Electronic Colloquium on Computational Complexity, Report No. 70 (2014)

The Complexity of Geometric Graph Isomorphism

V. Arvind * Gaurav Rattan *

Abstract

We study the complexity of *Geometric Graph Isomorphism*, in l_2 and other l_p metrics: given two sets of n points $A, B \subset \mathbb{Q}^k$ in k-dimensional euclidean space the problem is to decide if there is a bijection $\pi : A \to B$ such that for all $x, y \in A$, $d(x, y) = d(\pi(x), \pi(y))$, where d is the distance given by the metric. Our results are the following:

- We describe algorithms for isomorphism and canonization of point sets with running time $k^{O(k)}\text{poly}(nM)$, where M upper bounds the binary encoding length of numbers in the input. This is faster than previous algorithms for the problem.
- From a complexity-theoretic perspective, we show that the problem is in $\mathsf{NP}[O(k^2 \log^2 k)] \cap \mathsf{coIP}[O(k^2 \log^2 k)]$, where $O(k^2 \log^2 k)$ respectively bounds the nondeterministic witness length in NP and message length in the 2-round IP protocol.
- We also briefly discuss the isomorphism problem for other l_p metrics. We describe a deterministic logspace algorithm for point sets in $\mathbb{Q}^2$.

1 Introduction

Given two finite n-point sets A and B in a metric space (X, d), we say A and B are *isomorphic* if there is a *distance-preserving* bijection between A and B. The *Geometric Graph Isomorphism* problem, denoted GGI, is to decide if A and B are isomorphic.

A well-studied version of this general problem, also the main focus for us, is the *euclidean* setting where the metric space $(\mathbb{R}^k, l_2)$ is the standard k-dimensional euclidean space equipped with the l_2 distance metric. When k is constant, there is an easy polynomial-time algorithm for the problem [1]. When $k = n$, the problem is known to be polynomial-time equivalent to the usual Graph Isomorphism problem [2] and hence can be solved in time $2^{O(\sqrt{n \lg n})}\text{poly}(s)$ [3], where s is the size of the input encoded in binary. The interesting case is when the dimension k is much smaller than n. A randomized algorithm running in time $O(n^{\frac{k-1}{2}} \cdot \log n)$ was given in [4], which was improved to $O(n^{\lceil \frac{k}{3} \rceil} \cdot \log n)$ in [5]. Both these results are in a random access model of computation which allows for arbitrary precision real arithmetic.

In this paper we consider as input points with integer or rational coordinates in $\mathbb{R}^k$, with the l_2 metric and the more general l_p metric. In this setting, for the l_2 metric, considering the dimension k as parameter, there is a fixed-parameter tractable deterministic algorithm running in time $(e^{k^4} nM)^{O(1)}$ [6], where M bounds the binary encoding of entries in any point of A or B. The algorithm in [6] is based on nontrivial concepts from cellular algebras.

As our first result we obtain a $k^{O(k)}poly(nM)$ time algorithm for deciding Geometric Graph Isomorphism in the euclidean case. Indeed, we actually give a $k^{O(k)}poly(nM)$ time algorithm that computes *canonical forms* for point sets. This algorithm is more intuitive and

*The Institute of Mathematical Sciences (IMSc), Chennai, India. Emails: {arvind,grattan}@imsc.res.in

ISSN 1433-8092

is based on integer lattices. Specifically, we apply the recent lattice isomorphism algorithm of Haviv and Regev [8] to derive our canonization algorithm for point sets in $\mathbb{R}^k$.

At this point we recall some definitions. Computing canonical forms for structures is a fundamental algorithmic problem. *Graph Canonization*, which is the problem computing canonical forms for graphs, is closely connected to Graph Isomorphism. For a graph class $\mathcal{K}$, a mapping $f : \mathcal{K} \to \mathcal{K}$ is a *canonizing function* if $f(X)$ is isomorphic to X for each graph X in $\mathcal{K}$, and for any other graph X' in the class, $f(X) = f(X')$ if and only if X and X' are isomorphic. We say that $f(X)$ is the *canonical form* assigned by f to the isomorphism class containing X. For example, $f(X)$ can be defined as the lex-first graph in $\mathcal{K}$ isomorphic to X. This canonizing function is known to be NP-hard to compute. Whether there is a polynomial-time computable *some* canonizing function for graphs is open. It is also open if graph canonization is polynomial-time equivalent to graph isomorphism. Often, graph classes with efficient isomorphism tests have canonization algorithms [3] of comparable complexity (usually more sophisticated and involving additional work).

Analogously, we can define canonical forms and the canonization problem for point sets A contained in a metric space (X, d). A canonizing function $f : A \mapsto f(A)$ for a finite $A \subset X$ outputs an isomorphic point set $f(A)$ such that $f(A) = f(B)$ iff A and B are isomorphic point sets.

Theorem 1. *Given a finite point set $A \subset \mathbb{Q}^k$ of size n there is a deterministic $k^{O(k)}poly(nM)$ time algorithm that computes a canonizing function $f(A)$ for the l_2 metric. As a consequence, the* GGI *problem for n element point sets in $(\mathbb{Q}^k, l_2)$ has a deterministic $k^{O(k)}poly(nM)$ time algorithm B. Here, M bounds the binary encoding of entries in the input k-tuples for points in $A \cup B$.*

Efficient interactive proofs for $\overline{\text{GGI}}$

It is well-known that Graph Non-Isomorphism ($\overline{\text{GI}}$) has two-round Interactive Proof systems [9]. In the euclidean case we obtain two-round IP protocols for Geometric Graph Nonisomorphism ($\overline{\text{GGI}}$) with bounds on message lengths as a function of the parameter k.

Theorem 2. *There is a two-round interactive proof system which decides $\overline{\text{GGI}}$. Moreover, the randomness used by the verifier and the number of bits exchanged in the protocol is bounded by $O(k^2 \log^2 k)$. Hence k-dimensional euclidean Geometric Isomorphism is in $\mathsf{NP}[O(k^2 \log^2 k)] \cap \mathsf{coIP}[O(k^2 \log^2 k)]$, where $\mathsf{NP}[O(k^2 \log k)]$ denotes NP with at most $O(k^2 \log k)$ nondeterministic bits.*

Other metrics

In Section 5, we examine GGI for other l_p metrics. For the 2-dimensional case $\mathbb{Q}^2$ we show that the problem is in deterministic logspace (and hence in polynomial time). This is by a reduction to the problem of isomorphism of colored graphs with color classes of size 2 (BCGI_2), which is known to be solvable in deterministic logspace [14]. For higher dimensions we do not have any nontrivial upper bounds better than general Graph Isomorphism.

Theorem 3. *Given subsets A and B of $\mathbb{Q}^2$ as input, for any l_p metric, there is a deterministic logspace bounded algorithm for checking if A and B are isomorphic in that metric.*

2 Preliminaries

We denote the set $\{1, \ldots, k\}$ by $[k]$. We denote the k-dimensional euclidean space by $\mathbb{R}^k$. Since we consider points in $\mathbb{R}^k$ with rational coordinates, we are effectively working with $\mathbb{Q}^k$. Let the projection of a vector v on a subspace S be denoted by v_S. The *inner product* of two vectors $u = (u_1, \ldots, u_k)$ and $v = (v_1, \ldots, v_k)$ is $\langle u, v \rangle = \sum_{i \in [k]} u_i v_i$. The *euclidean norm* of a vector u, denoted by $\|u\|$, is defined to be $\sqrt{\langle u, u \rangle}$. The *distance* $d(u, v)$ between two points u and v in $\mathbb{R}^k$ is $\|u - v\|$. Two vectors u, v are *orthogonal* if $\langle u, v \rangle = 0$. In general, $\mathbb{R}^k$ can be equipped with different norms. For any $p \geq 1$, the p-norm of a vector $x \in \mathbb{R}^k$ is $\|x\|_p = (\|x_1\|^p + \cdots + \|x_k\|^p)^{1/p}$. Also we define $\|x\|_\infty$ as $max\{|x_1|, \ldots, |x_k|\}$. The euclidean norm is the 2-norm.

Given a set S of vectors $\{u_1, \ldots, u_n\}$, we define the $n \times n$ Gram matrix of S as $G(S)_{i,j} = \langle u_i, u_j \rangle$. It is well-known that two sets S and T have the same Gram matrix iff there is an orthogonal matrix O such that $T = OS$. Moreover, a Gram matrix G is known to be Cholesky decomposable as LL^T for a unique lower triangular matrix L. The factorization can be computed efficiently.

Given two point sets A and B in $\mathbb{Q}^k$, a bijection $\pi : A \to B$ is a *geometric isomorphism* if for every $x, y \in A$, $d(x, y) = d(\pi(x), \pi(y))$. Given two vector spaces U and V, a bijection $\tau : U \to V$ is called an *isometry* if for every $x, y \in U$, $d(x, y) = d(\pi(x), \pi(y))$. Additionally, if τ is a linear transformation, we call it *linear isometry*. It is natural to ask whether an isomorphism between point sets can also be extended to an isometry between the vector spaces which are spanned by these sets. In Section 3, we will show that this it true for Euclidean distances in Lemmata 2 and 3.

We recall some definitions and results from the theory of lattices [12]. A lattice $\mathcal{L}_B$ is the set of all integer linear combinations of a finite *basis* set of vectors $B = \{b_1, \ldots, b_m\} \subset \mathbb{R}^k$. The numbers k is the *dimension* of the lattice. We assume that the entries of the set B are rational and are described by bit vectors. Let M be the upper bound on the bit-size of any entry of b_i. Then, we can always recover a linearly independent basis of $r \leq k$ vectors for the lattice in time polynomial in k, M and m (using the Hermite Normal Form construction [12]). The number r is called the *rank* of the lattice $\mathcal{L}_B$.

A fundamental quantity for a lattice $\mathcal{L}$ is the length $\lambda_1(\mathcal{L})$ of a shortest vector in it. Computing shortest vectors in lattices is a well-known NP-hard computational problem. There have been several algorithms for exactly computing shortest vectors and for approximating them in the literature. For our purposes, we state a relatively recent algorithmic result [7] for enumerating all the shortest vectors in a given lattice.

Theorem 4 ([7], Corollary 5.8)**.** *There is a deterministic algorithm that takes as input a basis of some lattice $\Lambda \subset \mathbb{R}^k$, and a target vector $\boldsymbol{t} \in \mathbb{R}^k$, and an integer $p \geq 2$, and in time $\tilde{O}((4p)^k) \cdot poly(M, n)$ it outputs all vectors in Λ within distance $p\lambda_1(\Lambda)$ from $\boldsymbol{t}$. (The $\tilde{O}(\cdot)$ notation suppresses polylogarithmic factors).*

We also recall the following well-known fact about the number of short vectors in a lattice (see [7]).

Lemma 1. *In a lattice $\mathcal{L}$ of rank k, the number of vectors of length at most $p\lambda_1(\mathcal{L})$ is bounded by $(2p + 1)^k$.*

Haviv and Regev, in their interesting paper [8], showed a very general isolation lemma which they applied to lattices to give a $k^{O(k)}poly(nM)$ time algorithm for checking if two rank-k lattices are isomorphic under orthogonal transformations. They introduced the following notion of a linearly independent chain in a set which we recall as we will apply it to obtain our canonization algorithm for point sets. For a finite set $A \subseteq \mathbb{R}^k$ and a vector $v \in \mathbb{R}^k$, we say that v *uniquely defines a linearly independent chain of length* n in A if there are n vectors $x_1, \ldots, x_n \in A$ such that for every $1 \leq j \leq n$, the minimum inner product of v with vectors in $A \backslash Span(x_1, \ldots, x_{j-1})$ is uniquely achieved by x_j.

Given a lattice $\mathcal{L}$, its dual lattice $\mathcal{L}^*$ is defined as the set of vectors in $Span(\mathcal{L})$ such that they have an integer inner product with every vector in $\mathcal{L}$. The following theorem shows that there exists a suitably short vector in the dual lattice which defines a unique linearly independent chain in the set of shortest vectors of the lattice.

Theorem 5 ([8],Theorem 4.2)**.** *Let $\mathcal{L}$ be a lattice of rank k. Let S be the set of shortest vectors in $\mathcal{L}$. Suppose dimension of $Span(S)$ is k. Then, there exists a vector $v \in \mathcal{L}^*$ that uniquely defines a linearly independent chain of length k in S and satisfies $\|v\| \leq 5k^{17/2} \cdot \lambda_1(\mathcal{L}^*)$.*

3 Geometric Isomorphism and Canonization in l_2-metric

We prove Theorem 1 in this section. We start with some observations about isomorphisms of point sets. We first note that any isomorphism between point sets can be naturally extended to a linear isometry between the vector spaces spanned by these sets. Since this well-known fact can be considered folklore (e.g. see [4]) we summarize in the statements below. For simplicity we assume that the input sets A and B contain the element $\bar{0}$.

Lemma 2. *Suppose π is a geometric isomorphism between A and B such that $\pi(\bar{0}) = \bar{0}$. Then there exists a linear isometry $\mu : Span(A) \to Span(B)$ such that μ agrees with π on the set A.*

The proof of the above lemma follows from the following observations about geometric isomorphisms.

Lemma 3. *Let π be an isomorphism from A to B such that $\pi(\bar{0}) = \bar{0}$. Let $u_i, u_j \in A$.*

(a) π preserves inner products, i.e. $\langle u_i, u_j \rangle = \langle \pi(u_i), \pi(u_j) \rangle$.

(b) π preserves linear combinations inside the set, i.e. for any linear combination $\alpha u_i + \beta u_j \in A$, $\pi(\alpha u_i + \beta u_j) = \alpha\pi(u_i) + \beta\pi(u_j)$. Similarly, for any linear combination $\alpha v_i + \beta v_j \in B$, $\pi^{-1}(\alpha v_i + \beta v_j) = \alpha\pi^{-1}(v_i) + \beta\pi^{-1}(v_j)$.

(c) $U \subseteq A$ is a basis for $Span(A)$ iff $\pi(U) \subseteq B$ is a basis for $Span(B)$.

Proof. (a) For any two vectors $u_i, u_j \in A$, we must have $\|u_i\| = \|\pi(u_i)\|$ (since $\pi(\bar{0}) = \bar{0}$). Since $\|u_i - u_j\| = \|\pi(u_i) - \pi(u_j)\|$, squaring both sides and simplifying gives $\langle u_i, u_j \rangle = \langle \pi(u_i), \pi(u_j) \rangle$.

(b) The vector $x = \pi(\alpha_1 u_1 + \alpha_2 u_2) - (\alpha_1 \pi(u_1) + \alpha_2 \pi(u_2))$ is in $Span(B)$. Using part (a), the inner product of x with every vector in B can be easily verified to be zero. Therefore, x must be zero. The other case is symmetric.

(c) Since U is a basis for $Span(A)$, it is linearly independent. Part (b) implies that $\pi(U)$ is also linearly independent (since the linear combination $\bar{0} \in B$). Since U generates A, Part (b) implies that $\pi(U)$ must generate $\pi(A) = B$ and therefore $Span(B)$. Therefore, $\pi(U)$ must be a basis for $Span(B)$. The other direction is symmetric. □

Proof of Lemma 2. Fix a basis $J \subseteq A$ of $Span(A)$. By Lemma 3 (c), the set $\pi(J) \subseteq B$ must be a basis for $Span(B)$. Define the bijective linear transformation $\mu : Span(A) \to Span(B)$ which maps the basis vectors J to the basis vectors $\pi(J)$ (as ordered sets). Since μ agrees with π on $J \subseteq A$, Lemma 3 (b) implies that it must agree with π on A and therefore, $\mu(A) = B$. It remains to show that μ is an isometry. Inner product of any two vectors in $Span(A)$ is a linear combination of the inner products among the vectors in J. Therefore, it suffices to show that μ preserves the inner products between vectors in J and their images in $\pi(J)$. Since μ agrees with π on J, Lemma 1(a) implies that μ is an isometry. □

We assumed for the above lemmata that $\bar{0} \in A, B$ and $\bar{0}$ is fixed by the isometry. We now argue that it suffices to search for such isometries. It suffices to observe that the distance of point u_i in set A from the centroid of the points in A is:

$$\|u_i - \frac{1}{n}\sum_{j=i}^{n} u_j\|^2 = \frac{1}{n^2} \cdot \|\sum_{j=1}^{n}(u_i - u_j)\|^2 = \frac{1}{n^2} \cdot \sum_{j=1}^{n}\sum_{k=1}^{n}\langle u_i - u_j, u_i - u_k\rangle$$
$$= \frac{1}{n^2} \cdot \sum_{j=1}^{n}\sum_{k=1}^{n}\frac{1}{2} \cdot (\|u_i - u_j\|^2 + \|u_i - u_k\|^2 - \|u_j - u_k\|^2).$$

Therefore, if sets A and B are isomorphic via a permutation π, the distance of any point u_i from the centroid of set A must be equal to the distance of $\pi(u_i)$ from the centroid of set B. Now, in both A and B we do the following: (a) add its centroid to the set, and (b) translate the sets such that the centroid is mapped to origin. Clearly, the sets A and B are isomorphic if and only if the modified sets $\tilde{A}$ and $\tilde{B}$, obtained above, are isomorphic via a permutation that maps $\bar{0} \in \tilde{A}$ to $\bar{0} \in \tilde{B}$. Hence, it suffices to solve the following polynomial time equivalent problem: Given two point sets A and B in $\mathbb{Q}^k$, check if there exists an isomorphism mapping A to B that fixes $\bar{0}$.

Before we describe the canonization algorithm, we outline our algorithm for deciding geometric isomorphism. Consider the *lattices* $\mathcal{L}_A$ and $\mathcal{L}_B$ generated by sets A and B. By Lemma 3, any linear isometry μ that maps A bijectively to B also bijectively maps $\mathcal{L}_A$ to $\mathcal{L}_B$. Therefore, the set of shortest vectors in the lattice $\mathcal{L}_A$ must be mapped to the corresponding set in $\mathcal{L}_B$. Moreover, by Lemma 2 this linear isometry μ also maps the subspace spanned by the shortest vectors of $\mathcal{L}_A$ to the subspace spanned by the shortest vectors of $\mathcal{L}_B$. We follow a natural geometric approach. We fix a maximal linearly independent collection of shortest vectors S in lattice $\mathcal{L}_A$. Compute all possible (injectively mapped) images of S into the set of shortest vectors of the lattice $\mathcal{L}_B$ and branch on these choices. Project the set A to the orthogonal complement of the subspace spanned by S and B to the orthogonal complement of the subspace spanned by $\pi(S)$. Recursively continue to compute a geometric isomorphism for these projected point sets that are in subspaces of strictly smaller dimension. If A and B are isomorphic then, for one path of choices for the image set of S we can recover an isomorphism.

We will now directly describe our $k^{O(k)}poly(nM)$ time algorithm for computing canonical forms. Let A be the input set. The algorithm initially computes the set of shortest vectors in the lattice $\mathcal{L}_A$ generated by the basis A. Assume that this set spans $Span(A)$, otherwise we will proceed by subspace projections similar to the strategy explained above.

Using Theorem 5 of [8], we identify short vectors in the dual lattice which yield a unique linearly independent chain in the set of shortest vectors. Effectively, we have a small number of special bases for $Span(A)$. For each such basis B we generate a description of the set A

as follows. We compute the Gram matrix $G(B)$ for B. Also for each vector u_i in A, we compute the k-tuple Γ_i of the coordinates of u_i in basis B. The description of A is the tuple $(G(B), \Gamma_1, \ldots, \Gamma_n)$.

The important observation is that if two point-sets are isomorphic, then the sets of descriptions computed for each point-set are equal. This holds because the underlying linear isometry is an isometric map between the lattices $\mathcal{L}_A$ and $\mathcal{L}_B$ which preserves inner products. Therefore, the isometry sends a linear independent chain (a basis for $Span(A)$) in $\mathcal{L}_A$ to a linear independent chain in $\mathcal{L}_B$ (a basis for $Span(B)$). The descriptions generated for these bases will be identical since (a) Gram matrices for isometric bases are equal and (b) since the sets (A and B) and the bases (corresponding to the chains) are isometric, the coordinates of the sets in terms of the bases remain the same.

This suggests that the lexicographically least description is a canonical representation for a point set, and can be used to generate a canonical form. Now, we formally describe the algorithm, prove its correctness and analyze its time complexity.

Input: A set of vectors $A \subset \mathbb{Q}^k$ s.t. $|A| = n$ and $\bar{0} \in A$.
Output: A canonical set of vectors C_A.

1. While $dim(Span(A)) \neq 0$
   (a) Compute the set S_A of shortest vectors in $\mathcal{L}_A$ using Theorem 4.
   (b) Define the lattice $\Lambda_1 = \mathcal{L}_A \cap Span(S_A)$.
   (c) Compute the set of vectors W in the dual lattice Λ_1^* which are of length at most $5k^{17/2} \cdot \lambda_1(\Lambda_1^*)$ using Theorem 4.
   (d) For each vector in W, check if it defines a linearly independent chain in S_A. If yes, compute the chain. Otherwise, discard w from W.
   (e) Update set A to its component orthogonal to $Span(S_A)$. I.e. replace every $u \in A$ by $u - proj(u, S_A)$.
2. Let $W_1, \ldots, W_l$ be the sets computed during the l iterations of Step 1(c)-(d). For every tuple $(w_1, \ldots, w_l) \in W_1 \times \cdots \times W_l$,
   (a) Define the basis $B = C_1 \cup \cdots \cup C_l$, where C_i is the unique chain corresponding to vector w_i.
   (b) Compute the Gram matrix $G(B)$ for the set B.
   (c) For each u_i in the input set A, let $\Gamma_i = (\gamma_1, \ldots, \gamma_k)$ be the linear combination of the vectors in B which generates u_i. We can compute this tuple by solving a system of linear equations.
   (d) Define the string σ for the tuple $(w_1, \ldots, w_l)$ to be $(G(B), (\Gamma_1, \ldots, \Gamma_n))$.
3. Let Σ be the set of all strings generated in the previous step. Search the lexicographically least string σ_0 in Σ.
4. Given the string $\sigma_0 = (G, (\Gamma_1, \ldots, \Gamma_n))$,
   (a) Let L be the unique lower triangular matrix such that $G = LL^T$.

(b) Let B_0 be the set of rows of L.

(c) Compute the set C_A of vectors $\{u_1, \ldots, u_n\}$ where u_i is the Γ_i-linear-combination of B_0.

5. Output C_A as the canonical form for the set A.

The following two lemmas show that the algorithm indeed computes a canonical form.

Lemma 4. *Set A is isomorphic to set C_A.*

Proof. The lexicographically least description string $\sigma_0 = (G, (\Gamma_1, \ldots, \Gamma_n))$ used to construct C_A is generated using a certain basis $B = \{b_1, \ldots, b_l\}$. By construction, a vector $u_i \in A$ is a Γ_i-linear-combination of B. Similarly, a vector $v_i \in C_A$ is Γ_i-linear-combination of the set $L' = \{l_1, \ldots, l_k\}$, the rows of the unique matrix L obtained in Step 4 (a). Since the sets L and B have the same Gram matrix, there exists an orthogonal matrix O such that $b_i = Ol_i$ for all $i \in [k]$. By linearity, vector $u_i = Ov_i$ for each $i \in [n]$. Therefore, the set A can be obtained from set C_A by an orthogonal transformation. This shows that the two sets are isomorphic. □

Lemma 5. *Two sets A and B are isomorphic iff sets C_A and C_B are equal.*

Proof. Suppose the point sets A and B are isomorphic via a permutation π. It will suffice to show that the sets of all strings generated for A and B, denoted by Σ_A and Σ_B, are equal. The reason is that the lexicographically least description string will be equal for both sets, and the output sets C_A and C_B depend only on the string used to generate them. It further suffices to show that $\Sigma_A \subseteq \Sigma_B$ since the other containment is symmetric. We continue with the proof. Lemma 2 implies that there exists an orthogonal map $O : Span(A) \to Span(B)$ which agrees with π on A. Let $(w_1, \ldots, w_l)$ be a tuple processed in Step 2 in the computation of the canonical form of A and B_1 be the basis discovered in Step 2(a). We claim that $(Ow_1, \ldots, Ow_l)$ will be processed in the computation of the canonical form of B. This is true for Ow_1 because (a) $\mathcal{L}_B = O\mathcal{L}_A$ and therefore, (b) for any $v \in \mathcal{L}_B$, $\langle Ow_1, v\rangle = \langle Ow_1, Ou\rangle \in \mathbb{Z}$ for some u in $\mathcal{L}_A$. Also $\|w_1\| = \|Ow_1\|$. Therefore, Ow_1 is a vector in the dual lattice $\mathcal{L}_B^*$ and is short enough to be discovered. Moreover, for any b in the chain generated by w_1, $\langle w_1, b_j\rangle = \langle Ow_1, Ob_j\rangle$ which implies that Ob is in the chain generated by Ow_1. Hence, by uniqueness, the chain for set B is exactly the chain for set A transformed by the map O. In the next iteration, the computations for sets A and B proceeds by projecting the bases A and B out of the subspaces spanned by shortest vectors. Since O maps $Span(S_A)$ to $Span(S_B)$ and preserves inner products, it must map the updated lattice $\mathcal{L}_A$ to the updated lattice $\mathcal{L}_B$. Inductively, we can argue that $(Ow_1, \ldots, Ow_l)$ will be discovered in the computation for set B. Moreover, the basis B_2 obtained for this tuple must be OB_1. Therefore, the Gram matrices will be equal. Since O agrees with the isomorphism π, the linear combinations generated will also be equal. I.e. if $u_i \in A$ is equal to Γ_i-linear-combination of B_1, then $Ou_i \in B$ must be the Γ_i-linear-combination of $B_2 = OB_1$ as well. Therefore, the signatures generated in these computations will be equal. Therefore, $\Sigma_A \subseteq \Sigma_B$.

Conversely, let $C_A = C_B$. Then, there exist bases B_1 and B_2 such that the strings generated using them for set A and set B respectively are equal. Since this implies that B_1

and B_2 have the same Gram matrix, there must be an orthogonal map O such that $B_1 = OB_2$. Since the strings are equal, the points in A and B are identical linear combinations of vectors in B_1 and B_2 respectively. This implies that the set $A = OB$, and therefore A and B are isomorphic. □

Now, we are ready to finish the proof of Theorem 1.

Proof of Theorem 1. It suffices to verify that (a) the algorithm computes a canonical form on all inputs correctly and (b) the running time of the algorithm is bounded by $k^{O(k)}poly(nM)$. Theorem 5 ensures that the sets $W_1, \ldots, W_l$ are non-empty since by its construction, the lattice Λ_1 defined in Step 1(b) has as many linearly independent shortest vectors as its rank. Therefore, the algorithm always outputs a point set. Lemmata 4 and 5 show that the output set is a canonical form.

Next, we bound the running time of the algorithm on an input set of size n. Step 1 runs for at most k steps since the projection step (e) ensures that the dimension of the subspace $Span(A)$ strictly decreases. Claim 2 shows that the bit-size of the entries in S can increase by a multiplicative factor $O(k \log k)$ in each iteration of Step 1. (The proof is deferred to the Appendix). Therefore, the bit-size of any entry can increase from M to at most $M' = k^{O(k)}M$ during execution. Let us bound the time spent in an iteration of Step 1. Computing shortest vectors in Step (a) requires $2^{O(k)}poly(nM')$ time by Theorem 4. Computing the set W in Step (c) requires time $k^{O(k)}poly(nM')$ (set $p = 5k^{17/2}$ in Theorem 4). Checking for a chain in Step (d) can be done by scanning all the shortest vectors which are at most $2^{O(k)}$ in number by Lemma 1. Therefore, we spend at most $k^{O(k)}poly(nM)$ time in Step 1. Next, we bound the time spent in Step 2. By Lemma 1, the size of any set W_i must be at most $(25k^{17/2} + 1)^k = k^{O(k)}$. The number of tuples examined are at most $|W_1| \cdot \cdots \cdot |W_l|$ which is at most $k_1^{O(k_1)} \cdot \cdots \cdot k_l^{O(k_l)} \leq k^{O(k)}$. The operations inside Step 2 are usual polynomial time operations. Therefore, we spend at most $k^{O(k)}poly(nM)$ time in Step 2. Steps 3-5 are usual polynomial time operations. Therefore the overall running time is bounded by $k^{O(k)}poly(nM)$.

Clearly, we have a $k^{O(k)}poly(nM)$ running time canonization procedure. A $k^{O(k)}poly(nM)$ time isomorphism algorithm follows directly from Lemma 5: we compute the canonical forms for both the input sets, and accept iff the canonical forms are equal. □

Finally, we discuss a consequence of obtaining faster algorithms for Geometric Graph Isomorphism. As observed in the introduction, it is known that Graph Isomorphism is reducible to GGI, where, in the reduced instance, the output point sets are contained in $\mathbb{R}^n$. We first show a similar observation even for hypergraph isomorphism. Note that there is a standard reduction that reduces hypergraph isomorphism for n-vertex and m-edge hypergraphs to bipartite graph isomorphism on $n + m$ vertices. We can combine this with the reduction from Graph Isomorphism to GGI to obtain a reduction from hypergraph isomorphism to GGI. However, the point sets thus obtained will be in $\mathbb{R}^{n+m}$ and m could be much larger than n. The aim is to ensure that in the reduced GGI instance we have point sets in $O(n)$ dimensions.

Theorem 6. *Hypergraph Isomorphism can be reduced to Geometric Graph Isomorphism in polynomial time. More precisely, given a pair of hypergraphs (X_1, X_2) on n vertices the reduction outputs a pair of point sets (A, B), where $A, B \subset \mathbb{Q}^{5n}$, such that X_1 and X_2 are isomorphic if and only if A and B are isomorphic.*

Proof. Given a n-vertex hypergraph G having m hyperedges, we can construct a set A of $(n + m + 1)$ points in $\mathbb{Q}^{5n}$. First, add $\bar{0}$ to A. Then, add n vectors $e_1, \ldots, e_n$ as follows.

Set the first n coordinates of e_i to zero except the i^{th} coordinate. The next $2n$ coordinates are set to zero. The last $2n$ coordinates are set to 1. Finally, for each hyperedge E in the hypergraph, add a vector u_E as follows. Set the first n coordinates of u_E to 1 (or 0) if the vertex $i \in E$ (or not). The remaining $4n$ coordinates are set to 1. Clearly, the construction takes $poly(n, m)$ time.

Given two hypergraphs G and H, we can construct the corresponding sets A and B. We claim that G and H are isomorphic iff A and B are isomorphic. Suppose G and H are isomorphic via a mapping π. Then, the mapping μ sends $\bar{0}$ to $\bar{0}$, e_i to $e_{\pi(i)}$ for $i \in [n]$, and u_E to $u_{\pi(E)}$ for each hyperedge E. Here, $\pi(E) = \{\pi(i) \mid i \in E\}$. The map μ can be easily verified to be a natural isomorphism between sets A and B. Conversely, suppose that A and B are isomorphic via a mapping $\mu : A \to B$. We claim that μ maps $\bar{0}$ to $\bar{0}$. Indeed, $\bar{0} \in A$ has at least one point which is at distance at least $4n$ (any point corresponding to an hyperedge). If $\bar{0} \in A$ does not map to $\bar{0} \in B$, then its image can be seen to have no points at a distance at least $4n$ which is a contradiction. Therefore, $\mu(\bar{0}) = \bar{0}$. This implies that the distance preserving map μ maps the points $e_1, \ldots, e_n$ in A to $e_1, \ldots, e_n$ in B from which we recover a natural permutation $\pi : [n] \to [n]$. It is easy to verify that π is an isomorphism between the hypergraphs. □

The current best algorithm for Hypergraph Isomorphism is group theoretic. It uses a group-theoretic algorithm for the Coset-Intersection problem [15] and has running time $2^{O(n)}$.poly(m) for n-vertex hypergraphs. However, the only canonization algorithm for hypergraphs is $n^{O(n)}$ time. As a consequence of Theorem 6, obtaining a $2^{O(k)}$.poly(nM) canonization algorithm for GGI would imply a $2^{O(n)}$ time canonization algorithm for hypergraphs, which is a long standing open problem.

4 Interactive Proofs for Geometric Non-Isomorphism

In this section we prove Theorem 2. We first recall that the complexity class $\mathsf{IP}[2]$ consists of languages L that are accepted by 2-round interactive proof systems [9]. This class coincides with AM (it can also be defined as BP.NP). An interactive proof system for L consists of a Prover-Verifier protocol (where the verifier V is polynomial-time bounded and prover P is unrestricted) such that for all inputs x: (a) if $x \in L$, there is a prover P such that the verifier accepts x with probability at least $3/4$, and (b) if $x \notin L$, for all provers P, the verifier accepts the instance with probability at most $1/4$. More details can be found in a standard textbook [10].

It is well-known that $\overline{\text{GI}}$ has an $\mathsf{IP}[2]$ protocol that uses $O(n \log n)$ random bits to achieve constant success probability (n is the number of vertices in the graphs). In this section we give a more efficient $\mathsf{IP}[2]$ protocol for k-dimensional Geometric Graph Non-Isomorphism, where the number of random bits and message sizes can reduced to $O(k^2 \log^2 k)$.

Informally, the interactive proof system for $\overline{\text{GGI}}$ works as follows. Given sets A and B in $\mathbb{Q}^k$, the verifier first checks if $nM > k^k$. If so, the verifier can use the algorithm of Section 3 to solve such an instance of $\overline{\text{GGI}}$ in time polynomial in the input representation. Otherwise, $nM \leq k^k$. Then, the Verifier randomly picks one of the sets. The verifier randomly generates a description of that set, sends it to the prover and asks the prover to identify the set used. The descriptions are generated with the following property. If the sets are isomorphic, the descriptions generated are identically distributed and therefore, no prover can identify the set with probability more than $1/2$. If the sets are not isomorphic, then the distributions

generated for A and B have disjoint support. Hence, the prover can identify the distribution from a sample point with probability 1. For a small dimension k, we show that we can save the number of random bits used and the length of messages exchanged in the protocol.

First, we describe the randomized algorithm that samples the descriptions. Given a set S, the algorithm samples a basis B of $Span(S)$ for this purpose.

Input: Set $S = \{v_1, \ldots, v_n\}$ of points in $\mathbb{Q}^k$ s.t. $\bar{0} \in S$.

1. Let $B = \emptyset$. Denote $|S|$ by n.
2. Perform the following steps exhaustively.
   (a) Using $\log n$ random bits pick a point u from S uniformly at random and include it in the basis B.
   (b) Remove every vector in S which is in $Span(B)$. (This can be done using Gaussian Elimination).
3. Using the ordered set $B = (v_1, \ldots, v_l)$, $l \leq k$ obtained above.
   (a) define a $l \times l$ matrix K such that $K_{i,j} = \langle v_i, v_j \rangle$.
   (b) define a set $\tilde{S}$ as follows. For each $v \in S \backslash B$, add the tuple of rational numbers $(\langle v, v_1 \rangle, \ldots, \langle v, v_l \rangle)$ to the set $\tilde{S}$.
4. Output the signature $\sigma = (K, \tilde{S})$.

Given a set S, the above algorithm generates a distribution σ_S on the pairs $(K, \tilde{S})$ as above. We say that distributions σ_1 and σ_2 have *disjoint support* if their ranges do not intersect. We have the following simple claim about the distribution σ_U and σ_V generated by the above procedure for two point sets U and V.

Claim 1. *If sets U and V are isomorphic, the distributions σ_U and σ_V are identical. If they are not isomorphic, the distributions have disjoint supports.*

Proof. Suppose U and V are isomorphic via a permutation π. We observe that the algorithm samples the vectors from the set S independent of the indices of the vectors. This implies that if the basis B is sampled for set U with a certain probability, then the basis $\pi(B)$ is sampled for set V with equal probability and vice-versa. The signatures constructed using B and $\pi(B)$, for U and V respectively, must be equal by Lemma 3.

Next, suppose the two distributions do not have disjoint supports. In other words, there exist ordered bases $B_1 \subseteq U$ and $B_2 \subseteq V$ such that the signature $(K_1, \tilde{S}_1)$ constructed using B_1 for U and the signature $(K_2, \tilde{S}_2)$ constructed using B_2 for V are equal. Consider the mapping μ which sends i^{th} vector of B_1 to i^{th} vector of B_2. Since $K_1 = K_2$, μ preserves inner products between sets B_1 and B_2. Therefore, μ is a isomorphism between B_1 and B_2. Next, no two vectors in U can contribute the same tuple to the set $\tilde{S}_1$ since their difference must be a vector which has zero inner product with every vector in B_1. For any vector $u \in U$ which contributes the tuple $s_u = (\langle u, u_1 \rangle, \ldots, \langle u, u_l \rangle)$ to $\tilde{S}_1$, its representation in the basis B_1 can be verified to be $(K_1)^{-1} s_u$. Therefore, a vector $v \in U$ which contributes the same tuple to $\tilde{S}_2$

must have the identical representation $K_2^{-1}s_u = K_1^{-1}s_u$ in the basis B_2. Therefore, we define a map μ' which maps a vector $u \in U$ to a vector $v \in V$ iff they contribute the same tuples. Such a mapping is well-defined since $\tilde{S}_1 = \tilde{S}_2$. Using bilinearity of inner products and the identical representation of vectors under mapping by μ', we can verify that μ' preserves inner products. Therefore, the sets U and V must be isomorphic. □

Now, we can present the $\mathsf{IP}[2]$ protocol for $\overline{\mathrm{GGI}}$ and complete the proof of Theorem 2. We note that the signature generated by the algorithm above can be represented as a string in $\{0,1\}^r$ where r is upper bounded as follows. Let M be the bit-size of largest entry in a vector in S. Each of the l^2 entries in the matrix K is an inner product of k-dimensional vectors and will have size at most $2M + \log k$. Similarly, each of the $(n-l)$ k-tuples in $\tilde{S}$ has size at most $k \cdot (2M + \log k)$. Therefore, $r \leq k^2(2M + \log k) + nk(2M + \log k)$.

Input: Sets U and V in $\mathbb{R}^k$.

1. If $nM > k^k$, the Verifier runs the algorithm of Section 3 on sets U and V and **accepts** iff the algorithm accepts.
2. Otherwise, $nM \leq k^k$. Then, the Verifier gets a random bit b. If b is 0 (or 1), he samples a signature $\sigma \in \{0,1\}^r$ from the distribution μ_U (or μ_V). Let the number represented by the binary string σ be $R \in [0, 2^r - 1]$.
3. Verifier also picks a random prime p in the range $[0, \ldots, T]$ where $T = r^{3k}$. He sends the string $(R \mod p, p)$ to the prover.
4. Prover examines the string and sends back a bit b'.
5. Verifier **accepts** if and only if $b' = b$.

Proof of Theorem 2. If $nM > k^k$, the protocol runs in time $k^{O(k)} \cdot poly(M, n)$ which is polynomial in the input size n, M. Therefore, in polynomial time, the Verifier works correctly using zero randomness and interaction. Otherwise, $nM \leq k^k$. In this case, let us first verify the correctness of the protocol. If the sets U and V are isomorphic, the two distributions coincide. Therefore, no prover can determine the bit b with probability better than $1/2$. If the sets U and V are not isomorphic, the two original distributions are disjoint by Claim 1. We need to upper bound the probability (over the randomly picked prime p) that the modified distributions $(R \mod p, p)$ obtained in Step 3 do not remain disjoint. Since the algorithm for generating a basis uses $l \log n$ random bits, clearly R is bounded by n^k (since $l \leq k$). Now, by chinese remaindering and a union bound argument it follows that the distributions obtained in Step 3 for U and V have disjoint support with probability more than 0.9. Therefore, we have a protocol with completeness error 0.1 and soundness error 0.5. A standard parallel repetition can be used to guarantee a protocol with desired error probability for $\mathsf{IP}[2]$.

The verifier uses at most $k \log n$ random bits for the sampling algorithm and $\log T = 3k \log r = O(k \log k \cdot \log(n))$ random bits for sampling a prime. Since $nM \leq k^k$, the number of random bits and message lengths can be easily upper bounded by $O(k^2 \log^2 k)$. □

5 Geometric Isomorphism in other l_p metrics

In this section we include some observations about the GGI problem for other l_p metrics. We describe a deterministic logspace algorithm for the two-dimensional case (Theorem 3). We will prove the theorem for the case $p = \infty$ and explain how the algorithm can be easily adapted for all $p \neq \infty$. The algorithm works as follows. Given two point sets A and B of size n, we fix three points in set A and branch on their possible images in B under an isomorphism. Using these points, we will construct two colored graphs G and H such that (a) each graph has color class size at most two and (b) the point sets A and B are isomorphic iff the graphs G and H are isomorphic via a color-preserving isomorphism. This computation can be performed by a deterministic logspace transducer. The isomorphism problem for color class size two graphs, denoted by BCGI_2 is known to have a deterministic logspace algorithm [14]. By composing the logspace computations, we obtain a deterministic logspace algorithm for our problem.

Input: Two sets A and B of size n in $\mathbb{Q}^2$ (the l_∞ case).

1. Check if sets A and B are collinear by iterating over all triples and checking whether the three points are collinear.
   - If no, we store the first triple of non-collinear points $\{a, b, c\}$ that we discover.
   - If yes,
     - Construct two colored graphs G and H as follows. The graph G is $(A, \emptyset)$. The color of a vertex u_i is defined to be the set $\{d_1, d_2\}$ of the distances of u_i from the two extreme points in the set A. Similarly define H for set B.
     - Return **accept** iff G and H are isomorphic. The isomorphism can be decided using the logspace algorithm of [14].
2. Otherwise, assume w.l.o.g that $a, b, c \in A$ (the other case is symmetric). Branch on all possible images of $\{a, b, c\}$ in B, denoted by $\{a', b', c'\}$.
3. First, we compute a coloring of sets A and B. For set A, we color a point u by the ordered triple $(d_{u,a}, d_{u,b}, d_{u,c})$ of its distances from a, b, c.
4. Second, we refine these colorings and ensure that each color class is of size two.
   - If some points form a color class of size more than two, they will lie on a line segment parallel to x-axis or y-axis (proof is given later). Each such color class has two extreme points.
   - For each vertex $u \in A, B$, check whether it lies in a color class of size more than two. If yes, update the color of u, say C, with the color $(C, \{d_1, d_2\})$ where d_1, d_2 are the distances of u from the extreme points in the color class.
5. Third, we construct weighted colored graphs G' and H' over vertex sets A and B respectively. The graphs G' and H' are complete graphs. Every edge $\{u, v\}$ in G' or H' is labeled with the weight d_{uv}, the distance between points u and v. The coloring of the vertices have been already computed in Step 4.

6. Finally, we can use standard gadgets and modify the weighted graphs G' and H' to obtain unweighted graphs G and H such that G is isomorphic to H iff G' is isomorphic to H'.

7. Test whether G is isomorphic to H using the algorithm of [14]. If the answer is yes **accept**, else move to the next branch in Step 2. If all branches are exhausted, return **reject**.

Proof of Theorem 3, case $p = \infty$**.** It is easy to verify that the algorithm works correctly for the case when the sets are collinear. Therefore, we concentrate on the general case. If the above algorithm accepts, clearly the sets are isomorphic. Conversely, suppose there is a isomorphism π from A and B. In Step 2, one of the branches for the image of $\{a, b, c\}$ will coincide with $(\pi(a), \pi(b), \pi(c))$. Furthermore, π must respect the color classes defined by the algorithm based on the distance triples in Step 3. It also respects the color refinements in Step 4 due to the following fact which can be easily verified by induction. A color class of collinear points must map to another class of collinear points in a manner that preserves the order of vertices (therefore, in at most two possible ways). Hence, π respects the colors assigned by the algorithm.

Next, we verify the bound on the color class sizes. The set of points $S_{r,x}$ at l_∞-distance r from a point x is easily seen to be a square in $\mathbb{R}^2$ centered at x. It has sides of length $2r$ parallel to the coordinate axes. Consider the squares $S_{\alpha,a}$ and $S_{\beta,b}$. Their intersection is one of the following: (a) empty, or (b) at most two points, or (c) a common edge, or (d) two common incident edges. If we consider the third square $S_{\gamma,c}$, its intersection with $S_{\alpha,a} \cap S_{\beta,b}$ is therefore one of these cases: (a) empty or (b) at most two points or (c) a common edge. The last case is ruled our since three squares with non-collinear centres cannot have more than two edges common. Therefore, every color class is bounded by two unless the points in the color class lie on a common edge of three squares. Such a class is refined in Step 4 to have size at most two. Therefore, π must be an isomorphism between the weighted graphs G' and H' since it preserves mutual distances. By construction, the graphs G and H must be isomorphic and therefore, the algorithm accepts in Step 7.

It remains to verify that space complexity of the algorithm is upper bounded by a logarithmic function of the input size. It can be verified that (a) Step 1 can be performed by a logspace machine (b) Step 2 requires $O(\log n)$ space to store the indices of points $\{a, b, c, a', b', c'\}$ (c) Steps 3-6 can be seen to be implemented by logspace transducers and (d) Step 7 can be done in deterministic logspace [14]. □

We now briefly explain how the above algorithm can be adapted to solve the 2-dimensional GGI problem for any l_p-metric.

Proof of Theorem 3, for all $p \in [1, \infty)$**.** The set $S_{r,x}$ is a l_p-metric circle of radius r centered at the point x. For $p = 1$, such circles are squares of side $2r$ centered at x which have been rotated by $\pi/4$. The intersection of such squares is similar to the l_∞ case above. Hence, the above algorithm adapts to this case. For the case $p \in (1, \infty)$, it is known that l_p balls are *strictly convex* sets I.e., for any two distinct points u, v on the boundary of such a set, any convex combination $\theta u + (1 - \theta)v$ for $0 < \theta < 1$ is in the interior of the set. For $\mathbb{R}^2$,

this implies that any two l_p circles can intersect in at most two points ([13], Theorem 1). Therefore, any color class can be of size two and therefore, a similar algorithm which reduces the problem to BCGI_2 works. □

6 Concluding Remarks

We give a $k^{O(k)}.\mathrm{poly}(n, M)$ time algorithm for Geometric Isomorphism in the l_2 metric. It is asymptotically faster than previous algorithms. A natural question is to improve the running time. It would be interesting to obtain a "geometric" algorithm of running time $2^{O(k)}.\mathrm{poly}(nM)$ because that would imply a *non-group theoretic* algorithm for Graph Isomorphism of running time $2^{O(n)}$ (the simplest known algorithm for it requires solving the coset intersection problem in $2^{O(n)}$ time, which is a group-theoretic algorithm).

One approach to solving GGI for a metric space (X, d) is to try and efficiently embed the given points sets A and B *isometrically* into another metric space (X', d') for which we already know an efficient GGI algorithm. For instance it is easy to observe from known results about embedding metric spaces that there is a reduction from l_1^k-GGI to $l_\infty^{2^k}$-GGI in time $2^k \cdot poly(k, n, M)$, where l_p^k denotes the l_p metric on $\mathbb{R}^k$. We do not know of a reduction that avoids the blow-up from k to 2^k in dimension.

References

[1] H. Alt, K. Mehlhorn, H. Wagener, E. Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete Computational Geometry,* 3:237-256, 1988.

[2] Christos H. Papadimitriou and Shmuel Safra. The complexity of low-distortion embeddings between point sets. In *SODA*, pages 112–118, 2005.

[3] László Babai and Eugene M. Luks. Canonical labeling of graphs. In *STOC*, pages 171–183, 1983.

[4] Tatsuya Akutsu. On determining the congruence of point sets in d dimensions. *Comput. Geom.*, 9(4):247–256, 1998.

[5] Peter Braß and Christian Knauer. Testing the congruence of d-dimensional point sets. In *Symposium on Computational Geometry*, pages 310–314, 2000.

[6] S.A. Evdokimov and I.N. Ponomarenko. On the geometric graph isomorphism problem. *Pure and Applied Algebra*, 117-118:253–276, 1997.

[7] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In *STOC*, pages 351–358, 2010.

[8] Ishay Haviv and Oded Regev. On the lattice isomorphism problem. In *Proceedings of the SODA 2014 Conference,* to appear.

[9] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? *Inf. Process. Lett.*, 25(2):127–132, 1987.

[10] Sanjeev Arora and Boaz Barak. *Computational Complexity, a modern approach.* Cambridge University Press, 2009.

[11] Merrick L. Furst, John E. Hopcroft, and Eugene M. Luks. Polynomial-time algorithms for permutation groups. In *FOCS*, pages 36–41, 1980.

[12] Alexander Schrijver. *Theory of integer and linear programming.* Wiley-Interscience series in discrete mathematics and optimization, 1998.

[13] A. G. Corbalan, Marisa Mazon, and Tomas Recio. About Voronoi Diagrams for Strictly Convex Distances. In *9th European Workshop on Computational Geometry*, 1993.

[14] Birgit Jenner, Johannes Köbler, Pierre McKenzie and Jacobo Torán. Completeness results for graph isomorphism. *J. Comput. Syst. Sci.*, 66(3), 2003.

[15] Eugene M. Luks, Hypergraph Isomorphism and Structural Equivalence of Boolean Functions, STOC, 1999.

Appendix

Bounding the bit-sizes

Claim 2. *The binary encoding length of the largest entry in the set $A \subset \mathbb{Q}^k$ can increase by a multiplicative factor of at most $O(k \log k)$ during an iteration of Step 1 of Algorithm 3.*

Proof. In an iteration of Step 1, we have two main operations. In Step (a), we compute the shortest vector set in $\mathcal{L}_A$. In Step (b), we project set A. We will bound the increase in bit complexity in each step. Initially, the entries in A can be rewritten with a common denominator such that each entry has at most $M_1 = knM$ bits for denominator (multiply all the denominators) and $M_1 = knM$ bits for numerator. This allows us to think of A as a integral matrix times $(1/r)$ for some integer r.

Computing shortest vectors in $\mathcal{L}_A$ yields vectors which can be described by M_1 bits in denominator, and at most M_1 bits for the numerator (use Minkowski's bound along with the determinant upper bound: for any $n \times n$ matrix A, $det(A) \leq A_{11} \cdot \cdots \cdot A_{nn}$).

Next, we bound the increase in bit size in Step (b). Let the projection of a vector $u \in A$ onto $Span(S_A)$ be $p = \sum \alpha_i u_i$. We can compute α_i's from the linear system $G\bar{\alpha} = u'$ where the matrix G has $G_{i,j} = \langle u_i, u_j \rangle$, the column vector $\bar{\alpha} = (\alpha_1, \ldots, \alpha_k)$ and the column vector $u' = (\langle u, u_1 \rangle \ldots, \langle u, u_k \rangle)$. The entries in G and u' can be easily seen to have the bit sizes bounded by $M_2 = O(\log k) \cdot M_1$. By Cramer's rule, any α_i must be a ratio of determinants of G_i and G, where G_i is the matrix obtained by replacing i^{th} column of G by u'. The bit-size of these determinants is bounded by $M_3 = kM_2$ (use determinant upper bound). Overall, we increase the bit sizes by at most a factor of $O(k \log k)$. □

ECCC ISSN 1433-8092

http://eccc.hpi-web.de