists of nine congruent subdivision boxes,

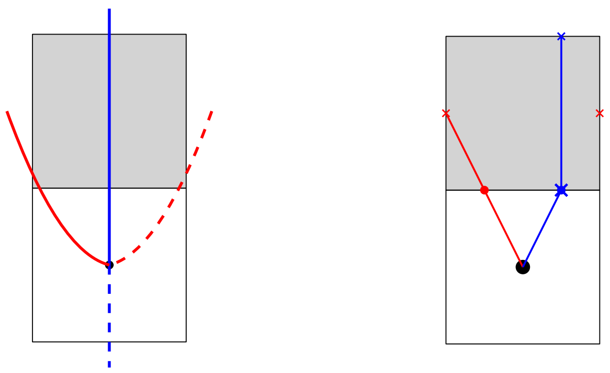


Figure 2.8: A root box B below a box $B' \in 3B$. In the left diagram, two bisectors are shown in red and blue respectively, with their active halves solid and their inactive halves dashed. (The combinatorial type corresponds to Case 2 or Case 3 in Figure 2.6; the third bisector is not shown.) The corresponding construction of $\widetilde{M}(F)$ in B and B' appears on the right. Bisector nodes placed by the root box construction appear as dots, and bisector nodes placed by the Plantinga-Vegter appear as crosses. For the blue bisector, the placement of these nodes is the same, but for the red bisector there are bisector nodes on three sides of B', and we discard the cross corresponding to its inactive half.

the Plantinga-Vegter construction for each bisector only adds two nodes in B'. In this situation we attach the node on the boundary of B and B' to the one bisector node placed by the Plantinga-Vegter construction which ensures correct topology. See Figure 2.8.

Call $S \subseteq M(\mathcal{F}) \cap 5B$ a *component* of $M(\mathcal{F})$ in $5B$ if S is connected, and a *principal component* if additionally $S \cap B \neq \emptyset$. We claim that there is a single principal component in $5B$. After the steps described in Section 2.3.4 and this section so far, we have completed construction of the principal component within $5B$. However, there may be other components in $5B$, as shown by the curve on the left in Figure 2.7. As the final step in our construction, we perform the Plantinga-Vegter construction on any such curves.

2.3.6 Removing Some Assumptions

We now sketch how to remove some of the assumptions on the input to our algorithm. First, we describe how to remove the assumption that no Voronoi bisectors $(f-g)^{-1}(0)$ intersect the corners of subdivision boxes. As mentioned in [PV04], we can simply assign $f(\boldsymbol{x})-g(\boldsymbol{x})$ to be positive whenever $f(\boldsymbol{x})-g(\boldsymbol{x})=0$ on a corner $\boldsymbol{x}$. Unfortunately, determining that $f(\boldsymbol{x})-g(\boldsymbol{x})=0$ requires exact computation, but we are already using this anyway.

Second, we describe how to remove the assumption that no Voronoi vertices lie on the boundary of subdivision boxes. Suppose that in fact a Voronoi vertex does lie on the boundary of a box B. The problems with this are two-fold: the MK test on B will generally fail to detect the vertex, and the constructions in Sections 2.3.4 and 2.3.5 are not valid.

To fix these problems, we use a technique used for root box construction in [LSVY14]. They observe that every Voronoi vertex lies in the interior of $2B$ for some subdivision box B. In particular, if a Voronoi vertex lies on the boundary of B then it lies in the interior of $2B$.

They therefore change the clause $\mathrm{MK}(B)$ to $\mathrm{MK}(2B)$ in Equation (2.8), and perform root box construction (analogous to our Section 2.3.4) on $2B$ rather than B, and extended root box construction on $10B$ instead of $5B$.[4] Since $2B$ is not a subdivision box (although it is a union of several such boxes), making the construction within $10B$ conformal with the external subdivision $B_0 \setminus 10B$ requires some additional work, which is described in [LSVY14].

Finally, we consider the assumption that the input functions are in general

[4][LSVY14] uses an extended root box of size $8B$ rather than $10B$, but the same idea works for $10B$ and is simpler.

position. Although we need this assumption to guarantee that we can isolate each Voronoi vertex, if we apply our algorithm to a family of functions $\mathcal{F}$ that is not in general position then we note that our algorithm still outputs a diagram which isolates all Voronoi vertices which are the intersection of three bisectors, and is within Hausdorff distance ε of $M(\mathcal{F})$.

2.4 Anisotropic Voronoi Diagrams

In this section we introduce the class of Voronoi diagrams of polygonal sites equipped with anisotropic norms in the plane. We will then show how to instantiate our framework to compute an isotopic ε-approximation of such a diagram. In particular in Section 2.4.1 we compute the separation function of a polygon under an anisotropic norm (and show that it is C^1), and in Section 2.4.2 we compute the Lipschitz constant of this separation function. Finally, we discuss our implementation and provide preliminary experimental results in Section 2.4.3.

The class of Voronoi diagrams of polygonal sites (which include point and line segment sites as degenerate special cases) equipped with anisotropic norms generalize anisotropic Voronoi diagrams for point sites, which were introduced by Labelle and Shewchuk [LS03].

Definition 2.4.1. We define the class of *anisotropic norms* to be those of the form $\|\boldsymbol{x}\|_M := \sqrt{\boldsymbol{x}^T M \boldsymbol{x}}$ for some symmetric positive definite matrix $M \in \mathbb{R}^{2\times 2}$.

We can define any norm in terms of its unit ball, which must be a centrally symmetric convex body. Anisotropic norms are those whose unit balls are ellipses. Using this view it is easy to see that anisotropic Voronoi diagrams generalize multiplicatively weighted Voronoi diagrams. The latter diagrams correspond to the special case when $M = c^2 \cdot I_2$ for some $c > 0$, which corresponds to a norm with a $(1/c)$-scaled Euclidean disk as its unit ball.

2.4.1 Separation Computations

In this section, we show how to compute the separation from point sites and line segment sites under an anisotropic norm $\|\cdot\|_M$. Because the boundary of a polygon consists of a chain of point and line segment features, these computations also suffice to determine the separation from a polygon. Namely, the separation from a polygon is the minimum separation to one of its constituent features.

Fix an anisotropic norm $\|\cdot\|_M$, and let $\boldsymbol{r}$ be a point in $\mathbb{R}^2$. Then the squared separation between $\boldsymbol{r}$ and a point site $\boldsymbol{p}$ equipped with $\|\cdot\|_M$ is

$$\mathrm{Sep}_{\boldsymbol{p}}(\boldsymbol{r})^2 = (\boldsymbol{r} - \boldsymbol{p})^T M (\boldsymbol{r} - \boldsymbol{p}), \tag{2.9}$$

which has gradient

$$\nabla \mathrm{Sep}_{\boldsymbol{p}}(\boldsymbol{r})^2 = 2M(\boldsymbol{r} - \boldsymbol{p}). \tag{2.10}$$

Next we compute the separation of r to a line L with parameterization $L(t)$. The separation of $\boldsymbol{r}$ from L is given by $\min\{\|L(t) - \boldsymbol{r}\|_M : t \in \mathbb{R}\}$. Let $t^* = t^*(\boldsymbol{r}) := \arg\min\{\|L(t) - \boldsymbol{r}\|_M : t \in \mathbb{R}\}$ denote the parameter value at which $L(t)$ achieves this minimum separation.

Computing t^* allows us to compute the separation not just from the line L, but to a line segment $[\boldsymbol{p}, \boldsymbol{q}]$, where $\boldsymbol{p} = L(0)$ and $\boldsymbol{q} = L(1)$. Namely, we define $\mathrm{Sep}_{[\boldsymbol{p},\boldsymbol{q}]}(\boldsymbol{r})$ piecewise in terms of $\boldsymbol{t}^*$, with the relevant case depending on whether $\boldsymbol{r}$ is closest to the interior or one of the endpoints of $[\boldsymbol{p}, \boldsymbol{q}]$. We first give a formula for t^*, and the separation to L.

Lemma 2.4.2. *Let $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r} \in \mathbb{R}^2$, with $\boldsymbol{p} \neq \boldsymbol{q}$, and let L be the line (parameterized by t) running through $\boldsymbol{p}, \boldsymbol{q}$ with $L(0) = \boldsymbol{p}, L(1) = \boldsymbol{q}$. Let $\boldsymbol{v} = \boldsymbol{q} - \boldsymbol{p}$ and let $\boldsymbol{w} = \boldsymbol{r} - \boldsymbol{p}$. Then the minimum separation of L equipped with $\|\cdot\|_M$ is achieved at the point $L(t^*(\boldsymbol{r}))$ where*

$$t^*(\boldsymbol{r}) = \frac{\boldsymbol{v}^T M \boldsymbol{w}}{\boldsymbol{v}^T M \boldsymbol{v}}.$$

Therefore, the separation between of $\boldsymbol{r}$ from L is

$$Sep_L(\boldsymbol{r})^2 = \boldsymbol{w}^T M \boldsymbol{w} - (\boldsymbol{v}^T M \boldsymbol{w})^2 / (\boldsymbol{v}^T M \boldsymbol{v}).$$

Proof. Note that $L(t) = L(0) + t(L(1) - L(0)) = \boldsymbol{p} + t\boldsymbol{v}$ so that $\boldsymbol{r} - L(t) = \boldsymbol{w} - t\boldsymbol{v}$. The squared separation of $\boldsymbol{r}$ from $L(t)$ is then $\mathrm{Sep}_{L(t)}(\boldsymbol{r})^2 = (r - L(t))^T M (r - L(t)) = (\boldsymbol{w} - t\boldsymbol{v})^T M (\boldsymbol{w} - t\boldsymbol{v})$, which for fixed $\boldsymbol{r}$ is a univariate quadratic polynomial in t and therefore has a single critical point. We get that the derivative of this separation function with respect to t is

$$\frac{d\,\mathrm{Sep}_{L(t)}(\boldsymbol{r})}{dt} = -2\boldsymbol{v}^T M (\boldsymbol{w} - t\boldsymbol{v}).$$

Setting the derivative equal to zero, we get that the minimum separation is achieved at

$$t^*(\boldsymbol{r}) = \frac{\boldsymbol{v}^T M \boldsymbol{w}}{\boldsymbol{v}^T M \boldsymbol{v}},$$

as desired. Plugging $t^*(\boldsymbol{r})$ in for t in $\mathrm{Sep}_{L(t)}(\boldsymbol{r})$, we get the desired expression for $\mathrm{Sep}_L(\boldsymbol{r})^2$. □

Lemma 2.4.3. *Let $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r} \in \mathbb{R}^2$, with $\boldsymbol{p} \neq \boldsymbol{q}$,and let L be the line (parameterized by* *t) running through* $\boldsymbol{p}, \boldsymbol{q}$ *with* $L(0) = \boldsymbol{p}, L(1) = \boldsymbol{q}$. *Let* $\boldsymbol{v} = \boldsymbol{q} - \boldsymbol{p}$ *and let* $\boldsymbol{w} = \boldsymbol{r} - \boldsymbol{p}$. *Let* $t^* = \arg\min\{\|L(t) - \boldsymbol{r}\|_M : t \in \mathbb{R}\}$.

Then the squared separation of the line segment site $[\boldsymbol{p}, \boldsymbol{q}]$ *equipped with* $\|\cdot\|_M$ *and its gradient are given by the following formulas.*

$$Sep_{[\boldsymbol{p},\boldsymbol{q}]}(\boldsymbol{r})^2 = \begin{cases} Sep_{\boldsymbol{p}}(\boldsymbol{r})^2 & \textit{if } t^*(\boldsymbol{r}) \leq 0, \\ \boldsymbol{w}^T M \boldsymbol{w} - (\boldsymbol{v}^T M \boldsymbol{w})^2 / (\boldsymbol{v}^T M \boldsymbol{v}) & \textit{if } t^*(\boldsymbol{r}) \in (0,1), \\ Sep_{\boldsymbol{q}}(\boldsymbol{r})^2 & \textit{if } t^*(\boldsymbol{r}) \geq 1. \end{cases}$$

$$\nabla(Sep_{[\boldsymbol{p},\boldsymbol{q}]}(\boldsymbol{r})^2) = \begin{cases} \nabla(Sep_{\boldsymbol{p}}(\boldsymbol{r})^2) & \textit{if } t^*(\boldsymbol{r}) \leq 0, \\ 2M(\boldsymbol{w} - \frac{\boldsymbol{v}^T M \boldsymbol{w}}{\boldsymbol{v}^T M \boldsymbol{v}} \cdot \boldsymbol{v}) & \textit{if } t^*(\boldsymbol{r}) \in (0,1), \\ \nabla(Sep_{\boldsymbol{q}}(\boldsymbol{r})^2) & \textit{if } t^*(\boldsymbol{r}) \geq 1. \end{cases}$$

Moreoever, the squared separation function is C^1*, i.e., it and its gradient are continuous.*

Proof. The formula for the separation function follows immediately from Lemma 2.4.2, and the formula for the gradient follows by direct computation. It remains to show that the squared separation function is C^1.

The piece-wise components of $\mathrm{Sep}_{L(t)}(\boldsymbol{r})^2$ and its gradient are all polynomials, and hence smooth. Therefore it suffices to check that the $\mathrm{Sep}_{\boldsymbol{p}}(\boldsymbol{r})^2 = \boldsymbol{w}^T M \boldsymbol{w} - (\boldsymbol{v}^T M \boldsymbol{w})^2 / (\boldsymbol{v}^T M \boldsymbol{v})$ when $t^*(\boldsymbol{r}) = 0$, that $\mathrm{Sep}_{\boldsymbol{q}}(\boldsymbol{r})^2 = \boldsymbol{w}^T M \boldsymbol{w} - (\boldsymbol{v}^T M \boldsymbol{w})^2 / (\boldsymbol{v}^T M \boldsymbol{v})$ when $t^*(\boldsymbol{r}) = 1$, and that the gradients of these pairs of expressions also agree at $t^*(\boldsymbol{r}) = 0$ and $t^*(\boldsymbol{r}) = 1$, respectively.

These equivalences are straightforward to check using the fact that by definition

$t^*(\boldsymbol{r}) = \frac{\boldsymbol{v}^T M \boldsymbol{w}}{\boldsymbol{v}^T M \boldsymbol{v}}$, which implies that $\boldsymbol{v}^T M \boldsymbol{w} = 0$ when $t^*(\boldsymbol{r}) = 0$ and $\boldsymbol{v}^T M \boldsymbol{w} = \boldsymbol{v}^T M \boldsymbol{v}$ when $t^*(\boldsymbol{r}) = 1$. $\square$

2.4.2 Lipschitz Constant Computations

In order to track active features as described in Section 2.2.4 we need an upper bound on the Lipschitz constants of anisotropic norms.

Lemma 2.4.4. *Let* $M = \begin{pmatrix} a & b \\ b & c \end{pmatrix} \in \mathbb{R}^{2\times 2}$ *be a symmetric positive definite matrix. Then for a site* S *equipped with* $\|\cdot\|_M$ *we have*

$$K(Sep_S) = \frac{1}{\sqrt{2}}\sqrt{a + c + \sqrt{(a-c)^2 + 4b^2}}. \tag{2.11}$$

Proof. Because M is symmetric positive definite, the maximizer $\boldsymbol{x}^*$ of $\|\boldsymbol{x}\|_M^2 = \boldsymbol{x}^T M \boldsymbol{x}$ is the eigenvector associated with the largest eigenvalue $\lambda_1(M)$ of M. Moreover, $(\boldsymbol{x}^*)^T M \boldsymbol{x}^* = \lambda_1(M) \cdot \|\boldsymbol{x}^*\|^2$.

A direct computation shows that the eigenvalues of M are $\frac{1}{2}(a+c\pm\sqrt{(a-c)^2+4b^2})$. Taking the square root of the larger eigenvalue gives the expression in Equation (2.11). $\square$

2.4.3 Implementation

We have implemented a prototype of our algorithm, called SubVor, for computing anisotropic Voronoi diagrams of polygons. The prototype is available on GitHub [BLPY16], and received the Replicability Stamp at the Eurographics Symposium on Geometry Processing (SGP) 2016 for being able to reproduce the figures in our conference paper.[5] SubVor follows our algorithm for tracking active features and using box predicates to achieve a topologically correct Voronoi diagram. The visualization component uses OpenGL, and supports basic interaction such as zooming.

However, we emphasize that the code is preliminary. In particular, it does not yet implement all of the details described in the construction phase of the algorithm, only computes up to machine (double) precision even when arbitrary precision is required, and has not been tested thoroughly enough.

The program takes two parameters, ε_a and ε_g, which control topological and geometric accuracy by either limiting or forcing quadtree boxes to split. The first, ε_a, specifies an "absolute" minimum radius for quadtree boxes. If a box's radius is smaller than ε_a and the box is still unresolved then the program ceases splitting and marks the box as unresolved. A box may be marked as unresolved either because it contains a degenerate Voronoi vertex (more than three Voronoi bisectors intersecting at a point), or because ε_a was not set to be small enough for our program to ensure topological correctness. If a box B is marked as unresolved then we do not guarantee anything about the topology of the Voronoi diagram inside B.

The second parameter, ε_g, specifies a bound on the desired "geometric" accuracy of the computation. If a box intersects an active Voronoi bisector then it will always be split down to radius $\varepsilon_g/2$, ensuring that the Hausdorff distance between the actual Voronoi diagram and the computed approximation is less than ε_g.

[5] See `http://www.replicabilitystamp.org/`.

2.4.3.1 Examples

We conclude by giving four examples of Voronoi diagrams computed by our program. Input sites are shown in black, the subdivision grid in gray, and the computed (approximate) Voronoi diagram in red. Unresolved boxes are shown in blue.

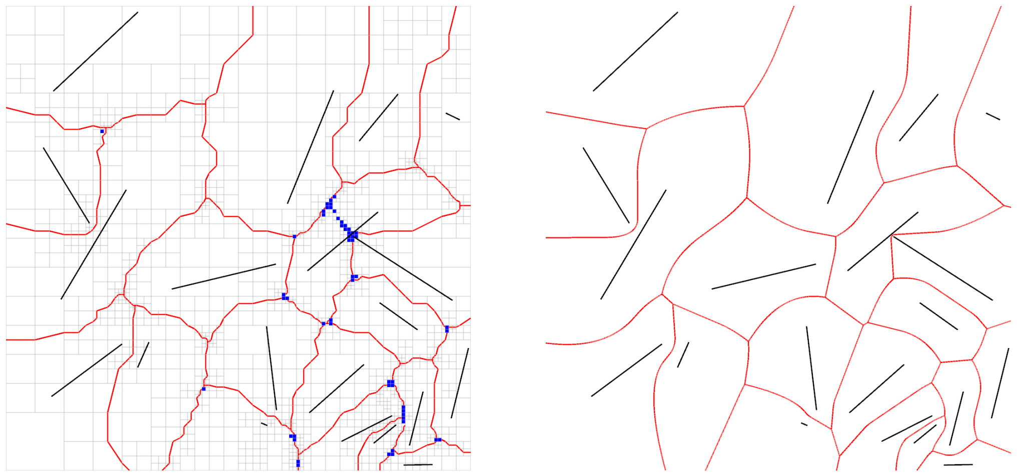


Figure 2.9: These two images show a Voronoi diagram computed on the same collection of line segments. The first image was produced with ε_a set to be relatively large, and with high ε_g, while the second image was produced with small ε_g. The first image shows that relatively little splitting is necessary to trace bisectors and confirm many Voronoi vertices. The second image (in which the subdivision grid is hidden) shows the effect of computing to high geometric precision (small ε_g).

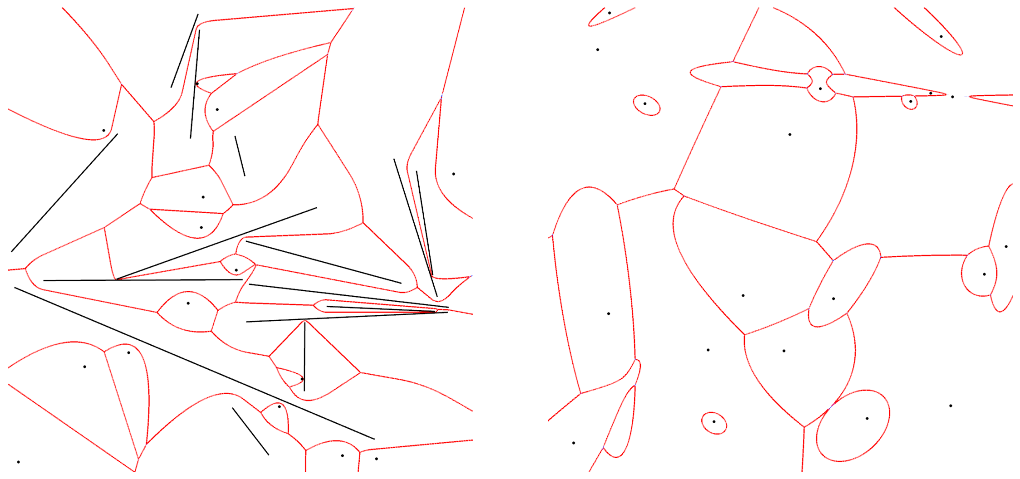


Figure 2.10: A Voronoi diagram with mixed point and line segment input sites with small ε_g (left), and a Voronoi diagram with point sites each equipped with a different anisotropic metric (right). Some of the anisotropic norms are very different in the figure on the right, leading to disconnected Voronoi regions.

Part II

Lattice Algorithms

Chapter 3

Background on Lattices

3.1 Introduction

Lattices are the primary object of study in Part II of this thesis. A lattice is a discrete additive subgroup of $\mathbb{R}^d$, or equivalently, the set of all integer combinations of some linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^d$.

Lattices are well-studied mathematical objects [CS98], and in the last few decades have found many applications within computer science including in integer programming (e.g. [Len83, Kan87, Dad12]), coding theory (e.g. [EZ04, LB14]), and especially cryptography (e.g. [Ajt96, AD97, GGH97, HPS98, GPV08, Reg09b, Gen09]).

In this thesis we present several results related to computational aspects of lattices. First, in this chapter we give background material about linear algebra and lattices. In Section 3.3 we also provide a novel exposition of ties between fundamental domains, CVP(P) algorithms, and basis reduction, which relates to notions of basis reduction that we use in later chapters. Next, in Chapter 4 we study the Lattice Distortion Problem (LDP). The rough goal of LDP is to compute how "similar" two given lattices $\mathcal{L}_1$ and $\mathcal{L}_2$ are. Finally, in Chapter 5 we present algorithms for computing nearly orthogonal and well-conditioned lattice bases.

3.2 Preliminaries

3.2.1 Linear Algebra

In this section we review and establish notation for a number of basic concepts in linear algebra. For a well-written exposition of most of the concepts, see [TI97].

For $1 \le p < \infty$, the ℓ_p norm of a vector $\boldsymbol{x} \in \mathbb{R}^n$ is defined as

$$\|\boldsymbol{x}\|_p := \Big(\sum_{i=1}^{n} |x_i|^p\Big)^{1/p},$$

and the ℓ_∞ norm is defined as $\|\boldsymbol{x}\|_\infty := \max_{i \in [n]} |x_i|$. We will most often work with ℓ_2, i.e. with the Euclidean norm, which we simply write as $\|\boldsymbol{x}\|$.

Call a matrix $O \in \mathbb{R}^{m \times n}$ *orthogonal* if $\|O\boldsymbol{x}\| = \|\boldsymbol{x}\|$ for every $\boldsymbol{x} \in \mathbb{R}^n$. Equivalently, O is orthogonal if $O^T O = I_n$, where I_n is the $n \times n$ identity matrix.

3.2.1.1 Gram-Schmidt Vectors

Given linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^m$, we define their *Gram-Schmidt orthogonalization* $\tilde{\boldsymbol{b}}_1, \ldots, \tilde{\boldsymbol{b}}_n$ as follows.

$$\tilde{\boldsymbol{b}}_1 := \boldsymbol{b}_1, \qquad \tilde{\boldsymbol{b}}_i := \boldsymbol{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \tilde{\boldsymbol{b}}_j \quad \text{for } 1 < i \le n, \tag{3.1}$$

where

$$\mu_{i,j} := \frac{\langle \boldsymbol{b}_i, \tilde{\boldsymbol{b}}_j \rangle}{\langle \tilde{\boldsymbol{b}}_j, \tilde{\boldsymbol{b}}_j \rangle}. \tag{3.2}$$

Given a matrix $B \in \mathbb{R}^{m \times n}$, we define its (reduced) *QR-decomposition* as $B = QR$, where $Q \in \mathbb{R}^{m \times n}$ is an orthogonal matrix, and $R \in \mathbb{R}^{n \times n}$ is an upper-triangular matrix. The matrices Q and R have a close correspondence to the Gram-Schmidt vectors; the QR-decomposition "writes the vectors $\boldsymbol{b}_i$ in the basis of the Gram-Schmidt vectors $\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_n$." The columns of Q are equal to the normalized Gram-Schmidt vectors, i.e., $\boldsymbol{q}_i = \tilde{\boldsymbol{b}}_i / \|\tilde{\boldsymbol{b}}_i\|$. The entries $r_{i,j}$ of R satisfy $r_{i,i} = \|\tilde{\boldsymbol{b}}_i\|$ for $1 \leq i \leq n$, and $r_{i,j} = \mu_{j,i} \cdot \|\tilde{\boldsymbol{b}}_i\|$ for $1 \leq i < j \leq n$.

3.2.1.2 Projection

We denote the *orthogonal projection* of a vector $\boldsymbol{x} \in \mathbb{R}^n$ onto a linear subspace $S \subseteq \mathbb{R}^n$ by $\pi_S(\boldsymbol{x})$. Given linearly independent vectors $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ and $0 \leq i \leq n$, we use the notation $\pi_i^{(B)}(\boldsymbol{x})$ to denote projection of $\boldsymbol{x}$ onto the subspace $\operatorname{span}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_i)^\perp$ (where we define $\pi_0^{(B)}$ to be the identity map). For example, the Gram-Schmidt vectors satisfy $\tilde{\boldsymbol{b}}_i = \pi_{i-1}^{(B)}(\boldsymbol{b}_i)$.

3.2.1.3 The Operator Norm and Condition Number

We define the *operator norm* of a full-rank matrix $A \in \mathbb{R}^{n \times n}$ as

$$\|A\| := \sup_{\boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}} \frac{\|A\boldsymbol{x}\|}{\|\boldsymbol{x}\|},$$

and the *condition number* of A as $\kappa(A) := \|A\| \|A^{-1}\|$. Alternatively, we may define $\|A\| = \sigma_1(A)$ and $\kappa(A) = \sigma_1(A)/\sigma_n(A)$, where $\sigma_1 \geq \cdots \geq \sigma_n$ denote the singular values of A. Because $\det(A) = \prod_{i=1}^n \sigma_i(A)$, it holds that $\kappa(A) \geq \|A\| / \det(A)^{1/n}$,

$$\begin{tikzcd}
 & \mathrm{GL}(n,\mathbb{R}) \arrow[dl, dash] \arrow[dr, dash] & \\
\text{Unimodular} = \mathrm{GL}(n,\mathbb{Z}) \arrow[d, dash] & & N(n,\mathbb{R}) \arrow[dl, dash] \\
\mathrm{SL}(n,\mathbb{Z}) \arrow[r, dash] & N(n,\mathbb{Z}) &
\end{tikzcd}$$

Figure 3.1: The partial ordering of certain subgroups of $\mathrm{GL}(n, \mathbb{R})$.

and in particular $\kappa(A) \geq \|A\|$ when $\det(A) \leq 1$.

3.2.1.4 Multiplicative Matrix Groups

We will consider several sets of $n \times n$ matrices which form groups with respect to matrix multiplication.

- The *general linear group* of matrices over the reals, $\mathrm{GL}(n, \mathbb{R})$. I.e., all real-valued invertible $n \times n$ matrices.

- *Unimodular* matrices, $\mathrm{GL}(n, \mathbb{Z})$. I.e., integer-valued $n \times n$ matrices with determinant ± 1.

- *Upper-triangular unipotent* matrices over the reals, $N(n, \mathbb{R})$. I.e., all upper-triangular, real-valued $n \times n$ matrices with 1s on the main diagonal. We also consider the subgroup of integer-valued matrices $N(n, \mathbb{Z})$.

3.2.2 Basic Lattice Definitions

In this section we review a number of standard definitions and facts about lattices. See the book by Micciancio and Goldwasser [MG02] and the notes by Regev [Reg09a] for comprehensive surveys about computational aspects of lattices, and also the notes by Dadush [Dad13] and Stephens-Davidowitz [Ste16] for useful expositions of select topics.

A lattice $\mathcal{L}$ is the set of all integer combinations of some linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^d$. We call the matrix B whose columns are $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ a *basis* of $\mathcal{L}$, and say that B *generates* $\mathcal{L}$. We write this as

$$\mathcal{L}(B) = \mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) := \Big\{ \sum_{i=1}^{n} a_i \boldsymbol{b}_i : a_i \in \mathbb{Z} \Big\}. \tag{3.3}$$

If a basis $B \in \mathbb{R}^{d \times n}$ generates a lattice $\mathcal{L}$, we say that $\mathcal{L}$ has *rank* n and *dimension* (or *ambient dimension*) d.

We next define several important geometric quantities of a lattice $\mathcal{L}$ of rank n and dimension d. Let $\|\boldsymbol{x}\|$ denote the Euclidean norm of a vector $\boldsymbol{x} \in \mathbb{R}^d$, and let $B_d(r)$ denote the closed Euclidean ball of radius r in d dimensions. For $1 \le i \le n$, we define the ith *successive minimum* of $\mathcal{L}$ as

$$\lambda_i(\mathcal{L}) := \min\{r \in \mathbb{R}^+ : \dim(\operatorname{span}(\mathcal{L} \cap B_d(r))) \ge i\}. \tag{3.4}$$

In particular, $\lambda_1(\mathcal{L})$ is the length of the shortest non-zero vector in $\mathcal{L}$.

Given a lattice $\mathcal{L}$, we define the *dual lattice* of $\mathcal{L}$ as $\mathcal{L}^* := \{x \in \operatorname{span}(\mathcal{L}) : \forall y \in \mathcal{L} \ \ \langle x, y \rangle \in \mathbb{Z}\}$. Given a basis $B \in \mathbb{R}^{d \times n}$ of $\mathcal{L}$, we define its *pseudo-inverse* as $B^+ := (B^T B)^{-1} B^T$, and its *dual basis* B^* as $B^* := (B^+)^T$ (note that we simply have $B^* = (B^{-1})^T$

when $d = n$). Given a basis B of $\mathcal{L}$, it holds that $\mathcal{L}^* = \mathcal{L}(B^*)$, i.e., that the dual basis generates the dual lattice. The following theorem of Banaszczyk relates the successive minima of a lattice to the successive minima of its dual.

Theorem 3.2.1 (Banaszczyk's Transference Theorem, [Ban93]). *For every lattice $\mathcal{L}$ of rank n and every $1 \le i \le n$, $1 \le \lambda_i(\mathcal{L}) \cdot \lambda_{n-i+1}(\mathcal{L}^*) \le n$.*

When the underlying lattice is clear from context, we write λ_i, λ_i^* to denote the ith successive minima of a lattice and its dual, respectively.

Let $\mathrm{dist}(\mathcal{L}, \boldsymbol{t}) := \min_{x \in \mathcal{L}} \|\boldsymbol{x} - \boldsymbol{t}\|$ denote the distance of $\boldsymbol{t}$ to $\mathcal{L}$. We define the *covering radius* of $\mathcal{L}$ as

$$\mu(\mathcal{L}) := \max_{\boldsymbol{x} \in \mathrm{span}(\mathcal{L})} \mathrm{dist}(\mathcal{L}, \boldsymbol{x}). \tag{3.5}$$

The following well-known bound relates the covering radius and successive minima of a lattice. See, e.g., [MG02].

Theorem 3.2.2. *For every lattice $\mathcal{L}$ of rank n, $\mu(\mathcal{L}) \le \frac{\sqrt{n}}{2} \cdot \lambda_n(\mathcal{L})$.*

The *determinant* of a lattice $\mathcal{L}$ with basis B is defined as $\det(\mathcal{L}) := \det(B^T B)^{1/2}$ (which is simply $|\det(B)|$ when B is full-rank). Any basis of $\mathcal{L}$ can be expressed as BU for some unimodular matrix U, so this quantity is well-defined.

3.2.3 Computational Lattice Problems

Below we define both search and decision variants of the two most important computational problems on lattices, the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP). We also define the Closest Vector Problem with Preprocessing (CVPP). For further details about standard computational lattice problems and a survey about rank-preserving reductions between them, see [Ste15]. The following definitions hold for any approximation factor $\gamma \geq 1$.

3.2.3.1 Search Problems

Definition 3.2.3. The γ-approximate Shortest Vector Problem (γ-SVP) is the search problem defined as follows. Given a lattice $\mathcal{L}$ (specified by a basis $B \in \mathbb{Q}^{d\times n}$) output a non-zero vector $\boldsymbol{v} \in \mathcal{L}$ such that $\|\boldsymbol{v}\| \leq \gamma \cdot \lambda_1(\mathcal{L})$.

Definition 3.2.4. The γ-approximate Closest Vector Problem (γ-CVP) is the search problem defined as follows. Given a lattice $\mathcal{L}$ (specified by a basis $B \in \mathbb{Q}^{d\times n}$) and a target vector $\boldsymbol{t} \in \mathbb{Q}^d$, output a vector $\boldsymbol{v} \in \mathcal{L}$ such that $\|\boldsymbol{v} - \boldsymbol{t}\| \leq \gamma \cdot \mathrm{dist}(\mathcal{L}, \boldsymbol{t})$.

Definition 3.2.5. The γ-approximate Closest Vector Problem with Preprocessing (γ-CVPP) is the problem of finding a preprocessing function P and an algorithm Q which work as follows. Given a lattice $\mathcal{L}$ (specified by a basis $B \in \mathbb{Q}^{d\times n}$, P outputs a new description of $\mathcal{L}$. Given $P(\mathcal{L})$ and a target vector $\boldsymbol{t} \in \mathbb{R}$, Q outputs a vector $\boldsymbol{v} \in \mathcal{L}$ such that $\|\boldsymbol{v} - \boldsymbol{t}\| \leq \gamma \cdot \mathrm{dist}(\mathcal{L}, \boldsymbol{t})$.

Often in CVPP the preprocessing algorithm P is required to output a description $P(\mathcal{L})$ of polynomial size. Note that a single preprocessing $P(\mathcal{L})$ of a lattice $\mathcal{L}$ can be used to answer multiple CVP queries on $\mathcal{L}$.

3.2.3.2 Decision Problems

Definition 3.2.6. The γ-approximate Gap Shortest Vector Problem (γ-GapSVP) is the decision problem defined as follows. The input is a lattice $\mathcal{L}$ (specified by a basis $B \in \mathbb{Q}^{d\times n}$) and a number $r > 0$. It is a 'YES' instance if $\lambda_1(\mathcal{L}) \leq r$, and a 'NO' instance if $\lambda_1(\mathcal{L}) > \gamma \cdot r$.

Definition 3.2.7. The γ-approximate Gap Closest Vector Problem (γ-GapCVP) is the decision problem defined as follows. The input is a lattice $\mathcal{L}$ (specified by a basis $B \in \mathbb{Q}^{d\times n}$), a target vector $\boldsymbol{t} \in \mathbb{Q}^d$, and a number $r > 0$. It is a 'YES' instance if $\operatorname{dist}(\mathcal{L}, \boldsymbol{t}) \le r$, and a 'NO' instance if $\operatorname{dist}(\mathcal{L}, \boldsymbol{t}) > \gamma \cdot r$.

When $\gamma > 1$, γ-GapSVP and γ-GapCVP are *promise problems.* An algorithm for solving a promise problem only needs to handle 'YES' and 'NO' instances correctly, and can have arbitrary behavior on other instances.

3.2.4 Basis Reduction

We review several standard notions of basis reduction, including Lenstra-Lenstra-Lovász-reduction (LLL-reduction) [LLL82] and Hermite-Korkine-Zolotareff-reduction (HKZ-reduction) [KZ73]. Basis reduction plays a key role in Chapters 4 and 5, and we discuss specific topics in more detail there. We note that two bases B and B' generate the same lattice if and only if there exists a unimodular matrix U such that $B' = BU$ (see, e.g., Lecture 1 in [Reg09a]), so one can view the task of basis reduction as finding a suitable unimodular matrix to right-multiply a basis by.

Call a lattice basis B *size-reduced* if $\mu_{i,j} \in [-\frac{1}{2}, \frac{1}{2})$ for all $i > j$ (where $\mu_{i,j}$ is as defined in Equation (3.2)).

Definition 3.2.8. A basis $B \in \mathbb{Q}^{n\times n}$ is *LLL-reduced* if

1. B is size-reduced,

2. For all $1 \le i \le n-1$, $\frac{3}{4}\|\tilde{\boldsymbol{b}}_i\|^2 \le \mu_{i+1,i}^2\|\tilde{\boldsymbol{b}}_i\|^2 + \|\tilde{\boldsymbol{b}}_{i+1}\|^2$.

Definition 3.2.9. A basis $B \in \mathbb{Q}^{n\times n}$ of a lattice $\mathcal{L}$ is *HKZ-reduced* if

1. $\|\boldsymbol{b}_1\| = \lambda_1(\mathcal{L})$,

2. B is size-reduced,

3. If $n > 1$ then $(\pi_1^{(B)}(\boldsymbol{b}_2), \ldots, \pi_1^{(B)}(\boldsymbol{b}_n))$ is an HKZ-reduced basis of $\pi_1(\mathcal{L})$.

The seminal LLL algorithm of [LLL82] gave a polynomial-time algorithm for computing LLL-reduced bases B which one can use to approximate many lattice problems. In particular, using the definition of an LLL-reduced basis together with the fact that $\lambda_1(\mathcal{L}) \geq \min_{i \in [n]} \|\tilde{\boldsymbol{b}}_i\|$ for any basis, it is straightforward to check that the first vector $\boldsymbol{b}_1$ in an LLL-reduced basis satisfies $\|\boldsymbol{b}_1\| \leq 2^{n/2} \cdot \lambda_1(\mathcal{L}(B))$. Therefore the LLL-algorithm gives a polynomial-time algorithm for $2^{n/2}$-SVP. Moreover, Babai [Bab86] showed how to extend this to a polynomial-time algorithm for solving $2^{n/2}$-CVP (see Section 3.3.1 for an outline of his algorithm).

HKZ-reduced bases give the strong guarantee that their first vector $\boldsymbol{b}_1$ is a shortest non-zero vector in the lattice. However, because exact SVP is NP-hard (under randomized reductions) [Ajt98], computing HKZ-reduced bases is intractable. In fact, as the definition indicates, computing an HKZ-reduced basis essentially amounts to solving n instances of SVP.

A natural question is whether it is possible to interpolate between LLL-reduced bases and HKZ-reduced bases, i.e., to get a trade-off between the running time and quality of lattice bases. Schnorr [Sch87] introduced Block Korkine-Zolotareff-reduced (BKZ-reduced) bases to address this question. The idea behind BKZ-reduced bases is that, although HKZ-reduced bases are intractable to compute in general, one can form a basis out of HKZ-reduced "blocks" of size k for some $k \leq n$. Indeed, using a $2^{O(n)}$-time algorithm for SVP (such as [MV13]), one can compute HKZ-reduced blocks of size $k = O(\log n)$ in polynomial time.

In [GN08], Gama and Nguyen defined *slide-reduced* bases, which refine the idea of BKZ-reduction. Slide-reduced bases play an important role in time-approximation quality trade-offs that arise in Chapters 4 and 5.

3.3 Relating Fundamental Domains, CVP(P) Algorithms, and Basis Reduction

In this section we describe a connection between fundamental domains of a lattice, algorithms for CVP and CVPP, and basis reduction which we summarize in Table 3.1 and Figure 3.2.

Applying size-reduction to a basis B is the standard way to make its vectors short while preserving its Gram-Schmidt vectors. In Chapters 4 and 5 we use two other such notions of "Gram-Schmidt preserving basis reductions." We build context for these notions of basis reduction here by describing their connections to fundamental domains and CVP(P) algorithms.

Given a lattice $\mathcal{L}$, a convex set $F \subseteq \text{span}(\mathcal{L})$ is a *fundamental domain* of $\mathcal{L}$ if (1) F is $\mathcal{L}$*-packing*, i.e., $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{L}, \boldsymbol{x} \neq \boldsymbol{y}, (F + \boldsymbol{x}) \cap (F + \boldsymbol{y}) = \emptyset$, and (2) F is $\mathcal{L}$*-covering*, i.e., $\mathcal{L} + F = \text{span}(\mathcal{L})$. See Lecture 3 in [Dad13] for a more thorough exposition.

In other words, a fundamental domain partitions $\text{span}(\mathcal{L})$ into disjoint regions $F + \boldsymbol{x}$ according to the vectors $\boldsymbol{x} \in \mathcal{L}$. This observation leads to a natural decoding-based class of algorithms for finding close lattice vectors. Namely, such an algorithm consists of first computing a fundamental domain F of $\mathcal{L}$ from some family of fundamental domains, and second decoding the target vector $\boldsymbol{t}$ to the unique lattice vector $\boldsymbol{x}$ satisfying $\boldsymbol{t} \in F + \boldsymbol{x}$.

One can naturally view this class of algorithms as solving CVPP. Indeed, one

Fundamental domain	CVP(P) algorithm	Basis reduction
Basis-induced box	Nearest plane	Size-reduction
Basis-induced parallelepiped	Rounding off	Seysen-reduction
Voronoi cell	Iterative slicer, MV-algorithm	CVP-reduction

Table 3.1: The correspondence between fundamental domains, CVP(P) algorithms, and basis reduction techniques.

may view computing F as the preprocessing step P, and the decoding step as the algorithm Q described in the definition of CVPP (Definition 3.2.5). However, an algorithm in this framework can also be used to solve CVP by first computing F and then applying the decoding algorithm. Several important algorithms for finding close lattice vectors fall into this framework, and relate to basis reduction.

We next describe a connection between CVP and basis reduction. We say that a basis reduction algorithm is *Gram-Schmidt preserving* if, on input a basis B, it outputs a basis B' satisfying $\mathcal{L}(B) = \mathcal{L}(B')$ and $\tilde{\boldsymbol{b}}_i = \tilde{\boldsymbol{b}}'_i$ for every i. Equivalently, an algorithm is Gram-Schmidt preserving if on input B it computes a basis $B' = BU$ for some $U \in N(n, \mathbb{Z})$. Note that this differs from general basis reduction in which case U is only required to be unimodular.

Given a basis $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ with such a "fixed" sequence of Gram-Schmidt vectors $\tilde{\boldsymbol{b}}_1, \ldots, \tilde{\boldsymbol{b}}_n$, one may view the problem of reducing $\boldsymbol{b}_i$ for $i = n, \ldots, 2$ as solving an instance of CVP on the lattice $\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{i-1})$ with target vector $\boldsymbol{b}_i$.

In the following subsections, we describe three fundamental domains and their connection to CVP and basis reduction.

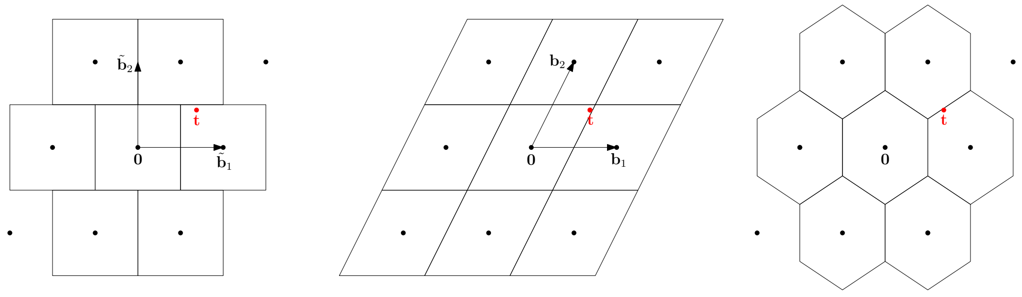


Figure 3.2: Three fundamental domains associated with a lattice $\mathcal{L}$ and its basis $(\boldsymbol{b}_1, \boldsymbol{b}_2)$ tiling space near the origin: the box induced by $\tilde{\boldsymbol{b}}_1, \tilde{\boldsymbol{b}}_2$ (left), the parallelepiped induced by $\boldsymbol{b}_1, \boldsymbol{b}_2$ (center), and the Voronoi cell of the lattice $\mathcal{V}(\mathcal{L})$ (right). The same target vector $\boldsymbol{t}$ appears in red in all three diagrams. Decoding to a lattice point according to the box (left) and the Voronoi cell (right) gives the correct closest vector of $\boldsymbol{b}_1$, while decoding acording to the parallelepiped (center) incorrectly gives $\mathbf{0}$.

3.3.1 Basis-induced Boxes

Babai's nearest plane algorithm for approximately solving CVP [Bab86] works as follows. On an input lattice $\mathcal{L}$ and target vector $\boldsymbol{t}$, it first computes a "good" basis B of $\mathcal{L}$ with Gram-Schmidt orthogonalization $\tilde{\boldsymbol{b}}_1, \ldots, \tilde{\boldsymbol{b}}_n$. Then, after initializing $\boldsymbol{s} := \boldsymbol{t}$, it computes the updates

$$\boldsymbol{s} := \boldsymbol{s} - \left\lfloor \frac{\langle \boldsymbol{s}, \tilde{\boldsymbol{b}}_i \rangle}{\langle \tilde{\boldsymbol{b}}_i, \tilde{\boldsymbol{b}}_i \rangle} \right\rceil \boldsymbol{b}_i$$

for $i = n, \ldots, 1$. Finally, it outputs $\boldsymbol{t} - \boldsymbol{s}$, which is in $\mathcal{L}$ since each update to $\boldsymbol{s}$ consists of addition by lattice vectors.

We note that for $j < i$ the update $\boldsymbol{s} - \left\lfloor \frac{\langle \boldsymbol{s}, \tilde{\boldsymbol{b}}_j \rangle}{\langle \tilde{\boldsymbol{b}}_j, \tilde{\boldsymbol{b}}_j \rangle} \right\rceil \boldsymbol{b}_j$ does not affect the ith coordinate of $\boldsymbol{s}$. Therefore, after all of the udpates, the ith coordinate of $\boldsymbol{s}$ is less than or equal to $\frac{1}{2}\|\tilde{\boldsymbol{b}}_i\|$ for all $1 \leq i \leq n$. I.e., $\langle \boldsymbol{s}, \tilde{\boldsymbol{b}}_i \rangle / \langle \tilde{\boldsymbol{b}}_i, \tilde{\boldsymbol{b}}_i \rangle \leq \frac{1}{2}$ for all $1 \leq i \leq n$.

It therefore holds that $\boldsymbol{s}$ lies in the box

$$(\tilde{\boldsymbol{b}}_1, \dots, \tilde{\boldsymbol{b}}_n) \cdot \big[-\frac{1}{2}, \frac{1}{2}\big)^n = \Big\{\sum_{i=1}^{n} a_i \tilde{\boldsymbol{b}}_i : -\frac{1}{2} \le a_i < \frac{1}{2}\Big\}. \tag{3.6}$$

This box is a fundamental domain of $\mathcal{L}$, and the vector output by Babai's algorithm is the unique lattice vector $\boldsymbol{x}$ such that $\boldsymbol{t} \in F + \boldsymbol{x}$. Therefore, Babai's algorithm fits into the "decoding from a fundamental domain" based framework for solving CVPP.

We note the direct correspondence between Babai's algorithm and size-reduction of a lattice basis. Indeed, the size-reduction algorithm is as follows. Given a basis B with Gram-Schmidt orthogonalization $\tilde{\boldsymbol{b}}_1, \dots, \tilde{\boldsymbol{b}}_n$, for $i = n, \dots, 2$, apply Babai's rounding algorithm with basis $\boldsymbol{b}_1, \dots, \boldsymbol{b}_{i-1}$ to $\boldsymbol{t} := \boldsymbol{b}_i$. Furthermore, both of these algorithms correspond to shifting a target vector into the box $(\tilde{\boldsymbol{b}}_1, \dots, \tilde{\boldsymbol{b}}_n) \cdot \big[-\frac{1}{2}, \frac{1}{2}\big)^n$ by adding lattice vectors.

3.3.2 Basis-induced Parallelepipeds

Given a basis B of $\mathcal{L}$, one may try an even simpler algorithm for finding a close lattice vector to $\boldsymbol{t}$ than Babai's algorithm. Namely, one can simply write $\boldsymbol{t}$ in the basis B and round each coefficient. I.e., one can output $\boldsymbol{x} := \sum_{i=1}^n \lfloor a_i \rceil \boldsymbol{b}_i$, where the coefficients a_i are uniquely defined by $\boldsymbol{t} = \sum_{i=1}^n a_i \boldsymbol{b}_i$. Although the approximation factor is worse than the nearest plane algorithm (which achieves an approximation factor of $2^{n/2}$ when B is an LLL-reduced basis), this simple algorithm still achieves a $2^{O(n)}$ approximation factor when B is an LLL-reduced basis. Indeed, Babai analyzed this algorithm alongside the nearest plane algorithm in his original work [Bab86], where he called it the "rounding off" algorithm.

Subtracting $\boldsymbol{x}$ from $\boldsymbol{t}$ amounts to shifting $\boldsymbol{t}$ into the parallelepiped induced by the basis B, namely

$$B \cdot \big[-\frac{1}{2}, \frac{1}{2}\big)^n = \Big\{ \sum_{i=1}^{n} a_i \boldsymbol{b}_i : -\frac{1}{2} \leq a_i < \frac{1}{2} \Big\}, \tag{3.7}$$

which is again a fundamental domain of the lattice.

Size-reduction is the standard technique for reducing a lattice basis with respect to a fixed sequence of Gram-Schmidt vectors. However, as we shall see in Chapters 4 and 5, shifting a vector into a parallelepiped corresponds to a notion of basis reduction introduced and analyzed by Seysen [Sey93], which ensures that both a basis B and its dual basis B^* contain short vectors simultaneously.

3.3.3 Lattice Voronoi Cells

The *Voronoi cell* $\mathcal{V}(\mathcal{L})$ of a lattice $\mathcal{L}$ is the set of points in $\operatorname{span}(\mathcal{L})$ that lie at least as close to the origin as to any other lattice point. Namely,

$$\mathcal{V}(\mathcal{L}) := \{\boldsymbol{x} \in \operatorname{span}(\mathcal{L}) : \forall \boldsymbol{y} \in \mathcal{L} \;\; \|\boldsymbol{x}\| \leq \|\boldsymbol{y} - \boldsymbol{x}\|\}. \tag{3.8}$$

By definition, deciding whether $\mathbf{0}$ is the closest lattice point to some $\boldsymbol{t} \in \operatorname{span}(\mathcal{L})$ is equivalent to deciding whether $\boldsymbol{t} \in \mathcal{V}(\mathcal{L})$. More generally, it is not hard to check that a shift $\mathcal{V}(\mathcal{L}) + \boldsymbol{x}$ of the Voronoi cell with $\boldsymbol{x} \in \mathcal{L}$ corresponds to the set of vectors closest to $\boldsymbol{x}$, and therefore that these shifts partition $\operatorname{span}(\mathcal{L})$ into sets of vectors closest to each lattice point. It follows that $\mathcal{V}(\mathcal{L})$ is a fundamental domain of the lattice, and that (exact, search) CVP corresponds to finding $\boldsymbol{x} \in \mathcal{L}$ such that $\boldsymbol{t} \in \mathcal{V}(\mathcal{L}) + \boldsymbol{x}$.[1]

Of course, getting an algorithm for CVP from this characterization requires specifying how $\mathcal{V}(\mathcal{L})$ is represented, and how one checks in which shift $\mathcal{V}(\mathcal{L}) + \boldsymbol{x}$ the target vector $\boldsymbol{t}$ lies. The first scheme for this came from the "iterative slicer" algorithm of Sommer et al. [SFS09]. Micciancio and Voulgaris [MV13] improved on this work to get an $\widetilde{O}(4^n)$-time algorithm for CVPP (and, because the preprocessing also requires $\widetilde{O}(4^n)$-time to compute, CVP as well).[2] This was the first singly-exponential time algorithm for CVP, and remains the fastest deterministic algorithm. In follow-up work, Bonifas and Dadush [DB15] gave a $\widetilde{O}(2^n)$-time Las Vegas algorithm for CVPP via the so-called "randomized straight line" algorithm.

There exists a related, natural notion of basis reduction which captures the "shortest possible" lattice basis with respect to a fixed sequence of Gram-Schmidt vectors. We define a basis $B = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_n)$ to be *CVP-reduced* if for every $1 < i \leq n$ it holds that $\|\boldsymbol{b}_i\| = \min_{\boldsymbol{x} \in \mathcal{L}} \{\|\boldsymbol{x}\| : \pi_{i-1}^{(B)}(\boldsymbol{x}) = \tilde{\boldsymbol{b}}_i\}$. Let $\mathrm{CVP}(\mathcal{L}, \boldsymbol{t})$ denote the closest vector to $\boldsymbol{t}$ in $\mathcal{L}$. To CVP-reduce a basis, it suffices to set $\boldsymbol{b}_i := \boldsymbol{b}_i - \mathrm{CVP}(\mathcal{L}(\boldsymbol{b}_1, \dots, \boldsymbol{b}_{i-1}), \boldsymbol{b}_i)$ for $i = 2, \dots, n$. This exactly corresponds to shifting $\pi_{\mathrm{span}(\boldsymbol{b}_1, \dots, \boldsymbol{b}_{i-1})}(\boldsymbol{b}_i)$ into $\mathcal{V}(\mathcal{L}(\boldsymbol{b}_1, \dots, \boldsymbol{b}_{i-1}))$.

Helfrich used the notion of CVP-reduction in her algorithm for computing Minkowski-reduced bases [Hel85], where she called it "correctly deprojecting" lattice vectors. We use it in our algorithm for computing bases with minimal orthogonality defect in Section 5.3. (See Chapter 5 for definitions and Section 5.3 for details of our algorithm.)

[1] Any pair of shifts $\mathcal{V}(\mathcal{L}) + \boldsymbol{x}$, $\mathcal{V}(\mathcal{L}) + \boldsymbol{y}$ with $\boldsymbol{x} \neq \boldsymbol{y}$ are interior-disjoint, but are not necessarily disjoint. So, technically we need to define a "half open" version of the Voronoi cell for it to be a fundamental domain. However, for simplicity we disregard this issue, and note that such non-empty intersections $\mathcal{V}(\mathcal{L}) + \boldsymbol{x} \cap \mathcal{V}(\mathcal{L}) + \boldsymbol{y}$ have a useful interpretation as sets of points with multiple closest lattice vectors.

[2] Recall that the $\widetilde{O}$ notation suppresses polylogarithmic factors in the argument.

Chapter 4

On The Lattice Distortion Problem

This chapter is based on the publication [BDS16], which was joint work with Daniel Dadush and Noah Stephens-Davidowitz.

4.1 Introduction

In this chapter we address a basic question: how "similar" are two lattices? We formalize this question in a natural way for studying the similarity of two geometric objects, namely, in terms of the minimum distortion of a mapping between them. I.e., given lattices $\mathcal{L}_1, \mathcal{L}_2$ does there exist a bijective linear mapping $T : \mathcal{L}_1 \to \mathcal{L}_2$ that nearly preserves distances between points? If we insist that T exactly preserves distances, then this is the *Lattice Isomorphism Problem* (LIP), which was studied in [PS97, SSV09, HR14, LS14]. We extend this study to the *Lattice Distortion Problem* (LDP), which asks how well such a mapping T can *approximately* preserve distances between points.

Given two lattices $\mathcal{L}_1, \mathcal{L}_2 \subseteq \mathbb{R}^n$, we define the *distortion* between them as

$$D(\mathcal{L}_1, \mathcal{L}_2) := \min_{T \in \mathbb{R}^{n \times n}} \{\|T\|\|T^{-1}\| : T(\mathcal{L}_1) = \mathcal{L}_2\},$$

where $\|T\| := \sup_{\boldsymbol{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}} \|T\boldsymbol{x}\|/\|\boldsymbol{x}\|$ is the *operator norm.* The quantity $\kappa(T) := \|T\|\|T^{-1}\|$ is the *condition number* of T, which measures how much T distorts distances. It is easy to see that distortion is invariant under scaling of the lattices, i.e., $D(\mathcal{L}_1, \mathcal{L}_2) = D(c_1\mathcal{L}_1, c_2\mathcal{L}_2)$ for all $c_1, c_2 > 0$. $D(\mathcal{L}_1, \mathcal{L}_2)$ bounds the ratio between most natural geometric parameters of $\mathcal{L}_1$ and $\mathcal{L}_2$ (up to scaling), and hence $D(\mathcal{L}_1, \mathcal{L}_2)$ is a strong measure of "similarity" between lattices. In particular, $\mathrm{dist}(\mathcal{L}_1, \mathcal{L}_2) = 1$ if and only if $\mathcal{L}_1, \mathcal{L}_2$ are isomorphic (i.e., if and only if they are related by a scaled orthogonal transformation).

The *Lattice Distortion Problem* LDP is then defined in the natural way. Namely, the input is a pair of lattices $\mathcal{L}_1, \mathcal{L}_2$ represented by bases, and the goal is to compute a bijective linear transformation T mapping $\mathcal{L}_1$ to $\mathcal{L}_2$ such that $\kappa(T) = D(\mathcal{L}_1, \mathcal{L}_2)$. In this work we study the approximate search and decision versions of this problem, which we refer to as γ-LDP and γ-GapLDP, respectively, for some approximation factor $\gamma = \gamma(n) \geq 1$. (See Section 4.2.3 for precise definitions.)

4.1.1 Our Contribution

As our first main contribution, we show that the distortion between any two lattices can be approximated by a natural function of geometric lattice parameters. Indeed, our proof techniques are constructive, leading to our second main contribution: an algorithm that computes low-distortion mappings, with a trade-off between the running time and the approximation factor. Finally, we show hardness of approximating lattice distortion.

A natural way to derive useful bounds on the distortion between two lattices is to study the "different scales over which the two lattices live." A natural notion of this is given by the successive minima. Since low-distortion mappings approximately preserve distances, it is intuitively clear that two lattices can only be related by a low-distortion mapping if their successive minima are close to each other (up to a fixed scaling).

Concretely, for two n-dimensional lattices $\mathcal{L}_1, \mathcal{L}_2$, we define

$$M(\mathcal{L}_1, \mathcal{L}_2) = \max_{i \in [n]} \frac{\lambda_i(\mathcal{L}_2)}{\lambda_i(\mathcal{L}_1)}, \tag{4.1}$$

which measures how much we need to scale up $\mathcal{L}_1$ so that its successive minima are at least as large as those of $\mathcal{L}_2$. For any linear map T from $\mathcal{L}_1$ to $\mathcal{L}_2$ and any $1 \leq i \leq n$, it is not hard to show that $\lambda_i(\mathcal{L}_2) \leq \|T\| \lambda_i(\mathcal{L}_1)$. Thus, by definition $M(\mathcal{L}_1, \mathcal{L}_2) \leq \|T\|$. Applying the same reasoning for T^{-1}, we derive the following simple lower bound on distortion.

$$D(\mathcal{L}_1, \mathcal{L}_2) \geq M(\mathcal{L}_1, \mathcal{L}_2) \cdot M(\mathcal{L}_2, \mathcal{L}_1). \tag{4.2}$$

We note that this lower bound is tight when $\mathcal{L}_1, \mathcal{L}_2$ are each generated by bases of orthogonal vectors. But, it is a priori unclear if any comparable upper bound should hold for general lattices, since the successive minima are a coarse characterization of the geometry of the lattice. Nevertheless, we show a corresponding upper bound.

Theorem 4.1.1. *Let $\mathcal{L}_1, \mathcal{L}_2$ be n-dimensional lattices. Then,*

$$M(\mathcal{L}_1, \mathcal{L}_2) \cdot M(\mathcal{L}_2, \mathcal{L}_1) \le D(\mathcal{L}_1, \mathcal{L}_2) \le n^{O(\log n)} \cdot M(\mathcal{L}_1, \mathcal{L}_2) \cdot M(\mathcal{L}_2, \mathcal{L}_1).$$

While the factor on the right-hand side of the theorem might be far from optimal, we show in Section 4.5.1 that it cannot be improved below $\Omega(\sqrt{n})$. This is because there exist lattices that are much more dense than $\mathbb{Z}^n$ over large scales but still have $\lambda_i(\mathcal{L}) = \Theta(1)$ for all i. I.e., there exist very dense lattice sphere packings (see, e.g., [Sie45]).

To prove the above theorem, we make use of the intuition that a low-distortion mapping T from $\mathcal{L}_1$ to $\mathcal{L}_2$ should map a "short" basis B_1 of $\mathcal{L}_1$ to a "short" basis B_2 of $\mathcal{L}_2$. (Note that the condition $TB_1 = B_2$ completely determines $T = B_2 B_1^{-1}$.) The difficulty here is that standard notions of "short" fail for the purpose of capturing low-distortion mappings. In particular, in Section 4.5.2, we show that Hermite-Korkine-Zolotarev (HKZ) reduced bases, one of the strongest notions of "shortest possible" lattice bases, do not suffice by themselves for building low-distortion mappings. (See Section 4.2.5 for the definition of HKZ-reduced bases.) In particular, we give a simple example of a lattice $\mathcal{L}$ where an HKZ-reduced basis of $\mathcal{L}$ misses the optimal distortion $D(\mathbb{Z}^n, \mathcal{L})$ by an exponential factor.

Fortunately, we show that a suitable notion of shortness does exist for building low-distortion mappings by making a novel connection between low-distortion mappings and a notion of basis reduction introduced by Seysen [Sey93]. In particular, for a basis $B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$ and dual basis $B^* = B^{-T} = [\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*]$, Seysen's condition number is defined as

$$S(B) = \max_{i \in [n]} \|\boldsymbol{b}_i\| \|\boldsymbol{b}_i^*\| \ .$$

Note that we always have $\langle \boldsymbol{b}_i, \boldsymbol{b}_i^* \rangle = 1$, so S measures how tight the Cauchy-Schwarz inequality is over all primal-dual basis-vector pairs. Another way of viewing $S(B)$ is as a condition number of B which is invariant under scaling of the columns (i.e. basis vectors) of B. We extend this notion and define $S(\mathcal{L})$ as the minimum of $S(B)$ over all bases B of $\mathcal{L}$. Call a basis $B = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_n]$ *sorted* if $\|\boldsymbol{b}_1\| \le \cdots \le \|\boldsymbol{b}_n\|$. Using this notion, we obtain the bounds in Theorem 4.1.1 as follows.

Theorem 4.1.2. *Let $\mathcal{L}_1, \mathcal{L}_2$ be n-dimensional lattices with sorted bases $B_1, B_2 \in \mathbb{R}^{n \times n}$, respectively. Then we have that*

$$M(\mathcal{L}_1, \mathcal{L}_2) M(\mathcal{L}_2, \mathcal{L}_1) \le \kappa(B_2 B_1^{-1}) \le n^2 S(B_1)^2 S(B_2)^2 \cdot M(\mathcal{L}_1, \mathcal{L}_2) M(\mathcal{L}_2, \mathcal{L}_1).$$

In particular, we have that

$$M(\mathcal{L}_1, \mathcal{L}_2) M(\mathcal{L}_2, \mathcal{L}_1) \le D(\mathcal{L}_1, \mathcal{L}_2) \le n^2 S(\mathcal{L}_1)^2 S(\mathcal{L}_2)^2 \cdot M(\mathcal{L}_1, \mathcal{L}_2) M(\mathcal{L}_2, \mathcal{L}_1).$$

From here, the bound in Theorem 4.1.1 follows directly from the following theorem of Seysen.

Theorem 4.1.3 (Seysen [Sey93])**.** *For any $\mathcal{L} \subset \mathbb{R}^n$, $S(\mathcal{L}) \le n^{O(\log n)}$.*

This immediately yields an algorithm for approximating the distortion between two lattices, by using standard lattice algorithms to approximate $M(\mathcal{L}_1, \mathcal{L}_2)$ and $M(\mathcal{L}_2, \mathcal{L}_1)$. But, Seysen's proof of the above theorem is actually constructive. In particular, he shows how to efficiently convert any suitably reduced lattice basis into a basis with a low Seysen condition number. (See Section 4.2.5.2 for details.) Using this methodology, combined with standard basis reduction techniques, we derive the following time-approximation trade-off for γ-LDP.

Theorem 4.1.4 (Algorithm for LDP)**.** *For any* $\log n \le k \le n$*, there is an algorithm solving* $k^{O(n/k + \log n)}$*-LDP in* $2^{O(k)}$*-time.*

In other words, using the bounds in Theorem 4.1.1 together with known algorithms, we are able to approximate the distortion between two lattices. But, because Theorem 4.1.3 is effective, we are able to solve *search* LDP by explicitly computing a low-distortion mapping between the input lattices.

We also prove the following lower bound for LDP.

Theorem 4.1.5 (Hardness of LDP)**.** γ*-GapLDP is NP-hard under randomized polynomial-time reductions for any constant* $\gamma \ge 1$*.*

In particular, we show a reduction from approximating the decisional Shortest Vector Problem (GapSVP) over lattices to γ-GapLDP, where the approximation factor that we obtain for GapSVP is $O(\gamma)$. Since hardness of GapSVP is quite well-studied [Ajt98, Mic01, Kho05, HR12], we are immediately able to import many hardness results to GapLDP. (See Corollary 4.4.5 and Theorem 4.4.7 for precise statements.)

4.1.2 Comparison to related work

The most related work to ours was the paper by Haviv and Regev [HR14] on the Lattice Isomorphism Problem (LIP). In their paper, they leverage the observation that an isomorphism from $\mathcal{L}_1$ to $\mathcal{L}_2$ must send shortest non-zero vectors of $\mathcal{L}_1$ to shortest non-zero vectors of $\mathcal{L}_2$ (and vice-versa) to get an $n^{O(n)}$-time algorithm for solving LIP.

LDP appears as the natural generalization of LIP, in which the goal is to compute a low-distortion mapping between the lattices, but which need not preserve distances exactly. I.e., the goal of LDP is to compute a mapping which approximately preserves distances.

One might expect that the approach of Haviv and Regev should also work for the purpose of solving LDP either exactly or for approximation factors below $n^{O(\log n)}$. However, the crucial assumption in LIP – that vectors in one lattice must be mapped to vectors of the same length in the other – breaks down in the current context, and we therefore do not know how to extend their techniques to LDP. On the other hand, our algorithm does use the intuition that vectors at "the same scale" should be mapped to each other as best as possible. Indeed, the mappings that our algorithm constructs are of the form $T = BA^{-1}$ for bases A, B (i.e. $T : \boldsymbol{a}_i \mapsto \boldsymbol{b}_i$ for $1 \leq i \leq n$), where the lengths of the basis vectors $\boldsymbol{a}_i, \boldsymbol{b}_i$ approximate the ith successive minima of $\mathcal{L}(A), \mathcal{L}(B)$, respectively. (Also, the lengths of $\boldsymbol{a}_i^*, \boldsymbol{b}_i^*$ approximate the $(n-i+1)$th successive minima of $\mathcal{L}(A)^*, \mathcal{L}(B)^*$, respectively.)

More generally, we note that LIP is the lattice analog of the Graph Isomorphism Problem (GI), and that indeed the problems are related. Both problems are in SZK but not known to be in P, and GI reduces to LIP [SSV09]. Therefore, LDP is analogous to the Approximate Graph Isomorphism Problem, which was studied by Arora, Frieze, and Kaplan [AFK02], who showed an upper bound, and Arvind, Köbler, Kuhnert, and Vasudev [AKKV12], who proved both upper and lower bounds. In particular, [AKKV12] showed that various versions of this problem are NP-hard to approximate to within a constant factor. Qualitatively, these hardness results are similar to our Theorem 4.1.5.

4.1.3 Open Problems

Our algorithm only works for solving γ-LDP with large approximation factors $\gamma = n^{\Omega(\log n)}$. The most obvious open problem is to find an algorithm which solves γ-LDP for smaller γ. We currently do not even know whether there exists a polynomial-time algorithm for γ-LDP on lattices of fixed rank n for any $\gamma = n^{o(\log n)}$.

Open Problem 4.1.6. *Find an algorithm for γ-LDP for some $\gamma = n^{o(\log n)}$ which runs in polynomial time for lattices of fixed rank n.*

The main problem with addressing Open Problem 4.1.6 is that we do not have any good characterization of nearly optimal distortion mappings between lattices.

Another major open problem is what the correct bound in Theorem 4.1.3 is. In particular, there are no known families of lattices for which the Seysen condition number is provably superpolynomial, and hence it is possible that $S(\mathcal{L}) = \mathrm{poly}(n)$ for every lattice $\mathcal{L}$ of rank n. Indeed, a better bound would immediately improve our Theorem 4.1.2 and give a better approximation factor for GapLDP, partially addressing Open Problem 4.1.6.

Open Problem 4.1.7. *Is $S(\mathcal{L}) = \mathrm{poly}(n)$ for every lattice $\mathcal{L}$ of rank n?*

Organization

In Section 4.2, we present necessary background material. In Section 4.3, we give our approximations for lattice distortion, proving Theorems 4.1.2 and 4.1.4. In

Section 4.4, we give the hardness for lattice distortion, proving Theorem 4.1.5. In Section 4.5, we give some illustrative example instances of lattice distortion.

Acknowledgments

We thank Oded Regev for pointing us to Seysen's paper and for many helpful conversations. The concise proof of Lemma 4.3.1, item 2 using the Transference Theorem is due to Michael Walter. We thank Paul Kirchner for telling us about Proposition 4.2.10, and for identifying a minor bug in an earlier version of [BDS16].

4.2 Preliminaries

We omit any mention of the bit length in the running time of our algorithms. In particular, all of our algorithms take as input vectors in $\mathbb{Q}^n$ and run in time $f(n) \cdot \mathrm{poly}(m)$ for some f, where m is the maximal bit length of an input vector. We therefore suppress the factor of $\mathrm{poly}(m)$.

4.2.1 Linear mappings between lattices

We next characterize linear mappings between lattices in terms of bases.

Lemma 4.2.1. *Let $\mathcal{L}_1, \mathcal{L}_2$ be full-rank lattices. Then a mapping $T : \mathcal{L}_1 \to \mathcal{L}_2$ is bijective and linear if and only if $T = BA^{-1}$ for some bases A, B of $\mathcal{L}_1, \mathcal{L}_2$ respectively. In particular, for any basis A of $\mathcal{L}_1$, $T(A)$ is a basis of $\mathcal{L}_2$.*

Proof. Let $T = BA^{-1}$ where $A = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n]$ and $B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$ are bases of $\mathcal{L}_1, \mathcal{L}_2$ respectively. We first show that such a mapping is a bijection from $\mathcal{L}_1$ to $\mathcal{L}_2$. Because T has full rank, it is injective as a mapping from $\mathbb{R}^n$ to $\mathbb{R}^n$, and it is therefore injective as a mapping from $\mathcal{L}_1$ to $\mathcal{L}_2$. To show that T is surjective, pick $\boldsymbol{w} \in \mathcal{L}_2$. It holds that $\boldsymbol{w} = \sum_{i=1}^n c_i \boldsymbol{b}_i$ for some $c_1, \ldots, c_n \in \mathbb{Z}$ since B is a basis of $\mathcal{L}_2$, and moreover $\sum_{i=1}^n c_i \boldsymbol{a}_i \in \mathcal{L}_1$ is a preimage of $\boldsymbol{w}$. Therefore, T is a bijection from $\mathcal{L}_1$ to $\mathcal{L}_2$.

We next show that any linear map T with $T(\mathcal{L}_1) = \mathcal{L}_2$ must have the form $T = BA^{-1}$. Let $A = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n]$ be a basis of $\mathcal{L}_1$, and let $B = T(A)$. We claim that $B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$ is a basis of $\mathcal{L}_2$.

Let $\boldsymbol{w} \in \mathcal{L}_2$. Because T is a bijection between $\mathcal{L}_1$ and $\mathcal{L}_2$, there exists $\boldsymbol{v} \in \mathcal{L}_1$ such that $T(\boldsymbol{v}) = w$. Using the definition of a basis and the linearity of T,

$$\boldsymbol{w} = T(\boldsymbol{v}) = T\big(\sum_{i=1}^n c_i \boldsymbol{a}_i\big) = \sum_{i=1}^n c_i \boldsymbol{b}_i,$$

for some $c_1, \ldots, c_n \in \mathbb{Z}$. Because $\boldsymbol{w}$ was picked arbitrarily, it follows that B is a basis of $\mathcal{L}_2$. $\square$

4.2.2 Seysen's condition number $S(B)$

Seysen shows how to take any basis with relatively low multiplicative drop in its Gram-Schmidt vectors and convert it into a basis with relatively low $S(B) = \max_i \|\boldsymbol{b}_i\| \|\boldsymbol{b}_i^*\|$ [Sey93]. By combining this with Gama and Nguyen's slide reduction technique [GN08], we obtain the following result.

Theorem 4.2.2. *For every $\log n \leq k \leq n$ there exists an algorithm that takes a lattice $\mathcal{L}$ as input and computes a basis B of $\mathcal{L}$ with $S(B) \leq k^{O(n/k + \log k)}$ in time $2^{O(k)}$.*

In particular, applying Seysen's procedure to slide-reduced bases suffices. We include a proof of Theorem 4.2.2 and a high-level description of Seysen's procedure in Section 4.2.5.

4.2.3 The Lattice Distortion Problem

Definition 4.2.3. For any $\gamma = \gamma(n) \geq 1$, the γ-Lattice Distortion Problem (γ-LDP) is the search problem defined as follows. The input consists of two lattices $\mathcal{L}_1, \mathcal{L}_2$ (represented by bases $B_1, B_2 \in \mathbb{Q}^{n \times n}$). The goal is to output a matrix $T \in \mathbb{Q}^{n \times n}$ such that $T(\mathcal{L}_1) = \mathcal{L}_2$ and $\kappa(T) \leq \gamma \cdot \mathrm{dist}(\mathcal{L}_1, \mathcal{L}_2)$.

Definition 4.2.4. For any $\gamma = \gamma(n) \geq 1$, the γ-GapLDP is the promise problem defined as follows. The input consists of two lattices $\mathcal{L}_1, \mathcal{L}_2$ (represented by bases $B_1, B_2 \in \mathbb{Q}^{n \times n}$) and a number $c \geq 1$. The goal is to decide between a 'YES' instance where $\mathrm{dist}(\mathcal{L}_1, \mathcal{L}_2) \leq c$ and a 'NO' instance where $\mathrm{dist}(\mathcal{L}_1, \mathcal{L}_2) > \gamma \cdot c$.

4.2.4 Complexity of LDP

We show some basic facts about the complexity of GapLDP. First, we show that the Lattice Isomorphism Problem (LIP) corresponds to the special case of GapLDP where $c = 1$. LIP takes bases of $\mathcal{L}_1, \mathcal{L}_2$ as input and asks if there exists an orthogonal linear transformation O such that $O(\mathcal{L}_1) = \mathcal{L}_2$. Haviv and Regev [HR14] show that there exists an $n^{O(n)}$-time algorithm for LIP, and that LIP is in the complexity class SZK.

Lemma 4.2.5. *There is a polynomial-time reduction from LIP to* 1*-GapLDP.*

Proof. Let $\mathcal{L}_1, \mathcal{L}_2$ be an LIP instance. First check that $\det(\mathcal{L}_1) = \det(\mathcal{L}_2)$. If not, then output a trivial 'NO' instance of 1-GapLDP. Otherwise, map the LIP instance to the 1-GapLDP instance with the same input bases and $c = 1$. For any $T : \mathcal{L}_1 \to \mathcal{L}_2$, we must have $\det(T) = 1$. Recalling that $\det(T) = \prod_{i=1}^{n} \sigma_i(T)$ and

$\kappa(T) = \sigma_1(T)/\sigma_n(T)$ (where σ_i is the ith largest singular value), we therefore have that $\kappa(T) = 1$ if and only if $\|T\| = \|T^{-1}\| = 1$. So, this is a 'YES' instance of GapLDP if and only if $\mathcal{L}_1, \mathcal{L}_2$ are isomorphic. $\square$

Lemma 4.2.6. *1-GapLDP is in* NP.

Proof. Let $I = (\mathcal{L}_1, \mathcal{L}_2, c)$ be an instance of GapLDP (where $\mathcal{L}_1$, $\mathcal{L}_2$ are specified by bases A, B, respectively), and let s be the length of I. We will show that for a 'YES' instance, there is a mapping $T : \mathcal{L}_1 \to \mathcal{L}_2$ that requires at most poly(s) bits to specify and satisfies $\kappa(T) \leq c$. Assume without loss of generality that $\mathcal{L}_1, \mathcal{L}_2 \subseteq \mathbb{Z}^n$. Otherwise, scale the input lattices to achieve this at the expense of a factor s blow-up in input size.

Using the definition of singular values and the fact that $\mathcal{L}_1, \mathcal{L}_2 \subseteq \mathbb{Z}^n$,

$$1/\|T^{-1}\| = \sigma_n(T) \leq (|\det(\mathcal{L}_2)/\det(\mathcal{L}_1)|)^{1/n} \leq \max\{1, |\det(\mathcal{L}_2)/\det(\mathcal{L}_1)|\} \leq |\det(\mathcal{L}_2)| \, .$$

Therefore, to satisfy $\|T\|\|T^{-1}\| \leq c$, we must have that $|t_{ij}| \leq \|T\| \leq c/\|T^{-1}\| \leq c \cdot |\det(\mathcal{L}_2)|$ for each entry t_{ij} of T. By Cramer's rule, each entry of T will be an integer multiple of $\det(\mathcal{L}_1)^{-1}$, so we can assume without loss of generality that the denominator of each entry of T is $\det(\mathcal{L}_1)$.

Combining these bounds and applying Hadamard's inequality, we get that $|t_{ij}|$ takes at most

$$\log(c \cdot \det(\mathcal{L}_1) \cdot \det(\mathcal{L}_2)) \leq \log\left(c \cdot \prod_{i=1}^{n} \|\boldsymbol{a}_i\| \cdot \prod_{i=1}^{n} \|\boldsymbol{b}_i\|\right)$$

bits to specify. Accounting for the sign of each t_{ij}, it follows that T takes at most $n^2 \cdot \log(2c \cdot \prod_{i=1}^{n} \|\boldsymbol{a}_i\|\|\boldsymbol{b}_i\|) \leq n^2 \cdot (s+1)$ bits to specify. $\square$

We remark that we can replace c with the quantity $n^{O(\log n)} \cdot M(\mathcal{L}_1, \mathcal{L}_2) M(\mathcal{L}_2, \mathcal{L}_1)$ (as given by the upper bound in Theorem 4.1.1) in the preceding argument to obtain an upper bound on the number of bits needed to represent an *optimal* mapping T.

4.2.5 Basis reduction

4.2.5.1 Slide-reduced bases

Gama and Nguyen (building on the work of Schnorr [Sch87]) introduced the notion of slide-reduced bases [GN08], which can be thought of as a relaxed notion of HKZ bases that can be computed more efficiently.

Definition 4.2.7 ([GN08, Definition 1])**.** Let B be a basis of $\mathcal{L} \subset \mathbb{Q}^n$ and $\varepsilon > 0$. We say that B is ε-DSVP (dual SVP) reduced if its corresponding dual basis $[\boldsymbol{b}_1^*, \dots, \boldsymbol{b}_n^*]$ satisfies $\|\boldsymbol{b}_n^*\| \leq (1 + \varepsilon) \cdot \lambda_1(\mathcal{L}^*)$. Then, for an integer $k \geq 2$ which divides n, we say that $B = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_n]$ is (ε, k)-slide reduced if

1. B is size-reduced;
2. $\forall\, 0 \leq i \leq n/k - 1$, the "projected truncated basis" $[\pi_{ik+1}^{(B)}(\boldsymbol{b}_{ik}), \dots, \pi_{ik}^{(B)}(\boldsymbol{b}_{ik+k})]$ is HKZ reduced; and
3. $\forall\, 0 \leq i \leq n/k - 2$, the "shifted projected truncated basis" $[\pi_{ik+1}^{(B)}(\boldsymbol{b}_{ik+2}), \dots, \pi_{ik+1}^{(B)}(\boldsymbol{b}_{ik+k+1})]$ is ε-DSVP reduced.

Theorem 4.2.8 ([GN08])**.** *There is an algorithm that takes as input a lattice $\mathcal{L} \subset \mathbb{Q}^n$, $\varepsilon > 0$, and an integer $k \geq \log n$ which divides n and outputs a (k, ε)-slide-reduced basis of $\mathcal{L}$ in time* $\mathrm{poly}(1/\varepsilon) \cdot 2^{O(k)}$.

We are primarily concerned with the ratios between the length of the Gram-Schmidt vectors of a given basis. We prefer bases whose Gram-Schmidt vectors do not "decay too quickly," and we measure this decay by

$$\eta(B) := \max_{i \le j} \frac{\|\tilde{\boldsymbol{b}}_i\|}{\|\tilde{\boldsymbol{b}}_j\|} .$$

Previous work bounded $\eta(B)$ for HKZ-reduced bases as follows.

Theorem 4.2.9 ([LLS90, Proposition 4.2])**.** *For any HKZ-reduced basis B over $\mathbb{Q}^n$, $\eta(B) \le n^{O(\log n)}$.*

We now use Theorem 4.2.9 and some of the results in [GN08] to bound $\eta(B)$ of slide-reduced bases.

Proposition 4.2.10. *For any integer $k \ge 2$ dividing n, if B is an $(1/n, k)$-slide-reduced basis for a lattice $\mathcal{L} \subset \mathbb{Q}^n$, then $\eta(B) \le k^{O(n/k + \log k)}$.*

Proof. We collect three simple inequalities that will together imply the result. First, from [GN08, Eq. (16)], we have $\|\tilde{\boldsymbol{b}}_1\| \le k^{O(n/k)} \cdot \|\tilde{\boldsymbol{b}}_{jk+1}\|$ for all $0 \le j \le n/k - 1$. Noting that the projection $[\pi_{ik}(\boldsymbol{b}_{ik+1}), \ldots, \pi_{ik}(\boldsymbol{b}_{ik+k})]$ of a slide-reduced basis is also slide reduced, we see that

$$\|\tilde{\boldsymbol{b}}_{ik+1}\| \le k^{O(n/k)} \cdot \|\tilde{\boldsymbol{b}}_{jk+1}\| , \tag{4.3}$$

for all $0 \le i \le j \le n/k - 1$. Next, from Theorem 4.2.9 and the fact that the "projected truncated bases" are HKZ reduced, we have that

$$\|\tilde{\boldsymbol{b}}_{ik+\ell}\| \le k^{O(\log k)} \cdot \|\tilde{\boldsymbol{b}}_{ik+\ell'}\| , \tag{4.4}$$

for all $1 \le \ell \le \ell' \le k$. Finally, [GN08] observe that

$$\|\tilde{\boldsymbol{b}}_{ik+k}\| \le C \cdot \|\tilde{\boldsymbol{b}}_{ik+k+1}\| , \tag{4.5}$$

for all $0 \le i \le n/k - 2$, where $C > 0$ is some universal constant.[1]

Now, let $0 \le i \le i' \le n/k-1$ and $1 \le \ell, \ell' \le k$ such that $1 \le ik+\ell < i'k+\ell' \le n$. If $i = i'$, then clearly $\|\tilde{\boldsymbol{b}}_{ik+\ell}\|/\|\tilde{\boldsymbol{b}}_{i'k+\ell'}\| \le k^{O(\log k)}$ by Eq. (4.4). Otherwise, $i < i'$ and

$$\begin{aligned}
\frac{\|\tilde{\boldsymbol{b}}_{ik+\ell}\|}{\|\tilde{\boldsymbol{b}}_{i'k+\ell'}\|} &\le k^{O(\log k)} \cdot \frac{\|\tilde{\boldsymbol{b}}_{ik+\ell}\|}{\|\tilde{\boldsymbol{b}}_{i'k+1}\|} && \text{(Eq. (4.4))}\\
&\le k^{O(n/k+\log k)} \cdot \frac{\|\tilde{\boldsymbol{b}}_{ik+\ell}\|}{\|\tilde{\boldsymbol{b}}_{ik+k+1}\|} && \text{(Eq. (4.3))}\\
&\le k^{O(n/k+\log k)} \cdot \frac{\|\tilde{\boldsymbol{b}}_{ik+\ell}\|}{\|\tilde{\boldsymbol{b}}_{ik+k}\|} && \text{(Eq. (4.5))}\\
&\le k^{O(n/k+\log k)} && \text{(Eq. (4.4)),}
\end{aligned}$$

as needed. □

Finally, we show how to get rid of the requirement that k divides n. We will do this by (1) extending an input basis $\hat{B}$ of a lattice $\mathcal{L}$ with n vectors to a basis with $n' := \lceil n/k \rceil \cdot k$ vectors by appending long, orthogonal vectors to $\hat{B}$, (2) running the slide-reduction algorithm on the new basis to obtain a slide-reduced basis B', and then (3) showing that the first n vectors B of B' form a basis of the original lattice $\mathcal{L}$.

Proposition 4.2.11. *For any $\log n \le k \le n$, there is an algorithm that takes as*

[1]They actually observe that a slide-reduced basis is LLL reduced, which immediately implies Eq. (4.5).

input a lattice $\mathcal{L} \subset \mathbb{Q}^n$ and outputs a basis B of $\mathcal{L}$ such that $\eta(B) \le k^{O(n/k+\log k)}$. Furthermore, the algorithm runs in $2^{O(k)}$-time.

Proof. We assume without loss of generality that k is an integer. Let $n' := \lceil n/k \rceil \cdot k$ be the smallest integer greater than or equal to n that is divisible by k. On input a basis $\hat{B} = [\hat{\boldsymbol{b}}_1, \ldots, \hat{\boldsymbol{b}}_n]$ for the lattice $\mathcal{L} \subset \mathbb{Q}^n$, the algorithm behaves as follows. Let $r := 2^{\Omega(n^2)} \cdot \max_i \|\hat{\boldsymbol{b}}_i\|$. Let $\mathcal{L}' := \mathcal{L} \oplus r\mathbb{Z}^{n'-n} = \mathcal{L}(\hat{\boldsymbol{b}}_1, \ldots, \hat{\boldsymbol{b}}_n, r \cdot \boldsymbol{e}_{n+1}, \ldots, r \cdot \boldsymbol{e}_{n'}) \subset \mathbb{Q}^{n'}$ be "the lattice generated by appending $n' - n$ orthogonal vectors of length r to $\hat{B}$." The algorithm then computes a $(1/n, k)$-slide reduced basis $B' = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n'}]$ of $\mathcal{L}'$ as in Theorem 4.2.8, and returns the basis $B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$ consisting of the first n entries of B'.

It follows immediately from Theorem 4.2.8 that the running time is as claimed, and from Proposition 4.2.10 we have that $\eta(B) \le \eta(B') \le k^{O(n'/k+\log k)} \le k^{O(n/k+\log k)}$. So, we only need to prove that B is in fact a basis for $\mathcal{L}$ (as opposed to some other sublattice of $\mathcal{L}'$).

Let i be the minimum index such that $\boldsymbol{b}_i \notin \operatorname{span}(\mathcal{L})$. It is clear that $i \le n+1$. We claim that $i = n+1$, i.e., that $\operatorname{span}(B) = \operatorname{span}(\mathcal{L})$ and therefore that $\mathcal{L}(B) = \mathcal{L}$. Suppose not. Then there exists $1 \le j \le n$ such that $\pi_{i-1}^{(B')}(\hat{\boldsymbol{b}}_j) \ne \boldsymbol{0}$ and $\pi_{i-1}^{(B')}(\hat{\boldsymbol{b}}_j) \in \pi_{i-1}^{(B')}(\mathcal{L}')$. Then,

$$\lambda_1(\pi_{i-1}^{(B')}(\mathcal{L}')) \le \|\pi_{i-1}^{(B')}(\hat{\boldsymbol{b}}_j)\| \le \|\hat{\boldsymbol{b}}_j\| \le 2^{-\Omega(n^2)} \cdot \pi_{i-1}^{(B')}(\boldsymbol{b}_i), \tag{4.6}$$

where the last inequality follows from the fact that $\|\pi_{i-1}^{(B')}(\boldsymbol{b}_i)\| \ge \|\pi_{\operatorname{span}(\mathcal{L})^\perp}(\boldsymbol{b}_i)\| \ge \lambda_1(\pi_{\operatorname{span}(\mathcal{L})^\perp}(\mathcal{L}')) = r$.

Slide-reduced bases are also LLL-reduced (see [GN08, Section 3.2]), so B' is LLL-reduced and therefore $[\pi_{i-1}^{(B')}(\boldsymbol{b}_i), \ldots, \pi_{i-1}^{(B')}(\boldsymbol{b}_{n'})]$ is also LLL-reduced. But because $[\pi_{i-1}^{(B')}(\boldsymbol{b}_i), \ldots, \pi_{i-1}^{(B')}(\boldsymbol{b}_{n'})]$ is LLL-reduced, it holds that $\|\pi_{i-1}^{(B')}(\boldsymbol{b}_i)\| \leq 2^{O(n)} \cdot \lambda_1(\pi_{i-1}^{(B')}(\mathcal{L}'))$ (see, e.g., [GN08, Equation (2)]), which contradicts Equation (4.6).

□

4.2.5.2 Seysen bases

Although slide-reduced bases B consist of short vectors and have bounded $\eta(B)$, they make only weak guarantees about the length of vectors in the dual basis B^*. Of course, one way to compute a basis whose dual will contain short dual basis is short is to simply compute B such that B^* is a suitably reduced basis of $\mathcal{L}^*$. Such a basis B is called a dual-reduced basis, and sees use in applications such as [HR14].

However, we would like to compute a basis such that the vectors in B and B^* are both short, which Seysen addressed in his work [Sey93]. Seysen's main result finds a basis B such that both B and B^* are short by dividing this problem into two subproblems. The first involves finding a basis with small $\eta(B)$, as in Section 4.2.5.1. The second subproblem, discussed in [Sey93, Section 3], involves conditioning unipotent matrices. Let $N(n, \mathbb{R})$ be the multiplicative group of unipotent $n \times n$-matrices. That is, a matrix $A \in N(n, \mathbb{R})$ if $a_{ii} = 1$ and $a_{ij} = 0$ for $i > j$ (i.e., A is upper triangular and has ones on the main diagonal). Let $N(n, \mathbb{Z})$ be the subgroup of $N(n, \mathbb{R})$ with integer entries. Because $N(n, \mathbb{Z})$ is a subset of $GL(n, \mathbb{Z})$, we trivially have that $\mathcal{L}(B) = \mathcal{L}(B \cdot U)$ for every $U \in N(n, \mathbb{Z})$.

Let $\|B\|_\infty := \max_{i,j \in [n]} |b_{ij}|$ denote the largest magnitude of an entry in B. We follow Seysen [Sey93] and define $S'(B) = \max\{\|B\|_\infty, \|B^{-1}\|_\infty\}$. We also let

$$\zeta(n) = \sup_{A \in N(n,\mathbb{R})} \big\{ \inf_{U \in N(n,\mathbb{Z})} \{S'(A \cdot U)\}\big\}.$$

Theorem 4.2.12 ([Sey93, Prop. 5 and Thm. 6])**.** *There exists an algorithm* Seysen *that takes as input $W \in N(n, \mathbb{R})$ and outputs $\hat{W} = W \cdot U$ where $U \in N(n, \mathbb{Z})$ and $S'(\hat{W}) \leq n^{O(\log n)}$ in time $O(n^3)$. In particular, $\zeta(n) \leq n^{O(\log n)}$.*

Let $B = QR$ be a QR-decomposition of B. We may further decompose R as $R = DW$, where $d_{ii} = \|\tilde{\boldsymbol{b}}_i\|$ and

$$w_{ij} = \begin{cases} 0 & \text{if } j < i, \\ 1 & \text{if } j = i, \\ \mu_{ji} & \text{if } j > i. \end{cases}$$

In particular, note that $W \in N(n, \mathbb{R})$.

For completeness, we reprove the following theorem which shows, given an input basis B', how to upper bound $S(\text{Seysen})$ as a function of $\eta(B')$ and $\zeta(n)$.

Theorem 4.2.13 ([Sey93, Theorem 7])**.** *Let $B = \text{Seysen}(B')$, where $B' \in \mathbb{Q}^{n \times n}$ is a full-rank matrix. Then $S(B) \leq n \cdot \eta(B') \cdot \zeta(n)^2$.*

Proof. Let $B' = QR = QDW$ where Q is an orthogonal matrix, D is a diagonal matrix, W is an upper-triangular unipotent matrix, and $R = DW$. Furthermore, let $U \in N(n, \mathbb{Z})$ be as in Theorem 4.2.12, i.e., so that $B = B'U$ and $S'(WU) \leq n^{O(\log n)}$. Let $\hat{R} = RU = DWU$ and let $\hat{W} = WU$. Then for all $1 \leq j \leq n$,

$$\begin{aligned}
\|\boldsymbol{b}_j\|^2\|\boldsymbol{b}_j^*\|^2 &= \|\hat{\boldsymbol{r}}_j\|^2\|\hat{\boldsymbol{r}}_j^*\|^2 && (Q \text{ is orthogonal})\\
&= \sum_{i=1}^{j}(d_{i,i}\cdot \hat{w}_{i,j})^2 \cdot \sum_{k=j}^{n}(\hat{w}_{k,j}^*/d_{k,k})^2\\
&= \sum_{i=1}^{j}\sum_{k=j}^{n}\left(\frac{d_{i,i}}{d_{k,k}}\right)^2\cdot(\hat{w}_{i,j}\hat{w}_{k,j}^*)^2\\
&\le n^2\cdot\max_{k\ge i}\left(\frac{d_{i,i}}{d_{k,k}}\right)^2\cdot\zeta(n)^4, && (\text{By Theorem 4.2.12})\\
&\le n^2\cdot\eta(B')^2\cdot\zeta(n)^4 && (\text{Since } \eta(B')=\eta(D))\\
&= n^2\cdot\eta(B')^2\cdot\zeta(n)^4,
\end{aligned}$$

which proves the claim. □

Proof of Theorem 4.2.2. Let $B = \textsc{Seysen}(B')$, where B' is a basis as computed in Proposition 4.2.11. We then have that

$$\begin{aligned}
S(B) &\le n\cdot\eta(B')\cdot\zeta(n)^2 && (\text{by Theorem 4.2.13})\\
&\le n\cdot k^{O(n/k+\log k)}\cdot\zeta(n)^2 && (\text{by Proposition 4.2.11})\\
&\le n\cdot k^{O(n/k+\log k)}\cdot(n^{O(\log n)})^2 && (\text{by Theorem 4.2.12})\\
&\le k^{O(n/k+\log k)}.
\end{aligned}$$

We can compute B' in $2^{O(k)}$ time using Proposition 4.2.11. Moreover, by Theorem 4.2.12, Seysen runs in $O(n^3)$ time. Therefore the algorithm runs in $2^{O(k)}$ time. □

4.2.6 Lower bounding $S(B)$

Call a basis $B = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_n]$ *sorted* if $\|\boldsymbol{b}_1\| \leq \cdots \leq \|\boldsymbol{b}_n\|$. Clearly, $\|\boldsymbol{b}_i\|/\lambda_i \geq 1$ for a sorted basis B. Note that sorting B does not change $S(B)$, since $S(\cdot)$ is invariant under permutations of the basis vectors.

A fundamental question (stated in Open Problem 4.1.7) is whether the upper bound $S(\mathcal{L}) \leq n^{O(\log n)}$ on $S(\mathcal{L})$ can be improved for every lattice $\mathcal{L}$ of rank n. We conclude this section by showing that for sorted bases B, $\eta(B)$ lower bounds $S(B)$, and therefore improving the upper bound on $S(\mathcal{L})$ would require finding a family of bases with lower η than HKZ-reduced bases (i.e. better than the bases obtained via the algorithm in Proposition 4.2.11 with $k = n$).

Proposition 4.2.14. *For sorted bases B, $\eta(B) \leq S(B)$.*

Proof. We use the fact that for any basis B of rank n and any $1 \leq i \leq n$,

$$\|\tilde{\boldsymbol{b}}_i\|^{-1} \leq \|\boldsymbol{b}_i^*\|. \tag{4.7}$$

Note that $\boldsymbol{b}_i = \tilde{\boldsymbol{b}}_i + \boldsymbol{w}_i$ for some $\boldsymbol{w}_i \in \operatorname{span}(\boldsymbol{b}_1, \dots, \boldsymbol{b}_{i-1})$. Equation (4.7) then follows from the fact that

$$1 = \langle \boldsymbol{b}_i, \boldsymbol{b}_i^* \rangle = \langle \tilde{\boldsymbol{b}}_i + \boldsymbol{w}_i, \boldsymbol{b}_i^* \rangle = \langle \tilde{\boldsymbol{b}}_i, \boldsymbol{b}_i^* \rangle \leq \|\tilde{\boldsymbol{b}}_i\| \|\boldsymbol{b}_i^*\|,$$

where the equalities hold because $\langle \boldsymbol{b}_i, \boldsymbol{b}_j^* \rangle$ is equal to 1 if $i = j$ and is equal to 0 otherwise, and the inequality holds because of the Cauchy-Schwarz inequality.

Therefore,

$$\eta(B) = \max_{i \leq j} \frac{\|\tilde{\boldsymbol{b}}_i\|}{\|\tilde{\boldsymbol{b}}_j\|} \leq \max_{i \leq j} \|\tilde{\boldsymbol{b}}_i\| \|\boldsymbol{b}_j^*\| \leq \max_{i \leq j} \|\boldsymbol{b}_i\| \|\boldsymbol{b}_j^*\| \leq \max_j \|\boldsymbol{b}_j\| \|\boldsymbol{b}_j^*\| = S(B),$$

where the first inequality holds by Equation (4.7) and the third inequality holds since B is sorted. □

It is easy to see that Proposition 4.2.14 is false for unsorted bases. Indeed, an unsorted diagonal basis B always has $S(B) = 1$ but may have arbitrarily large $\eta(B)$.

4.3 Approximating Lattice Distortion

In this section, we show how to compute low-distortion mappings between lattices by using bases B with low $S(B)$.

4.3.1 Basis length bounds in terms of $S(B)$

A natural way to quantify the "shortness" of a lattice basis is to upper bound $\|\boldsymbol{b}_k\|/\lambda_k$ for all $k \in [n]$. For example, [LLS90] shows that $\|\boldsymbol{b}_k\|/\lambda_k \leq \sqrt{n}$ when B is an HKZ basis. We give a characterization of Seysen bases showing that in fact *both* the primal basis vectors and the dual basis vectors are not much longer than the successive minima. Namely, $S(B)$ is an upper bound on both $\|\boldsymbol{b}_k\|/\lambda_k$ and $\|\boldsymbol{b}_k^*\|/\lambda_{n-k+1}^*$ for sorted bases B. Although we only use the fact that $S(B) \geq \|\boldsymbol{b}_k\|/\lambda_k$ we show both bounds. Seysen [Sey93] gave essentially the same characterization, but we state and prove it here in a slightly different form.

Lemma 4.3.1 (Theorem 8 in [Sey93])**.** *Let B be a sorted basis of $\mathcal{L}$. Then for all $k \in [n]$,*

1. $\|\boldsymbol{b}_k\|/\lambda_k(\mathcal{L}) \leq S(B)$.

2. $\|\boldsymbol{b}_k^*\|/\lambda_{n-k+1}^*(\mathcal{L}) \leq S(B)$.

Proof. For every $k \in [n]$ we have

$$\begin{aligned}
\|\boldsymbol{b}_k\|/\lambda_k &\le \|\boldsymbol{b}_k\|\lambda^*_{n-k+1} && \text{(by the lower bound in Theorem 3.2.1)} \\
&\le \|\boldsymbol{b}_k\| \max_{k\le i\le n}\|\boldsymbol{b}^*_i\| && \text{(the } \boldsymbol{b}^*_i\text{'s are linearly independent)} \\
&\le \max_{k\le i\le n}\|\boldsymbol{b}_i\|\|\boldsymbol{b}^*_i\| && (B \text{ is sorted}) \\
&\le S(B).
\end{aligned}$$

This proves Item 1. Furthermore, for every $k \in [n]$ we have

$$\frac{\|\boldsymbol{b}^*_k\|}{\lambda^*_{n-k+1}} \le \frac{\|\boldsymbol{b}_k\|\|\boldsymbol{b}^*_k\|}{\lambda_k\lambda^*_{n-k+1}} \le \max_{i\in[n]}\|\boldsymbol{b}_i\|\|\boldsymbol{b}^*_i\| = S(B).$$

The first inequality follows from the assumption that B is sorted, and the second follows from the lower bound in Theorem 3.2.1. This proves Item 2. $\square$

4.3.2 Approximating LDP using Seysen bases

In this section, we bound the distortion $D(\mathcal{L}_1, \mathcal{L}_2)$ between lattices $\mathcal{L}_1, \mathcal{L}_2$. The upper bound is constructive and depends on $S(B_1), S(B_2)$, which naturally leads to Theorem 4.1.4.

The proof uses two linear algebraic identities. First, it uses the fact that one can write the product XY of two matrices X, Y as a sum of *outer products*. I.e., it holds that

$$XY = \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{y}_i^T, \tag{4.8}$$

where $\boldsymbol{x}_i$ is the ith column of X and $\boldsymbol{y}_i$ is the ith row of Y. Second, we use the following identity about the operator norm of a rank-one matrix defined as an outer product. Namely, given vectors $\boldsymbol{x}, \boldsymbol{y}$,

$$\|\boldsymbol{x}\boldsymbol{y}^T\| = \|\boldsymbol{x}\|\|\boldsymbol{y}\|. \tag{4.9}$$

Lemma 4.3.2. *Let $A = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n]$ and $B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$ be sorted bases of $\mathcal{L}_1, \mathcal{L}_2$ respectively. Then,*

$$\|BA^{-1}\| \leq n \cdot S(A)S(B) \cdot M(\mathcal{L}_1, \mathcal{L}_2).$$

Proof.

$$\begin{aligned}
\|BA^{-1}\| &= \left\| \sum_{i=1}^{n} \boldsymbol{b}_i(\boldsymbol{a}_i^*)^T \right\| && \text{(by Equation (4.8))} \\
&\leq \sum_{i=1}^{n} \left\| \boldsymbol{b}_i(\boldsymbol{a}_i^*)^T \right\| && \text{(by triangle inequality)} \\
&= \sum_{i=1}^{n} \|\boldsymbol{b}_i\|\|\boldsymbol{a}_i^*\| && \text{(by Equation (4.9))} \\
&\leq n \cdot \max_{i\in[n]} \|\boldsymbol{b}_i\|\|\boldsymbol{a}_i^*\| \\
&\leq n \cdot S(B) \cdot \max_{i\in[n]} \lambda_i(\mathcal{L}_2)\|\boldsymbol{a}_i^*\| && \text{(by Item 1 in Lemma 4.3.1)} \\
&\leq n \cdot S(A)S(B) \cdot \max_{i\in[n]} \lambda_i(\mathcal{L}_2)/\|\boldsymbol{a}_i\| && \text{(by definition of } S(A)\text{)} \\
&\leq n \cdot S(A)S(B) \cdot \max_{i\in[n]} \lambda_i(\mathcal{L}_2)/\lambda_i(\mathcal{L}_1) && (A \text{ is sorted}) \\
&= n \cdot S(A)S(B) \cdot M(\mathcal{L}_1, \mathcal{L}_2).
\end{aligned}$$

□

We can now prove the bounds on distortion given in Theorem 4.1.2.

Proof of Theorem 4.1.2. Note that for $i = 1, 2$ there always exists a basis B_i of $\mathcal{L}_i$ which achieves $S(B_i) = S(\mathcal{L}_i)$. Indeed, this follows from the fact that for every lattice $\mathcal{L}$ and $r > 0$ there are finitely many bases B of $\mathcal{L}$ with $S(B) \leq r$, which was shown by Seysen [Sey93, Corollary 9] (see also Corollary 5.4.1 in the next chapter). Therefore, applying Lemma 4.3.2 twice to bound both $\|B_2 B_1^{-1}\|$ and $\|B_1 B_2^{-1}\|$, we get the upper bound.

For the lower bound, let $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathcal{L}_1$ be linearly independent vectors such that $\|\boldsymbol{v}_i\| = \lambda_i(\mathcal{L}_1)$ for every i. Then, for every i,

$$\lambda_i(\mathcal{L}_2) \leq \max_{j \in [i]} \|T\boldsymbol{v}_j\| \leq \|T\| \max_{j \in [i]} \|\boldsymbol{v}_j\| = \|T\| \lambda_i(\mathcal{L}_1).$$

Rearranging, we get that $\lambda_i(\mathcal{L}_2)/\lambda_i(\mathcal{L}_1) \leq \|T\|$. This holds for arbitrary i, so in particular $\max_{i \in [n]} \lambda_i(\mathcal{L}_2)/\lambda_i(\mathcal{L}_1) = M(\mathcal{L}_1, \mathcal{L}_2) \leq \|T\|$. The same computation with $\mathcal{L}_1, \mathcal{L}_2$ reversed shows that $M(\mathcal{L}_2, \mathcal{L}_1) \leq \|T^{-1}\|$. Multiplying these bounds together implies the lower bound in the theorem statement. $\square$

Finally, we can prove Theorem 4.1.4, which gives an algorithm with a time-approximation trade-off for computing low-distortion mappings.

Proof of Theorem 4.1.4. Let $(\mathcal{L}_1, \mathcal{L}_2)$ be an instance of LDP. For $i = 1, 2$, compute a basis B_i of $\mathcal{L}_i$ using the algorithm described in Theorem 4.2.2 with parameter k. We have that $S(B_i) \leq k^{O(n/k + \log k)}$. This computation takes $2^{O(k)}$ time. The algorithm then simply outputs $T = B_2 B_1^{-1}$.

By Lemma 4.3.2 and the upper bounds on $S(B_i)$, we get that $\kappa(T) \leq k^{O(n/k+\log k)} \cdot M(\mathcal{L}_1, \mathcal{L}_2) \cdot M(\mathcal{L}_2, \mathcal{L}_1)$, which is within a $k^{O(n/k+\log k)}$ factor of $D(\mathcal{L}_1, \mathcal{L}_2)$ by Theorem 4.1.1. So, the algorithm is correct.

$\square$

4.4 Hardness of LDP

In this section, we prove the hardness of γ-GapLDP. (See Theorem 4.4.7.) Our reduction works in two steps. First, we a variant of GapCVP that we call γ-GapCVP$'$ to GapLDP. (See Definition 4.4.1 and Theorem 4.4.3.) Given a CVP instance consisting of a lattice $\mathcal{L}$ and a target vector $\boldsymbol{t}$, our idea is to compare "$\mathcal{L}$ with $\boldsymbol{t}$ appended to it" to "$\mathcal{L}$ with an extra orthogonal vector appended to it." (See Eq. (4.10).) We show that, if $\mathrm{dist}(\boldsymbol{t}, \mathcal{L})$ is small, then these lattices will be similar. On the other hand, if (1) $\mathrm{dist}(k\boldsymbol{t}, \mathcal{L})$ is large for all integers k with small magnitude, and (2) $\lambda_1(\mathcal{L})$ is not too small, then the two lattices must be quite dissimilar.

We next show that γ-GapCVP$'$ is as hard as GapSVP using a variant of the celebrated reduction of [GMSS99]. (See Theorem 4.4.4.) It differs from the original in that it "works in base p" instead of in base two. We show that this is sufficient to satisfy the promises required by γ-GapCVP$'$.

4.4.1 Reduction from a variant of CVP

We first define γ-GapCVP$'$, a variant of GapCVP which differs from GapCVP in two ways. Namely, it requires for a 'NO' instance the additional promises (1) that $d < \gamma \cdot \lambda_1(\mathcal{L})$, and (2) that all non-zero integer multiples $k\boldsymbol{t}$ of the target vector $\boldsymbol{t}$ with $|k| \le \gamma$ are far from the lattice.

Definition 4.4.1. For any $\gamma = \gamma(n) \ge 1$, γ-GapCVP$'$ is the promise problem defined as follows. The input is a lattice $\mathcal{L} \subset \mathbb{Q}^n$ (specified by a basis $B \in \mathbb{Q}^{n \times n}$), a target $\boldsymbol{t} \in \mathbb{Q}^n$, and a distance $d > 0$. It is a 'YES' instance if $\mathrm{dist}(\boldsymbol{t}, \mathcal{L}) \le d$ and a 'NO' instance if $d < \lambda_1(\mathcal{L})/\gamma$ and $\mathrm{dist}(k\boldsymbol{t}, \mathcal{L}) > \gamma d$ for integers k with $1 \le |k| \le \gamma$.

We will need the following characterization of the operator norm of a matrix in terms of its behavior over a lattice. Intuitively, this says that "a lattice has a point in every direction."

Fact 4.4.2. *For any matrix $A \in \mathbb{R}^{n \times n}$ and (full-rank) lattice $\mathcal{L} \subset \mathbb{R}^n$,*

$$\|A\| = \sup_{\boldsymbol{y} \in \mathcal{L} \setminus \{\boldsymbol{0}\}} \frac{\|A\boldsymbol{y}\|}{\|\boldsymbol{y}\|}.$$

Proof. It suffices to note that, for any $\boldsymbol{x} \in \mathbb{R}^n$ with $\|\boldsymbol{x}\| = 1$ and any full-rank lattice $\mathcal{L} \subset \mathbb{R}^n$, there is a sequence $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots$ of vectors $\boldsymbol{y}_i \in \mathcal{L}$ such that

$$\lim_{m \to \infty} \frac{\boldsymbol{y}_m}{\|\boldsymbol{y}_m\|} = \boldsymbol{x}.$$

This follows by taking $\boldsymbol{y}_m$ to be the closest vector in $\mathcal{L}$ to $m\boldsymbol{x}$, and by noting that the covering radius of $\mathcal{L}$ is finite.

$\square$

Recall that a polynomial-time, many-one mapping from an instance of problem A to an instance problem B which preserves 'YES' and 'NO' instances is called a *Karp reduction* from A to B. A polynomial-time algorithm for solving a problem A given access to an oracle for a problem B is called a *Cook reduction* from A to B.

Theorem 4.4.3. *For any $\gamma = \gamma(n) \geq 1$, there is a Karp reduction from* 6γ-GapCVP$'$ *to γ-GapLDP.*

Proof. On input $\mathcal{L} \subset \mathbb{Q}^n$ with basis $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$, $\boldsymbol{t} \in \mathbb{Q}^n$, and $d > 0$, the reduction behaves as follows.

Let $B_1 := [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n, 2d \cdot \boldsymbol{e}_{n+1}]$. Let $B_2 := [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n, \boldsymbol{t} + 2d \cdot \boldsymbol{e}_{n+1}]$. I.e.,

$$B_1 = \begin{pmatrix} B & \mathbf{0} \\ 0 & 2d \end{pmatrix}, \qquad B_2 = \begin{pmatrix} B & \boldsymbol{t} \\ 0 & 2d \end{pmatrix}. \tag{4.10}$$

(Formally, we must embed the $\boldsymbol{b}_i$ and $\boldsymbol{t}$ in $\mathbb{Q}^{n+1}$ under the natural embedding, but we ignore this for simplicity.) Let $\mathcal{L}_1 := \mathcal{L}(B_1)$ and $\mathcal{L}_2 := \mathcal{L}(B_2)$. The reduction then outputs the γ-GapLDP instance $\mathcal{L}_1$, $\mathcal{L}_2$, (specified by the bases B_1, B_2) and $c > 0$, for some c which will be set in the analysis.

It is clear that the reduction runs in polynomial time. Suppose that $\operatorname{dist}(\boldsymbol{t}, \mathcal{L}) \leq d$. We note that $\mathcal{L}_2$ does not change if we shift $\boldsymbol{t}$ by a lattice vector. So, we may assume without loss of generality that $\mathbf{0}$ is a closest lattice vector to $\boldsymbol{t}$ and therefore $\|\boldsymbol{t}\| \leq d$.

Indeed, for any $\boldsymbol{y} \in \mathcal{L}_1$, we can write $\boldsymbol{y} = (\boldsymbol{y}', 2dk)$ for some $\boldsymbol{y}' \in \mathcal{L}$ and $k \in \mathbb{Z}$. Then, we have

$$\|B_2 B_1^{-1} \boldsymbol{y}\| = \|(\boldsymbol{y}' + k\boldsymbol{t}, 2dk)\| \leq \|(\boldsymbol{y}', 2dk)\| + |k|\|\boldsymbol{t}\| \leq 3\|\boldsymbol{y}\|/2.$$

Similarly, $\|B_2 B_1^{-1}\boldsymbol{y}\| \geq \|\boldsymbol{y}\| - |k|\|\boldsymbol{t}\| \geq \|\boldsymbol{y}\|/2$. Therefore, by Fact 4.4.2, $\kappa(B_2 B_1^{-1}) \leq (3\|\boldsymbol{y}\|/2)/(\|\boldsymbol{y}\|/2) = 3$. So, we take $c := 3$, and the resulting GapLDP instance is a 'YES' instance.

Now, suppose $\operatorname{dist}(z\boldsymbol{t}, \mathcal{L}) > 6\gamma d$ for integers z with $1 \leq |z| \leq 6\gamma$, and $\lambda_1(\mathcal{L}) > 6\gamma d$. Let A be a linear map with $A(\mathcal{L}_1) = \mathcal{L}_2$. Recall that $\kappa(A) \geq \|A\| \geq \max_{\boldsymbol{x} \in \mathcal{L}_1 \setminus \{\mathbf{0}\}} \frac{\|A\boldsymbol{x}\|}{\|\boldsymbol{x}\|}$, where the first inequality holds because A has determinant one.

We have that $A(\mathbf{0}, 2d) = (\boldsymbol{y}'', 2dk)$ for some $\boldsymbol{y}'' \in \mathcal{L} + k\boldsymbol{t}$ and some $k \in \mathbb{Z}$. We consider three cases. If $k = 0$, then $\boldsymbol{y}'' \in \mathcal{L} \setminus \{\mathbf{0}\}$ and $\|A(\mathbf{0}, 2d)\| = \|(\boldsymbol{y}'', 0)\| \geq \lambda_1(\mathcal{L}) > 6\gamma d$, so that we have $\kappa(A) \geq \|A(\mathbf{0}, 2d)\|/2d > 3\gamma$. If $1 \leq |k| \leq 6\gamma$, then $\|A(\mathbf{0}, 2d)\| \geq \operatorname{dist}(k\boldsymbol{t}, \mathcal{L}) > 6\gamma d$, so $\kappa(A) \geq \|A(\mathbf{0}, 2d)\|/2d > 3\gamma$. Finally, if $|k| > 6\gamma$, then $\|A(\mathbf{0}, 2d)\| \geq |2dk| > 12d\gamma$, so again $\kappa(A) \geq \|A(\mathbf{0}, 2d)\|/2d > 3\gamma$.

In each case, $\kappa(A) > 3\gamma = \gamma \cdot c$, so the output GapLDP instance is a 'NO' instance.

□

4.4.2 Hardness of This Variant of GapCVP

We next prove the hardness of γ-*GapCVP*$'$.

Theorem 4.4.4. *For any* $1 \leq \gamma = \gamma(n) \leq \operatorname{poly}(n)$*, there is a Cook reduction from* γ-*GapSVP to* γ-CVP$'$.

Proof. Let p be a prime with $10\gamma \leq p \leq 20\gamma$. On input a basis $B := [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$ for a lattice $\mathcal{L} \subset \mathbb{Q}^n$, and $d > 0$, the reduction behaves as follows. For $i = 1, \ldots, n$, let $\mathcal{L}_i := \mathcal{L}(\boldsymbol{b}_1, \ldots, p\boldsymbol{b}_i, \ldots, \boldsymbol{b}_n)$ be "$\mathcal{L}$ with its ith basis vector multiplied by p." And, for all i and $1 \leq j < p$, let $\boldsymbol{t}_{i,j} := j\boldsymbol{b}_i$. For each i, j, the reduction calls its γ-GapCVP$'$ oracle on input $\mathcal{L}_i$, $\boldsymbol{t}_{i,j}$, and distance d. Finally, it outputs 'YES' if the oracle answered 'YES' for any query. Otherwise, it outputs 'NO'.

The algorithm makes an oracle call for each $1 \leq i \leq n$ and each $1 \leq j < p$, for a total of $O(\gamma n)$ oracle calls. It follows that the reduction runs in polynomial time.

We next prove the correctness of the reduction. Note that

$$\operatorname{dist}(j\boldsymbol{b}_i, \mathcal{L}_i) = \min\left\{\left\| \sum_{\ell=1}^{n} a_\ell \boldsymbol{b}_\ell \right\| \ : \ a_\ell \in \mathbb{Z},\ a_i \equiv j \bmod p\right\} .$$

In particular, $\lambda_1(\mathcal{L}) = \min_{i,j} \operatorname{dist}(j\boldsymbol{b}_i, \mathcal{L}_i)$. So, suppose that $\lambda_1(\mathcal{L}) \leq d$. Then there must be some i, j such that $\operatorname{dist}(\boldsymbol{t}_{i,j}, \mathcal{L}_i)^2 \leq \lambda_1(\mathcal{L})^2 \leq d^2$, and therefore the oracle answers 'YES' at least once.

Now, suppose that $\lambda_1(\mathcal{L}) > \gamma d$. Since $\mathcal{L}_i \subset \mathcal{L}$, we have $\lambda_1(\mathcal{L}_i) \geq \lambda_1(\mathcal{L}) > \gamma d$, and therefore $d < \lambda_1(\mathcal{L}_i)/\gamma$, as needed. And, by the above observation, we have $\mathrm{dist}(j\boldsymbol{b}_i, \mathcal{L}_i) \geq \lambda_1(\mathcal{L}) > \gamma d$ for all $1 \leq i \leq n$ and $1 \leq j < p$. Furthermore, for any integer $1 \leq z < p$, we have $\mathrm{dist}(zj\boldsymbol{b}_i, \mathcal{L}_i) = \mathrm{dist}((zj \bmod p) \cdot \boldsymbol{b}_i, \mathcal{L}_i) > \gamma d$, where we have used the fact that p is prime so that $zj \not\equiv 0 \bmod p$. It follows that $\mathrm{dist}(z\boldsymbol{t}_{i,j}, \mathcal{L}_i) > \mathrm{dist}(zj\boldsymbol{b}_i, \mathcal{L}_i) > \gamma d$ for each integer z with $1 \leq |z| < \gamma$. So, the oracle will always answer 'NO'. □

Corollary 4.4.5. *For any $1 \leq \gamma = \gamma(n) \leq \mathrm{poly}(n)$, there is a reduction from 6γ-GapSVP to γ-GapLDP. Furthermore, the reduction runs in polynomial time.*

Proof. Combine Theorems 4.4.3 and 4.4.4. □

Haviv and Regev (building on work of Ajtai, Micciancio, and Khot [Ajt98, Mic01, Kho05]) proved the following strong hardness result for γ-GapSVP [HR12].

Theorem 4.4.6 ([HR12, Theorem 1.1])**.**

1. *γ-GapSVP is NP-hard under randomized polynomial-time reductions for any constant $\gamma \geq 1$. I.e., there is no randomized polynomial-time algorithm for γ-GapSVP unless $\mathsf{NP} \subseteq \mathsf{RP}$.*

2. *$2^{\log^{1-\varepsilon} n}$-GapSVP is NP-hard under randomized quasipolynomial-time reductions for any constant $\varepsilon > 0$. I.e., there is no randomized polynomial-time algorithm for $2^{\log^{1-\varepsilon} n}$-GapSVP unless $\mathsf{NP} \subseteq \mathsf{RTIME}(2^{\mathrm{polylog}(n)})$.*

3. *$n^{c/\log\log n}$-GapSVP is NP-hard under randomized subexponential-time reductions for some universal constant $c > 0$. I.e., there is no randomized polynomial-time algorithm for $n^{c/\log\log n}$-GapSVP unless $\mathsf{NP} \subseteq \mathsf{RSUBEXP} := \bigcap_{\delta>0} \mathsf{RTIME}(2^{n^\delta})$.*

With this, Theorem 4.1.5 and additional hardness results follow immediately.

Theorem 4.4.7. *The three hardness results in Theorem 4.4.6 hold with GapLDP in place of GapSVP.*

Proof. Combine Theorem 4.4.6 with Corollary 4.4.5. □

4.5 Some illustrative examples

4.5.1 Separating distortion from the successive minima

We now show that, for every n, there exists a lattice $\mathcal{L}$ such that $D(\mathcal{L}, \mathbb{Z}^n) \geq \Omega(n) \cdot M(\mathcal{L}, \mathbb{Z}^n) \cdot M(\mathbb{Z}^n, \mathcal{L})$ using a simple argument by Regev [Reg17].[2] Indeed, to show this bound it suffices to take any lattice $\mathcal{L}$ with $\lambda_i(\mathcal{L}) = \Theta(\sqrt{n})$ and $\lambda_i(\mathcal{L}^*) = \Theta(\sqrt{n})$ for all $i \in [n]$. This is true for almost all lattices in a certain precise sense; see [Sie45].

Lemma 4.5.1. *For any $n \geq 1$, there is a lattice $\mathcal{L} \subset \mathbb{Q}^n$ such that $\lambda_i(\mathcal{L}) = \Theta(\sqrt{n})$ and $\lambda_i(\mathcal{L}^*) = \Theta(\sqrt{n})$ for every $i \in [n]$.*

Proposition 4.5.2 ([Reg17])**.** *For any $n \geq 1$, there exists a lattice $\mathcal{L} \subset \mathbb{Q}^n$ such that*

$$D(\mathcal{L}, \mathbb{Z}^n) \geq \Omega(n) \cdot M(\mathcal{L}, \mathbb{Z}^n) \cdot M(\mathbb{Z}^n, \mathcal{L}) .$$

Proof. We note that to lower bound $D(\mathcal{L}, \mathbb{Z}^n)$ for any $\mathcal{L}$ it suffices to lower bound the condition number $\kappa(B) = \|B\|\|B^{-1}\|$ of every basis B of $\mathcal{L}$. This is because every linear bijection from $\mathbb{Z}^n$ to $\mathcal{L}$ must map I_n to a basis of $\mathcal{L}$.

[2] In [BDS16], we gave a weaker version of this bound with $\Omega(\sqrt{n})$ in place of $\Omega(n)$. We showed this by arguing about the determinant and successive minima of a random lattice (in the sense of [Sie45]) compared to $\mathbb{Z}^n$.

Let $\mathcal{L} \subset \mathbb{Q}^n$ be any lattice as in Lemma 4.5.1, and let $B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$ be a basis $\mathcal{L}$. Then

$$\|B\| \geq \max_{i \in [n]} \|\boldsymbol{b}_i\| \geq \lambda_1(\mathcal{L}) \geq \Omega(\sqrt{n}). \tag{4.11}$$

Similarly,

$$\|B^{-1}\| = \|B^*\| \geq \max_{i \in [n]} \|\boldsymbol{b}_i^*\| \geq \lambda_1(\mathcal{L}^*) \geq \Omega(\sqrt{n}). \tag{4.12}$$

Moreover, because $\lambda_i(\mathbb{Z}_n) = 1$ and $\lambda_i(\mathcal{L}) = \Theta(\sqrt{n})$ for every $i \in [n]$ we have that $M(\mathcal{L}, \mathbb{Z}^n) \cdot M(\mathbb{Z}^n, \mathcal{L}) = \Theta(1)$. Combining this with Equations (4.11) and (4.12) proves the claim. □

4.5.2 Non-optimality of HKZ bases for distortion

We show an example demonstrating that mappings between lattices built using HKZ bases are non-optimal in terms of their distortion. Let B_n be the $n \times n$ upper-triangular matrix with diagonal entries equal to 1 and upper triangular off-diagonal entries equal to $-\frac{1}{2}$. I.e., B_n has entries

$$b_{ij} = \begin{cases} 0 & \text{if } j < i, \\ 1 & \text{if } j = i, \\ -\frac{1}{2} & \text{if } j > i. \end{cases}$$

Luk and Tracy [LT08] use the family $\{B_n\}$ as an example of bases that are well-reduced but poorly conditioned. Indeed it is not hard to show that $\{B_n\}$ are HKZ bases that nevertheless have $\kappa(B_n) = \Omega(1.5^n)$ (see [LT08], Example 2). We use these bases to show the necessity of using Seysen reduction even on HKZ bases.

Theorem 4.5.3. *For every $n \geq 1$, there exists an $n \times n$ HKZ basis B such that $\operatorname{dist}(\mathbb{Z}^n, \mathcal{L}(B)) \leq n^{O(\log n)}$, but $\kappa(B) \geq \Omega(1.5^n)$.*

Proof. Let $B' = B_n$ be an HKZ basis in the family described above, and take I_n as the basis of $\mathbb{Z}_n$. Then $\kappa(B' \cdot I_n) = \Omega(1.5^n)$.

On the other hand, let $B = \textsc{Seysen}(B')$. Then, because $\eta(B') = 1$, $S(B) = n^{O(\log n)}$ by Theorem 4.2.13. Clearly, $\lambda_i(\mathbb{Z}_n) = 1$ for all $i \in [n]$. On the other hand, $1 \leq \lambda_i(\mathcal{L}(B)) \leq \sqrt{n}$ for all $i \in [n]$, where the lower bound holds because $\min \|\tilde{\boldsymbol{b}}_i\| = 1$, and the upper bound comes from the fact that $\|\boldsymbol{b}'_i\| \leq \sqrt{n}$ for all $i \in [n]$.[3] It follows that $M(\mathbb{Z}^n, \mathcal{L}(B)) \leq \sqrt{n}$ and $M(\mathcal{L}(B), \mathbb{Z}^n) \leq 1$. Applying Lemma 4.3.2 to B and B^{-1}, we then get that $\kappa(B \cdot I_n) \leq n^{O(\log n)}$. $\square$

[3]In fact, $\lambda_n(\mathcal{L}(B)) = O(1)$.

Chapter 5

Algorithms for Computing Nearly Orthogonal and Well-Conditioned Lattice Bases

This chapter is based on the publication [Ben17].

5.1 Introduction

Any given basis $B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$ of a lattice $\mathcal{L}$ is not unique, and a common goal is to compute a *reduced* basis of $\mathcal{L}$, i.e., one which satisfies useful properties such as having short and nearly orthogonal vectors. The theory of basis reduction is intimately related to solving lattice problems, and is therefore a major area of study.

In terms of approximation algorithms, the seminal LLL algorithm [LLL82] efficiently computes a basis which yields an approximate solution to the shortest vector problem (SVP). Such LLL-reduced bases can also be used to approximately solve the closest vector problem (CVP) efficiently [Bab86], and have many other applications. In terms of slower but exact algorithms, Kannan's algorithm for exact SVP and CVP [Kan87] relies on computing HKZ-reduced bases [KZ73], which give a greedy way of formalizing of what it means to be a shortest-possible lattice basis.

A general way of formalizing what it means for a basis B to be short and orthogonal is according to its *orthogonality defect*, defined as

$$\delta(B) := \prod_{i=1}^{n} (\|\boldsymbol{b}_i\| / \|\tilde{\boldsymbol{b}}_i\|) = \Big(\prod_{i=1}^{n} \|\boldsymbol{b}_i\|\Big) / \det(\mathcal{L}), \tag{5.1}$$

where $\tilde{\boldsymbol{b}}_i$ is the ith Gram-Schmidt vector of B. The problem of computing bases with minimum orthogonality defect is called the Quasi Orthogonal Basis Problem (QOB). See [MG02] Chapter 7, Section 2 for a survey.

The orthogonality defect is a widely-used measure of the quality of lattice bases, and captures the quality of standard notions of reduced bases. It holds that LLL-reduced bases B have $\delta(B) \leq 2^{n(n-1)/4}$ (see, e.g., [Vaz01]), and that HKZ-reduced bases B have $\delta(B) \leq n^n$ and are within a $n^{n/2}$ factor of optimal (see [MG02] and Theorem 5.2.3). Furthermore, Minkowski-reduced bases (another greedy way of formalizing shortest-possible lattice bases) have orthogonality defect at most $2^{O(n^2)}$ [vdWG68], a characterization which is crucial to Helfrich's algorithm for computing them [Hel85].

The orthogonality defect also appears directly in applications. For example, the original security analysis of the well-known GGH encryption and signature schemes [GGH97] depends on the difficulty of computing a basis with low orthogonality defect.[1]

Standard notions of basis reduction including LLL-reduction and HKZ-reduction guarantee that the vectors in a reduced basis B are relatively short, but make no explicit guarantees about the lengths of vectors in the *dual basis* B^*. Some applications require short primal bases B, some require short dual bases B^*, and some require B to be well-conditioned so that B and B^* both have short vectors simultaneously. In particular, in Chapter 4 we used the existence of such well-conditioned bases to upper bound the distortion between two lattices.

To study the question of finding well-conditioned lattice bases, Seysen [Sey93] defined the matrix condition number $S(B) := \max_{i \in [n]} \|\boldsymbol{b}_i\| \|\boldsymbol{b}_i^*\|$. Trying to find lattice bases that are well-conditioned in the sense of Seysen is a natural problem in its own right, and recently has found applications such as ours. We call the problem of finding bases B which minimize $S(B)$ the Seysen basis problem.

For any full-rank matrix B, Hadamard's inequality asserts that $\delta(B) \geq 1$, while by the Cauchy-Schwarz inequality $S(B) = \max_{i \in [n]} \|\boldsymbol{b}_i\| \|\boldsymbol{b}_i^*\| \geq \max_{i \in [n]} |\langle \boldsymbol{b}_i, \boldsymbol{b}_i^* \rangle| = 1$. Therefore, one can view δ and S as measuring how tight Hadamard's inequality and the Cauchy-Schwarz inequality are for a basis B, respectively. The quantities δ and S are also related by the simple inequality $S(B) = \max_{i \in [n]} \|\boldsymbol{b}_i\| \|\boldsymbol{b}_i^*\| \geq \max_{i \in [n]} \|\boldsymbol{b}_i\| / \|\tilde{\boldsymbol{b}}_i\| \geq \delta(B)^{1/n}$.[2] I.e., $S(B)$ upper bounds the normalized orthogonality defect $\delta(B)^{1/n}$.

As his main result, Seysen [Sey93] showed that every lattice has a basis B such that $S(B) \leq n^{O(\log n)}$, and moreover that one can compute such a basis in $2^{O(n)}$-time. Seysen also implicitly showed that a basis B minimizing $S(B)$ lies inside

[1]Although the original GGH and related NTRU signature scheme [HPS98] have been cryptanalyzed [Ngu99, NR09], they continue to inspire related new schemes.

[2]This holds by Equation (4.7).

a ball of radius $n^{O(\log n)} \cdot \lambda_n$, where λ_n denotes the largest successive minimum of $\mathcal{L}$. This characterization yields an algorithm for computing an optimal Seysen basis: simply enumerate all bases B lying inside a ball of radius $n^{O(\log n)} \cdot \lambda_n$ and output the one which minimizes $S(B)$. However, the number of vectors lying inside such a ball depends on the parameter λ_n/λ_1 (the ratio of the largest and smallest successive minima of $\mathcal{L}$), so this algorithm's runtime may be exponential even for lattices of constant rank n. A similar characterization and algorithm work for QOB, but again the algorithm's runtime depends on λ_n/λ_1.

The orthogonality defect and Seysen's condition number are fundamental geometric quantities, and as such the problem of finding bases which minimize them is important. In this chapter we give algorithms which minimize δ and $(1+\varepsilon)$-approximately minimize S and run in time depending only on the rank n of the lattice times a polynomial in the input length. To the best of our knowledge, no such algorithms were previously known even for computing bases which approximately minimize either quantity within a poly(n) factor for lattices of rank n.

Although our algorithms have high enough runtime that they are mainly of theoretical interest, there are a number of ways in which they may be useful for applications. First, spending a large amount of time reducing a basis makes sense as a pre-processing step in contexts such as cryptography and coding theory where the goal is often to answer multiple CVP queries on the same lattice. Second, our main algorithmic technique of breaking a lattice into pieces according to its successive minima seems natural and likely has a number of other applications. Third, we describe directions for potentially getting faster runtimes while still using essentially the same algorithms.

	Approximation	Runtime	Notes
QOB	$n^{n/2}$	$2^{O(n)}$	HKZ bases.
	$k^{O(n(n/k+\log k))}$	$2^{O(k)}$	For any $\log n \le k \le n$. [GN08], this chapter.
	1	$n^{O(n^4)}$	This chapter.
Seysen	$k^{O(n/k+\log k)}$	$2^{O(k)}$	For any $\log n \le k \le n$. [Sey93, BDS16].
	1	$f(n, \lambda_n/\lambda_1)$	Implicit in [Sey93].
	$1+\varepsilon$	$(n/\varepsilon)^{O(n^3 \log n)}$	This chapter.

Table 5.1: A summary of algorithms for the approximate Quasi Orthogonal Basis (QOB) and approximate Seysen Basis problems, which correspond to finding bases that minimize δ and S respectively. Here n denotes the rank of the input lattice; the listed runtimes suppress polynomial dependence on the input length. f denotes an explicit function depending on n and λ_n/λ_1.

5.1.1 Summary of Results

In this chapter we show how to compute bases B which achieve minimal (resp. $(1+\varepsilon)$-approximately minimal) $\delta(B)$ (resp. $S(B)$) over all bases of $\mathcal{L}(B)$ in time that does not depend on λ_n/λ_1 and in polynomial space. Our main results are the algorithms summarized in the following pair of theorems.

Theorem 5.1.1 (QOB exact algorithm, informal)**.** *There exists an algorithm which given a lattice $\mathcal{L}$ of rank n outputs a basis B of $\mathcal{L}$ with $\delta(B) \le \delta(B')$ for all bases B' of $\mathcal{L}$. The algorithm runs in polynomial time for every fixed n and in polynomial space.*

Theorem 5.1.2 (Seysen Basis approximation scheme, informal)**.** *There exists an algorithm which given a lattice $\mathcal{L}$ of rank n and an $\varepsilon > 0$ outputs a basis B of $\mathcal{L}$ with $S(B) \le (1+\varepsilon) \cdot S(B')$ for all bases B' of $\mathcal{L}$. The algorithms runs in polynomial time for every fixed n and ε, and in polynomial space.*

Table 5.1 summarizes these and other algorithms for the Seysen basis and QOB problems. There and throughout the remainder of the paper we suppress polynomial dependence on the input length when analyzing the runtimes of algorithms.

We then show that a single convex body associated with a given lattice $\mathcal{L}$, namely a scaling of the Minkowski Ellipsoid $E(\mathcal{L})$, contains the nearly optimal basis output by Theorem 5.1.2. Let $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n \in \mathcal{L}$ denote linearly independent vectors achieving the successive minima $\lambda_1, \dots, \lambda_n$ of $\mathcal{L}$ (i.e. $\|\boldsymbol{v}_i\| = \lambda_i(\mathcal{L})$), let $\tilde{\boldsymbol{v}}_1, \dots, \tilde{\boldsymbol{v}}_n$ denote their Gram-Schmidt orthogonalization. The Minkowski Ellipsoid $E(\mathcal{L})$ is the ellipsoid whose ith axis is aligned with $\tilde{\boldsymbol{v}}_i$ and whose ith radius has length $\lambda_i(\mathcal{L})$.

Because such an ellipsoid contains relatively few additional lattice vectors, this in turn leads to a conceptually simpler algorithm for computing good Seysen bases which consists of enumerating all bases B within this ellipsoid and outputting the one with minimal $S(B)$. In fact, one can view our first algorithm for the Seysen basis problem as a constructive proof that a nearly optimal Seysen basis lies inside scaled Minkowski Ellipsoids.

Theorem 5.1.3 (Basis in scaled Minkowski Ellipsoid, informal)**.** *For every lattice $\mathcal{L}$ of rank n and every $\varepsilon > 0$ there exists a basis $B = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_n]$ of $\mathcal{L}$ such that $S(B) \leq (1 + \varepsilon) \cdot S(B')$ for all bases B' of $\mathcal{L}$, and $\boldsymbol{b}_1, \dots, \boldsymbol{b}_n \in t \cdot E(\mathcal{L})$ for some t depending only on n and ε.*

In order to prove our main theorems we first show a number of lemmas about the successive minima of a lattice in Section 5.2.3. Although their proofs are straightforward, some have not appeared before to the best of our knowledge and may be of independent interest. For example, Lemma 5.2.7 states that a sufficiently large gap in the successive minima of a lattice implies that the span of vectors achieving the first few successive minima in a lattice is orthogonal to the span of vectors achieving the first few successive minima in the corresponding dual lattice. The idea behind this lemma – that one can use gaps in the successive minima to decompose a lattice – is also the main idea behind our algorithms.

Finally, in Section 5.3.1 we observe that the slide-reduced bases of Gama and Nguyen [GN08] have relatively low orthogonality defect as a consequence of their low Gram-Schmidt decay. Slide-reduced bases are also relatively efficient to compute compared to the bases output by our exact algorithm, and provide a time-approximation quality tradeoff.

5.1.2 Techniques

We give a brief outline of our results and the ideas used in our algorithm while deferring definitions and formal statements. Let $V_k = \text{span}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)$, and let $\pi_k(\boldsymbol{x})$ denote the projection of $\boldsymbol{x}$ onto $V_k^{\perp}$.

The main idea behind our algorithms is to split a lattice into pieces according to large gaps in its successive minima, compute a basis for each of these pieces, and then lift the bases for each piece to form a basis of the whole lattice. Namely, our algorithms use three observations: (1) if there are no large gaps in the successive minima then we can simply enumerate an optimal basis in time depending only on n, (2) if $\mathcal{L} = \mathcal{L}_1 \oplus \mathcal{L}_2$ then an optimal basis B of $\mathcal{L}$ has the form $B = B_1 \oplus B_2$ where B_1, B_2 are bases of $\mathcal{L}_1, \mathcal{L}_2$ respectively, and (3) if there is a large multiplicative gap in the successive minima, i.e. λ_{k+1}/λ_k is large, then "$\mathcal{L} \approx (\mathcal{L} \cap V_k) \oplus \pi_k(\mathcal{L})$". Because of observation (3), a large gap in the successive minima allows us to take advantage of observation (2) and reduce the problem of finding a good basis for $\mathcal{L}$ to the subproblems of finding good bases for $(\mathcal{L} \cap V_k), \pi_k(\mathcal{L})$. In particular, our algorithms work by computing sub-bases $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k$ of B whose spans agree with the successive minima of $\mathcal{L}$ (i.e. bases satisfying $\text{span}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k) = V_k$) whenever λ_{k+1}/λ_k is sufficiently large.

The classic enumeration-based algorithms of Kannan [Kan87] for computing HKZ-reduced bases and Helfrich [Hel85] for computing Minkowski-reduced bases work by "repeatedly enumerating the next Gram-Schmidt vector" of a basis. Our algorithms extend this idea by enumerating basis blocks and then lifting the blocks to form a full basis, in a similar manner to the Block Korkine-Zolotareff-reduced (BKZ-reduced) bases of Schnorr [Sch87]. Helfrich's algorithm is the most similar to ours of any previous algorithm in that it uses repeated enumeration and lifting. It also runs in $2^{O(n^3)}$-time, which is comparable to the running time of our algorithms, showing that hard basis reduction problems may require high runtimes.

The technique of splitting a lattice into pieces according to its successive minima seems natural, and should have further applications. Similar ideas have appeared in other work. In particular, an algorithm by Haviv and Regev [HR14] for determining whether two lattices are isomorphic inspired our algorithm. Their algorithm works by splitting each lattice $\mathcal{L}$ into the sublattice $\mathcal{L} \cap V_k$ and the projected lattice $\pi_k(\mathcal{L})$ whenever there is *any* gap in the successive minima ($\lambda_{k+1} > \lambda_k$); our algorithm only does so when there is a large gap ($\lambda_{k+1} \gg \lambda_k$).

5.1.3 Open Questions

There are several natural open questions related to our work. The first is whether we can turn our approximation scheme for Seysen Bases into an exact algorithm. Our algorithm already enumerates optimal bases for projections of the lattice, but it's unclear how to lift these bases to a basis for the whole lattice without incurring small error.

Open Problem 5.1.4 (Exact Seysen basis FPT algorithm). *Find an algorithm which, on input a lattice $\mathcal{L}$, computes a basis B of $\mathcal{L}$ which achieves $S(B) = S(\mathcal{L})$*

in polynomial time for lattices of fixed rank.

The second question is whether the runtimes in our algorithms can be improved. One direction is to improve the runtime's dependence on the gaps in successive minima *inside* sublattices. Although the dependence is bounded as a function of n, it is still quite large.

The third question is whether our techniques yield algorithms for related problems. In particular, it seems that similar enumeration-based techniques may yield algorithms for other basis quality measures, and for the lattice distortion problem (LDP) studied in [BDS16].

Open Problem 5.1.5 (LDP approximation scheme)**.** *Can the techniques in this chapter be extended to give an approximation scheme or exact algorithm for the lattice distortion problem which runs in polynomial time for lattices of fixed rank?*

The approximation factor given by Babai's algorithm for CVP (described in Section 3.3.1) is a function of the basis used in his algorithm. A natural question is whether one can use our techniques to compute a basis which minimizes this quantity for any given lattice.

Open Problem 5.1.6 (Optimal Babai basis)**.** *Can the techniques in this chapter be extended to give an algorithm for computing a basis B which minimizes the quantity $(1 + \max_{i \in [n]}(\sum_{j=1}^{i} \|\tilde{\boldsymbol{b}}_j\|^2)/(\|\tilde{\boldsymbol{b}}_i\|^2))^{1/2}$ and runs in polynomial time for lattices of fixed rank? Is there such an algorithm for computing a basis which minimizes the similar quantity $\eta(B) := \max_{i \leq j} \frac{\|\tilde{\boldsymbol{b}}_i\|}{\|\tilde{\boldsymbol{b}}_j\|}$?*

5.1.4 Organization

In Section 5.2 we present background material about lattices, and prove a number of basic lemmas which will be useful in our subsequent analysis. In Section 5.3 we study bases with low orthogonality defect, and present the algorithm corresponding to Theorem 5.1.1. In Section 5.4 we study bases that are well-conditioned in the sense of Seysen, and present the algorithm corresponding to Theorem 5.1.2 and the proof of Theorem 5.1.3.

5.1.5 Acknowledgments

I would like to thank Daniel Dadush for many useful suggestions, and especially for suggesting a potential connection between Seysen bases and Minkowski Ellipsoids. I would like to thank Michael Walter for suggesting a potential connection between Seysen-reduction and the orthogonality defect of a basis. Finally, I would like to thank Oded Regev and Noah Stephens-Davidowitz for helpful comments.

5.2 Preliminaries

We will need the following theorem, which shows how to enumerate lattice points inside a Euclidean ball. We use the formulation of [HR14], which uses Kannan's algorithm to find a dual HKZ basis using low space, and observes that short lattice vectors have small coefficients when written in such a basis.

Theorem 5.2.1 (Lattice point enumeration, Corollary 2.16 in [HR14]). *Given a number $t \geq 1$ and an n-dimensional lattice $\mathcal{L}$, there exists an algorithm that enumerates all vectors $\boldsymbol{w} \in \mathcal{L}$ such that $\|\boldsymbol{w}\| \leq t \cdot \lambda_1(\mathcal{L})$ in $(t \cdot n)^{O(n)}$-time and using polynomial space.*

5.2.1 The Basis Quality Measures δ and S

5.2.1.1 The Quasi Orthogonal Basis Problem

Following [MG02] Chapter 7, Section 2, we define the problem of finding a basis which minimizes δ as the Quasi Orthogonal Basis problem (QOB). Recall that $\delta(B) := \prod_{i=1}^{n}(\|\boldsymbol{b}_i\|/\|\tilde{\boldsymbol{b}}_i\|)$, and let $\delta(\mathcal{L}) := \min_{B:\mathcal{L}(B)=\mathcal{L}} \delta(B)$ denote the minimal value of δ over all bases B of $\mathcal{L}$. Let $\delta(n) := \sup\{\delta(\mathcal{L}) : \mathcal{L} \text{ of rank } n\}$.

Definition 5.2.2. For any $\gamma = \gamma(n)$, the γ-approximate Quasi Orthogonal Basis problem is the search problem defined as follows. The input consists of a lattice $\mathcal{L}$ (specified by a basis $B' \in \mathbb{Q}^{m\times n}$). The goal is to output a basis B of $\mathcal{L}$ such that $\delta(B) \leq \gamma \cdot \delta(\mathcal{L})$.

The next fact follows directly from Minkowski's Second Theorem and the fact that $\|\boldsymbol{b}_i\| \leq \sqrt{i} \cdot \lambda_i$ for HKZ bases B (as proved by Lagarias et al. [LLS90]).

Theorem 5.2.3. *Let B be an HKZ basis of a lattice $\mathcal{L}$ of rank n. Then $\delta(B) \leq n^n$ and in particular $\delta(n) \leq n^n$.*

Micciancio and Goldwasser [MG02] use a very similar argument to show the "in particular" part, and that $\delta(B) \leq n^{n/2} \cdot \delta(\mathcal{L}(B))$ for HKZ bases B.

5.2.1.2 The Seysen Basis Problem

Recall that $S(B) := \max_{i\in[n]} \|\boldsymbol{b}_i\|\|\boldsymbol{b}_i^*\|$, and let $S(\mathcal{L}) := \min_{B:\mathcal{L}(B)=\mathcal{L}} S(B)$ denote the minimal value of S over all bases B of $\mathcal{L}$. Let $s(n) := \sup\{S(\mathcal{L}) : \mathcal{L} \text{ of rank } n\}$.

Definition 5.2.4. For any $\gamma = \gamma(n)$, the γ-approximate Seysen Basis problem is the search problem defined as follows. The input consists of a lattice $\mathcal{L}$ (specified by a basis $B' \in \mathbb{Q}^{m\times n}$). The goal is to output a basis B of $\mathcal{L}$ such that $S(B) \leq \gamma \cdot S(\mathcal{L})$.

We recall the upper bound from Theorem 4.2.13 which showed that $s(n) \leq n^{O(\log n)}$. Because of this theorem and the Cauchy-Schwarz inequality we get that $1 \leq S(\mathcal{L}) \leq s(n) = n^{O(\log n)}$ for every lattice $\mathcal{L}$. Therefore the *decision* variant of the Seysen basis problem is trivial for $\gamma \geq n^{\omega(\log n)}$.

5.2.1.3 Basic Properties of δ and S

It is not hard to show the following basic properties of δ and S.

Fact 5.2.5. *Let $B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n] \in \mathbb{R}^{m \times n}$. Then:*

1. *$\delta([\boldsymbol{b}_{\pi(1)}, \ldots, \boldsymbol{b}_{\pi(n)}]) = \delta(B)$ and $S([\boldsymbol{b}_{\pi(1)}, \ldots, \boldsymbol{b}_{\pi(n)}]) = S(B)$ for every permutation $\pi : [n] \to [n]$.*
2. *$\delta(OB) = \delta(B)$ and $S(OB) = S(B)$ for every orthogonal $O \in \mathbb{R}^{m \times n}$.*
3. *$S(B) = S(B^*)$ and therefore $S(\mathcal{L}) = S(\mathcal{L}^*)$.*

Call a basis B *sorted* if $\|\boldsymbol{b}_1\| \leq \cdots \leq \|\boldsymbol{b}_n\|$. By item 1 there always exists a sorted basis B which satisfies $S(B) = S(\mathcal{L}(B))$. By item 2, δ and S are invariant under an orthogonal change of basis.

5.2.2 Non-Optimality of HKZ Bases

HKZ-reduced bases give one way of formalizing what it means to be a shortest possible lattice basis. Nevertheless, there are HKZ bases B that do not minimize either δ or S. In fact, we show that an example of a poorly conditioned HKZ basis previously given in [LT08] and in Section 4.5.2 also has poor orthogonality defect. Therefore, HKZ reduction alone does not suffice to minimize δ or S.

Let B be the $n \times n$ upper triangular basis with diagonal entries equal to 1 and off-diagonal upper triangular entries equal to $-\frac{1}{2}$. I.e., B has entries

$$b_{ij} = \begin{cases} 0 & \text{if } j < i, \\ 1 & \text{if } j = i, \\ -\frac{1}{2} & \text{if } j > i. \end{cases}$$

Let B' be the $n \times n$ bidiagonal basis with entries equal to 1 on the main diagonal, and entries equal to $-\frac{3}{2}$ on the diagonal above. I.e., B' has entries

$$b'_{ij} = \begin{cases} 1 & \text{if } j = i, \\ -\frac{3}{2} & \text{if } j = i + 1, \\ 0 & \text{otherwise.} \end{cases}$$

It is not hard to show that B is an HKZ basis, and that $\mathcal{L}(B) = \mathcal{L}(B')$. It is also not hard to show that $S(B) \geq \Omega(1.5^n)$, that $\delta(B) \geq n^{\Omega(n)}$, that $S(B') \geq \Omega(1.5^n)$ and that $\delta(B') \leq 2^{O(n)}$. Furthermore, Theorem 4.2.13 asserts that there exists a basis B'' of $\mathcal{L}(B)$ with $S(B'') \leq n^{O(\log n)}$. Comparing $\delta(B)$ with $\delta(B')$ and $S(B)$ with $S(B'')$ we then have that HKZ bases may be exponentially far from optimal in terms of minimizing both S and δ. Moreover, comparing $S(B')$ with $S(B'')$ shows that bases with low orthogonality defect may still be poorly conditioned.

The non-optimality of B comes from its off-diagonal elements both for minimizing δ and for minimizing S. I.e., the above examples show that size-reduction can be non-optimal. It is an interesting question whether there always exist bases minimizing δ and S whose Gram-Schmidt vectors are the same as some HKZ basis.

Open Problem 5.2.6. *For every lattice $\mathcal{L}$ of rank n, is there an HKZ-basis B of*

$\mathcal{L}$ and some $U \in N(n, \mathbb{Z})$ such that $\delta(BU) = \delta(\mathcal{L})$? Are there always such B and U so that $S(BU) = S(\mathcal{L})$?

5.2.3 The Successive Minima of Sublattices and Projected Lattices

By *vectors that achieve the successive minima of $\mathcal{L}$* we mean linearly independent vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathcal{L}$ that satisfy $\|\boldsymbol{v}_i\| = \lambda_i(\mathcal{L})$ for a lattice $\mathcal{L}$ of rank n. When the underlying lattice $\mathcal{L}$ is clear from context, we use $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ to denote vectors that achieve the successive minima of $\mathcal{L}$, and let $V_k = \text{span}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)$. Similarly, we use $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n$ to denote vectors that achieve the successive minima of $\mathcal{L}^*$, and let $W_k = \text{span}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k)$.

We write the projection π_k as shorthand for $\pi_k^{(V)}$, i.e. projection onto the orthogonal complement of V_k. Given a projection π and a matrix $B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n] \in \mathbb{R}^{m \times n}$, let $\pi(B) = [\pi(\boldsymbol{b}_1), \ldots, \pi(\boldsymbol{b}_n)]$. Note that the projections $\pi(\boldsymbol{b}_i)$ still lie inside the ambient space $\mathbb{R}^m$.

In this section we show several useful facts about the lattices $\mathcal{L} \cap V_k$ and $\pi_k(\mathcal{L})$. We first show that a sufficiently large gap in the successive minima implies useful structure in the subspaces V_k, W_{n-k}.

Lemma 5.2.7. *Let $\mathcal{L}$ be a lattice of rank n, and assume that $\lambda_{k+1}(\mathcal{L})/\lambda_k(\mathcal{L}) > n$ for some $k \in [n-1]$. Then $V_k \perp W_{n-k}$.*

Proof. Let $i \in [k]$ and $j \in [n-k]$. Using the Cauchy-Schwarz inequality and the upper bound in Theorem 3.2.1,

$$|\langle \boldsymbol{v}_i, \boldsymbol{w}_j \rangle| \leq \|\boldsymbol{v}_i\| \|\boldsymbol{w}_j\| \leq \lambda_k \cdot \lambda_{n-k}^* < \frac{\lambda_{k+1}}{n} \cdot \lambda_{n-k}^* \leq 1. \tag{5.2}$$

Because primal and dual vectors must have integral inner product $|\langle \boldsymbol{v}_i, \boldsymbol{w}_j \rangle| < 1$ implies that $\langle \boldsymbol{v}_i, \boldsymbol{w}_j \rangle = 0$. Because Equation (5.2) holds for all $i \in [k]$, $j \in [n-k]$, it follows that $V_k \perp W_{n-k}$. $\square$

The following lemma establishes relations between the successive minima of a lattice $\mathcal{L}$ and the lattices $\mathcal{L} \cap V_k$ and $\pi_k(\mathcal{L})$. These bounds are folklore; the upper bound in Equation (5.3) has appeared, e.g., in [LLS90]. These bounds and those in the following pair of lemmas roughly say that $\lambda_{k+j}(\mathcal{L}) \approx \lambda_j(\pi_k(\mathcal{L}))$.

Lemma 5.2.8. *Let $\mathcal{L}$ be a lattice of rank n, and let $k \in [n-1]$. Then:*

1. *For every $j \in [k]$, $\lambda_j(\mathcal{L} \cap V_k) = \lambda_j(\mathcal{L})$.*
2. *For every $j \in [n-k]$,*

$$\lambda_{k+j}(\mathcal{L}) - \frac{\sqrt{k}}{2}\lambda_k(\mathcal{L}) \leq \lambda_j(\pi_k(\mathcal{L})) \leq \lambda_{k+j}(\mathcal{L}) \tag{5.3}$$

Proof. For every $j \in [k]$, we have that $\boldsymbol{v}_1, \dots, \boldsymbol{v}_j \in \mathcal{L} \cap V_k$ so $\lambda_j(\mathcal{L} \cap V_k) \leq \lambda_j(\mathcal{L})$. On the other hand, $\mathcal{L} \cap V_k \subseteq \mathcal{L}$, so $\lambda_j(\mathcal{L} \cap V_k) \geq \lambda_j(\mathcal{L})$. This proves item 1.

We have that $\pi_k(\boldsymbol{v}_{k+1}), \dots, \pi_k(\boldsymbol{v}_{k+j}) \in \pi_k(\mathcal{L})$ are linearly independent by the linear independence of $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$. Therefore $\lambda_j(\pi_k(\mathcal{L})) \leq \max_{\ell \in [j]} \|\pi_k(\boldsymbol{v}_{k+\ell})\| \leq \lambda_{k+j}(\mathcal{L})$, proving the upper bound in item 2.

Let $\boldsymbol{u}_1, \dots, \boldsymbol{u}_{n-k} \in \pi_k(\mathcal{L})$ be vectors achieving the successive minima of $\pi_k(\mathcal{L})$, and let $j \in [n-k]$. By the triangle inequality and the definition of the covering radius, there exist liftings $\boldsymbol{x}_1, \dots, \boldsymbol{x}_j \in \mathcal{L}$ of $\boldsymbol{u}_1, \dots, \boldsymbol{u}_j$ such that $\pi_k(\boldsymbol{x}_\ell) = \boldsymbol{u}_\ell$, and $\|\boldsymbol{x}_\ell\| \leq \|\boldsymbol{u}_\ell\| + \mu(\mathcal{L} \cap V_k)$ for every $\ell \in [j]$. By the linear independence of $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{n-k}$, we therefore have that $\lambda_{k+j}(\mathcal{L}) \le \max_{\ell \in [j]} \|\boldsymbol{x}_\ell\| \le \max_{\ell\in[j]} \|\boldsymbol{u}_\ell\| + \mu(\mathcal{L} \cap V_k) = \lambda_j(\pi_k(\mathcal{L})) + \mu(\mathcal{L} \cap V_k)$. Finally, using Theorem 3.2.2 and item 1, $\mu(\mathcal{L} \cap V_k) \le \frac{\sqrt{k}}{2}\lambda_k(\mathcal{L} \cap V_k) = \frac{\sqrt{k}}{2}\lambda_k(\mathcal{L})$. We then have $\lambda_{k+j}(\mathcal{L}) \le \lambda_j(\pi_k(\mathcal{L})) + \frac{\sqrt{k}}{2}\lambda_k(\mathcal{L})$. Subtracting $\frac{\sqrt{k}}{2}\lambda_k(\mathcal{L})$ from both sides proves the lower bound in item 2. $\square$

By applying the lower and upper bounds in Equation (5.3), we get the following bounds for $k \in [n-1]$ and $j \in [n-k-1]$, which says that the gaps in the successive minima in the projection of a lattice are close to the corresponding gaps in the original lattice.

$$\frac{\lambda_{k+j+1}(\mathcal{L}) - \frac{\sqrt{k}}{2}\lambda_k(\mathcal{L})}{\lambda_{k+j}(\mathcal{L})} \le \frac{\lambda_{j+1}(\pi_k(\mathcal{L}))}{\lambda_j(\pi_k(\mathcal{L}))} \le \frac{\lambda_{k+j+1}(\mathcal{L})}{\lambda_{k+j}(\mathcal{L}) - \frac{\sqrt{k}}{2}\lambda_k(\mathcal{L})}. \tag{5.4}$$

We also get the following pair of lemmas.

Lemma 5.2.9. *Assume that $\lambda_{k+1}/\lambda_k > c$ for some $1 \le k \le n-1$ and $c = c(n) \ge \sqrt{n}$. Let $1 \le j \le n-k$. Then $\lambda_{k+j}(\mathcal{L})/\lambda_j(\pi_k(\mathcal{L})) \le \left(1 + \frac{1}{2c/\sqrt{k}-1}\right) \le 2$.*

Proof. By applying the lower bound in Equation (5.3) to the denominator, and then dividing the numerator and denominator by $\lambda_{k+j}(\mathcal{L})$,

$$\frac{\lambda_{k+j}(\mathcal{L})}{\lambda_j(\pi_k(\mathcal{L}))} \le \frac{\lambda_{k+j}(\mathcal{L})}{\lambda_{k+j}(\mathcal{L}) - \frac{\sqrt{k}}{2}\lambda_k(\mathcal{L})} \le \frac{1}{1 - \frac{\sqrt{k}}{2c}} = \left(1 + \frac{1}{2c/\sqrt{k} - 1}\right) \le 2.$$

$\square$

Lemma 5.2.10. *Assume that $\lambda_{k+1}/\lambda_k > c$ and that $\lambda_{k+j+1}/\lambda_{k+j} \le c$ for some $1 \le k \le n-1$, $1 \le j \le n-k-1$, and $c = c(n) \ge \sqrt{n}$. Then $\lambda_{j+1}(\pi_k(\mathcal{L}))/\lambda_j(\pi_k(\mathcal{L})) \le \left(1 + \frac{1}{2c/\sqrt{k}-1}\right) \cdot c \le 2c$.*

Proof. By applying the upper bound in Equation (5.4), and then dividing the numerator and denominator by $\lambda_{k+j}(\mathcal{L})$,

$$\frac{\lambda_{j+1}(\pi_k(\mathcal{L}))}{\lambda_j(\pi_k(\mathcal{L}))} \leq \frac{\lambda_{k+j+1}(\mathcal{L})}{\lambda_{k+j}(\mathcal{L}) - \frac{\sqrt{k}}{2}\lambda_k(\mathcal{L})} \leq \frac{c}{1-\frac{\sqrt{k}}{2c}} = c \cdot \left(1 + \frac{1}{2c/\sqrt{k}-1}\right) \leq 2c.$$

□

We next show a correspondence between sublattices and projected lattices related to the successive minima when $V_k \perp W_{n-k}$. We say that a linear subspace S is a *lattice subspace* of $\mathcal{L}$ if there exist $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in \mathcal{L}$ such that $\mathrm{span}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) = S$.

Lemma 5.2.11. *Let $\mathcal{L}$ be a lattice of rank n, and assume that $V_k \perp W_{n-k}$ for some $k \in [n-1]$. Then:*

1. $(\mathcal{L} \cap V_k)^* = \pi^{(W)}_{n-k}(\mathcal{L}^*)$.

2. $\mathcal{L}^* \cap W_{n-k} = \pi_k(\mathcal{L})^*$.

Proof. Let $\mathcal{L}$ be a lattice and let S be a lattice subspace of $\mathcal{L}$. Then it holds that $(\mathcal{L} \cap S)^* = \pi_S(\mathcal{L}^*)$ (see, e.g., [Dad12, Lemma 2.4.1]).

Applying this fact to $\mathcal{L}$ with $S = V_k$, $(\mathcal{L} \cap V_k)^* = \pi_{V_k}(\mathcal{L}^*)$. Using the assumption that $V_k \perp W_{n-k}$ we additionally have $\pi_{V_k}(\mathcal{L}^*) = \pi^{(W)}_{n-k}(\mathcal{L}^*)$, which proves item 1. Applying the same argument to $\mathcal{L}^*$ with $S = W_{n-k}$ we get that $(\mathcal{L}^* \cap W_{n-k})^* = \pi_{W_{n-k}}(\mathcal{L}) = \pi_k(\mathcal{L})$. Item 2 then follows by taking duals. □

Finally we show an equivalence between the subspaces spanned by vectors achieving the successive minima of a lattice, and vectors achieving the successive minima of a projection of the lattice.

Lemma 5.2.12. *Let $\mathcal{L}$ be a lattice of rank n, let $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n \in \mathcal{L}$ be vectors that achieve the successive minima of $\mathcal{L}$, and let $\boldsymbol{u}_1, \dots, \boldsymbol{u}_{n-i} \in \pi_i(\mathcal{L})$ be vectors that achieve the successive minima of $\pi_i(\mathcal{L})$ for some $i \in [n-1]$. Assume that $\lambda_{k+1}(\mathcal{L})/\lambda_k(\mathcal{L}) > \frac{\sqrt{k}}{2} + 1$ for some $k > i$. Then* $\operatorname{span}(\tilde{\boldsymbol{v}}_{i+1}, \dots, \tilde{\boldsymbol{v}}_k) = \operatorname{span}(\boldsymbol{u}_1, \dots, \boldsymbol{u}_{k-i})$.

Proof. By definition $\boldsymbol{u}_1, \dots, \boldsymbol{u}_{k-i} \notin V_i$ so it suffices to show that $\boldsymbol{u}_1, \dots, \boldsymbol{u}_{k-i} \in V_k$. Suppose not. Then $\boldsymbol{u}_j \notin V_k$ for some $j \in [k-i]$. Using the triangle inequality and the definition of the covering radius, there exists a lifting $\boldsymbol{x} \in \mathcal{L} \setminus V_k$ of $\boldsymbol{u}_j$ such that $\pi_i(\boldsymbol{x}) = \boldsymbol{u}_j$, and $\|\boldsymbol{x}\| \le \|\boldsymbol{u}_j\| + \mu(\mathcal{L} \cap V_i)$. Using Theorem 3.2.2, Lemma 5.2.8 item 1, and the upper bound in Equation (5.3), $\|\boldsymbol{u}_j\| + \mu(\mathcal{L} \cap V_i) \le \|\boldsymbol{u}_j\| + \frac{\sqrt{i}}{2} \cdot \lambda_i(\mathcal{L} \cap V_i) \le \lambda_{i+j}(\mathcal{L}) + \frac{\sqrt{i}}{2} \cdot \lambda_i(\mathcal{L}) \le (\frac{\sqrt{k}}{2} + 1) \cdot \lambda_k(\mathcal{L})$. But because $\boldsymbol{x} \notin V_k$, this implies that $\lambda_{k+1}(\mathcal{L}) \le \|\boldsymbol{x}\| \le (\frac{\sqrt{k}}{2} + 1) \cdot \lambda_k(\mathcal{L})$, which is a contradiction. □

5.3 Algorithms for QOB

5.3.1 Approximation Algorithms

We first show that the slide-reduced bases of Gama and Nguyen [GN08] give a time-approximation quality tradeoff for QOB. Let $\eta(B) := \max_{1 \le i \le j \le n} \|\tilde{\boldsymbol{b}}_i\| / \|\tilde{\boldsymbol{b}}_j\|$ denote the Gram-Schmidt decay of a basis. In Section 4.2.5.1 we showed how to bound the Gram-Schmidt decay of slide-reduced bases. Here we use these bounds to conclude that slide-reduced bases have low orthogonality defect as well.

Lemma 5.3.1. *Let B be a size-reduced basis of rank n. Then $\delta(B) \le \sqrt{n!} \cdot \eta(B)^n$.*

Proof. For every $i \in [n]$,

$$\|\boldsymbol{b}_i\|^2 \le \|\tilde{\boldsymbol{b}}_i\|^2 + \frac{1}{4}\sum_{j=1}^{i-1}\|\tilde{\boldsymbol{b}}_j\|^2 \le \|\tilde{\boldsymbol{b}}_i\|^2 + \frac{(i-1)\cdot\eta(B)^2}{4}\|\tilde{\boldsymbol{b}}_i\|^2 \le \frac{(i+3)\cdot\eta(B)^2}{4}\|\tilde{\boldsymbol{b}}_i\|^2.$$

Therefore, $\delta(B) = \prod_{i=1}^{n}(\|\boldsymbol{b}_i\|/\|\tilde{\boldsymbol{b}}_i\|) \le \sqrt{n!}\cdot\eta(B)^n$. □

A bound on the orthogonality defect of slide-reduced bases follows immediately.

Proposition 5.3.2. *For every $\log n \le k \le n$ there exists an algorithm that takes as input a lattice $\mathcal{L}$ of rank n and outputs a basis B of $\mathcal{L}$ satisfying $\delta(B) \le k^{O(n(n/k+\log k))}$. The algorithm runs in $2^{O(k)}$ time.*

Proof. Combine Proposition 4.2.11 and Lemma 5.3.1. □

In the $k = n$ regime, Proposition 5.3.2 yields an upper bound of $\delta(B) \le n^{O(n\log n)}$, which is worse than the n^n bound for HKZ bases whose proof uses properties of HKZ-reduced bases other than their Gram-Schmidt decay. In the $k = \log n$ regime, Proposition 5.3.2 shows that there is a polynomial time algorithm which yields an upper bound of $\delta(B) \le 2^{O(n^2\log\log n/\log n)}$, which is slightly better than the $2^{O(n^2)}$ bound guaranteed by LLL-reduced bases.

5.3.2 An Exact Algorithm

The following enumeration-based algorithm computes a basis B that achieves $\delta(B) = \delta(\mathcal{L})$ in time depending only on n. We will use the same idea of "enumerating blocks according to gaps in the successive minima" in our approximation scheme for Seysen bases described in Section 5.4.4. The main differences are that here (1) the enumeration of later blocks depends on previous blocks, and (2) the algorithm lifts blocks to a full basis in a different way, which allows us to get an exact algorithm.

We recall the definition of CVP-reduction from Section 3.3.3. The *CVP-reduction* of a vector CVP-Red$(\boldsymbol{v}, \mathcal{L})$ denote the vector $\boldsymbol{v}' := \boldsymbol{v} - \boldsymbol{x}$, where $\boldsymbol{x} := \arg\min_{\boldsymbol{y} \in \mathcal{L}} \|\boldsymbol{v} - \boldsymbol{y}\|$.

```
Algorithm 2: OrthDefectMin($\mathcal{L}$)
    Input: A lattice $\mathcal{L}$ of rank $n$ (specified by a basis $B' \in \mathbb{Q}^{n \times n}$).
    Output: A basis $B$ of $\mathcal{L}$ achieving $\delta(B) = \delta(\mathcal{L})$.
    $K \leftarrow \{k \in [n-1] : \lambda_{k+1}(\mathcal{L})/\lambda_k(\mathcal{L}) > \delta(n)\} \cup \{n\}$
    return OrthDefectAux($\mathcal{L}, \emptyset, K$)
```

Theorem 5.3.3. OrthDefectMin$(\mathcal{L})$ *computes a basis* B *of* $\mathcal{L}$ *satisfying* $\delta(B) = \delta(\mathcal{L})$ *in* $n^{O(n^4)}$*-time and polynomial space.*

Proof. Because $\det(\mathcal{L}) \leq \prod_{i=1}^n \lambda_i(\mathcal{L})$, $\delta(B) \geq \prod_{i=1}^n \|\boldsymbol{b}_i\|/\lambda_i(\mathcal{L})$. Combining this with the fact that $\delta(n) \leq n^n$ from Theorem 5.2.3, we have that if B is sorted and $\|\boldsymbol{b}_i\|/\lambda_i > n^n$ for some i then B is non-optimal. We use this fact to prove the correctness of OrthDefectMin by induction.

Let $K := \{k \in [n-1] : \lambda_{k+1}(\mathcal{L})/\lambda_k(\mathcal{L}) > n^n\} \cup \{n\}$, and let $k' := \min K$. By the preceding argument in the base case an optimal basis B of $\mathcal{L}$ must contain k' vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{k'} \in \mathcal{L} \cap V_{k'}$, which we can assume without loss of generality come first since δ is invariant under permutation of basis vectors.

Further, suppose that $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k \in \mathcal{L} \cap V_k$ is a prefix of an optimal basis B, and that $k' \in K$ with $k' > k$. Then similarly there must exist vectors $\boldsymbol{b}_{k+1}, \ldots, \boldsymbol{b}_{k'} \in \mathcal{L} \cap (V_{k'} \setminus V_k)$ such that $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k, \boldsymbol{b}_{k+1}, \ldots, \boldsymbol{b}_{k'}$ can be extended to an optimal basis of $\mathcal{L}$. Moreover, for an optimal basis, we must have (1) that $\|\pi_k(\boldsymbol{b}_i)\| \leq \|\boldsymbol{b}_i\| \leq n^n \lambda_{k'}(\mathcal{L})$ and (2) that $\|\boldsymbol{b}_i\| = \|$CVP-Red$(\pi_k(\boldsymbol{b}_i), \mathcal{L} \cap V_k)\|$. It follows that

Algorithm 3: $\textsc{OrthDefectAux}(\mathcal{L}, S, K)$

```
Input: A lattice $\mathcal{L}$ of rank $n$ (specified by a basis $B' \in \mathbb{Q}^{n\times n}$), a set of $k$
       linearly independent vectors $S = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k\} \subseteq \mathcal{L}$ which can be
       extended to a basis of $\mathcal{L}$, the set $K \subseteq [n]$ of all indices $k'$ such that
       $k < k' < n$ and $\lambda_{k'+1}(\mathcal{L})/\lambda_{k'}(\mathcal{L}) \leq \delta(n)$, and the index $n$.
Output: Vectors $\boldsymbol{b}_{k+1}, \ldots, \boldsymbol{b}_n$ such that $B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$ is a basis of $\mathcal{L}$
        which satisfies $\delta(B) \leq \delta(B')$ among all bases $B'$ of $\mathcal{L}$ prefixed with
        $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k$.
$k' \leftarrow \min K$; $d \leftarrow \infty$
if $k = 0$ then
    $X \leftarrow \{\boldsymbol{x} \in \mathcal{L} : \|\boldsymbol{x}\| \leq n^n \cdot \lambda_{k'}(\mathcal{L})\}$
else
    $X \leftarrow \{\text{CVP-}\textsc{Red}(\boldsymbol{x}, \mathcal{L} \cap V_k) : \boldsymbol{x} \in \pi_k(\mathcal{L}) \text{ and } \|\boldsymbol{x}\| \leq \delta(n) \cdot \lambda_{k'}(\mathcal{L})\}$
end
for $\boldsymbol{b}'_{k+1}, \ldots, \boldsymbol{b}'_{k'} \in X^{k'-k}$ s.t. $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k, \boldsymbol{b}'_{k+1}, \ldots, \boldsymbol{b}'_{k'}]$ can be extended to a
  basis of $\mathcal{L}$ do
    if $|K| > 1$ then
        $\boldsymbol{b}'_{k'+1}, \ldots, \boldsymbol{b}'_n \leftarrow \textsc{OrthDefectAux}(\mathcal{L}, S \cup \{\boldsymbol{b}'_{k+1}, \ldots, \boldsymbol{b}'_{k'}\}, K \setminus \{k'\})$
    end
    if $\delta([\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k, \boldsymbol{b}'_{k+1}, \ldots, \boldsymbol{b}'_n]) < d$ then
        $d \leftarrow \delta([\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k, \boldsymbol{b}'_{k+1}, \ldots, \boldsymbol{b}'_n])$
        $\boldsymbol{b}_{k+1}, \ldots, \boldsymbol{b}_n \leftarrow \boldsymbol{b}'_{k+1}, \ldots, \boldsymbol{b}'_n$
    end
end
return $\boldsymbol{b}_{k+1}, \ldots, \boldsymbol{b}_n$
```

$\boldsymbol{b}_{k+1}, \ldots, \boldsymbol{b}_{k'} = \boldsymbol{b}'_{k+1}, \ldots, \boldsymbol{b}'_{k'}$ for one of the tuples $\boldsymbol{b}'_{k+1}, \ldots, \boldsymbol{b}'_{k'} \in X^{k'-k}$, and the correctness of OrthDefectMin then follows inductively.

We next bound the time complexity of OrthDefectMin. Let $\ell = k' - k$ denote the length of a block $\boldsymbol{b}_{k+1}, \ldots, \boldsymbol{b}_{k'}$ of vectors enumerated at some stage of OrthDefectMin. In the case where $k = 0$, we have that $\lambda_{k'}(\mathcal{L})/\lambda_1(\mathcal{L}) \leq (n^n)^{k'-1} = n^{(\ell-1)n}$. In the case where $k > 0$, because $\lambda_{k+1}/\lambda_k > n^n$ we have by Lemma 5.2.9 that $\lambda_{k+j}(\mathcal{L})/\lambda_j(\pi_k(\mathcal{L})) \leq 2$ for $1 \leq j \leq \ell$. Furthermore, because $\lambda_{k+j+1}/\lambda_{k+j} \leq n^n$ for $1 \leq j \leq \ell - 1$, we have by Lemma 5.2.10 that $\lambda_{j+1}(\pi_k(\mathcal{L}))/\lambda_j(\pi_k(\mathcal{L})) \leq 2n^n$. Therefore,

$$\lambda_{k'}(\mathcal{L})/\lambda_1(\pi_k(\mathcal{L})) \leq 2\lambda_\ell(\pi_k(\mathcal{L}))/\lambda_1(\pi_k(\mathcal{L})) \leq 2(2n^n)^{\ell-1} \leq n^{O(\ell n)}.$$

It follows that for every $\boldsymbol{x} \in X$ the vector $\pi_k(\boldsymbol{x})$ lies in a ball of radius $n^{O(\ell n)} \cdot \lambda_1(\pi_k(\mathcal{L}))$. At each stage we need only enumerate points in the ℓ-dimensional lattice $\pi_k(\mathcal{L}) \cap V_{k'}$, so by Theorem 5.2.1 we can enumerate all such vectors in $n^{O(\ell^2 n)}$-time and polynomial space, and therefore all ℓ-tuples of such vectors in $n^{O(\ell^3 n)}$-time and polynomial space. Lifting each enumerated vector via CVP-Red amounts to solving CVP on a k-dimensional lattice and therefore takes $k^{O(k)} \leq n^{O(n)}$-time and polynomial space. Therefore, $n^{O(\ell^3 n)}$ also bounds the total of amount of time and space required to compute X^ℓ.

Let $T_n(m) := \max_{1 \leq \ell \leq m} n^{O(\ell^3 n)} \cdot T_n(m - \ell)$. Then the total running time of OrthDefectMin is bounded by $T_n(n)$, which solves to $T_n(n) = n^{O(n^4)}$. □

We remark on one simple optimization to OrthDefectMin. Namely, when setting X in the "else" branch in OrthDefectMinAux, it suffices to consider vectors $\boldsymbol{x}$ with $\|\boldsymbol{x}\| \leq \delta(n)/\delta([\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k]) \cdot \lambda_{k'}(\mathcal{L})$. The factor of $\delta([\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k])$ in the denominator accounts for the orthogonality defect of the prefix of the basis computed so far.

5.4 Approximation Schemes for Seysen Bases

5.4.1 Finding Optimal Seysen Bases via Enumeration

In this section we present a simple algorithm for enumerating bases which minimize S. However, its runtime depends on the parameter λ_n/λ_1, and therefore may be unbounded in n. Nevertheless it will serve as a useful subroutine in our subsequent algorithm for computing blocks of a basis. (We previously used a similar enumeration technique for finding the blocks in an orthogonality defect minimizing basis in Section 5.3.2.) Seysen [Sey93] used a similar line of reasoning to upper bound the number of bases B with $S(B)$ smaller than a given value.

We recall Lemma 4.3.1, which says that for a sorted basis $B = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_n]$ of a lattice $\mathcal{L}$, $S(B) \geq \|\boldsymbol{b}_k\|/\lambda_k(\mathcal{L})$ for all $k \in [n]$. We get the following corollary.

Corollary 5.4.1. *Let $B = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_n]$ be a basis of $\mathcal{L}$ satisfying $S(B) \leq c \cdot S(\mathcal{L})$ for some $c \geq 1$. Then for every $k \in [n]$, there exist k basis vectors $\boldsymbol{b}_{i_1}, \dots, \boldsymbol{b}_{i_k}$ such that $\|\boldsymbol{b}_{i_j}\| \leq c \cdot s(n) \cdot \lambda_k(\mathcal{L})$ for every $j \in [k]$. In particular, $\|\boldsymbol{b}_i\| \leq c \cdot s(n) \cdot \lambda_n(\mathcal{L})$ for every $i \in [n]$.*

Corollary 5.4.1 and Theorem 5.2.1 yield a simple enumeration-based algorithm for computing an optimal Seysen basis.

Proposition 5.4.2. *There exists an algorithm* EnumerateSeysenOpt *which takes a lattice $\mathcal{L}$ as input and outputs a basis B of $\mathcal{L}$ satisfying $S(B) = S(\mathcal{L})$ in $(s(n) \cdot \lambda_n(\mathcal{L})/\lambda_1(\mathcal{L}))^{O(n^2)}$ time and polynomial space.*

Proof. There is a polynomial time, dimension-preserving reduction from the successive minima problem to CVP [Mic08], and therefore λ_n/λ_1 can be computed in $n^{O(n)}$-time and polynomial space using Kannan's algorithm [Kan87].

By Corollary 5.4.1 every optimal Seysen basis $B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$ is such that $\|\boldsymbol{b}_i\| \leq s(n) \cdot \lambda_n(\mathcal{L})$ for every i. By Theorem 5.2.1 we can enumerate all vectors $\boldsymbol{w} \in \mathcal{L}$ such that $\|\boldsymbol{w}\| \leq s(n) \cdot \lambda_n(\mathcal{L})$ in $(s(n) \cdot \lambda_n(\mathcal{L})/\lambda_1(\mathcal{L}))^{O(n)}$ time, and therefore we can enumerate all n-tuples of such vectors in $(s(n) \cdot \lambda_n(\mathcal{L})/\lambda_1(\mathcal{L}))^{O(n^2)}$ time. We therefore obtain an optimal Seysen basis by taking the n-tuple of such vectors $B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$ which achieves minimal $S(B)$ among all those that are bases of $\mathcal{L}$. □

5.4.2 A Lower Bound on $S(\mathcal{L})$

Lemma 5.4.3. *Let $\mathcal{L}$ be a lattice of rank n with $\lambda_{k+1}(\mathcal{L})/\lambda_k(\mathcal{L}) > s(n)$. Then* $\max\{S(\mathcal{L} \cap V_k), S(\pi_k(\mathcal{L}))\} \leq S(\mathcal{L})$.

Proof. Let B be a sorted basis of $\mathcal{L}$ which achieves $S(B) = S(\mathcal{L})$. Because $\lambda_{k+1}(\mathcal{L})/\lambda_k(\mathcal{L}) > s(n)$ we have by Corollary 5.4.1 that $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k \in \mathcal{L} \cap V_k$.

Let $C = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k] = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k]$. Then C is a basis of $\mathcal{L} \cap V_k$, and we claim that $\boldsymbol{c}_i^* = \pi_{V_k}(\boldsymbol{b}_i^*)$ for $i \in [k]$. Indeed for $i, j \in [k]$, $\pi_{V_k}(\boldsymbol{b}_j^*) \in \operatorname{span}(C)$ and $\langle \boldsymbol{c}_i, \pi_{V_k}(\boldsymbol{b}_j^*) \rangle = \langle \boldsymbol{b}_i, \pi_{V_k}(\boldsymbol{b}_j^*) \rangle = \langle \boldsymbol{b}_i, \boldsymbol{b}_j^* \rangle$. The last expression is equal to 1 if $i = j$ and 0 otherwise as required. Then for every $i \in [k]$ we have that $\|\boldsymbol{c}_i\| \|\boldsymbol{c}_i^*\| \leq \|\boldsymbol{b}_i\| \|\boldsymbol{b}_i^*\|$ since $\boldsymbol{c}_i = \boldsymbol{b}_i$ and $\boldsymbol{c}_i^* = \pi_{V_k}(\boldsymbol{b}_i^*)$. Therefore, $S(\mathcal{L} \cap V_k) \leq S(C) = \max_{i \in [k]} \|\boldsymbol{c}_i\| \|\boldsymbol{c}_i^*\| \leq \max_{i \in [k]} \|\boldsymbol{b}_i\| \|\boldsymbol{b}_i^*\| \leq S(B) = S(\mathcal{L})$.

A similar argument shows that, taking $D = [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_{n-k}] = [\pi_k(\boldsymbol{b}_{k+1}), \ldots, \pi_k(\boldsymbol{b}_n)]$, D is a basis of $\pi_k(\mathcal{L})$ and $S(\pi_k(\mathcal{L})) \leq S(D) \leq S(B) = S(\mathcal{L})$. The result follows by combining the lower bounds on $S(\mathcal{L})$.

□

5.4.3 Seysen Reduction

The following lemma uses essentially the same analysis as Proposition 5 in [Sey93], which shows how to build a well-conditioned basis using well-conditioned blocks.

We recall from Section 3.3 that geometrically Seysen reduction amounts to shifting a vector to lie inside a parallelepiped $[\boldsymbol{b}_1, \dots, \boldsymbol{b}_k] \cdot [-\frac{1}{2}, \frac{1}{2}]^k$. This contrasts with size-reduction, which amounts to shifting a vector to lie inside a box $[\tilde{\boldsymbol{b}}_1, \dots, \tilde{\boldsymbol{b}}_k] \cdot [-\frac{1}{2}, \frac{1}{2}]^k$. Although size-reduction gives a stronger guarantee about the size of entries in the primal basis, using Seysen reduction is necessary to ensure that entries in *both* the primal and dual bases are small simultaneously. Indeed, this was Seysen's key insight in [Sey93].

Let $\lfloor X \rceil$ denote component-wise rounding of a real-valued matrix X. Let $\|X\|_\infty := \max_{i,j} |X_{ij}|$ denote the largest magnitude of an entry in X. For a matrix $B = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_n]$, let $m^+(B) := \max_{i \in [n]} \|\boldsymbol{b}_i\|$,[3] and $m^-(B) := \min_{i \in [n]} \|\boldsymbol{b}_i\|$ denote the largest and smallest norms of columns of B respectively.

Lemma 5.4.4. *Let $B \in \mathbb{R}^{m \times n}$ be a basis, let $C = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_k]$, and let $D = [\pi_k^{(B)}(\boldsymbol{b}_{k+1}), \dots, \pi_k^{(B)}(\boldsymbol{b}_n)]$ so that $B = [C, Z + D]$ for some Z with* $\operatorname{span}(Z) \perp \operatorname{span}(D)$*. Then there exists a polynomial-time computable, unimodular matrix $T = T(B,k)$ and $X \in \mathbb{R}^{m \times (n-k)}, Y \in \mathbb{R}^{m \times k}$ satisfying*

1. *$BT = [C, X + D]$ with* $\operatorname{span}(X) \subseteq \operatorname{span}(C)$ *and* $\operatorname{span}(X) \perp \operatorname{span}(D)$,
2. *$(BT)^* = [C^* + Y, D^*]$ with* $\operatorname{span}(Y) \subseteq \operatorname{span}(D^*)$ *and* $\operatorname{span}(Y) \perp \operatorname{span}(C^*)$,

[3]Micciancio and Goldwasser [MG02] define $m^+(B)$ for bases as $\mu(B)$ and call the problem of finding a basis with small $\mu(B)$ the Shortest Basis Problem (SBP).

3. $m^+(X) \leq \frac{k}{2} \cdot m^+(C)$,

4. $m^+(Y) \leq \frac{n-k}{2} \cdot m^+(D^*)$.

Proof. Let $B = QB'$ be the QR-decomposition of B. Then B' has the form:

$$B' = \begin{pmatrix} C' & Z' \\ 0 & D' \end{pmatrix}$$

with blocks $C' \in \mathrm{GL}(k, \mathbb{R}), D' \in \mathrm{GL}(n-k, \mathbb{R})$. Let

$$T = T(B,k) := \begin{pmatrix} I_k & -\lfloor (C')^{-1} Z' \rceil \\ 0 & I_{n-k} \end{pmatrix}. \tag{5.5}$$

Then T has integer entries and $\det(T) = 1$, so T is unimodular. Furthermore, $B'T, (B'T)^*$ are of the form

$$B'T = \begin{pmatrix} C' & X' \\ 0 & D' \end{pmatrix}, \quad (B'T)^* = \begin{pmatrix} (C')^* & 0 \\ Y' & (D')^* \end{pmatrix},$$

for some X', Y'. Let $X := Q^{-1} \cdot [X', 0]^T$, let $Y := Q^{-1} \cdot [0, Y']^T$. Using the orthogonality of Q and the definitions of X and Y, it is clear that items 1 and 2 hold.

Let $W = (C')^{-1} Z' - \lfloor (C')^{-1} Z' \rceil$. A straightforward computation shows that $X' = C'W$ and $Y' = -(D')^* W$. Using the orthogonality of Q and the fact that $\|W\|_\infty \leq \frac{1}{2}$, $m^+(X) = m^+(X') = m^+(C'W) \leq \frac{1}{2} \sum_{i=1}^{k} \|\boldsymbol{c}_i'\| = \frac{1}{2} \sum_{i=1}^{k} \|\boldsymbol{c}_i\| \leq \frac{k}{2} \cdot m^+(C)$, which implies item 3. Similarly, $m^+(Y) = m^+(Y') = m^+(-(D')^* W) \leq \frac{1}{2} \sum_{i=1}^{n-k} \|(\boldsymbol{d}_i')^*\| = \frac{1}{2} \sum_{i=1}^{n-k} \|\boldsymbol{d}_i^*\| \leq \frac{n-k}{2} \cdot m^+(D^*)$, which implies item 4. $\square$

The following corollary analyzes how Seysen reduction affects the conditioning of bases in terms of the conditioning of its blocks.

Corollary 5.4.5. *Let $B \in \mathbb{R}^{m \times n}$ be a basis, let $C = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k]$, and let $D = [\pi_k^{(B)}(\boldsymbol{b}_{k+1}), \ldots, \pi_k^{(B)}(\boldsymbol{b}_n)]$. Let $T = T(B, k)$ denote the matrix defined in Equation* (5.5)*, and let $A = BT$. Then $S(A) \leq \max\{\beta_1, \beta_2\}$, where*

$$\begin{aligned} \beta_1 = \beta_1(C, D) &= \Big(1 + \frac{n-k}{2} \cdot \frac{m^+(D^*)}{m^-(C^*)}\Big) S(C), \\ \beta_2 = \beta_2(C, D) &= \Big(1 + \frac{k}{2} \cdot \frac{m^+(C)}{m^-(D)}\Big) S(D). \end{aligned} \tag{5.6}$$

Proof. Fix $i \in [k]$. We have by Lemma 5.4.4 that $\boldsymbol{a}_i = \boldsymbol{c}_i$ and that $\boldsymbol{a}_i^* = \boldsymbol{c}_i^* + \boldsymbol{y}_i$ for some $\boldsymbol{y}_i$ with $\|\boldsymbol{y}_i\| \leq \frac{n-k}{2} \cdot m^+(D^*)$. Therefore,

$$\frac{\|\boldsymbol{a}_i\|\|\boldsymbol{a}_i^*\|}{S(C)} \leq \frac{\|\boldsymbol{a}_i\|\|\boldsymbol{a}_i^*\|}{\|\boldsymbol{c}_i\|\|\boldsymbol{c}_i^*\|} \leq \frac{\|\boldsymbol{c}_i^*\| + \|\boldsymbol{y}_i\|}{\|\boldsymbol{c}_i^*\|} \leq 1 + \frac{\|\boldsymbol{y}_i\|}{m^-(C^*)} \leq 1 + \frac{(n-k) \cdot m^+(D^*)}{2m^-(C^*)}.$$

It follows that $\|\boldsymbol{a}_i\|\|\boldsymbol{a}_i^*\| \leq \beta_1$.

Fix $i \in \{k+1, \ldots, n\}$. We have by Lemma 5.4.4 that $\boldsymbol{a}_i^* = \boldsymbol{d}_{i-k}^*$ and that $\boldsymbol{a}_i = \boldsymbol{x}_{i-k} + \boldsymbol{d}_{i-k}$ for some $\boldsymbol{x}_{i-k}$ with $\|\boldsymbol{x}_{i-k}\| \leq \frac{k}{2} \cdot m^+(C)$. Therefore,

$$\frac{\|\boldsymbol{a}_i\|\|\boldsymbol{a}_i^*\|}{S(D)} \leq \frac{\|\boldsymbol{a}_i\|\|\boldsymbol{a}_i^*\|}{\|\boldsymbol{d}_{i-k}\|\|\boldsymbol{d}_{i-k}^*\|} \leq \frac{\|\boldsymbol{x}_{i-k}\| + \|\boldsymbol{d}_{i-k}\|}{\|\boldsymbol{d}_{i-k}\|} \leq 1 + \frac{\|\boldsymbol{x}_{i-k}\|}{m^-(D)} \leq 1 + \frac{k \cdot m^+(C)}{2m^-(D)},$$

It follows that $\|\boldsymbol{a}_i\|\|\boldsymbol{a}_i^*\| \leq \beta_2$. Therefore for all $i \in [n]$, $\|\boldsymbol{a}_i\|\|\boldsymbol{a}_i^*\| \leq \max\{\beta_1, \beta_2\}$, which proves the claim. □

Note that $1 \leq m^+(B) \cdot m^-(B^*) \leq S(B)$ by the Cauchy-Schwarz inequality. Using the lower bound, one could merge the expressions in Equation (5.6) into a single expression depending on $m^+(C) \cdot m^+(D^*)$, but this would lead to worse bounds in our subsequent analysis.

5.4.4 A First Approximation Scheme for Seysen Bases

We now present an algorithm for computing a $(1+\varepsilon)$-approximately optimal Seysen basis. The main idea is to break the lattice into blocks according to large gaps in its successive minima, and to enumerate an optimal basis for each block. In GoodSeysen $g(n, \varepsilon)$ quantifies the threshold for such a large gap; $g(n, \varepsilon)$ will be set in the analysis.

The vectors $\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_n$ denote the Gram-Schmidt vectors associated with vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathcal{L}$ which achieve the successive minima of $\mathcal{L}$. In LiftAndReduce, $T(B, k)$ denotes the Seysen reduction matrix defined in Equation (5.5).

```
Algorithm 4: GoodSeysen($\mathcal{L}, \varepsilon$)
  Input: A lattice $\mathcal{L}$ of rank $n$ (specified by a basis $B' \in \mathbb{Q}^{m \times n}$), and a
         number $\varepsilon \in (0, 1)$.
  Output: A basis $B$ of $\mathcal{L}$ such that $S(B) \leq (1+\varepsilon) \cdot S(\mathcal{L})$.
  $k_1 < \cdots < k_m \leftarrow \{k \in [n-1] : \lambda_{k+1}(\mathcal{L})/\lambda_k(\mathcal{L}) > g(n, \varepsilon)\} \cup \{0\}$
  $B_m \leftarrow$ EnumerateSeysenOpt($\pi_{k_m}(\mathcal{L})$)          /* With $A_m = B_m$ for
   analysis */
  for $i = m-1$ to $1$ do
      $A_i \leftarrow$ EnumerateSeysenOpt($\pi_{k_i}(\mathcal{L}) \cap \mathrm{span}(\tilde{\boldsymbol{v}}_{k_i+1}, \ldots, \tilde{\boldsymbol{v}}_{k_{(i+1)}})$)
      $B_i \leftarrow$ LiftAndReduce($\pi_{k_i}(\mathcal{L}), k_{(i+1)} - k_i, A_i, B_{i+1}$)
  end
  return $B_1$
```

Note that $k_1 = 0$ in GoodSeysen. The following lemma analyzes the Seysen condition number $S(B_i)$ for each intermediate basis B_i computed in GoodSeysen.

Lemma 5.4.6. *Let $\mathcal{L}$ be a lattice of rank n with $\lambda_{k+1}(\mathcal{L})/\lambda_k(\mathcal{L}) > n$ for some $k \in [n-1]$. Let C be a basis of $\mathcal{L} \cap V_k$ that satisfies $S(C) = S(\mathcal{L} \cap V_k)$, and*

```
Algorithm 5: LiftAndReduce($\mathcal{L}, k, C, D$)
    Input: A lattice $\mathcal{L}$ of rank $n$ (specified by a basis $B' \in \mathbb{Q}^{m \times n}$), an index
           $k \in [n-1]$ such that $\lambda_{k+1}(\mathcal{L})/\lambda_k(\mathcal{L}) > n$, a basis $C = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k]$ of
           $\mathcal{L} \cap V_k$, and a basis $D = [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_{n-k}]$ of $\pi_k(\mathcal{L})$.
    Output: A basis $A$ of $\mathcal{L}$ such that $S(A) \leq (1+t) \cdot \max\{S(C), S(D)\}$ where
            $t = t(\mathcal{L}, k)$ is defined as in Lemma 5.4.6.
    $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k \leftarrow \boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$
    $\boldsymbol{b}_{k+1}, \ldots, \boldsymbol{b}_n \leftarrow$ Liftings of $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_{n-k}$ such that $\boldsymbol{b}_i \in \mathcal{L}$ and $\pi_k(\boldsymbol{b}_i) = \boldsymbol{d}_{i-k}$
     for $i \in \{k+1, \ldots, n\}$
    $B \leftarrow [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$
    return $B \cdot T(B, k)$
```

let D be a basis of $\pi_k(\mathcal{L})$ that satisfies $S(D) \leq c \cdot S(\pi_k(\mathcal{L}))$ for some $c \geq 1$. Let $A =$ LiftAndReduce$(\mathcal{L}, k, C, D)$. Then A is a basis of $\mathcal{L}$ that satisfies $S(A) \leq c \cdot (1+t) \cdot \max\{S(\mathcal{L} \cap V_k), S(\pi_k(\mathcal{L}))\}$ where $t = t(\mathcal{L}, k) = \dfrac{n^2 \cdot s(n)}{2} \cdot \lambda_k(\mathcal{L})/\lambda_{k+1}(\mathcal{L})$.

Proof. We have that $A = B \cdot T(B, k)$, where B is a basis of $\mathcal{L}$ by construction and $T(B, k)$ is unimodular by Lemma 5.4.4. So, A is a basis of $\mathcal{L}$ as well. We prove the upper bound on $S(A)$ by upper bounding the quantities β_1 and β_2 in Equation (5.6), and applying Corollary 5.4.5.

Because $\lambda_{k+1}(\mathcal{L})/\lambda_k(\mathcal{L}) > n$ we have that $V_k \perp W_{n-k}$ by Lemma 5.2.7, and therefore $\mathcal{L}(D^*) = \pi_k(\mathcal{L})^* = \mathcal{L}^* \cap W_{n-k}$ and $\mathcal{L}(C^*) = (\mathcal{L} \cap V_k)^* = \pi^{(W)}_{n-k}(\mathcal{L}^*)$ by Lemma 5.2.11. By Lemma 5.2.8 item 1 we have that $\lambda_k(\mathcal{L} \cap V_k) = \lambda_k(\mathcal{L})$ and $\lambda_{n-k}(\mathcal{L}^* \cap W_{n-k}) = \lambda_{n-k}(\mathcal{L}^*)$. We will use all of these identities freely.

We first upper bound β_2. Using the assumption that $S(C) = S(\mathcal{L} \cap V_k)$, we have by Corollary 5.4.1 that

$$m^+(C) \leq s(k) \cdot \lambda_k(\mathcal{L} \cap V_k) = s(k) \cdot \lambda_k(\mathcal{L}). \tag{5.7}$$

Using the lower bound in Theorem 3.2.1,

$$m^-(D) \geq \lambda_1(\mathcal{L}(D)) = \lambda_1(\pi_k(\mathcal{L})) \geq \frac{1}{\lambda_{n-k}(\pi_k(\mathcal{L})^*)} = \frac{1}{\lambda_{n-k}(\mathcal{L}^* \cap W_{n-k})} = \frac{1}{\lambda_{n-k}(\mathcal{L}^*)}. \tag{5.8}$$

Therefore by Equations (5.7) and (5.8), the upper bound in Theorem 3.2.1, and the assumption that $S(D) \leq c \cdot S(\pi_k(\mathcal{L}))$,

$$\begin{aligned}\beta_2(C, D) &= \Big(1 + \frac{k}{2} \cdot \frac{m^+(C)}{m^-(D)}\Big) \cdot S(D) \\ &\leq \Big(1 + \frac{k}{2} \cdot s(k) \cdot \lambda_k(\mathcal{L}) \cdot \lambda_{n-k}(\mathcal{L}^*)\Big) \cdot S(D) \\ &\leq c \cdot \Big(1 + \frac{k \cdot n}{2} \cdot s(k) \cdot \lambda_k(\mathcal{L})/\lambda_{k+1}(\mathcal{L})\Big) \cdot S(\pi_k(\mathcal{L})).\end{aligned}$$

We next upper bound β_1. Using the assumption that $S(D) \leq c \cdot S(\pi_k(\mathcal{L}))$ and the identities $S(D^*) = S(D)$, $S(\pi_k(\mathcal{L})^*) = S(\pi_k(\mathcal{L}))$ we have that $S(D^*) \leq c \cdot S(\mathcal{L}^* \cap W_{n-k})$. Therefore by Corollary 5.4.1 we have that

$$m^+(D^*) \leq c \cdot s(n-k) \cdot \lambda_{n-k}(\mathcal{L}^* \cap W_{n-k}) = c \cdot s(n-k) \cdot \lambda_{n-k}(\mathcal{L}^*). \tag{5.9}$$

Using the lower bound in Theorem 3.2.1,

$$m^-(C^*) \geq \lambda_1(\mathcal{L}(C)^*) = \lambda_1(\pi_{n-k}^{(W)}(\mathcal{L}^*)) \geq 1/\lambda_k(\pi_{n-k}^{(W)}(\mathcal{L}^*)^*) = 1/\lambda_k(\mathcal{L} \cap V_k) = 1/\lambda_k(\mathcal{L}). \tag{5.10}$$

Therefore by Equations (5.9) and (5.10), the upper bound in Theorem 3.2.1, and the assumption that $S(C) = S(\mathcal{L} \cap V_k)$,

$$\begin{aligned}
\beta_1(C,D) &= \Big(1 + \frac{n-k}{2} \cdot \frac{m^+(D^*)}{m^-(C^*)}\Big) \cdot S(C) \\
&\le \Big(1 + c \cdot \frac{n-k}{2} \cdot s(n-k) \cdot \lambda_k(\mathcal{L}) \cdot \lambda_{n-k}(\mathcal{L}^*)\Big) \cdot S(C) \\
&\le c \cdot \Big(1 + \frac{(n-k) \cdot n}{2} \cdot s(n-k) \cdot \lambda_k(\mathcal{L})/\lambda_{k+1}(\mathcal{L})\Big) \cdot S(\mathcal{L} \cap V_k).
\end{aligned}$$

□

We now prove the main theorem which ensures the approximation quality and runtime of GoodSeysen$(\mathcal{L}, \varepsilon)$. The main idea in the analysis is that the large gaps in successive minima between blocks ensure good approximation quality, while the small gaps within blocks ensure good runtime.

Theorem 5.4.7. *Let* $\varepsilon \in (0,1)$ *and let* $g(n,\varepsilon) := n^3 \cdot s(n)/\varepsilon + 1$*. Then* GoodSeysen$(\mathcal{L}, \varepsilon)$ *outputs a basis* B *of* $\mathcal{L}$ *satisfying* $S(B) \le (1+\varepsilon) \cdot S(\mathcal{L})$ *in* $(\mathrm{poly}(n) \cdot s(n)/\varepsilon)^{O(n^3)} \le (n/\varepsilon)^{O(n^3 \log n)}$*-time and polynomial space.*

Proof. We first bound the approximation quality of the basis returned by GoodSeysen$(\mathcal{L}, \varepsilon)$. We have that B_i is a basis of $\pi_{k_i}(\mathcal{L})$, and we will prove by induction that $S(B_i) \le (1+\varepsilon)^{(m-i)/n} \cdot S(\pi_{k_i}(\mathcal{L}))$. In the base case, $S(B_m) = S(\pi_{k_m}(\mathcal{L}))$ by Proposition 5.4.2. For the inductive case we will use Lemma 5.4.6 to analyze the quality of the basis B_i of $\pi_{k_i}(\mathcal{L})$ lifted from the basis A_i of $\pi_{k_i}(\mathcal{L}) \cap \mathrm{span}(\tilde{\boldsymbol{v}}_{k_i+1}, \ldots, \tilde{\boldsymbol{v}}_{k_{(i+1)}})$ and the basis B_{i+1} of $\pi_{k_{(i+1)}}(\mathcal{L})$.

We will repeatedly use the fact that $\lambda_{k_i+1}(\mathcal{L})/\lambda_{k_i}(\mathcal{L}) \ge g(n,\varepsilon) > n^3 + 1$ for $i \in \{2, \ldots, m\}$. Fix $i \in [m-1]$, and let $\ell_i = k_{i+1} - k_i$ for $i \in [m-1]$ (again recall the $k_1 = 0$). Let $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{n-k_i} \in \pi_{k_i}(\mathcal{L})$ denote vectors that achieve the successive minima of $\pi_{k_i}(\mathcal{L})$. Since $\lambda_{k_{i+1}+1}(\mathcal{L})/\lambda_{k_{i+1}}(\mathcal{L}) > \frac{\sqrt{n}}{2} + 1$, we have that

$\mathrm{span}(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{\ell_i}) = \mathrm{span}(\tilde{\boldsymbol{v}}_{k_i+1}, \ldots, \tilde{\boldsymbol{v}}_{k_{i+1}})$ by Lemma 5.2.12. Moreover, $S(A_i) = S(\mathcal{L}(A_i))$ by Proposition 5.4.2, and $S(B_{i+1}) \leq c_{i+1} \cdot S(\mathcal{L}(B_{i+1}))$, where $c_{i+1} = (1+\varepsilon)^{(m-i-1)/n}$ by the induction hypothesis. Therefore, A_i, B_{i+1} satisfy the conditions for C, D in Lemma 5.4.6, respectively.

For $i = 1$, $\lambda_{\ell_i+1}(\pi_{k_i}(\mathcal{L}))/\lambda_{\ell_i}(\pi_{k_i}(\mathcal{L})) = \lambda_{k_2+1}(\mathcal{L})/\lambda_{k_2}(\mathcal{L}) \geq g(n, \varepsilon)$. Furthermore, for every $i \in \{2, \ldots, m-1\}$, using the lower bound in Equation (5.4) it holds that

$$\frac{\lambda_{\ell_i+1}(\pi_{k_i}(\mathcal{L}))}{\lambda_{\ell_i}(\pi_{k_i}(\mathcal{L}))} \geq \frac{\lambda_{k_{(i+1)}+1}(\mathcal{L}) - \frac{\sqrt{k_i}}{2}\lambda_{k_i}(\mathcal{L})}{\lambda_{k_{(i+1)}}(\mathcal{L})} \geq g(n,\varepsilon) - \frac{\sqrt{k_i}}{2g(n,\varepsilon)} \geq g(n,\varepsilon) - 1. \quad (5.11)$$

We then have that

$$\begin{aligned} S(B_i) \leq{}& (1+t) \cdot (1+\varepsilon)^{(m-i-1)/n} \\ &\cdot \max\{S(\pi_{k_i}(\mathcal{L}) \cap \mathrm{span}(\tilde{\boldsymbol{v}}_{k_i+1}, \ldots, \tilde{\boldsymbol{v}}_{k_{i+1}})), S(\pi_{k_{(i+1)}}(\mathcal{L}))\} \\ \leq{}& (1+t) \cdot (1+\varepsilon)^{(m-i-1)/n} \cdot S(\pi_{k_i}(\mathcal{L})), \end{aligned}$$

where $t = t(\pi_{k_i}(\mathcal{L}), \ell_i)$ is as defined in Lemma 5.4.6. The first inequality follows from Lemma 5.4.6 and the second from Lemma 5.4.3. It remains to upper bound t. Indeed by Equation (5.11),

$$t = \frac{n^2 \cdot s(n)}{2} \cdot \lambda_{\ell_i}(\pi_{k_i}(\mathcal{L}))/\lambda_{\ell_{i+1}}(\pi_{k_i}(\mathcal{L})) \leq \frac{n^2 \cdot s(n)}{2(g(n,\varepsilon) - 1)} \leq \varepsilon/(2n).$$

It's not hard to show that $x/2 \leq \ln(1+x)$ for $x \in (0,1)$ using the Taylor expansion of $\ln(1+x)$. Furthermore, $1+x \leq e^x$ for all $x \in \mathbb{R}$. Therefore, $1+t \leq 1+\varepsilon/(2n) \leq 1 + \ln(1+\varepsilon)/n \leq (1+\varepsilon)^{1/n}$ as desired.

Next we bound the runtime of GoodSeysen. All operations except for computing the successive minima of the input lattice and enumerating an optimal Seysen basis require at most polynomial time in n and are performed at most $O(n)$ times each. As mentioned in the proof of Proposition 5.4.2 the successive minima problem can be solved in $n^{O(n)}$-time and polynomial space using Kannan's algorithm [Kan87].

It remains to bound the runtime of computing
$A_i = \textsc{EnumerateSeysenOpt}(\pi_{k_i}(\mathcal{L}) \cap \operatorname{span}(\tilde{\boldsymbol{v}}_{k_i+1}, \ldots, \tilde{\boldsymbol{v}}_{k_{(i+1)}}))$ for each i. For $i = 1$, $\mathcal{L} = \mathcal{L}(A_i)$ and $\lambda_{j+1}(\mathcal{L})/\lambda_j(\mathcal{L}) \leq g(n, \varepsilon)$ for $1 \leq j \leq \ell_i - 1$. For $i \in \{2, \ldots, m\}$ we have by Lemma 5.2.8 item 1 and Lemma 5.2.10 that for every $1 \leq j \leq \ell_i - 1$,

$$\frac{\lambda_{j+1}(\mathcal{L}(A_i))}{\lambda_j(\mathcal{L}(A_i))} = \frac{\lambda_{j+1}(\pi_{k_i}(\mathcal{L}))}{\lambda_j(\pi_{k_i}(\mathcal{L}))} \leq 2g(n, \varepsilon).$$

Therefore, by Proposition 5.4.2, computing A_i takes at most

$$\begin{aligned}(s(\ell_i) \cdot \lambda_{\ell_i}(\pi_{k_i}(\mathcal{L}))/\lambda_1(\pi_{k_i}(\mathcal{L})))^{O(\ell_i^2)} &\leq (s(\ell_i) \cdot (2g(n, \varepsilon))^{\ell_i - 1})^{O(\ell_i^2)} \\ &\leq (s(n) \cdot \operatorname{poly}(n)/\varepsilon)^{O(\ell_i^3)}\end{aligned}$$

time. The overall time spent on calls to EnumerateSeysenOpt is then at most $\sum_{i=1}^{m} (s(n) \cdot \operatorname{poly}(n)/\varepsilon)^{O(\ell_i^3)} \leq (s(n) \cdot \operatorname{poly}(n)/\varepsilon)^{O(n^3)} \leq (n/\varepsilon)^{O(n^3 \log n)}$, which dominates the overall runtime of GoodSeysen. Computing each A_i also dominates the space complexity of GoodSeysen, and takes polynomial space by Proposition 5.4.2. □

5.4.5 Minkowski Ellipsoids Contain Good Seysen Bases

In this section we characterize the $(1+\varepsilon)$-approximately optimal Seysen bases computed by GoodSeysen by showing that they lie inside a scaled Minkowski Ellipsoid $t \cdot E(\mathcal{L})$ for some t depending only on n and ε. This characterization in turn yields a simpler approximation scheme for computing Seysen bases which consists of enumerating all bases lying inside such an ellipsoid.

Let $\mathcal{L}$ be a lattice of rank n. Recall that the Minkowski Ellipsoid $E(\mathcal{L})$ is the ellipsoid whose ith axis is aligned with $\tilde{\boldsymbol{v}}_i$ and whose ith radius has length $\lambda_i(\mathcal{L})$. More formally, define the closed Minkowski Ellipsoid associated with $\mathcal{L}$ as

$$E(\mathcal{L}) := \Big\{\boldsymbol{x} \in \operatorname{span}(\mathcal{L}) : \sum_{i=1}^{n} \Big(\frac{\langle \boldsymbol{x}, \tilde{\boldsymbol{v}}_i \rangle}{\|\tilde{\boldsymbol{v}}_i\| \cdot \lambda_i}\Big)^2 \leq 1\Big\}. \tag{5.12}$$

The interior of $E(\mathcal{L})$ contains no non-zero lattice points, a fact which can be used to prove Minkowski's Second Theorem (see, e.g., [Reg09a]).

Lemma 5.4.8. *Let $B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n] =$ GoodSeysen$(\mathcal{L}, \varepsilon)$, and let A_i, k_i, and ℓ_i be as defined in* GoodSeysen. *Let $\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_n$ denote the Gram-Schmidt orthogonalization of vectors achieving the successive minima of $\mathcal{L}$, let $r, j \in [n]$, and let $i \in [m]$ be the maximum index such that $j > k_i$. Then $|\langle \boldsymbol{b}_r, \tilde{\boldsymbol{v}}_j \rangle| / \|\tilde{\boldsymbol{v}}_j\| \leq \ell_i \cdot m^+(A_i)$.*

Proof. If $A =$ LiftAndReduce$(\mathcal{L}, k, C, D)$, then
$m^+(\pi_{\operatorname{span}(C)}(A)) \leq \max\{1, \operatorname{rank}(C)/2\} \cdot m^+(C)$, and $m^+(\pi_{\operatorname{span}(D)}(A)) = m^+(D)$ by Lemma 5.4.4. Applying this observation recursively, and noting that $A_i = B_i$ in the base case when $i = m$, we get that $m^+(\pi_{\operatorname{span}(A_i)}(B)) = m^+(\pi_{\operatorname{span}(A_i)}(B_i)) \leq \max\{1, \ell_i/2\} \cdot m^+(A_i)$, where B_i is as defined in GoodSeysen.

Furthermore, $\tilde{\boldsymbol{v}}_j \in \mathrm{span}(\tilde{\boldsymbol{v}}_{k_i+1}, \ldots, \tilde{\boldsymbol{v}}_{k_i+\ell_i}) = \mathrm{span}(A_i)$, so $|\langle \boldsymbol{b}_r, \tilde{\boldsymbol{v}}_j\rangle| / \|\tilde{\boldsymbol{v}}_j\| \leq \|\pi_{\mathrm{span}(A_i)}(\boldsymbol{b}_r)\|$. Clearly, $\|\pi_{\mathrm{span}(A_i)}(\boldsymbol{b}_r)\| \leq m^+(\pi_{\mathrm{span}(A_i)}(B))$, and the claim follows. $\square$

Theorem 5.4.9. *Let $\mathcal{L}$ be a lattice of rank n, and let*
$B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n] = \textsc{GoodSeysen}(\mathcal{L}, \varepsilon)$. *Then $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in t \cdot E(\mathcal{L})$ where $t = (\mathrm{poly}(n) \cdot s(n)/\varepsilon)^n \leq (n/\varepsilon)^{O(n \log n)}$.*

Proof. Let A_i, k_i, ℓ_i and $g(n, \varepsilon) = n^3 \cdot s(n)/\varepsilon + 1$ be as defined in $\textsc{GoodSeysen}$. Fix $j \in [n]$, and let $i \in [m]$ be the maximum index such that $j > k_i$. I.e., $\tilde{\boldsymbol{v}}_j \in \mathrm{span}(A_i)$. Let $\ell_i = k_{i+1} - k_i$ if $i \in [m-1]$, and $\ell_i = n - k_m$ if $i = m$. By Corollary 5.4.1 and Lemma 5.2.8,

$$m^+(A_i) \leq s(\ell_i) \cdot \lambda_{\ell_i}(\mathcal{L}(A_i)) = s(\ell_i) \cdot \lambda_{\ell_i}(\pi_{k_i}(\mathcal{L})) \leq s(\ell_i) \cdot \lambda_{k_{(i+1)}}(\mathcal{L}). \tag{5.13}$$

Therefore, combining Lemma 5.4.8, Equation (5.13), and the fact that
$\lambda_{k_i+\ell_i}(\mathcal{L})/\lambda_{k_i+1}(\mathcal{L}) \leq g(n, \varepsilon)^{\ell_i - 1}$,

$$\begin{aligned}
|\langle \boldsymbol{b}_r, \tilde{\boldsymbol{v}}_j\rangle / \|\tilde{\boldsymbol{v}}_j\|| &\leq \ell_i \cdot m^+(A_i) \\
&\leq \ell_i \cdot s(\ell_i) \cdot \lambda_{k_{(i+1)}}(\mathcal{L}) \\
&\leq \ell_i \cdot s(\ell_i) \cdot g(n, \varepsilon)^{\ell_i - 1} \cdot \lambda_j(\mathcal{L}) \\
&\leq (\mathrm{poly}(n) \cdot s(n)/\varepsilon)^n \cdot \lambda_j(\mathcal{L}).
\end{aligned}$$

Then for all $r \in [n]$,

$$\sum_{j=1}^{n} \Big(\frac{\langle \boldsymbol{b}_r, \tilde{\boldsymbol{v}}_j \rangle}{\|\tilde{\boldsymbol{v}}_j\| \cdot \lambda_j} \Big)^2 \leq n \cdot (\mathrm{poly}(n) \cdot s(n)/\varepsilon)^{2n}.$$

Recalling the definition of $E(\mathcal{L})$ from Equation (5.12) and the fact that $s(n) \leq n^{O(\log n)}$ from Theorem 4.2.13, it follows that $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in t \cdot E(\mathcal{L})$ where $t = (\mathrm{poly}(n) \cdot s(n)/\varepsilon)^n \leq (n/\varepsilon)^{O(n \log n)}$. $\square$

Lemma 5.4.10 (Enumerating lattice points inside a scaled Minkowski Ellipsoid)**.** *Given a number $t \geq 1$ and a lattice $\mathcal{L}$ of rank n, there exists an algorithm running in $(t \cdot n)^{O(n)}$-time and polynomial space that enumerates all vectors $x \in t \cdot E(\mathcal{L}) \cap \mathcal{L}$.*

Proof. We reduce the problem of enumerating lattice points inside a scaled Minkowski Ellipsoid to the problem of enumerating lattice points inside a ball. Let Λ be the diagonal matrix with $\mathrm{diag}(\Lambda) = (\lambda_1(\mathcal{L}), \ldots, \lambda_n(\mathcal{L}))$, let $\overline{V} = [\tilde{\boldsymbol{v}}_1/\|\tilde{\boldsymbol{v}}_1\|, \ldots, \tilde{\boldsymbol{v}}_n/\|\tilde{\boldsymbol{v}}_n\|]$, and let $T = \overline{V} \cdot \Lambda$.

Let B_2^n denote the closed Euclidean ball, and let $\mathrm{int}(S)$ denote the interior of a set S. Note that $T(B_2^n) = E(\mathcal{L})$, so that T^{-1} is a bijection between $t \cdot E(\mathcal{L}) \cap \mathcal{L}$ and $t \cdot B_2^n \cap T^{-1}(\mathcal{L})$ for every $t > 0$. In particular the fact that there are no non-zero lattice points in the interior of $E(\mathcal{L})$ implies that $T^{-1}(\mathcal{L}) \cap \mathrm{int}(B_2^n) = \{\mathbf{0}\}$, so $\lambda_1(T^{-1}(\mathcal{L})) \geq 1$.

We can therefore output all points in $t \cdot E(\mathcal{L}) \cap \mathcal{L}$ by enumerating each point $\boldsymbol{x} \in T^{-1}(\mathcal{L})$ with $\|\boldsymbol{x}\| \leq t$ and then outputting $T\boldsymbol{x}$. Computing T amounts to computing vectors achieving the successive minima of $\mathcal{L}$, which can be done using $n^{O(n)}$ time and polynomial space. Finally, because $\lambda_1(T^{-1}(\mathcal{L})) \geq 1$, we can enumerate all $\boldsymbol{x} \in T^{-1}(\mathcal{L})$ with $\|\boldsymbol{x}\| \leq t$ in $(t \cdot n)^{O(n)}$-time and polynomial space by Theorem 5.2.1. $\square$

Let $t = (\text{poly}(n) \cdot s(n)/\varepsilon)^n \leq (n/\varepsilon)^{O(n \log n)}$, and let ENUMERATESEYSENELLIPSOID$(\mathcal{L}, \varepsilon)$ denote the algorithm which enumerates all n-tuples $B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$ of vectors in $t \cdot E(\mathcal{L})$, and outputs the basis B' which achieves minimal $S(B')$ among all the n-tuples B that are bases of $\mathcal{L}$.

Corollary 5.4.11. *Let $\mathcal{L}$ be a lattice of rank n. Then* ENUMERATESEYSENELLIPSOID$(\mathcal{L}, \varepsilon)$ *computes a basis B of $\mathcal{L}$ such that $S(B) \leq (1+\varepsilon) \cdot S(\mathcal{L})$ in $(s(n) \cdot \text{poly}(n)/\varepsilon)^{O(n^3)} \leq (n/\varepsilon)^{O(n^3 \log n)}$-time and polynomial space.*

Proof. Combine Theorem 5.4.9 with Lemma 5.4.10. $\square$

We remark that ENUMERATESEYSENELLIPSOID$(\mathcal{L}, \varepsilon)$ achieves the same approximation quality and runtime as GOODSEYSEN$(\mathcal{L}, \varepsilon)$, and is conceptually simpler. We emphasize again that one may in fact view GOODSEYSEN$(\mathcal{L}, \varepsilon)$ as a constructive proof of correctness for ENUMERATESEYSENELLIPSOID$(\mathcal{L}, \varepsilon)$.

Bibliography

[Abe07] Oliver Aberth. *Introduction to Precise Numerical Methods, Second Edition*. Academic Press, Inc., Orlando, FL, USA, 2007.

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 284–293, 1997.

[AFK02] Sanjeev Arora, Alan Frieze, and Haim Kaplan. A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. *Mathematical Programming*, 2002.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 99–108, 1996.

[Ajt98] Miklós Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 10–19, 1998.

[AKKV12] Vikraman Arvind, Johannes Köbler, Sebastian Kuhnert, and Yadu Vasudev. Approximate Graph Isomorphism. In *Mathematical Foundations of Computer Science*, 2012.

[AKL13] Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific, 2013.

[AM06] F. Betul Atalay and David M. Mount. The cost of compatible refinement of simplex decomposition trees. In *Proceedings of the 15th International Meshing Roundtable, Birmingham, Alabama, USA, September 17-20, 2006, Proceedings*, pages 57–69, 2006.

[Bab86] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[Ban93] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, 1993.

[BDS16] Huck Bennett, Daniel Dadush, and Noah Stephens-Davidowitz. On the lattice distortion problem. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 9:1–9:17, 2016.

[BEG94] Marshall W. Bern, David Eppstein, and John R. Gilbert. Provably good mesh generation. *J. Comput. Syst. Sci.*, 48(3):384–409, 1994.

[Ben17] Huck Bennett. Algorithms for computing nearly orthogonal and well-conditioned lattice bases, 2017. Preprint.

[BKY09] Michael A. Burr, Felix Krahmer, and Chee Yap. Continuous amortization: A non-probabilistic adaptive analysis technique. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:136, 2009.

[BLPY16] Huck Bennett, Jyh-Ming Lien, Evanthia Papadopoulou, and Chee Yap. Subvor, a program for computing voronoi diagrams via subdivision., 2016. Available at `https://github.com/hbennett/SubVor`.

[BPY16] Huck Bennett, Evanthia Papadopoulou, and Chee Yap. Planar minimization diagrams via subdivision with applications to anisotropic voronoi diagrams. *Comput. Graph. Forum*, 35(5):229–247, 2016.

[Bur16] Michael A. Burr. Continuous amortization and extensions: With applications to bisection-based root isolation. *J. Symb. Comput.*, 77:78–126, 2016.

[BY14] Huck Bennett and Chee Yap. Amortized analysis of smooth quadtrees in all dimensions. In *Algorithm Theory - SWAT 2014 - 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July 2-4, 2014. Proceedings*, pages 38–49, 2014.

[BY17] Huck Bennett and Chee Yap. Amortized analysis of smooth quadtrees in all dimensions. *Computational Geometry*, 63:20–39, 2017. Preliminary version in SWAT 2014.

[CCK+06] Ee-Chien Chang, Sung Woo Choi, DoYong Kwon, Hyungju Park, and Chee-Keng Yap. Shortest path amidst disc obstacles is computable. *Int. J. Comput. Geometry Appl.*, 16(5-6):567–590, 2006.

[Cor] Core Library homepage. Software download, source, documentation and links: `http://cs.nyu.edu/exact/core/`.

[CS98] John Conway and Neil J. A. Sloane. *Sphere Packings, Lattices and Groups*. Springer, 1998.

[Dad12] Daniel Dadush. *Integer Programming, Lattice Algorithms, and Deterministic Volume Estimation.* PhD thesis, Georgia Institute of Technology, 2012.

[Dad13] Daniel Dadush. Lecture notes from course on Lattices, Convexity and Algorithms, 2013. Available at `https://www.cs.nyu.edu/courses/spring13/CSCI-GA.3033-013/index.html`.

[DB15] Daniel Dadush and Nicolas Bonifas. Short paths on the Voronoi graph and closest vector problem with preprocessing. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 295–314, 2015.

[dBCvKO08] Mark de Berg, Otfried Cheong, Mark van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications.* Springer, Third edition, 2008.

[dBRS12] Mark de Berg, Marcel Roeloffzen, and Bettina Speckmann. Kinetic compressed quadtrees in the black-box model with applications to collision detection for low-density scenes. In *Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*, pages 383–394, 2012.

[EMM13] Ioannis Z. Emiris, Angelos Mantzaflaris, and Bernard Mourrain. Voronoi diagrams of algebraic distance fields. *Computer-Aided Design*, 45(2):511–516, 2013.

[ES86] Herbert Edelsbrunner and Raimund Seidel. Voronoi diagrams and arrangements. *Discrete & Computational Geometry*, 1:25–44, 1986.

[EZ04] Uri Erez and Ram Zamir. Achieving 1/2 log (1+SNR) on the AWGN channel with lattice encoding and decoding. *IEEE Trans. Information Theory*, 50(10):2293–2314, 2004.

[FB74] Raphael A. Finkel and Jon Louis Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Inf.*, 4:1–9, 1974.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178, 2009.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, pages 112–131, 1997.

[GMSS99] Oded Goldreich, Daniele Micciancio, Shmuel Safra, and Jean-Paul Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters*, 71(2):55 – 61, 1999.

[GN08] Nicolas Gama and Phong Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 207–216, 2008.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 197–206, 2008.

[Har01] Sariel Har-Peled. A replacement for Voronoi diagrams of near linear size. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 94–103, 2001.

[Hel85] Bettina Helfrich. Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. *Theor. Comput. Sci.*, 41:125–139, 1985.

[HK15] Sariel Har-Peled and Nirman Kumar. Approximating minimization diagrams and generalized proximity search. *SIAM J. Comput.*, 44(4):944–974, 2015.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph Silverman. NTRU: A ring-based public key cryptosystem. *Algorithmic Number Theory*, pages 267–288, 1998.

[HR12] Ishay Haviv and Oded Regev. Tensor-based hardness of the Shortest Vector Problem to within almost polynomial factors. *Theory of Computing*, 8(23):513–531, 2012. Preliminary version in STOC'07.

[HR14] Ishay Haviv and Oded Regev. On the lattice isomorphism problem. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 391–404, 2014.

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.

[Kho05] Subhash Khot. Hardness of approximating the Shortest Vector Problem in lattices. *Journal of the ACM*, 52(5):789–808, September 2005. Preliminary version in FOCS'04.

[Kle89] Rolf Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes in Computer Science*. Springer, 1989.

[KS04] Mohammad R. Kolahdouzan and Cyrus Shahabi. Voronoi-based K nearest neighbor search for spatial network databases. In *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 840–851, 2004.

[Kul97] Wladyslaw Kulpa. The Poincaré-Miranda theorem. *The American Mathematical Monthly*, 104(6):545–550, 1997.

[KZ73] A. Korkine and G. Zolotareff. Sur les formes quadratiques. *Mathematische Annalen*, 6(3):366–389, 1873.

[LB14] Cong Ling and Jean-Claude Belfiore. Achieving AWGN channel capacity with lattice gaussian coding. *IEEE Trans. Information Theory*, 60(10):5918–5929, 2014.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1987, Anaheim, California, USA, July 27-31, 1987*, pages 163–169, 1987.

[Lee82] Der-Tsai Lee. On k-nearest neighbor Voronoi diagrams in the plane. *IEEE Trans. Computers*, 31(6):478–487, 1982.

[Len83] Hendrik W. Lenstra Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983.

[LLL82] A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[LLS90] J. C. Lagarias, Hendrik W. Lenstra Jr., and Claus-Peter Schnorr. Korkin-Zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica*, 10(4):333–348, 1990.

[LS03] François Labelle and Jonathan Richard Shewchuk. Anisotropic voronoi diagrams and guaranteed-quality anisotropic mesh generation. In *Proceedings of the 19th ACM Symposium on Computational Geometry, San Diego, CA, USA, June 8-10, 2003*, pages 191–200, 2003.

[LS14] Hendrik W. Lenstra Jr. and Alice Silverberg. Lattices with symmetry, 2014. `http://arxiv.org/abs/1501.00178`.

[LSS13] Maarten Löffler, Joseph A. Simons, and Darren Strash. Dynamic planar point location with sub-logarithmic local updates. In *Algorithms and Data Structures - 13th International Symposium, WADS 2013, London, ON, Canada, August 12-14, 2013. Proceedings*, pages 499–511, 2013.

[LSVY14] Jyh-Ming Lien, Vikram Sharma, Gert Vegter, and Chee Yap. Isotopic arrangement of simple curves: An exact numerical approach based on subdivision. In *Mathematical Software - ICMS 2014 - 4th International Congress, Seoul, South Korea, August 5-9, 2014. Proceedings*, pages 277–282, 2014.

[LT08] Franklin T. Luk and Daniel M. Tracy. An improved LLL algorithm. *Linear Algebra and its Applications*, 428(2):441 – 452, 2008.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems*. Springer Science+Business Media, 2002.

[Mic01] Daniele Micciancio. The Shortest Vector Problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, March 2001. Preliminary version in FOCS 1998.

[Mic08] Daniele Micciancio. Efficient reductions among lattice problems. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 84–93, 2008.

[MK80] R. E. Moore and J. B. Kioustelidis. A simple test for accuracy of approximate solutions to nonlinear (or linear) systems. *SIAM Journal on Numerical Analysis*, 17(4):521–529, 1980.

[MMT11] Angelos Mantzaflaris, Bernard Mourrain, and Elias P. Tsigaridas. On continued fraction expansion of real roots of polynomial systems, complexity and condition numbers. *Theor. Comput. Sci.*, 412(22):2312–2330, 2011.

[Moo92] Doug Moore. *Simplicial Mesh Generation with Applications*. PhD thesis, Cornell University, 1992.

[Moo95] Doug Moore. The cost of balancing generalized quadtrees. In *Symposium on Solid Modeling and Applications*, pages 305–312, 1995.

[MP09] Bernard Mourrain and Jean Pascal Pavone. Subdivision methods for solving polynomial equations. *J. Symb. Comput.*, 44(3):292–306, 2009.

[MV13] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. *SIAM J. Comput.*, 42(3):1364–1391, 2013.

[Ngu99] Phong Q. Nguyen. Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto '97. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 288–304, 1999.

[NR09] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *J. Cryptology*, 22(2):139–160, 2009.

[OBSC00] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, 2000.

[OvL82] Mark H. Overmars and Jan van Leeuwen. Dynamic multi-dimensional data structures based on quad- and *K-D* trees. *Acta Inf.*, 17:267–285, 1982.

[ÓY85] Colm Ó'Dúnlaing and Chee-Keng Yap. A "retraction" method for planning the motion of a disc. *J. Algorithms*, 6(1):104–111, 1985.

[PM12] Eunhui Park and David M. Mount. A self-adjusting data structure for multidimensional point sets. In *Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*, pages 778–789, 2012.

[PS97] W. Plesken and B. Souvignier. Computing isometries of lattices. *J. Symbolic Comput.*, 24(3-4):327–334, 1997. Computational algebra and number theory (London, 1993).

[PV04] Simon Plantinga and Gert Vegter. Isotopic approximation of implicit curves and surfaces. In *Second Eurographics Symposium on Geometry Processing, Nice, France, July 8-10, 2004*, pages 245–254, 2004.

[Reg09a] Oded Regev. Lecture notes from course on "Lattices in Computer Science", 2009. Available at `http://www.cims.nyu.edu/~regev/teaching/lattices_fall_2009/index.html`.

[Reg09b] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.

[Reg17] Oded Regev. Personal communication, 2017.

[Rup93] Jim Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. In *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas.*, pages 83–92, 1993.

[Sam90] Hanan Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1990.

[Sch87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53:201–224, 1987.

[Sey93] Martin Seysen. Simultaneous reduction of a lattice basis and its reciprocal basis. *Combinatorica*, 13(3):363–376, 1993.

[SFS09] Naftali Sommer, Meir Feder, and Ofir Shalvi. Finding the closest lattice point by iterative slicing. *SIAM J. Discrete Math.*, 23(2):715–731, 2009.

[She] Donald R. Sheehy. Personal correspondence.

[She12] Donald R. Sheehy. New Bounds on the Size of Optimal Meshes. *Computer Graphics Forum*, 31(5):1627–1635, 2012.

[Sie45] Carl Ludwig Siegel. A mean value theorem in geometry of numbers. *Annals of Mathematics*, 46(2):pp. 340–347, 1945.

[SSV09] Mathieu Dutour Sikiric, Achill Schürmann, and Frank Vallentin. Complexity and algorithms for computing Voronoi cells of lattices. *Math. Comput.*, 78(267):1713–1731, 2009.

[Ste15] Noah Stephens-Davidowitz. Dimension-preserving reductions between lattice problems. `http://noahsd.com/latticeproblems.pdf`, 2015.

[Ste16] Noah Stephens-Davidowitz. Lecture notes from minicourse on lattices., 2016. Available at `http://www.noahsd.com/mini_lattices_course/`.

[TI97] Lloyd N. Trefethen and David Bau III. *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, 1997.

[TS89] Osamu Takahashi and Robert J. Schilling. Motion planning in a plane using generalized Voronoi diagrams. *IEEE Trans. Robotics and Automation*, 5(2):143–150, 1989.

[Vaz01] Vijay V. Vazirani. *Approximation algorithms*. Springer, 2001.

[vdWG68] Bartel L. van der Waerden and Herbert Gross. *Studien zur Theorie der quadratischen Formen*. Birkhuser, 1968.

[WCY13] Cong Wang, Yi-Jen Chiang, and Chee Yap. On soft predicates in subdivision motion planning. In *Proceedings of the twenty-ninth an-*

nual Symposium on Computational Geometry, SoCG '13, pages 349–358, New York, NY, USA, 2013. ACM.

[Yap09] Chee-Keng Yap. In praise of numerical computation. In *Efficient Algorithms, Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, pages 380–407, 2009.

[YSL12] Chee-Keng Yap, Vikram Sharma, and Jyh-Ming Lien. Towards exact numerical Voronoi diagrams. In *Ninth International Symposium on Voronoi Diagrams in Science and Engineering, ISVD 2012, New Brunswick, NJ, USA, June 27-29, 2012*, pages 2–16, 2012.