Universiteit Leiden

A Geometric Approach to Finding Lattice Isomorphisms

E. Riedeman (s4090780)

Supervisors:
prof.dr. L. Ducas & prof.dr. V. Dunjko

Bachelor Thesis– Mathematics and Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

Mathematical Institute (MI)
universiteitleiden.nl/en/science/mathematics

1 July 2026

Abstract

The Lattice Isomorphism Problem (LIP) asks whether an isometry exists between two n-dimensional lattices. Existing algorithms either lack strong theoretical runtime guarantees despite practicality, or fail to fully utilize the available geometric information. In this thesis, we address this gap by presenting a geometric approach that involves a reduction to the Spherical Code Isomorphism Problem (SCIP). By leveraging bounds on spherical codes, we develop an algorithm achieving an $n^{\mathcal{O}(n)}$ time complexity for a specific class of exponentially sized codes, highlighting the potential of this approach. By examining the failing cases of this initial algorithm, we introduce an additional structural bound. This allows us to generalize our approach into an algorithm capable of handling any arbitrary spherical code, achieving a runtime of $n^{\mathcal{O}(n^{3/2})}$.

Contents

1 Introduction **1**
1.1 Overview 1

2 Preliminaries **2**
2.1 Linear Algebra 2
2.2 Lattices and the Lattice Isomorphism Problem 3
2.3 Spherical Codes 5
2.4 The Symmetric Component of a Tensor Power 7

3 Reduction to Spherical Codes **8**

4 Bounds on Spherical Codes **11**
4.1 Linear Programming Bounds 11
4.2 The Tensor Power Trick 13

5 Algorithms **15**
5.1 The Naive Approach 15
5.2 The Special Case 16
5.3 The General Case 19

1 Introduction

A common algorithmic task in mathematics is to decide whether two objects are isomorphic. In the context of lattices, this problem is known as the Lattice Isomorphism Problem (LIP), which asks whether there exists an isometry that maps one n-dimensional lattice to another. The LIP is a seemingly hard problem, as most known algorithms require computing the shortest nonzero lattice point, which itself is known to be NP-hard [Kho04]. These include the theoretically best-known algorithm, discovered by Haviv and Regev [HR13], and the best practical algorithms.

One such practical algorithm is to reduce it to an instance of the Graph Isomorphism Problem (GIP) [Sik+20], which, despite having a worse provable time complexity, due to the exponential size of the constructed graphs, performs well in practice. This is likely attributable to hidden structures in these graphs, which make GIP easier to solve than in any arbitrary pair of graphs.

Haviv and Regev's algorithm enumerates all $n^{n+o(n)}$ solutions to a LIP (Corollary 2.8) in time $n^{10n+o(n)}$. While this runtime is optimal up to a constant in the exponent, its sheer magnitude seems to make the algorithm impractical for actual computation. Furthermore, the algorithm is based on the generalized isolation lemma, which does not appear to utilize all of the available geometric information of the given lattices.

These observations reveal a significant gap in the literature, and raise the question of whether there exists an alternative algorithm that leverages more geometric information, making it more suitable for practical applications, whilst also maintaining a strong theoretical runtime guarantee. In this thesis, we present one such geometric approach to the LIP. We do this by reducing the problem to finding isometries between spherical codes, which we call the Spherical Code Isomorphism Problem (SCIP), and demonstrating algorithms solving this particular task.

1.1 Overview

As shown in Proposition 2.15, the set of shortest vectors of a lattice induces a spherical code: a collection of points on a hypersphere with a given minimal distance between them. Using this fact, in Section 3, we reframe the first computational step of Haviv and Regev's lattice isomorphism algorithm ([HR13, Algorithm 2]) as an explicit reduction from LIP to SCIP.

In order to analyze the time complexity of our algorithms, we first investigate bounds on the size of spherical codes in Section 4. We begin by exploring theory related to the linear programming bound by Delsarte, Goethals, and Seidel [DGS77]. In particular, we will be restating the sphere packing bound by Cohn and Elkies

[CE03]. Afterwards, we amplify the spherical code bound by Delsarte, Goethals, and Seidel to obtain an explicit bound that applies to larger codes. We do this via a straightforward adaptation of the "tensor power trick" by Tao [Tao13].

Subsequently, in Section 5, we first give a naive algorithm establishing a baseline time complexity of $2^{\mathcal{O}(n^2)}$, followed by a more thorough approach, presented in Algorithm 3. The strategy involves recursing on lower-dimensional spherical codes, by taking specific hyperplane intersections or "slices" of the input codes. We choose these slices in such a way that their induced spherical codes get repeatedly sparser every recursive step. This geometric property constrains the code sizes each step, pruning the search space drastically to obtain a time complexity of $n^{\mathcal{O}(n)}$. Despite reaching the theoretical limit up to a constant, it is only applicable to a certain class of exponentially sized spherical codes, since the algorithm may fail whenever it cannot find a suitable slice. Nevertheless, this result shows the potential of this geometric approach.

In Section 5.3, we present a similar yet more complicated algorithm applicable to any arbitrary pair of spherical codes, with a worst-case runtime of $n^{\mathcal{O}(n^{3/2})}$. We motivate this strategy by showing an example of a spherical code which fails the special case algorithm, and highlight why this should be one of the easier cases. We formalize this observation in a new structural bounding method, leading to a recursive scheme that splits the dimension of the codes to create two distinct branches. The first branch mirrors the geometric pruning of the special case algorithm, whereas the other branch leverages the new structural bound effectively. Since trying to force the search space in one branch results in an expansion in the other, we balance these competing penalties to end up with the $n^{\mathcal{O}(n^{3/2})}$ time complexity.

2 Preliminaries

2.1 Linear Algebra

An *Euclidean vector space* is a finite-dimensional real inner product space $(V, \langle \cdot, \cdot \rangle)$. For a set $S \subset V$, we write $\operatorname{span} S$ for the smallest linear subspace of V containing S and $S^\perp$ to be the orthogonal complement of S in V. For a subspace $U \subset V$, we denote $\pi_U : V \to U$ as the orthogonal projection onto U, and similarly $\pi_U^\perp := \pi_{U^\perp}$ as the orthogonal projection onto $U^\perp$. An *isometry* $O : V_1 \to V_2$ between two Euclidean vector spaces is a map satisfying $\langle O(v), O(w) \rangle = \langle v, w \rangle$ for all $v, w \in V_1$. The set of all isometries from V_1 to V_2 is denoted by $\mathcal{O}(V_1, V_2)$.

2.2 Lattices and the Lattice Isomorphism Problem

Definition 2.1. A *lattice* $\mathcal{L}$ is a discrete subgroup of an Euclidean vector space. We call $\dim \mathcal{L} := \dim \operatorname{span} \mathcal{L}$ the *dimension* of $\mathcal{L}$.

Definition 2.2. Two lattices $\mathcal{L}_1$ and $\mathcal{L}_2$ are *isomorphic* if there exists an isometry $O : \operatorname{span} \mathcal{L}_1 \to \operatorname{span} \mathcal{L}_2$ such that $O(\mathcal{L}_1) = \mathcal{L}_2$. The isometry O is called an *isomorphism* between $\mathcal{L}_1$ and $\mathcal{L}_2$.

There are various formulations of the Lattice Isomorphism Problem. The simplest to state is the decision problem of determining whether two given lattices are isomorphic. We will use the following version.

Problem 2.3 (Lattice Isomorphism Problem (LIP))**.** Given two n-dimensional lattices $\mathcal{L}_1$ and $\mathcal{L}_2$, enumerate the set

$$\operatorname{Isom}(\mathcal{L}_1, \mathcal{L}_2) := \{O \in \mathcal{O}(\operatorname{span} \mathcal{L}_1, \operatorname{span} \mathcal{L}_2) \mid O(\mathcal{L}_1) = \mathcal{L}_2\}\,.$$

Indeed, the decision version of the LIP reduces to Problem 2.3 by simply checking whether $\operatorname{Isom}(\mathcal{L}_1, \mathcal{L}_2)$ is empty. Haviv and Regev [HR13] have presented an algorithm that solves Problem 2.3 in time $n^{10n+o(n)}$. [1] This runtime is theoretically optimal up to a constant in the exponent, as the integer lattice $\mathbb{Z}^n$, for example, has precisely $2^n \cdot n! = n^{n+o(n)}$ automorphisms. It turns out that this is also the asymptotic upper bound for the size of $\operatorname{Isom}(\mathcal{L}_1, \mathcal{L}_2)$ for any pair of lattices, as we will establish in Corollary 2.8. Unlike the decision version, this enumeration variant provides a strict upper limit on the output complexity, making optimal performance a well-defined target to reach.

We emphasize that Problem 2.3 asks for an enumeration of the set of isomorphisms, not merely a representation, as $\operatorname{Isom}(\mathcal{L}_1, \mathcal{L}_2)$ can be also be identified as a group whenever $\mathcal{L}_1$ and $\mathcal{L}_2$ are isomorphic.

Fact 2.4. Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be isomorphic lattices and let $\phi \in \operatorname{Isom}(\mathcal{L}_1, \mathcal{L}_2)$ be an isomorphism between $\mathcal{L}_1$ and $\mathcal{L}_2$. Then $(\operatorname{Isom}(\mathcal{L}_1, \mathcal{L}_2), \circ_\phi)$ is a group, where

$$\begin{aligned} \circ_\phi : \operatorname{Isom}(\mathcal{L}_1, \mathcal{L}_2) \times \operatorname{Isom}(\mathcal{L}_1, \mathcal{L}_2) &\longrightarrow \operatorname{Isom}(\mathcal{L}_1, \mathcal{L}_2) \\ (O_1, O_2) &\longmapsto O_1 \circ \phi^{-1} \circ O_2, \end{aligned}$$

with identity element ϕ, and inverses given by $O^\dagger = \phi \circ O^{-1} \circ \phi$ for $O \in \operatorname{Isom}(\mathcal{L}_1, \mathcal{L}_2)$.

[1] This exponent is derived from computing the sets W_i in Algorithm 1 of [HR13], which utilizes the algorithm from Corollary 2.16 with a target dual norm multiplier of $t = 5n^{17/2}$, requiring the evaluation of $(2t \cdot n^{3/2} + 1)^n = (10n^{10} + 1)^n$ integer coefficient combinations.

Evidently, a lattice $\mathcal{L}$ is always isomorphic to itself via the identity map $\mathrm{id} : \operatorname{span}\mathcal{L} \to \operatorname{span}\mathcal{L}$. $\mathrm{Aut}(\mathcal{L}) := \mathrm{Isom}(\mathcal{L}, \mathcal{L})$ is thus naturally a group, called the *automorphism group* of $\mathcal{L}$. You can easily show that

$$\mathrm{Isom}(\mathcal{L}_1, \mathcal{L}_2) \cong \mathrm{Aut}(\mathcal{L}_1) \cong \mathrm{Aut}(\mathcal{L}_2). \tag{1}$$

By a computational standpoint, any algorithm must take an input of finite size. As most lattices consist of infinitely many elements, we will implicitly represent them with a basis in each of the stated algorithms.

Definition 2.5. Let $B = (b_1, \ldots, b_n)$ be a tuple of $\mathbb{R}$-linearly independent elements of an n-dimensional lattice $\mathcal{L}$. The tuple B is called a *basis* of $\mathcal{L}$ if

$$\mathcal{L} = \left\{ \sum_{i=1}^{n} c_i b_i \mid c \in \mathbb{Z}^n \right\}.$$

Fact 2.6. Every lattice admits a basis.

Moreover, using this representation of a lattice, Corollary 2.8 is a direct consequence of the following result by Friedland.

Theorem 2.7 ([Fri97, Theorem 3])**.** Let G be a finite subgroup of $\mathrm{GL}_n(\mathbb{Q})$. Then the order of G is asymptotically bounded by $2^n \cdot n! = n^{n+o(n)}$.

Corollary 2.8. The size of $\mathrm{Isom}(\mathcal{L}_1, \mathcal{L}_2)$ for two n-dimensional isomorphic lattices $\mathcal{L}_1, \mathcal{L}_2$ is asymptotically bounded by $n^{n+o(n)}$.

Proof. By the isomorphism relation from 1, we may assume without loss of generality that $\mathcal{L} := \mathcal{L}_1 = \mathcal{L}_2$. Let $B = (b_1, \ldots, b_n)$ be a basis of $\mathcal{L}$. Every automorphism of $\mathcal{L}$ maps B to another basis B' of the same lattice, meaning that $\mathrm{Aut}(\mathcal{L})$ is isomorphic to a subgroup of $\mathrm{GL}_n(\mathbb{Z}) \subset \mathrm{GL}_n(\mathbb{Q})$. Because the discreteness of $\mathcal{L}$ restricts the basis image $O(B)$ of any $O \in \mathrm{Aut}(\mathcal{L})$ to the finite set

$$\{x \in \mathcal{L} \mid \|x\| = \|b_1\|\} \times \cdots \times \{x \in \mathcal{L} \mid \|x\| = \|b_n\|\},$$

the subgroup $\mathrm{Aut}(\mathcal{L})$ is finite. We conclude that $|\mathrm{Aut}(\mathcal{L})| = n^{n+o(n)}$ by Theorem 2.7. □

Similarly to vector spaces, we can define the dual of a lattice as all linear forms that take integer values on the lattice. As we work in Euclidean vector spaces, we can naturally identify linear forms with vectors via the inner product, and thus we define the dual as follows.

Definition 2.9. The *dual* of a lattice $\mathcal{L}$ is defined as

$$\mathcal{L}^* := \{v \in \operatorname{span}\mathcal{L} \mid \langle v, w \rangle \in \mathbb{Z} \text{ for all } w \in \mathcal{L}\}.$$

We will also be referring to the covolume of a lattice.

Definition 2.10. The *covolume* of a lattice $\mathcal{L} \subset V$ is defined as

$$\det \mathcal{L} := |\det B| ,$$

where B is a matrix represention of a basis of $\mathcal{L}$.

One can also view the covolume as the volume of a fundamental cell: a parallelepiped enclosed by a lattice basis which tiles its ambient vector space.

Lastly, we will use the fact that for every $v \in \operatorname{span} \mathcal{L}$, there is always a closest lattice point $x \in \mathcal{L}$. Their distance $\|v - x\|$ is bounded by the covering radius of the lattice.

Definition 2.11. The *covering radius* of a lattice $\mathcal{L}$ is defined as

$$\rho(\mathcal{L}) := \max_{v \in \operatorname{span} \mathcal{L}} \min_{x \in \mathcal{L}} \|v - x\| .$$

2.3 Spherical Codes

The first step of our algorithm is to compute the set of nonzero shortest vectors.

Definition 2.12. The set of *shortest vectors* of a lattice $\mathcal{L}$ is defined as

$$\mathcal{S}(\mathcal{L}) := \{v \in \mathcal{L} \mid \|v\| = \lambda(\mathcal{L})\} ,$$

where

$$\lambda(\mathcal{L}) := \min_{v \in \mathcal{L} \setminus \{0\}} \|v\| .$$

In our algorithms we will use the following result by Micciancio and Voulgaris.

Theorem 2.13 ([MV10])**.** There exists an algorithm that, given an n-dimensional lattice $\mathcal{L}$, enumerates $\mathcal{S}(\mathcal{L})$ in time $2^{\mathcal{O}(n)}$.

Now we show that $\mathcal{S}(\mathcal{L})$ induces a spherical code.

Definition 2.14. A finite set S is called a *spherical* (n, α)*-code* if

1. every element of S is a unit vector in some Euclidean vector space;
2. $\dim \operatorname{span} S$ is equal to n; and
3. $\max \{\langle x, y \rangle \mid \text{distinct } x, y \in S\} =: \alpha(S)$ is equal to α.[2]

[2]We use the convention $\max \varnothing = -\infty$ to cover the case $|S| \leq 1$.

Proposition 2.15. The set of nonzero shortest vectors $\mathcal{S}$ of an n-dimensional lattice $\mathcal{L}$ induces a spherical (d, α)-code

$$\hat{\mathcal{S}} := \left\{ \frac{v}{\|v\|} \mid v \in \mathcal{S} \right\},$$

satisfying $d \leq n$ and $\alpha \leq 1/2$.

Proof. By the discreteness of $\mathcal{L}$, $\hat{\mathcal{S}}$ is a finite set of unit vectors, making it a spherical (d, α)-code for some d and α. Clearly, $d \leq n$ as $\hat{\mathcal{S}} \subset \operatorname{span} \mathcal{L}$. We may assume without loss of generality that $\mathcal{S} = \hat{\mathcal{S}}$ by scaling $\mathcal{L}$ appropriately. Let $v, w \in \mathcal{S} \subset \mathcal{L}$ be distinct shortest vectors, such that their inner product $\langle v, w \rangle$ is equal to α. By the fact that $v - w \in \mathcal{L}$, we obtain the inequality

$$1 = \lambda(\mathcal{L})^2 \leq \|v - w\|^2 = \|v\|^2 + \|w\|^2 - 2\langle v, w \rangle = 2 - 2\langle v, w \rangle,$$

and thus $\alpha = \langle v, w \rangle \leq 1/2$. □

Bounds on spherical codes have been studied intensively in literature. Exact values

$$A(n, \alpha) := \max_{\text{spherical } (n,\alpha)\text{-code } S} |S|$$

are known for some pairs (n, α), but in general, only upper and lower bounds have been established. The famous kissing number problem is to determine $A(n, 1/2)$ for every $n \in \mathbb{N}$, and is still open for $n \in \mathbb{N} \setminus \{1, 2, 3, 4, 8, 24\}$ [Mus06]. Many general bounds on $A(n, \alpha)$ derived in the literature [CS88, sec. 1.2.6, pp.27–28] assume a fixed maximum inner product α as the dimension n tends to infinity. Such asymptotic bounds are unsuitable for the complexity analysis of our algorithms in Section 5, where α varies throughout the search. Delsarte, Goethals, and Seidel [DGS77], on the other hand, gave the explicit upper bound

$$A(n, \alpha) \leq \frac{n\,(1 - \alpha)\,(2 + (n + 1)\,\alpha)}{1 - n\alpha^2} \tag{2}$$

when $0 < \alpha \leq 1/\sqrt{n}$. This bound falls under the category of linear programming bounds, which we will review in Section 4.1.

Akin to lattices, we can also define isomorphisms between spherical codes.

Definition 2.16. Let S_1 and S_2 be spherical codes. An *isomorphism* from S_1 to S_2 is an isometry $O : \operatorname{span} S_1 \to \operatorname{span} S_2$ such that $O(S_1) = S_2$. We denote the set of all isomorphisms from S_1 to S_2 by $\operatorname{Isom}(S_1, S_2)$.

Problem 2.17 (Spherical Code Isomorphism Problem (SCIP))**.** Given two spherical (n, α)-codes with $\alpha \leq 1/2$, enumerate $\operatorname{Isom}(S_1, S_2)$.

2.4 The Symmetric Component of a Tensor Power

We will work in the k-th tensor power

$$V^{\otimes k} := \underbrace{V \otimes \cdots \otimes V}_{k \text{ times}}$$

of an n-dimensional Euclidean vector space $(V, \langle \cdot, \cdot \rangle_V)$. This is again a Euclidean vector space with inner product $\langle \cdot, \cdot \rangle$ defined by

$$\langle v_1 \otimes \cdots \otimes v_k, w_1 \otimes \cdots \otimes w_k \rangle := \prod_{i=1}^{k} \langle v_i, w_i \rangle_V \tag{3}$$

for $v_1, \ldots, v_k, w_1, \ldots, w_k \in V$. In particular, we will work in the following subspace of this n^k-dimensional space.

Definition 2.18. A tensor $x \in V^{\otimes k}$ is *symmetric* if $s_\sigma(x) = x$ for every permutation $\sigma \in S_k$, where $s_\sigma : V^{\otimes k} \to V^{\otimes k}$ is the linear map defined by

$$s_\sigma : v_1 \otimes \cdots \otimes v_k \longmapsto v_{\sigma(1)} \otimes \cdots \otimes v_{\sigma(k)}$$

for $v_1, \ldots, v_k \in V$. The set of all symmetric tensors in $V^{\otimes k}$ is denoted by $\mathrm{Sym}(V^{\otimes k})$, and is called the *symmetric component* of $V^{\otimes k}$.

Proposition 2.19. Let V be an n-dimensional real vector space and let k be a positive integer. Then $\mathrm{Sym}(V^{\otimes k})$ is a $\binom{n+k-1}{k}$-dimensional subspace of $V^{\otimes k}$.

Proof. The set forms a subspace as it is equal to the intersection of kernels

$$\bigcap_{\sigma \in S_k} \ker(s_\sigma - \mathrm{id}).$$

Let $v_1, \ldots, v_n$ be a basis of V. Now we will show that the elements

$$\sum_{\sigma \in S_k} v_{i_{\sigma(1)}} \otimes \cdots \otimes v_{i_{\sigma(k)}}$$

for $1 \le i_1 \le \cdots \le i_k \le n$ form a basis of $\mathrm{Sym}(V^{\otimes k})$. Let $x \in \mathrm{Sym}(V^{\otimes k}) \subset V^{\otimes k}$ be a symmetric tensor and write

$$x = \sum_{1 \le i_1, \ldots, i_k \le n} a_{i_1, \ldots, i_k} \cdot v_{i_1} \otimes \cdots \otimes v_{i_k}$$

for some coefficients $a_{i_1,\ldots,i_k} \in \mathbb{R}$. The symmetry of x implies $a_{i_1,\ldots,i_k} = a_{i_{\sigma(1)},\ldots,i_{\sigma(k)}}$ for every $\sigma \in S_k$ and thus x is a scalar multiple of

$$\sum_{1 \le i_1 \le \cdots \le i_k \le n} a_{i_1,\ldots,i_k} \cdot \sum_{\sigma \in S_k} v_{i_{\sigma(1)}} \otimes \cdots \otimes v_{i_{\sigma(k)}},$$

proving that the above elements span $\mathrm{Sym}(V^{\otimes k})$. The elements are linearly independent as no two of them share a common basis vector $v_{i_1} \otimes \cdots \otimes v_{i_k}$ of $V^{\otimes k}$ for $1 \leq i_1, \ldots, i_k \leq n$. Finally, there are $\binom{n+k-1}{k}$ ways to choose $1 \leq i_1 \leq \cdots \leq i_k \leq n$, thereby establishing the dimension. □

3 Reduction to Spherical Codes

The core algorithm of Haviv and Regev [HR13, Section 4.1] assumes that the input lattices are *well-rounded*, i.e., their shortest vectors span the ambient space. Later, in [HR13, Section 4.2], they show an algorithm that handles general lattices by recursively reducing the problem to this well-rounded case. In this section, we reframe this general lattice isomorphism algorithm as an explicit reduction to the SCIP (Problem 2.17).

This reduction relies on the fact that isometries strictly preserve the shortest vectors of a lattice. Ideally, in the well-rounded case, this property completely reduces the problem to finding isometries between these finite sets. For general lattices, however, a full isometry cannot be recovered from this set alone. To overcome this, we project away from the subspace spanned by the shortest vectors to find and combine isometries on the remaining dimensions, leading to the recursion scheme in Algorithm 1. We begin by proving two properties of orthogonal projections.

Lemma 3.1. Let $\mathcal{L}' \subset \mathcal{L}$ be lattices and let $\pi : V \to U^\perp$ be the orthogonal projection $\pi_U^\perp$, where $U := \operatorname{span} \mathcal{L}' \subset \operatorname{span} \mathcal{L} =: V$. Then $\pi(\mathcal{L})$ is a lattice.

Proof. By the linearity of π, $\pi(\mathcal{L})$ is an additive subgroup of V. Suppose $\pi(\mathcal{L})$ is not discrete. This means 0 is an accumulation point of $\pi(\mathcal{L})$, so we can find a sequence of points $\{x_n\}_{n\in\mathbb{N}} \subset \mathcal{L}$ and assume without loss of generality that the projections $\pi(x_n)$ are pairwise distinct and bounded by $\|\pi(x_n)\| \leq 1$ for all $n \in \mathbb{N}$.

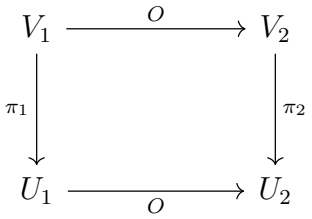
For each $n \in \mathbb{N}$, we can decompose x_n as $x_n = \pi(x_n) + u_n$, where $u_n \in U$. Because $\mathcal{L}'$ is a lattice in U, the distance from any point in U to its closest lattice point in $\mathcal{L}'$ is bounded by the covering radius $\rho(\mathcal{L}')$. Thus, we can choose $x'_n \in \mathcal{L}'$ such that $\|u_n - x'_n\| \leq \rho(\mathcal{L}') =: C$ for all $n \in \mathbb{N}$.

Now, define $v_n = x_n - x'_n \in \mathcal{L}$. Since $x'_n \in U$, we have $\pi(v_n) = \pi(x_n)$, meaning the constructed lattice points v_n must be pairwise distinct. However, by the triangle inequality, the norm of v_n is bounded by

$$\|v_n\| = \|\pi(x_n) + u_n - x'_n\| \leq \|\pi(x_n)\| + \|u_n - x'_n\| \leq 1 + C$$

for all $n \in \mathbb{N}$. This yields infinitely many distinct points $v_n \in \mathcal{L}$ within a bounded ball, contradicting the discreteness of $\mathcal{L}$. Hence, $\pi(\mathcal{L})$ must be discrete. □

Lemma 3.2. Let $O : V_1 \to V_2$ be a surjective isometry between two Euclidean vector spaces. For $i \in \{1, 2\}$, let $U_i \subset V_i$ be a subspace such that $O(U_1) = U_2$, and let $\pi_i := \pi_{U_i} : V_i \to U_i$ be the orthogonal projection onto U_i. Then the following diagram commutes.



Proof. Since the isometry O preserves orthogonality, it maps the orthogonal decomposition $V_1 = U_1 \oplus U_1^\perp$ naturally to $V_2 = U_2 \oplus U_2^\perp$. Because the projections π_i act precisely by isolating the U_i-component of these respective decompositions, the diagram commutes. □

Lemma 3.3. Let O be an isomorphism between two lattices $\mathcal{L}_1$ and $\mathcal{L}_2$. Define $V_i := \operatorname{span} \mathcal{L}_i$, $\mathcal{S}_i := \mathcal{S}(\mathcal{L}_i)$, $U_i := \operatorname{span} \mathcal{S}_i$, and $\pi_i : V_i \to U_i^\perp$ the orthogonal projection $\pi_{U_i}^\perp$ for $i \in \{1, 2\}$. Then the decomposition $O = O\big|_{U_1} \oplus O\big|_{U_1^\perp}$ induces

(i) a spherical code isomorphism $O\big|_{U_1} : U_1 \to U_2$ between the spherical codes $\hat{\mathcal{S}}_1$ and $\hat{\mathcal{S}}_2$, and

(ii) a lattice isomorphism $O\big|_{U_1^\perp} : U_1^\perp \to U_2^\perp$ between the lattices $\pi_1(\mathcal{L}_1)$ and $\pi_2(\mathcal{L}_2)$.

Proof. For the first part, we have $O(\mathcal{S}_1) = \mathcal{S}_2$ as isometries are norm-preserving, and thus $O(\hat{\mathcal{S}}_1) = \hat{\mathcal{S}}_2$ as they are linear as well.

For the second part, note that $\mathcal{L}_i \cap U_i$ is a sublattice of $\mathcal{L}_i$ for $i \in \{1, 2\}$, so by Lemma 3.1, $\pi_i(\mathcal{L}_i)$ are lattices as well. Additionally, by Lemma 3.2, we have $O(\pi_1(\mathcal{L}_1)) = \pi_2(O(\mathcal{L}_1)) = \pi_2(\mathcal{L}_2)$, so indeed $O\big|_{U_1^\perp}$ is a lattice isomorphism. □

Algorithm 1 Lattice Isomorphism Enumeration Reduction to Spherical Codes

Input: Two lattices $\mathcal{L}_1$ and $\mathcal{L}_2$ and an oracle $\mathcal{A}$ for enumerating isomorphisms between spherical codes.

Output: $\mathrm{Isom}(\mathcal{L}_1, \mathcal{L}_2)$.

1: **if** $\mathcal{L}_1 = \{0\}$ and $\mathcal{L}_2 = \{0\}$ **then**
2: **return** $\{0 \mapsto 0\}$
3: **for** $i \in \{1, 2\}$ **do**
4: $\mathcal{S}_i \leftarrow \mathcal{S}(\mathcal{L}_i)$ using Theorem 2.13.
5: $\pi_i \leftarrow$ the orthogonal projection from span $\mathcal{L}_i$ onto the orthogonal complement of span $\mathcal{S}_i$.
6: **if** $\lambda(\mathcal{L}_1) \neq \lambda(\mathcal{L}_2)$ **then**
7: **return** $\varnothing$
8: $\mathrm{Isom}_{\mathrm{SC}} \leftarrow \mathcal{A}(\hat{\mathcal{S}}_1, \hat{\mathcal{S}}_2)$.
9: $\mathrm{Isom}_{\pi} \leftarrow \textsc{Recurse}(\pi_1(\mathcal{L}_1), \pi_2(\mathcal{L}_2), \mathcal{A})$.
10: **for** each $O_{\mathrm{SC}} \in \mathrm{Isom}_{\mathrm{SC}}$ and $O_{\pi} \in \mathrm{Isom}_{\pi}$ **do**
11: $O \leftarrow O_{\mathrm{SC}} \oplus O_{\pi}$.
12: **if** $O(\mathcal{L}_1) = \mathcal{L}_2$ **then**
13: **yield** O.

Theorem 3.4. Algorithm 1 works correctly as specified and its time complexity is asymptotically bounded by that of the oracle $\mathcal{A}$.

Proof. Correctness follows directly by Lemma 3.3. By Corollary 2.8, the oracle $\mathcal{A}$ runs in time $\Omega\left(n^{n+o(n)}\right)$ on any isomorphic pair of n-dimensional lattice-induced spherical codes. Therefore, we must show that accumulated runtime of the loops at lines 3 and 10 is $n^{n+o(n)}$.

For each of the r recursive calls, let n_t denote the dimension of the spherical codes induced by the input lattices, and note that their sum is equal to n. Computing the sets of shortest vectors in the loop at line 3, takes time $2^{\mathcal{O}(n)}$ by Theorem 2.13 each call, so $r \cdot 2^{\mathcal{O}(n)} = 2^{\mathcal{O}(n)} \leq n^{n+o(n)}$ in total.

For $i \in \{1, 2\}$, let $\hat{\mathcal{L}}_i$ denote the smallest sublattice of $\mathcal{L}_i$ containing its shortest vectors $\hat{\mathcal{S}}_i$. Observe that these lattices are n_t-dimensional and that the set of lattice isomorphisms $\mathrm{Isom}(\hat{\mathcal{L}}_1, \hat{\mathcal{L}}_2)$ is equal to $\mathrm{Isom}_{\mathrm{SC}}$ from line 8, meaning that by Corollary 2.8 this set is bounded by $n_t^{n_t+o(n_t)}$ each call. This implies that the total runtime of the loop at line 10 is

$$\prod_{t=1}^{r} n_t^{n_t+o(n_t)} \leq \prod_{t=1}^{r} n^{n_t+o(n)} = n^{n+o(n)},$$

as required. □

4 Bounds on Spherical Codes

In order to analyze the complexity of our algorithms, we first need to establish some bounds on $A(n, \alpha)$.

4.1 Linear Programming Bounds

Recall that for small α we have the following bound.

Theorem 4.1 ([DGS77, Example 4.6][3])**.** Let S be a spherical (n, α)-code satisfying $0 < \alpha \leq 1/\left(1 + \sqrt{n+3}\right)$. Then

$$|S| \leq \tfrac{1}{2}n^2 + \tfrac{3}{2}n.$$

This bound by Delsarte is an example of a linear programming bound. The charm of LP bounds is that they are obtained through translating a discrete geometry problem into a continuous analytical one. The underlying theory is deeply rooted in harmonic analysis, which is not straightforward on the hypersphere as it lacks a natural group structure. Rather than delving into how this exact bound is achieved, we can illustrate the core mechanics of linear programming bounds by temporarily deviating to sphere packings in Euclidean space $\mathbb{R}^n$. Since $\mathbb{R}^n$ is a locally compact abelian group, we can utilize classical, much more manageable Fourier analysis.

Specifically, we will present the proof of the sphere packing bound established by Cohn and Elkies [CE03], which was later used by Viazovska to prove the optimal sphere packing in dimensions 8 and 24 [Via17; Coh+17]. As seen in Theorem 4.3, finding a bound on an optimal packing reduces to finding an admissible function $f : \mathbb{R}^n \to \mathbb{R}$ that satisfies a set of linear constraints, much like the theorem behind Delsarte's bound.

The goal in sphere packings is to pack $\mathbb{R}^n$ with non-overlapping equal-sized balls as densely as possible. We restrict ourselves to periodic sphere packings, as any general packing can be apprioximated by a periodic one [CE03, Appendix A].

Definition 4.2. A *periodic sphere packing* $\mathcal{P}$ in $\mathbb{R}^n$ is a union of balls $B_{r/2}$ of radius $r/2$ centered at $x + t$ for every point x in a lattice $\mathcal{L}$ and for every offset $t \in \{t_1, \ldots, t_m\}$, such that for any two distinct offsets t_j, t_k $(1 \leq j < k \leq m)$ we have

1. $t_j - t_k \notin \mathcal{L}$, and
2. $\|x + t_j - t_k\| \geq r$ for every $x \in \mathcal{L}$.

[3] Delsarte gave a marginally stronger bound as shown in Equation (2), but the variant presented here suffices for simplicity.

The *packing density* $\Delta(\mathcal{P})$ is defined as

$$\frac{m \cdot \operatorname{vol} B_{r/2}}{\det \mathcal{L}},$$

where vol denotes the Lebesgue measure.

Theorem 4.3 ([CE03, Theorem 3.2])**.** Let $f : \mathbb{R}^n \to \mathbb{R}$ be an *admissible* function and r a positive real number, such that

1. $f(0) = \hat{f}(0) > 0$,
2. $f(x) \leq 0$ for all $x \in \mathbb{R}^n$ with $\|x\| \geq r$, and
3. $\hat{f}(t) \geq 0$ for all $t \in \mathbb{R}^n$.

Then the packing density $\Delta(\mathcal{P})$ of any periodic sphere packing $\mathcal{P}$ in $\mathbb{R}^n$ is at most $\operatorname{vol} B_{r/2}$.

The admissibility of f is a technical condition ensuring absolute convergence, which allows us to invoke Proposition 4.5 for any lattice $\mathcal{L}$.

Definition 4.4. A function $f : \mathbb{R}^n \to \mathbb{R}$ is admissible if there exist constants $C, \delta > 0$ such that

$$|f(x)|,\ |\hat{f}(x)| \leq C(1 + \|x\|)^{-n-\delta}$$

for all $x \in \mathbb{R}^n$.

Before we prove Theorem 4.3, we will state the generalization of the Poisson Summation Formula.

Proposition 4.5 (Generalized Poisson Summation Formula)**.** Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice, $f : \mathbb{R}^n \to \mathbb{R}$ an *admissible* function, and $v \in \mathbb{R}^n$. Then

$$\sum_{x \in \mathcal{L}} f(x + v) = \frac{1}{\det \mathcal{L}} \sum_{x^* \in \mathcal{L}^*} \hat{f}(x^*) e^{2\pi i \langle x^*, v \rangle}.$$

Proof of Theorem 4.3. Let $\mathcal{P}$ be a periodic sphere packing in $\mathbb{R}^n$ defined by a lattice $\mathcal{L} \subset \mathbb{R}^n$ and offsets $t_1, \ldots, t_m \in \mathbb{R}^n$. By the scale-invariance of the density, we may assume without loss of generality that the spheres have radius $r/2$.

Summing twice over all offsets and using Proposition 4.5 yields

$$\begin{aligned}
\sum_{j=1}^{m} \sum_{k=1}^{m} \sum_{x \in \mathcal{L}} f(x + t_j - t_k) &= \sum_{j=1}^{m} \sum_{k=1}^{m} \frac{1}{\det \mathcal{L}} \sum_{x^* \in \mathcal{L}^*} \hat{f}(x^*) e^{2\pi i \langle x^*, t_j - t_k \rangle} \\
&= \frac{1}{\det \mathcal{L}} \sum_{x^* \in \mathcal{L}^*} \hat{f}(x^*) \left(\sum_{j=1}^{m} e^{2\pi i \langle x^*, t_j \rangle} \right) \left(\sum_{k=1}^{m} e^{-2\pi i \langle x^*, t_k \rangle} \right) \\
&= \frac{1}{\det \mathcal{L}} \sum_{x^* \in \mathcal{L}^*} \hat{f}(x^*) \left| \sum_{j=1}^{m} e^{2\pi i \langle x^*, t_j \rangle} \right|^2.
\end{aligned}$$

Note that the right-hand side is at least $m^2 \cdot \hat{f}(0)/\det \mathcal{L}$ by the non-negativity of $\hat{f}$. On the other hand, we can upper bound the left-hand side by observing that

- when $x = 0$ and $j = k$, we have $f(x + t_j - t_k) = f(0) > 0$, and
- otherwise, when $x \neq 0$ or $j \neq k$, we have $\|x + t_j - t_k\| \geq r$, ensuring $f(x + t_j - t_k) \leq 0$.

This means the only positive contributions come from the m instances where $x = 0$ and $j = k$, yielding an overall upper bound of $m \cdot f(0)$. Combining these inequalities we get

$$m \cdot f(0) \geq m^2 \cdot \hat{f}(0)/\det \mathcal{L},$$

which, after simplification, gives the desired bound on the density of $\mathcal{P}$. □

4.2 The Tensor Power Trick

Theorem 4.1 is too weak for our purposes, as the maximum inner product of our lattice-induced spherical codes will reach $1/2$ (see Proposition 2.15). Fortunately, we can amplify Theorem 4.1 almost effortlessly using the "tensor power trick", akin to [Tao13].

Proposition 4.6. Let V be a real vector space and let k be an odd positive integer. Then

$$\begin{aligned} V &\longrightarrow V^{\otimes k} \\ v &\longmapsto v^{\otimes k} \end{aligned}$$

is an injective map.

Proof. Let $v, w \in V$ such that $v^{\otimes k} = w^{\otimes k}$. For any linear form $f : V \to \mathbb{R}$, we have

$$f(v)^k = f^{\otimes k}(v^{\otimes k}) = f^{\otimes k}(w^{\otimes k}) = f(w)^k,$$

implying $f(v) = f(w)$ since $x \mapsto x^k$ is injective over $\mathbb{R}$ for odd k. In other words, $f(v - w) = 0$ for all linear forms $f : V \to \mathbb{R}$, hence $v = w$. □

Corollary 4.7. Let k be an odd positive integer and let S be a spherical (n, α)-code satisfying

$$0 < \alpha \leq \left(1 + \sqrt{N + 3}\right)^{-1/k}, \quad \text{where} \quad N := \binom{n + k - 1}{k}.$$

Then

$$|S| \leq \tfrac{1}{2}N^2 + \tfrac{3}{2}N.$$

Proof. Let

$$S' := \left\{ v^{\otimes k} \in V^{\otimes k} \mid v \in S \right\}$$

where $V = \operatorname{span} S$. Then $S' \subset \operatorname{Sym}(V^{\otimes k})$ is a spherical $\left(N', \alpha^k\right)$-code, by the identity

$$\langle v^{\otimes k}, w^{\otimes k} \rangle = \langle v, w \rangle^k$$

for all $v, w \in V$, and the fact that $x \mapsto x^k$ is increasing for odd k. Moreover, we have $N' \leq \binom{n+k-1}{k} = N$ by Proposition 2.19. Hence, applying Theorem 4.1 to S' yields the desired bound on $|S'|$, and thus on $|S|$ by Proposition 4.6. □

Choosing k appropriately gives the following explicit exponential bound.

Theorem 4.8. Let S be a spherical (n, α)-code satisfying $1/\left(1 + \sqrt{n+3}\right) < \alpha \leq 1/2$. Then

$$|S| \leq \left(\frac{C}{\alpha^2}\right)^{C\alpha^2 n}$$

for some absolute constant $C > 0$.

Proof. Note that

$$1 + \sqrt{N+3} \leq 3\sqrt{N}$$

for $N := \binom{n+k-1}{k} \geq 1$. Hence, using the bound $\binom{a}{b} \leq (ea/b)^b$, we have

$$\left(1 + \sqrt{N+3}\right)^{-1/k} \geq \left(3\sqrt{N}\right)^{-1/k} = 3^{-1/k} N^{-1/(2k)} \geq 3^{-1/k} \sqrt{\frac{k}{e(n+k-1)}}.$$

We require this lower bound to be at least α, which is equivalent to

$$k \geq \left(\frac{e \cdot 3^{2/k}}{1 - e \cdot 3^{2/k} \cdot \alpha^2}\right) \alpha^2 (n-1),$$

provided the denominator is positive. Since $\alpha \leq 1/2$, we have $e \cdot \alpha^2 \leq e/4 < 1$. Therefore, for sufficiently large k, the term $3^{2/k}$ approaches 1, ensuring the denominator remains strictly positive. In addition, this implies the fractional factor is bounded from above by an absolute constant. Furthermore, the assumption $\alpha > 1/\left(1 + \sqrt{n+3}\right)$ implies $\alpha^2 n > 1/9$, bounding it strictly away from zero. This guarantees that setting k as the largest odd integer not exceeding $C'\alpha^2 n$, for a sufficiently large absolute constant $C' > 0$, yields a valid k large enough to satisfy

the inequality. The condition of Corollary 4.7 is thus met, and gives

$$\begin{aligned}
|S| &\leq \tfrac{1}{2}N^2 + \tfrac{3}{2}N \leq 2N^2 \\
&= 2\binom{n+k-1}{k}^2 \\
&\leq 2\left(e\left(n+k-1\right)/k\right)^{2k} \\
&\leq 2\left(ne\left(1+C'/4\right)/k\right)^{2k} \\
&\leq \left(\frac{C}{\alpha^2}\right)^{C\alpha^2 n}
\end{aligned}$$

for some absolute constant $C > 0$ if C' is chosen sufficiently large. □

5 Algorithms

5.1 The Naive Approach

In Algorithm 1, we reduced LIP, which requires finding certain linear maps between *infinite* sets, to SCIP, which requires finding maps between *finite* sets. This reduction immediately yields a finite-time solution to Problem 2.3 via the naive algorithm presented in Algorithm 2.

Algorithm 2 Naive Spherical Code Isomorphism Enumeration

Input: S_1, S_2 two spherical (n, α)-codes, satisfying $\alpha \leq 1/2$.
Output: $\mathrm{Isom}(S_1, S_2)$.

1: choose an arbitrary linearly independent n-tuple $(v_1, \ldots, v_n) \in S_1^n$.
2: **for** each linearly independent n-tuple $(w_1, \ldots, w_n) \in S_2^n$ **do**
3: $\quad O \leftarrow$ the linear map $\operatorname{span} S_1 \to S_2$ defined by $v_1 \mapsto w_1, \ldots, v_n \mapsto w_n$.
4: $\quad$ **if** O is an isometry satisfying $O(S_1) = S_2$ **then**
5: $\quad\quad$ **yield** O.

Moreover, Theorem 4.1 establishes that for sufficiently small α, the size of a spherical code is polynomial in its dimension. Consequently, on this restricted class of spherical codes, the naive algorithm already achieves our target complexity of $n^{\mathcal{O}(n)}$.

Theorem 5.1. Algorithm 2 correctly enumerates $\mathrm{Isom}(S_1, S_2)$ in time $2^{\mathcal{O}\left(n^2\right)}$. Furthermore, if $\alpha \leq 1/\left(1+\sqrt{n+3}\right)$, then the algorithm runs in time $n^{\mathcal{O}(n)}$.

Proof. Correctness is clear, and the time complexity is asymptotically bounded by the number of iterations of the loop in line 2. The number of tuples checked is at

most $|S_2|^n$. In general, Theorem 4.8 implies $|S_2|^n \leq \left(2^{\mathcal{O}(n)}\right)^n = 2^{\mathcal{O}\left(n^2\right)}$. Additionally, if $\alpha \leq 1/\left(1+\sqrt{n+3}\right)$, then Theorem 4.1 gives $|S_2|^n \leq \left(\mathcal{O}\left(n^2\right)\right)^n = n^{\mathcal{O}(n)}$. □

5.2 The Special Case

The main idea behind Algorithm 3 is to find smaller dimensional spherical codes, induced by taking a "slice" from the two given codes. Then, we recursively find the isomorphisms of these smaller codes. A repeated decrease in dimension alone still results in a $2^{\mathcal{O}\left(n^2\right)}$ time complexity, however. To actually achieve the goal complexity of $n^{\mathcal{O}(n)}$, we choose these slices such that the maximal inner product of their induced spherical codes decrease as much as possible at each recursive step.

Definition 5.2. Let S be a spherical (n,α)-code. We define a *slice* as

$$H_S(v,\beta) := \{h \in S \mid \langle v,h\rangle = \beta\},$$

and its induced spherical code as

$$\hat{S}(v,\beta) := \left\{ \frac{1}{\sqrt{1-\beta^2}} \cdot \pi(h) \mid h \in H_S(v,\beta) \right\}$$

for $v \in S$ and $\beta \in (-1,\alpha]$, where π is defined as projection onto the orthogonal complement $\{v\}^\perp$.

Proposition 5.3. Let S be a spherical (n,α)-code and let $v \in S$ and $\beta \in (-1,\alpha]$. Then $\hat{S}(v,\beta)$ is a spherical (n^*,α^*)-code satisfying $n^* \leq n-1$ and

$$\alpha^* = \frac{\gamma-\beta^2}{1-\beta^2} \quad \text{with} \quad \gamma = \max\{\langle h_1,h_2\rangle \in \mathbb{R} \mid \text{distinct } h_1,h_2 \in H_S(v,\beta)\} \leq \alpha.$$

Proof. Per definition, we have $\hat{S}(v,\beta) \subset \{v\}^\perp$, proving $n^* \leq n-1$. Let $h_1,h_2 \in H_S(v,\beta)$ be distinct points and define $h_i^\perp := \pi^\perp_{\operatorname{span} v}(h_i) = h_i - \beta v$ for $i \in \{1,2\}$. Working out the inner product of the corresponding points in $\hat{S}(v,\beta)$, we obtain

$$\frac{1}{1-\beta^2} \cdot \langle h_1^\perp, h_2^\perp\rangle = \frac{1}{1-\beta^2} \cdot \left(\langle h_1,h_2\rangle - \beta^2\right) \leq \frac{\gamma-\beta^2}{1-\beta^2} = \alpha^*.$$ □

Note that if $\beta = \alpha$, we have $\alpha^* \leq \alpha/(1+\alpha)$. The decrease of the maximum inner product (or equivalently the increase in angle) can visually be observed in Figure 1.

Proposition 5.4. Let S_1 be a spherical (n,α)-code with a point $v \in S_1$ such that $\dim \operatorname{span} S_1^* = n-1$ where $S_1^* := \hat{S}_1(v,\beta)$ for some $\beta \in (-1,\alpha]$. Let S_2 be another spherical (n,α)-code isomorphic to S_1 via $O : \operatorname{span} S_1 \to \operatorname{span} S_2$. Then $O\big|_{\{v\}^\perp}$ is an isomorphism between the codes S_1^* and $S_2^* := \hat{S}_2(O(v),\beta)$.

Proof. This immediately follows from Lemma 3.2. □

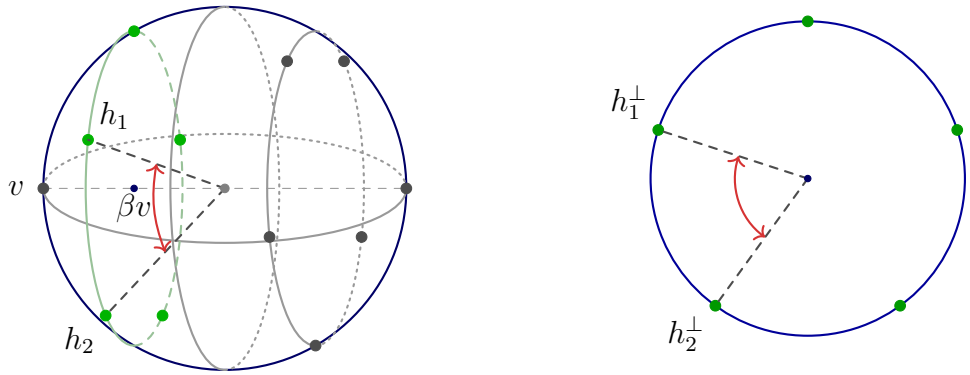


Figure 1: On the left, a 3-dimensional spherical code S is displayed, with a slice $H_S(v, \beta)$ highlighted in green. On the right, its induced 2-dimensional spherical code $\hat{S}(v, \beta)$ is shown. One may visually observe that the angle between h_1 and h_2 in the original code is slightly smaller than the angle between the corresponding $h_1^\perp$ and $h_2^\perp$ in the slice-induced code.

Algorithm 3 Spherical Code Isomorphism Enumeration – Special Case

Input: S_1, S_2 two spherical (n, α)-codes, satisfying $\alpha \leq 1/2$.
Output: $\mathrm{Isom}(S_1, S_2)$ or Fail.

if $\alpha \leq 1/\left(1 + \sqrt{n+3}\right)$ **then**
  return Naive Spherical Code Isomorphism Enumeration(S_1, S_2)
$v_1 \leftarrow$ an arbitrary element in S_1 such that $\dim \operatorname{span} \hat{S}_1(v_1, \alpha) = n - 1$ or **throw** Fail if no such element exists.
$S_1^* \leftarrow \hat{S}_1(v_1, \alpha), \quad \alpha^* \leftarrow \alpha(S_1^*)$. ▷ $\alpha^* \leq \alpha/(1+\alpha)$ *by Proposition 5.3*
for each $v_2 \in S_2$ such that $\dim \operatorname{span} \hat{S}_2(v_2, \alpha) = n - 1$ **do**
  $S_2^* \leftarrow \hat{S}_2(v_2, \alpha)$.
  if $\alpha^* = \alpha(S_2^*)$ **then**
    $\mathrm{Isom}^* \leftarrow$ Recurse(S_1^*, S_2^*). ▷ S_1^* *and* S_2^* *are spherical* $(n-1, \alpha^*)$*-codes*
    for each $O^* \in \mathrm{Isom}^*$ **do**
      $O \leftarrow$ the extension of O^* by $v_1 \mapsto v_2$. ▷ *By Proposition 5.4*
      if $O(S_1) = S_2$ **then**
        yield O.

Theorem 5.5. Algorithm 3 works correctly as specified and runs in time $n^{\mathcal{O}(n)}$.

Proof. Correctness follows directly from Proposition 5.4 and the time complexity is asymptotically dominated by the size of the recursion tree. From every function call, the number of candidates v_2 in line 5 is at most $|S_2|$, and for each candidate, we make a recursive call at line 8 on spherical codes of dimension $n-1$ and inner product at most $\alpha/(1+\alpha)$. Thus, assuming the worst case, our recursion tree is bounded by the product

$$\prod_{i=1}^{n} A(d_i, \alpha_i),$$

where $d_1 = n$, $\alpha_1 = 1/2$, and for each $i \in \{2, \dots, n\}$,

$$d_i = d_{i-1} - 1 = n - i + 1, \quad \alpha_i = \frac{\alpha_{i-1}}{1+\alpha_{i-1}} = \frac{1}{i+1}.$$

We next claim that this tree size (and thus the runtime complexity) is asymptotically bounded by $n^{\mathcal{O}(n)}$.

Up to some index $j \in \{1, \dots, n\}$, the maximum inner products α_i (for $i \le j$) satisfy the conditions required to apply Theorem 4.8, while for the remaining indices $i > j$, the polynomial bound from Theorem 4.1 applies. Rather than decomposing the product at this threshold index j, we exploit the non-negativity of the logarithms of both bounds to sum over all indices simultaneously, yielding the upper bound

$$\log\left(\prod_{i=1}^{n} A(d_i, \alpha_i)\right) = \sum_{i=1}^{n} \log A(d_i, \alpha_i) \le \sum_{i=1}^{n} \log\left(d_i^{\mathcal{O}(1)}\right) + \sum_{i=1}^{n} C\alpha_i^2 d_i \log\left(C/\alpha_i^2\right).$$

The first sum is trivially bounded by $\mathcal{O}(n \log n)$. Since the summands of the second sum are non-increasing with respect to i over the interval $[1, n]$, bounding the sum by its corresponding integral[4] and omitting the constant factor C produces

$$\begin{aligned}
\sum_{i=1}^{n} \alpha_i^2 d_i \log\left(1/\alpha_i^2\right) &\le \frac{n \log 2}{4} + \int_1^n \frac{n-x+1}{(x+1)^2} \cdot \log(x+1)\,\mathrm{d}x \\
&= \frac{n \log 2}{4} + \left[-\frac{(n+2)(\log x + 1)}{x} - \frac{1}{2}(\log x)^2\right]_2^{n+1} \\
&= \mathcal{O}(n).
\end{aligned}$$

[4]Here we use the fact that if $f : [a, b] \to \mathbb{R}$ is a decreasing function, then $\sum_{i=a}^{b} f(i) \le f(a) + \int_a^b f(t)\,\mathrm{d}t$.

Combining these two evaluations, we conclude

$$\prod_{i=1}^{n} A(d_i, \alpha_i) = 2^{\mathcal{O}(n \log n)+\mathcal{O}(n)} = n^{\mathcal{O}(n)}. \qquad \square$$

5.3 The General Case

The issue with Algorithm 3 is that the slice at $\beta = \alpha$ in lines 3 and 5, may not always have dimension $n-1$. It could even be one-dimensional from every point, as is the case with the spherical $(2n, 1/2)$-code $H^{\oplus n}$, constructed as a direct sum of n orthogonal copies of the 2-dimensional kissing number configuration H, as visualized in Figure 2. Nevertheless, this should be actually one of the easier cases, as the size of this code is only polynomial in n. We generalize this observation in Lemma 5.7, by considering spherical caps.

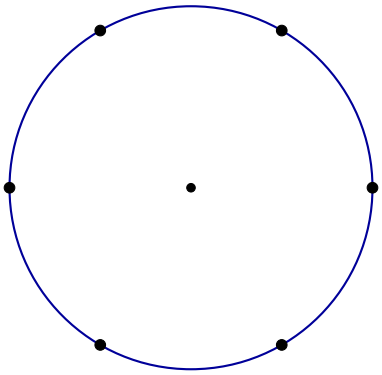


Figure 2: A projection of the spherical $(2n, 1/2)$-code $H^{\oplus n}$ onto one component H, the 2-dimensional kissing number configuration. This shows that from any fixed point, there are just 2 points that are at angle 60° (or inner product 1/2) with this point, while the vast majority of other points are orthogonal.

Definition 5.6. Let S be a spherical (n, α)-code. We define a *spherical cap* as

$$H_S^{\varepsilon}(v) := \{h \in S \mid \langle v, h \rangle > \varepsilon\}$$

for some point $v \in S$ and $\varepsilon \in [0, \alpha)$.

Lemma 5.7. Let S be a spherical (n, α)-code. Define

$$d = \max_{v \in S} \dim \operatorname{span} H_S^{\varepsilon}(v)$$

for some $\varepsilon \in [0, \alpha)$. Then

$$|S| \leq A(n, \varepsilon) \cdot A(d, \alpha).$$

Proof. Let $S' \subset S$ be a subset of points such that

- the collection of the spherical caps $S_v := H^\varepsilon_S(v)$ forms a covering

$$\bigcup_{v \in S'} S_v = S,$$

  and

- $v \in S_w$ if and only if $v = w$ for all $v, w \in S'$, which directly implies that the inner product of any two distinct vectors in S' is at most ε.

Consequently, S' forms a spherical (n', α')-code where $n' \leq n$ and $\alpha' < \varepsilon$, while each patch S_v forms a spherical (d_v, α_v)-code where

$$d_v = \dim \operatorname{span} H^\varepsilon_S(v) \leq d$$

and $\alpha_v \leq \alpha$ for all $v \in S'$. Using the covering property of these patches, we obtain the bound

$$|S| \leq \sum_{v \in S'} |S_v| \leq |S'| \cdot \max_{v \in S'} |S_v| \leq A(n, \varepsilon) \cdot A(d, \alpha). \qquad \square$$

Applying this lemma to $S = H^{\oplus n}$ and $\varepsilon = 0$, we indeed get a polynomial bound $A(2n, 0) \cdot A(2, 1/2) = 4n \cdot 6$ on $|S|$.

We will now present Algorithm 4 which utilizes this new bounding method, running in time $n^{\mathcal{O}(n^{3/2})}$. The overall strategy is to find a point $v_1 \in S_1$ such that the dimension d of its spherical cap $H^\varepsilon_{S_1}(v_1)$ is maximized, for a thoughtfully chosen ε that depends on the current α and n. Selecting our points this way allows us to invoke Lemma 5.7 to bound the codes when analyzing the algorithm's complexity. We then enumerate all points $v_2 \in S_2$ whose spherical cap has the same dimension d. For every such point v_2, we create a partial candidate isomorphism $v_1 \mapsto v_2$ and find all its extensions to the whole code in two steps.

First, we find its extensions on the d-dimensional subcodes $H_1 := H^\varepsilon_{S_1}(v_1)$ and $H_2 := H^\varepsilon_{S_2}(v_2)$. Since we have already fixed one dimension, we recurse on spherical codes of dimension at most $d - 1$, and with a decreased maximum inner product as all points in H_1 and H_2 lie close to v_1 and v_2 respectively. Second, we extend the partial isomorphisms found in the first step to the whole code by recursing on slices of dimension at most $n - d$, but now without guaranteeing a decrease in the maximum inner product.

This means that, if d is relatively small each recursive step, we recurse on spherical codes with only a marginally smaller dimension and, in the worst-case, a mostly unchanged maximum inner product. While this naively suggests a $2^{\mathcal{O}(n^2)}$ time complexity, utilizing the structural bound from Lemma 5.7 mitigates this.

Conversely, if the dimension d is large in every recursive step, we recurse on spherical codes with a strictly smaller maximal inner product. This precisely replicates the effect of the slicing strategy of Algorithm 3, ensuring the branching factor shrinks rapidly enough to break the baseline $2^{\mathcal{O}(n^2)}$ complexity.

In order to formalize the mentioned extending process, we first need to generalize the slices from Definition 5.2 in the following way.

Definition 5.8. Let S be a spherical (n, α)-code. We define the *generalized slice* as

$$H_S(U, h) := \{h^* \in S \mid \pi_U(h^*) = h\}$$

and its induced spherical code

$$\hat{S}(U, h) := \frac{1}{\sqrt{1 - \|h\|^2}} \cdot \pi_U^{\perp}(H_S(U, h))$$

for a subspace U of $\operatorname{span} S$ and $h \in U$ with norm less than 1.

Note that Definition 5.8 coincides with Definition 5.2 when setting $U = \operatorname{span} v$ and $h = \beta \cdot v$. We also generalize Propositions 5.3 and 5.4 easily and omit the proofs, as they are analogous to the proofs of the special case.

Proposition 5.9. Let S be a spherical (n, α)-code. Let U be a subspace of $\operatorname{span} S$ and $h \in U$ with $\|h\| < 1$. Then $\hat{S}(U, h)$ is a spherical (n^*, α^*)-code satisfying $n^* \leq n - \dim U$ and

$$\alpha^* = \frac{\gamma - \|h\|^2}{1 - \|h\|^2} \quad \text{with} \quad \gamma = \max\{\langle h_1, h_2 \rangle \in \mathbb{R} \mid \text{distinct } h_1, h_2 \in H_S(U, h)\} \leq \alpha.$$

Proposition 5.10. Let S_1 be a spherical (n, α)-code. Let U be a subspace of $\operatorname{span} S_1$ and $h \in U$ with $\|h\| < 1$. Define $U' := \operatorname{span} S_1^* \subset U^{\perp}$ where $S_1^* := \hat{S}_1(U, h)$. Let S_2 be another spherical (n, α)-code isomorphic to S_1 via $O : \operatorname{span} S_1 \to \operatorname{span} S_2$. Then $O\big|_{U'}$ is an isomorphism between the codes S_1^* and $S_2^* := \hat{S}_2(O(U), O(h))$.

Algorithm 4 Spherical Code Isomorphism Enumeration – General Case

Input: S_1, S_2 two spherical (n, α)-codes, satisfying $\alpha \leq 1/2$.
Output: $\mathrm{Isom}(S_1, S_2)$.

function FIND ISOMORPHISMS(S_1, S_2)
  $\varepsilon \leftarrow \alpha / n^{1/4}$.
  $d \leftarrow \max_{v \in S_1} \dim \operatorname{span} H^{\varepsilon}_{S_1}(v)$.
  if $d \neq \max_{v \in S_2} \dim \operatorname{span} H^{\varepsilon}_{S_2}(v)$ **then**
    return $\varnothing$.
  $v_1 \leftarrow$ an arbitrary element in S_1 such that $\dim \operatorname{span} H^{\varepsilon}_{S_1}(v_1) = d$.
  for each $v_2 \in S_2$ such that $\dim \operatorname{span} H^{\varepsilon}_{S_2}(v_2) = d$ **do**
    $H_1 \leftarrow H^{\varepsilon}_{S_1}(v_1), \quad H_2 \leftarrow H^{\varepsilon}_{S_2}(v_2)$. ▷ *Two d-dimensional spherical codes*
    $\mathrm{Isom} \leftarrow$ EXTEND ISOMORPHISM($H_1, H_2, v_1 \mapsto v_2$).
    for each $O \in \mathrm{Isom}$ **do**
      if $O(S_1 \cap \operatorname{span} H_1) = S_2 \cap \operatorname{span} H_2$ **then**
        yield from EXTEND ISOMORPHISM(S_1, S_2, O).

Input: Two spherical codes S_1, S_2 and an isometry $O : U_1 \to U_2$ such that $O(S_1 \cap U_1) = S_2 \cap U_2$.
Output: The set $\left\{ O^* \in \mathrm{Isom}(S_1, S_2) \mid O^*\big|_{U_1} = O \right\}$.

function EXTEND ISOMORPHISM($S_1, S_2, O : U_1 \to U_2$)
  $\mathrm{Isom} \leftarrow \{O\}$.
  $U_1^* \leftarrow U_1, \quad U_2^* \leftarrow U_2$.
  while $U_1^* \neq \operatorname{span} S_1$ **do**
    $h^* \leftarrow$ an arbitrary element in $S_1 \setminus U_1^*, \quad h \leftarrow \pi_{U_1}(h^*)$.
    $S_1^* \leftarrow \hat{S}_1(U_1, h), \quad S_2^* \leftarrow \hat{S}_2(U_2, O(h))$. ▷ *Definition 5.8*
    if $\dim \operatorname{span} S_1^* \neq \dim \operatorname{span} S_2^*$ or $\alpha(S_1^*) \neq \alpha(S_2^*)$ **then**
      return $\varnothing$.
    $\mathrm{Isom}_{\mathrm{new}} \leftarrow$ FIND ISOMORPHISMS(S_1^*, S_2^*).
    $W \leftarrow U_1^* \cap \operatorname{span} S_1^*$. ▷ *Determine subspace overlap*
    $U_1^* \leftarrow U_1^* + \operatorname{span} S_1^*, \quad U_2^* \leftarrow U_2^* + \operatorname{span} S_2^*$.
    $\mathrm{Isom} \leftarrow$ GLUE ISOMORPHISMS($\mathrm{Isom}, \mathrm{Isom}_{\mathrm{new}}, W, U_1^*, U_2^*, S_1, S_2$).
  return Isom.

Input: $\mathcal{F}, \mathcal{G}$ are sets of isometries defined on respective domains V, V', with intersection $W = V \cap V'$.
Output: The set of glued isometries on $U_1^* = V + V'$ that map $U_1^* \cap S_1$ to $U_2^* \cap S_2$.

function GLUE ISOMORPHISMS($\mathcal{F}, \mathcal{G}, W, U_1^*, U_2^*, S_1, S_2$)
  $\mathcal{H} \leftarrow \left\{ f \cup g \mid f \in \mathcal{F}, g \in \mathcal{G} \text{ such that } f\big|_W = g\big|_W \right\}$.
  return $\{h \in \mathcal{H} \mid h(U_1^* \cap S_1) = U_2^* \cap S_2\}$.

Theorem 5.11. Algorithm 4 works correctly as specified.

Proof. Evidently, Glue Isomorphisms is correct for all inputs. Additionally, Find Isomorphisms and Extend Isomorphism are clearly correct when $n = 1$. We proceed by strong induction on n by assuming these functions are correct for spherical codes of dimension less than $n > 1$.

Let $O : \operatorname{span} S_1 \to \operatorname{span} S_2$ be an isomorphism between two n-dimensional spherical codes S_1 and S_2 and let v_1 be an element in S_1 such that

$$\dim \operatorname{span} H^{\varepsilon}_{S_1}(v_1) = \max_{v \in S_1} \dim \operatorname{span} H^{\varepsilon}_{S_1}(v) =: d.$$

Let $v_2 := O(v_1)$. Trivially, we have $\dim \operatorname{span} H^{\varepsilon}_{S_2}(v_2) = d$, as $O(H^{\varepsilon}_{S_1}(v_1)) = H^{\varepsilon}_{O(S_1)}(O(v_1)) = H^{\varepsilon}_{S_2}(v_2)$. This equality also confirms that $O\big|_{\operatorname{span} H^{\varepsilon}_{S_1}(v_1)}$ is an isomorphism between the d-dimensional spherical codes $H^{\varepsilon}_{S_1}(v_1)$ and $H^{\varepsilon}_{S_2}(v_2)$, proving the correctness of Find Isomorphisms by our inductive hypothesis.

To show the correctness of Extend Isomorphism, we first note that the while loop at line 16 always terminates, as each iteration strictly increases the dimension of U_1^* by at least one, via the choice of $h^* \in S_1 \setminus U_1^*$. Next, we prove that at each iteration, Isom holds the set

$$\left\{ O^* \in \operatorname{Isom}(S_1 \cap U_1^*, S_2 \cap U_2^*) \mid O^*\big|_{U_1} = O \right\},$$

which precisely coincides with our desired output after the last iteration, when $U_1^* = \operatorname{span} S_1$ and $U_2^* = \operatorname{span} S_2$. At initialization, Isom indeed holds this set by construction. Now assume this is the case at the start of an iteration. Define S_1^* and S_2^* as in line 18 and let

$$O' \in \operatorname{Isom}(S_1 \cap (U_1^* + \operatorname{span} S_1^*), S_2 \cap (U_2^* + \operatorname{span} S_2^*))$$

such that $O'\big|_{U_1} = O$. By Proposition 5.10, $O'\big|_{\operatorname{span} S_1^*}$ is an isomorphism between S_1^* and S_2^*, discovered by Find Isomorphisms and thus included in $\operatorname{Isom}_{\text{new}}$ at line 21. Simultaneously, because O' extends the original input O and correctly maps the respective spherical codes, its restriction $O'\big|_{U_1^*}$ extends O and maps $S_1 \cap U_1^*$ to $S_2 \cap U_2^*$. By the inductive hypothesis, this restriction is already contained in the current set Isom. The set produced by "glueing" these partial isomorphisms in line 24 therefore contains O'. Conversely, every output element of Glue Isomorphisms is an isometry satisfying the same properties as O' by construction, confirming that Isom is updated correctly at each iteration.

This completes the inductive proof of correctness for both Find Isomorphisms and Extend Isomorphism on all dimensions $n \geq 1$. □

In the remainder of this section, we analyse the algorithm's time complexity.

Theorem 5.12. Algorithm 4 enumerates $\mathrm{Isom}(S_1, S_2)$ in time $n^{\mathcal{O}\left(n^{3/2}\right)}$.

Similar to Theorem 5.5, the runtime is asymptotically bounded by the size of the recursion tree. We let $T_d(n, \alpha)$ denote an upper bound of the recursion tree size of Find Isomorphisms in Algorithm 4 when inputted any two spherical (n, α)-codes where d is the maximal spherical cap dimension from line 3. The overall upper bound $T(n, \alpha)$ is thus defined as

$$T(n, \alpha) := \max_{1 \le d \le n} T_d(n, \alpha).$$

Lemma 5.13. The upper bound $T_d(n, \alpha)$ follows the recurrence relation

$$T_d(n, \alpha) \le n^2 \cdot A(n, \varepsilon) \cdot A(d, \alpha) \cdot T(d-1, \alpha^*) \cdot T(n-d, \alpha).$$

where

$$\alpha^* := \frac{\alpha - \alpha^2/\sqrt{n}}{1 - \alpha^2/\sqrt{n}}.$$

Proof. Note that the recursion of Find Isomorphisms occurs at line 21 when calling Extend Isomorphism in lines 9 and 12.

In the first call, the linear spans U_1 and U_2 are one-dimensional subspaces of the d-dimensional codes H_1 and H_2 defined in line 8. Consequently, the slice induced spherical codes S_1^* and S_2^* are at most $(d-1)$-dimensional by Proposition 5.9. Moreover, by $h^* \in H_1$ in line 17 implying that

$$\|h\| = \langle v_1, h \rangle = \langle v_1, h^* \rangle > \varepsilon = \alpha / n^{1/4},$$

we find that the maximum inner product of these codes is strictly bounded by

$$\frac{\alpha - \varepsilon^2}{1 - \varepsilon^2} = \frac{\alpha - \alpha^2/\sqrt{n}}{1 - \alpha^2/\sqrt{n}} = \alpha^*.$$

Lastly, the while loop in line 16 terminates after at most $\dim \operatorname{span} H_1 - \dim U_1 = d-1$ iterations, resulting in a worst-case tree size of

$$(d-1) \cdot T(d-1, \alpha^*)$$

in this first recursive branch.

Likewise, for the second call of Extend Isomorphism (line 12), the slice induced spherical codes S_1^* and S_2^* are at most $(n-d)$-dimensional, as U_1 and U_2 are d-dimensional while S_1 and S_2 are of dimension n. As mentioned earlier, the maximum inner product of these codes are tightly bounded by α, since in the extreme case we may have $h = \pi_{U_1}(h^*) = 0$ at line 17. This yields a worst-case tree size of

$$(n-d) \cdot T(n-d, \alpha)$$

in the second recursive branch.

Combining these two branching factors with the structural code bound $|S_2| \leq A(n, \varepsilon) \cdot A(d, \alpha)$ from Lemma 5.7 for the amount of candidates in line 7, we obtain the desired recurrence after bounding $(n - d) \cdot (d - 1)$ by n^2. □

To simplify the analysis, we continue with an auxiliary recurrence $t_d(n, \alpha)$ defined as

$$t_d(n, \alpha) := \log A(n, \varepsilon) + \log A(d, \alpha) + t(n - d, \alpha) + t(d - 1, \alpha^*) \tag{4}$$

where

$$t(n, \alpha) := \max_{1 \leq d \leq n} t_d(n, \alpha).$$

Note that the recurring factor n^2 from Lemma 5.13 accumulates to $n^{\mathcal{O}(n)}$, thereby establishing the relation

$$\log T(n, \alpha) \leq t(n, \alpha) + \mathcal{O}(n \log n).$$

Next, we demonstrate that the maximum branching factor always occurs at the extremes of our dimension split. We define two auxiliary recurrences representing these extreme paths:

- $t^{(1)}(n, \alpha)$ for the case where $d = 1$ every recursive step, and
- $t^{(n)}(n, \alpha)$ for the case where $d = n$ every recursive step.

Lemma 5.14. For all $n \geq 1$ and $\alpha \leq 1/2$, the recurrence $t(n, \alpha)$ is bounded by the maximum of its boundary cases. In other words,

$$t(n, \alpha) \leq \max\left(t^{(1)}(n, \alpha), t^{(n)}(n, \alpha)\right).$$

Proof. We proceed by strong induction on n. Let $B(n, \alpha)$ denote the maximum of the closed-form upper bounds for $t^{(1)}(n, \alpha)$ and $t^{(n)}(n, \alpha)$ which we will establish in Propositions 5.15 and 5.16. Assume that for all $k < n$ and $\gamma \leq \alpha$, we have $t(k, \gamma) \leq B(k, \gamma)$.

By substituting our inductive hypothesis into the definition of $t_d(n, \alpha)$, we must maximize the continuous relaxation of the right-hand side as a function of d on the interval $[1, n]$, given by

$$f(d) = \log A(n, \varepsilon) + \log A(d, \alpha) + B(n - d, \alpha) + B(d - 1, \alpha^*).$$

The term $\log A(n, \varepsilon)$ is constant with respect to d, thus trivially convex. The term $\log A(d, \alpha)$ is convex in d when substituting the bounds from Theorems 4.1 and 4.8. Carefully inspecting the proofs of Propositions 5.15 and 5.16 confirms that the

overarching bound $B(x, \gamma)$ is convex in x for all $x \geq 1$ and fixed $\gamma \leq 1/2$. This means that the remaining two terms of f are also convex. Therefore, f is convex on $[1, n]$ and its maximum must occur at one of the boundaries, i.e.

$$\max_{1 \leq d \leq n} f(d) = \max(f(1), f(n)).$$

As $f(1)$ exactly mirrors the step for $t^{(1)}$ and $f(n)$ mirrors $t^{(n)}$, both are bounded by $B(n, \alpha)$ by definition, completing the induction. □

We now evaluate the closed-form bounds for these two extreme recurrence sequences.

Proposition 5.15. The extreme recurrence path $t^{(n)}(n, \alpha)$, generated when $d = n$ at every recursive step, is bounded by $\mathcal{O}\left(n^{3/2}\right)$.

Proof. Note that, when substituting $d = n$ into the recurrence of $t_d(n, \alpha)$ in Equation (4), the terms $\log A(d, \alpha)$ and $t(d-1, \alpha^*)$ are asymptotically dominant. Omitting these terms and unrolling the recurrence yields the asymptotic bound

$$\sum_{i=1}^{n} \log A(d_i, \alpha_i),$$

where $d_1 = n$ and $\alpha_1 = 1/2$, and

$$d_i = d_{i-1} - 1 = n - i + 1, \quad \alpha_{i+1} = \frac{\alpha_i - \alpha_i^2/\sqrt{d_i}}{1 - \alpha_i^2/\sqrt{d_i}}$$

for all $i \in \{1, \ldots, n-1\}$. To obtain a closed-form bound on α_i, note that rearranging the recursive definition of α_{i+1} yields

$$\frac{1}{\alpha_{i+1}} - \frac{1}{\alpha_i} \geq \frac{1}{2\sqrt{d_i}}.$$

We can therefore telescope

$$\frac{1}{\alpha_i} - \frac{1}{\alpha_1} \geq \frac{1}{2} \sum_{j=1}^{i-1} 1/\sqrt{d_j} > \sqrt{n+1} - \sqrt{n-i+2}$$

using the fact that $1/\sqrt{x} > 2/(\sqrt{x+1} + \sqrt{x})$ for all $x > 0$. This gives our desired closed-form bound

$$\alpha_i \leq \frac{1}{2 + \sqrt{n+1} - \sqrt{n-i+2}}.$$

Following the proof of Theorem 5.5, we avoid decomposing the sum at some threshold index by exploiting the non-negativity of the logarithms and summing both bounds simultaneously over all indices. This yields the upper bound

$$\sum_{i=1}^{n} \log A(d_i, \alpha_i) \leq \sum_{i=1}^{n} \log \left(d_i^{\mathcal{O}(1)} \right) + \sum_{i=1}^{n} C\alpha_i^2 d_i \log \left(C/\alpha_i^2 \right).$$

The first sum trivially evaluates to $\mathcal{O}(n \log n)$. For the second sum, we substitute the closed-form bound $\alpha_i \leq 1/(A - \sqrt{d_i + 1})$, where $A = 2 + \sqrt{n+1}$. Because the dimension sequence d_i decreases from n to 1, we bound the sum by an integral over a continuous variable $x \in [1, n]$, similar to before. Applying the substitution $u = A - \sqrt{x+1}$, which yields $x = (A-u)^2 - 1$ and $\mathrm{d}x = -2(A-u)\,\mathrm{d}u$, the limits of integration invert from $[1, n]$ to $[A - \sqrt{2}, 2]$. Swapping the bounds to absorb the negative sign, and omitting the constant C and the first term of the integral approximation for asymptotic clarity, produces the asymptotic bound on the second sum, given by

$$\int_1^n \frac{x}{\left(A - \sqrt{x+1}\right)^2} \log \left(\left(A - \sqrt{x+1} \right)^2 \right) \mathrm{d}x \leq 2A^3 \int_2^A \frac{2 \log u}{u^2}\,\mathrm{d}u.$$

Since $A = 2 + \sqrt{n+1} = \mathcal{O}(\sqrt{n})$, the leading coefficient $2A^3$ scales tightly as $\mathcal{O}\left(n^{3/2}\right)$. The remaining integral evaluates to an absolute constant bounded independently of n, as

$$\int_2^A \frac{2 \log u}{u^2}\,\mathrm{d}u = \left[-\frac{2 \log u + 2}{u} \right]_2^A = \mathcal{O}(1).$$

Combining these evaluations yields

$$\sum_{i=1}^{n} \log A(d_i, \alpha_i) = \mathcal{O}(n \log n) + \mathcal{O}\left(n^{3/2}\right) = \mathcal{O}\left(n^{3/2}\right). \qquad \square$$

Proposition 5.16. The extreme recurrence path $t^{(1)}(n, \alpha)$, generated when $d = 1$ at every recursive step, is bounded by $\mathcal{O}\left(n^{3/2} \log n\right)$.

Proof. Substituting $d = 1$ at every step and unrolling the recurrence yields the sum

$$\sum_{k=1}^{n} \log A(k, \varepsilon_k), \quad \text{where} \quad \varepsilon_k = \frac{1}{2k^{1/4}},$$

in which we assume the worst-case inner product $\alpha = 1/2$. Following the analogous decomposition from Theorem 5.5 and Proposition 5.15, we apply the piecewise upper bound to obtain

$$\sum_{k=1}^{n} \log A(k, \varepsilon_k) \leq \sum_{k=1}^{n} \log \left(k^{\mathcal{O}(1)} \right) + \sum_{k=1}^{n} C\varepsilon_k^2 k \log \left(C/\varepsilon_k^2 \right).$$

The first sum evaluates to $\mathcal{O}(n \log n)$. For the second sum, substituting $\varepsilon_k = 1/(2k^{1/4})$ simplifies the summand to $\mathcal{O}\left(\sqrt{k} \log k\right)$. Therefore, the recurrence path $t^{(1)}(n, \alpha)$ is asymptotically bounded by

$$\mathcal{O}(n \log n) + \sum_{k=1}^{n} \mathcal{O}\left(\sqrt{k} \log k\right) = \mathcal{O}\left(n^{3/2} \log n\right). \qquad \square$$

Proof of Theorem 5.12. As established, the runtime of Algorithm 4 is bounded by the recursion tree size $T(n, 1/2)$, which satisfies $\log T(n, 1/2) \leq t(n, 1/2) + \mathcal{O}(n \log n)$. By Lemma 5.14, $t(n, 1/2)$ is bounded by the maximum of the extreme paths $t^{(1)}$ and $t^{(n)}$. Since Propositions 5.15 and 5.16 show that both paths are at most $\mathcal{O}\left(n^{3/2} \log n\right)$, it follows that $t(n, 1/2) \leq \mathcal{O}\left(n^{3/2} \log n\right)$. Therefore, the total runtime is bounded by

$$T(n, 1/2) \leq 2^{\mathcal{O}\left(n^{3/2} \log n\right)} = n^{\mathcal{O}\left(n^{3/2}\right)}. \qquad \square$$

References

[CE03] Henry Cohn and Noam Elkies. "New upper bounds on sphere packings I". In: *Annals of Mathematics* 157.2 (Mar. 2003), pp. 689–714. ISSN: 0003-486X. DOI: 10.4007/annals.2003.157.689. URL: http://dx.doi.org/10.4007/annals.2003.157.689.

[Coh+17] Henry Cohn et al. "The sphere packing problem in dimension 24". In: *Annals of Mathematics* 185.3 (May 2017). ISSN: 0003-486X. DOI: 10.4007/annals.2017.185.3.8. URL: http://dx.doi.org/10.4007/annals.2017.185.3.8.

[CS88] John Conway and N. Sloane. *Sphere Packings, Lattices and Groups*. Vol. 290. Jan. 1988. ISBN: 978-1-4757-2018-1. DOI: 10.1007/978-1-4757-2016-7.

[DGS77] P. Delsarte, J. M. Goethals, and J. J. Seidel. "Spherical codes and designs". In: *Geometriae Dedicata* 6.3 (Sept. 1977), pp. 363–388. ISSN: 1572-9168. DOI: 10.1007/BF03187604. URL: https://doi.org/10.1007/BF03187604.

[Fri97] Shmuel Friedland. "The Maximal Orders of Finite Subgroups in GLn(Q)". In: *Proceedings of the American Mathematical Society* 125.12 (1997), pp. 3519–3526. ISSN: 00029939, 10886826. URL: http://www.jstor.org/stable/2162249.

[HR13] Ishay Haviv and Oded Regev. *On the Lattice Isomorphism Problem*. 2013. arXiv: 1311.0366 [cs.DS]. URL: https://arxiv.org/abs/1311.0366.

[Kho04] S. Khot. "Hardness of approximating the shortest vector problem in lattices". In: *45th Annual IEEE Symposium on Foundations of Computer Science*. 2004, pp. 126–135. DOI: 10.1109/FOCS.2004.31.

[Mus06] Oleg R. Musin. *The kissing number in four dimensions*. 2006. arXiv: math/0309430 [math.MG]. URL: https://arxiv.org/abs/math/0309430.

[MV10] Daniele Micciancio and Panagiotis Voulgaris. "A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations". In: *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*. STOC '10. Cambridge, Massachusetts, USA: Association for Computing Machinery, 2010, pp. 351–358. ISBN: 9781450300506. DOI: 10.1145/1806689.1806739. URL: https://doi.org/10.1145/1806689.1806739.

[Sik+20] Mathieu Dutour Sikirić et al. *A Canonical Form for Positive Definite Matrices*. 2020. arXiv: 2004.14022 [math.NT]. URL: https://arxiv.org/abs/2004.14022.

[Tao13] Terence Tao. *A cheap version of the Kabatjanskii-Levenshtein bound for almost orthogonal vectors*. 2013. URL: https://terrytao.wordpress.com/2013/07/18/a-cheap-version-of-the-kabatjanskii-levenshtein-bound-for-almost-orthogonal-vectors/.

[Via17] Maryna Viazovska. "The sphere packing problem in dimension 8". In: *Annals of Mathematics* 185.3 (May 2017). ISSN: 0003-486X. DOI: 10.4007/annals.2017.185.3.7. URL: http://dx.doi.org/10.4007/annals.2017.185.3.7.