Faster FPT Algorithm for Graph Isomorphism
Parameterized by Eigenvalue Multiplicity

V. Arvind and Gaurav Rattan
The Institute of Mathematical Sciences
C.I.T. Campus
Chennai 600 113, India
{arvind,grattan}@imsc.res.in

Abstract
We give a \( O^*(k^{O(k)}) \) time isomorphism testing algorithm for graphs of
eigenvalue multiplicity bounded by \( k \) which improves on the previous best
running time bound of \( O^*(2^{O(k^2/\log k)}) \) \cite{EP97a}.

1 Introduction

Two simple undirected graphs \( X = (V, E) \) and \( X' = (V', E') \) are said to be
isomorphic if there is a bijection \( \varphi : V \to V' \) such that for all pairs \( \{u, v\} \in \binom{V}{2} \),
\( \{u, v\} \in E \) if and only if \( \{\varphi(u), \varphi(v)\} \in E' \). Given two graphs \( X \) and \( X' \) as
input the decision problem Graph Isomorphism asks whether \( X \) is isomorphic to
\( X' \). An outstanding open problem in the field of algorithms and complexity is
whether the Graph Isomorphism problem has a polynomial-time algorithm. The
asymptotically fastest known algorithm for Graph Isomorphism has worst-case
running time time \( 2^{O(\sqrt{n \log n})} \) on \( n \)-vertex graphs \cite{BL83}. On the other hand,
the problem is unlikely to be NP-complete as it is in \( \text{NP} \cap \text{coAM} \) \cite{BHZ87}.

However, efficient algorithms for Graph Isomorphism have been discovered
over the years for various interesting subclasses of graphs, like, for example,
bounded degree graphs \cite{Luks80}, bounded genus graphs \cite{Mil80, GM12}, bounded
eigenvalue multiplicity graphs \cite{BGM82, EP97a}.

The focus of the present paper is Graph Isomorphism for bounded eigenvalue
multiplicity graphs. This was first studied by Babai et al \cite{BGM82} who gave an
\( n^{O(k)} \) time algorithm for it. There is also an NC algorithm \cite{Bab86} for the problem for constant \( k \) due to Babai \cite{Bab86}. Using an approach based on cellular algebras
and some nontrivial group theory, Evdokimov and Ponomarenko \cite{EP97a} gave
an \( O^*(2^{O(k^2/\log k)}) \) algorithm for it. This puts the problem in FPT, which is the
class of fixed parameter tractable problems. The parameter in question here is
the bound \( k \) on the eigenvalue multiplicity of the input graphs.

\(^1\)Throughout the paper, we use the \( O^*(\cdot) \) notation to suppress multiplicative factors that
are polynomial in input size.

\(^2\)NC denotes the class of problems that can be solved in in the parallel-RAM model in
polylogarithmic time using polynomially many processors.
In this paper we obtain a $O^∗(k^{O(k)})$ time isomorphism algorithm for graphs of eigenvalue multiplicity bounded by $k$. We follow a relatively simple geometric approach to the problem using integer lattices. Recently, we obtained an $O^∗(k^{O(k)})$ time algorithm for Point Set Congruence (abbreviated GGI) in $\mathbb{Q}^k$ in the $\ell_2$ metric [AR14]. Our algorithm is based on a lattice isomorphism algorithm of running time $O^∗(k^{O(k)})$, due to Haviv and Regev [HR14]. They design an $O^∗(n^{O(n)})$ time algorithm for checking if two integer lattices in $\mathbb{R}^n$ are isomorphic under an orthogonal transformation. In [AR14] we adapt their technique to solve the Point Set Congruence problem, GGI, in $O^∗(k^{O(k)})$ time.

Now, in this paper, building on our previous algorithm for GGI [AR14], combined with some permutation group algorithms, we first give a $O^∗(k^{O(k)})$ time algorithm for a suitable geometric automorphism problem, defined in Section 4. It turns out that the bounded eigenvalue multiplicity Graph Isomorphism can be efficiently reduced to this geometric automorphism problem, which yields the $O^∗(k^{O(k)})$ time algorithm for it.

2 Preliminaries

Let $[n]$ denote the set $\{1,\ldots,n\}$. We assume basic familiarity with the notions of vector spaces, linear transformations and matrices. The projection of a vector $v \in \mathbb{R}^n$ on a subspace $S \subseteq \mathbb{R}^n$ is denoted as $\text{proj}_S(v)$. The inner product of vectors $u = (u_1,\ldots,u_n)$ and $v = (v_1,\ldots,v_n)$ is $\langle u,v \rangle = \sum_{i \in [n]} u_i v_i$. The euclidean norm, $\|u\|$, of a vector $u$, is $\sqrt{\langle u,u \rangle}$, and the distance between two points $u$ and $v$ in $\mathbb{R}^n$ is $\|u - v\|$. Vectors $u,v$ are orthogonal if $\langle u,v \rangle = 0$. Subspaces $U,V$ are orthogonal if for every $u \in U, v \in V$, $u,v$ are orthogonal. A set of subspaces $W_1,\ldots,W_r$ is said to be an orthogonal decomposition of $\mathbb{R}^n$ if each pair of subspaces are mutually orthogonal, and they span $\mathbb{R}^n$. A square matrix $M$ is orthogonal if $M^T M = I$. A linear transformation $T$ stabilizes a subspace $S$ if $T(S) \subseteq S$. Given a matrix $M$, we call $\lambda$ to be an eigenvalue of $M$ if there exists a vector $v$ such that $Mv = \lambda v$. We call $v$ to be an eigenvector of $M$ of eigenvalue $\lambda$. The set of all eigenvectors of $M$ of eigenvalue $\lambda$ is a subspace of $\mathbb{R}^n$. The following well-known fact about $n \times n$ symmetric matrices will be useful.

**Fact 1.** All eigenvalues of a symmetric matrix are real. Moreover, the eigenspaces form an orthogonal decomposition of $\mathbb{R}^n$.

We use Sym$(V)$ to denote group of all permutations on a finite set $V$. Given a graph $X = (V,E)$, a permutation $\pi \in \text{Sym}(V)$ is an automorphism of the graph $X$ if for all pairs $\{u,v\}$ of vertices, $\{u,v\} \in E$ iff $\{\pi(u),\pi(v)\} \in E$. In other words, $\pi$ preserves adjacency in $X$. The set of all automorphisms of a graph $X$, denoted by Aut$(X)$, is a subgroup of Sym$(V)$, which is denoted by Aut$(X) \leq \text{Sym}(V)$.

We can similarly talk of automorphisms of hypergraphs: Let $X = (V,E)$ be a hypergraph with vertex set $V$ and edge set $E \subseteq 2^V$. A permutation $\pi \in \text{Sym}(V)$
is an automorphism of the hypergraph \( X \) if for every subset \( e \subseteq V, e \in E \) if and only if \( \pi(e) = \{ \pi(v) \mid v \in e \} \).

Given an undirected graph \( X = (V, E) \), the set indexed by \([n]\), we define its adjacency matrix \( A_X \) is defined as follows: \( A_X(i, j) = 1 \) if \( \{v_i, v_j\} \in E \) and 0 otherwise. Clearly, the adjacency matrix \( A_X \) of an undirected graph is symmetric. Given a permutation \( \pi : [n] \to [n] \), we can associate a natural permutation matrix \( M_\pi \) with it. It is easy to verify that \( \pi \) is an automorphism of a graph \( G \) iff \( M_\pi^T A_X M_\pi = A_X \). Since permutation matrices are orthogonal matrices, the following simple folklore lemma characterizes the automorphisms of a graph through the action of the associated matrix on the eigenspaces of its adjacency matrix.

**Lemma 1.** Let \( X \) be the adjacency matrix of a graph \( G = (V, E) \). Then, a permutation \( \pi \in \text{Sym}(V) \) is an automorphism of \( G \) iff the associated linear map \( M_\pi \) preserves the eigenspaces of \( X \).

**Proof.** Suppose \( \pi \in \text{Aut}(G) \). Then \( M_\pi A_X = A_X M_\pi \) and therefore, for any eigenvector \( v \) in eigenspace \( W_i \) of eigenvalue \( \lambda_i \), \( A_X M_\pi v = M_\pi A_X v = \lambda_i M_\pi v \) which shows that \( M_\pi v \in W_i \). Conversely, suppose \( M_\pi \) preserves eigenspaces \( W_i \) of \( X \). Then, for any \( v \in W_i \), \( A_X M_\pi v = \lambda_i M_\pi x = M_\pi A_X v \). Since eigenvectors of the symmetric matrix \( A_X \) span \( \mathbb{R}^n \), this implies that \( A_X M_\pi = M_\pi A_X \). Therefore, \( \pi \) must be an automorphism of \( G \).

**Remark 1.** Our approach to solving Graph Isomorphism for bounded eigenvalue multiplicity is based on a variation of this lemma, as described in Proposition 2.

We first map the graph \( G \) into a point set \( \mathcal{P} \) in the \( n \)-dimensional space \( \mathbb{R}^n \). Then, we project \( \mathcal{P} \) into eigenspace \( W_i \) of \( G \), to obtain \( \mathcal{P}_i \) for each eigenspace \( W_i \). It turns out that \( \pi \) is an automorphism of \( G \) if and only if \( \pi \), in its induced action, is a congruence for the point set \( \mathcal{P}_i \) for each eigenspace \( W_i \). When the eigenspaces \( W_i \) are of dimension bounded by the parameter \( k \), it creates the setting for application of the \( O^*(k^{O(k)}) \)-time algorithm for GGI [AIR13].

Next, we recall some useful results about permutation group algorithms. Further details can be found in the excellent text of Seress [Ser].

A permutation group is a subgroup \( G \subseteq \text{Sym}(\Omega) \) of the group of all permutations on a finite domain \( \Omega \). A subset \( A \subseteq G \) of a permutation group \( G \) is a generating set for \( G \) if every element of \( G \) can be expressed as a product of elements of \( A \). Every permutation group \( G \leq \text{Sym}(\Omega) \) has a generating set of size \( \log |G| \leq n \log n \) where \( n = |\Omega| \). Thus, algorithmically, a compact input representation for permutation groups is by a generating set of size at most \( n \log n \). With this input representation, it turns out there several natural permutation group problems have efficient polynomial-time algorithms. A fundamental problem here is membership testing: Given a permutation \( \pi \in \text{Sym}(\Omega) \) and permutation group \( G \) by a generating set, there is a polynomial-time algorithm (the Schreier-Sims algorithm [Ser]) to check if in \( \pi \in G \). The pointwise stabilizer of a subset \( \Delta \in \Omega \) in a permutation group \( G \leq \text{Sym}(\Omega) \) is the subgroup

\[
G_{\{\Delta\}} = \{ \pi \in G \mid \forall \gamma \in \Gamma, \pi(\gamma) = \gamma \}.
\]
Given a permutation group \( G \leq \text{Sym}(\Omega) \) by a generating set, a generating set for \( G \) \( \{ \Delta \} \) in polynomial time using ideas from the Schreier-Sims algorithm \cite{Ser}. More generally, suppose \( G \leq \text{Sym}(\Omega) \) is given by a generating set and \( \sigma \in \text{Sym}(\Omega) \) is a permutation. The subset of permutations \( (G\sigma) \Delta \) \( \{ \pi \in G\sigma \mid \pi(\gamma) = \gamma \forall \gamma \in \Delta \} \) that pointwise fix \( \Delta \) is a right coset \( G \{ \pi^{-1}(\Delta) \} \) and a generating set for \( G \{ \pi^{-1}(\Delta) \} \) and such a coset representative \( \tau \) can be computed in polynomial time \cite{Ser}. We often use the following group-theoretic fact.

**Fact 2.** Let \( H_i \leq \text{Sym}(\Omega), 1 \leq i \leq t \) and \( \sigma_i \in \text{Sym}(\Omega), 1 \leq i \leq t \), where each \( H_i \) is given by a generating set \( A_i \). Suppose the union of the right cosets \( \bigcup_{i=1}^{t} H_i \sigma_i \) is a coset \( G\sigma \) for some subgroup \( G \leq \text{Sym}(\Omega) \). Then, we can choose the coset representative \( \sigma \) to be \( \sigma_1 \) and the set \( \bigcup_{i=1}^{t} A_i \cup \{ \sigma_i \sigma_1^{-1} \mid 2 \leq i \leq t \} \) is a generating set for \( G \).

### 3 Algorithm Overview

Before we give an overview of the main result of this paper, we recall the Point Set Congruence problem (also known as the geometric isomorphism problem) GGI \cite{AMW88,Ak98,BK00}.

Given two finite \( n \)-point sets \( A \) and \( B \) in \( \mathbb{Q}^k \), we say \( A \) and \( B \) are isomorphic if there is a distance-preserving bijection between \( A \) and \( B \), where the distance is in the \( l_2 \) metric. The Geometric Graph Isomorphism problem, denoted GGI, is to decide if \( A \) and \( B \) are isomorphic. This problem is also known as Point Set Congruence in the computational geometry literature \cite{Ak98,BK00,AMW88}. It is called “Geometric Graph Isomorphism” by Evdokimov and Ponomarenko in \cite{EP97b}, which we find more suitable as the problem is closely related to Graph Isomorphism. In \cite{AR14} we obtained a \( O^*(k^{O(k)}) \) time algorithm for this problem.

We now begin with a definition.

**Definition 1.** Let \( \mathcal{P} = \{ p_1, p_2, \ldots, p_m \} \subset \mathbb{Q}^n \) be a finite point set. A geometric automorphism of \( \mathcal{P} \) is a permutation \( \pi \) of the point set \( \mathcal{P} \) such that for each pair of points \( p_i, p_j \in \mathcal{P} \) we have

\[
\|p_i\| = \|\pi(p_i)\|, \quad \text{and} \quad \|p_i - p_j\| = \|\pi(p_i) - \pi(p_j)\|,
\]

where \( p_i \) denotes, by abuse of notation, also the position vector of the point \( p_i \).

Let \( \mathcal{P} = \{ p_1, p_2, \ldots, p_m \} \subset \mathbb{Q}^n \) be a finite point set such that their set of position vectors \( \{ p_i \} \) spans \( \mathbb{R}^n \). We refer to \( \mathcal{P} \) as a full-dimensional point set in \( \mathbb{R}^n \).

**Proposition 1.** Let \( \mathcal{P} = \{ p_1, p_2, \ldots, p_m \} \subset \mathbb{Q}^n \) be a full-dimensional point set. Then there is a unique orthogonal \( n \times n \) matrix \( A_\pi \) such that \( A_\pi(p_i) = \pi(p_i) \) for each \( p_i \in \mathcal{P} \).
Proof. As $\mathcal{P}$ is full dimensional, we can define a unique matrix $A_\pi$ by extending $\pi$ linearly to all of $\mathbb{R}^n$. $A_\pi$ can be shown to be orthogonal as follows. Any vector $x \in \mathbb{R}^n$, $x$ is a linear combination $\sum_{i=1}^n \sigma_i v_i$ where $v_i \in \mathcal{P}$. Then, $\|Ax\|^2 = \sum_{i,j} \sigma_i \sigma_j v_i A^T v_j$. It suffices to observe that $2v_i A^T A v_j = \|A(v_i - v_j)\|^2 - \|A v_i\|^2 - \|A v_j\|^2 = \|v_i - v_j\|^2 - \|v_i\|^2 - \|v_j\|^2 = 2v_i^T v_j$ for any vectors $v_i, v_j \in \mathcal{P}$.

The geometric automorphism problem is defined below:

**Problem 1 (Geom-AUT$_k$).**

**Input:** A point set $\{p_1, p_2, \ldots, p_m\} \subseteq \mathbb{Q}^n$ and an orthogonal decomposition of $\mathbb{R}^n = W_1 \oplus W_2 \oplus \cdots \oplus W_r$, where $\dim(W_i) \leq k$ and $W_i \perp W_j$ for all $i \neq j$.

**Parameter:** $k$.

**Output:** The subgroup $G \leq S_m$ consisting of all automorphisms $\pi$ of the input point set such that the orthogonal matrix $A_\pi$ stabilizes each subspace $W_i$.

The $O^*(k^{O(1)})$ time algorithm for EVGI$_k$ has the following three steps.

1. We give a polynomial-time reduction from EVGI$_k$ to Geom-AUT$_{2k}$.

2. We apply the $O^*(k^{O(1)})$ time algorithm for GG1 [AR14] to give a $O^*(k^{O(1)})$ time reduction from Geom-AUT$_{2k}$ to a special hypergraph automorphism problem Hyp-AUT.

3. We give a polynomial-time dynamic programming algorithm for Hyp-AUT by adapting the hypergraph isomorphism algorithm for bounded color classes in [ADKT10].

**Proposition 2.** There is a deterministic polynomial-time reduction from EVGI$_k$ with parameter $k$ to Geom-AUT$_{2k}$ with parameter $2k$.

**Proof.** Let $X = X_1 \cup X_2$ be the disjoint union of the input instance $(X_1, X_2)$ of EVGI$_k$. The adjacency matrix $A_X$ of $X$ is block diagonal and has the adjacency $A_{X_1}$ and $A_{X_2}$ as its two blocks along the diagonal. Thus, $A_X$ has the same set of eigenvalues as $A_{X_1}$ and $A_{X_2}$, and the multiplicity at most doubles.

Clearly, we can decide whether $X_1$ and $X_2$ are isomorphic by computing $\text{Aut}(X)$ and checking if there exists a $\pi \in \text{Aut}(X)$ such that $\pi(X_1) = X_2$ and vice-versa.

Furthermore, by Lemma 3 a permutation $\pi \in \text{Sym}(V(X))$ is an automorphism of $X$ if and only if $\pi$ (considered as a linear map on $\mathbb{R}^{2n}$) preserves each eigenspace of $X$. Let $\lambda_1, \lambda_2, \ldots, \lambda_r$ be the $r$ eigenvalues of $X$ and $W_1, W_2, \ldots, W_r$ be the corresponding eigenspaces.

Next, we compute the point set $\mathcal{P} = \{p_1, p_2, \ldots, p_{m+2n}\}$ corresponding to the graph $X = (V, E)$, where $|V| = 2n$ and $|E| = m$. The points $p_1, p_2, \ldots, p_{2n}$

---

3 We can assume w.l.o.g. that $A_{X_1}$ and $A_{X_2}$ have the same eigenvalues with the same multiplicity as we can check that in polynomial time.

4 By applying suitable numerical methods we can compute each $\lambda_i$ and basis for each $W_i$ to polynomially many bits of accuracy in polynomial time. This suffices for our algorithms.
are defined by the elementary \( n \)-dimensional vectors \( e_i \in \mathbb{R}^{2n}, 1 \leq i \leq 2n \). The points \( p_{2n+1}, \ldots, p_{2n+m} \) are defined by vectors corresponding to the edges in \( E \) as follows: For each edge \( e = \{i, j\} \in E \) the corresponding point has 1 in the \( i^{th} \) and \( j^{th} \) locations and zeros elsewhere.

We claim that \( \pi \in \mathrm{Aut}(X) \) iff \( \pi \) is a geometric automorphism of \( \mathcal{P} \). Let \( \pi \) be any permutation on the vertex set \( V(X) \). The action of the permutation \( \pi \) extends (uniquely) to the edge set, and hence to the point set \( \mathcal{P} \) as well. If \( \pi \in \mathrm{Aut}(X) \) then, clearly, \( \pi \) is a geometric automorphism for the point set \( \mathcal{P} \). Conversely, if \( \pi \) is geometric automorphism of the point set \( \mathcal{P} \) then it stabilizes the subset of points \( \{p_1, \ldots, p_{2n}\} \) encoding vertices and the subset \( \{p_{2n+1}, \ldots, p_{2n+m}\} \) encoding edges which means \( \pi \in \mathrm{Aut}(X) \). This completes the reduction and its correctness proof.

\[ \square \]

## 4 The Geometric Automorphism Problem

**GEOM-AUT\(_k\)**

In this section, we introduce some necessary definitions and state a useful characterization of a geometric isomorphism of a set of points. This will lead to our \( O^*(k^{O(1)}) \) time algorithm for GEOM-AUT\(_k\) which yields the main result for EVGI\(_k\) by Proposition 2.

Let \((\mathcal{P}, W_1, W_2, \ldots, W_r)\) be the instance of GEOM-AUT\(_k\). W.l.o.g. we can assume that \( \mathcal{P} \) is full dimensional. Otherwise, we can cut down the dimensional of the ambient space \( \mathbb{R}^n \) to the dimension of the point set \( \mathcal{P} \).

We can assume w.l.o.g. that each \( W_\ell \) is given by a basis \( u_{\ell 1}, u_{\ell 2}, \ldots, u_{\ell k_\ell} \) where \( k_\ell \leq k \) for all \( \ell \in [r] \).

Each point \( p_i \in \mathcal{P} \) has its projection \( \text{proj}_\ell(p_i) \) in the subspace \( W_\ell \) defining the projection \( \mathcal{P}_\ell = \text{proj}_\ell(\mathcal{P}) \) inside \( W_\ell \) of the point set \( \mathcal{P} \). For each \( p_i \in \mathcal{P} \) we can uniquely express it as

\[ p_i = \sum_{\ell=1}^r \text{proj}_\ell(p_i). \]

Thus we have the projections \( \mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_r \) of the input point set \( \mathcal{P} \) into the orthogonal subspaces \( W_1, W_2, \ldots, W_r \), respectively. These projections naturally define equivalence relations on the point set \( \mathcal{P} \) as follows.

**Definition 2.** Two points \( p_i, p_j \in \mathcal{P} \) are \((\ell)\)-equivalent if \( \text{proj}_\ell(p_i) = \text{proj}_\ell(p_j) \), and they are \([\ell]\)-equivalent if \( \text{proj}_t(p_i) = \text{proj}_t(p_j), 1 \leq t \leq \ell \).

Since \( \mathbb{R}^n = W_1 \oplus W_2 \oplus \cdots \oplus W_r \) we observe the following.

**Fact 3.** For any two \( p_i, p_j \in \mathcal{P} \) we have \( p_i = p_j \) iff \( p_i \) and \( p_j \) are \([r]\)-equivalent.

In other words, the common refinement of the \((\ell)\)-equivalence relations, \( 1 \leq \ell \leq r \), is the identity relation on \( \mathcal{P} \), and the equivalence classes of this refinement are the singleton sets. Given a permutation \( \pi \) on the point set \( \mathcal{P} \) we can ask whether it induces an automorphism on the projection \( \mathcal{P}_\ell \) in the following sense.

A subset \( \Delta \subset \mathcal{P} \) of points is an \((\ell)\)-equivalence class of \( \mathcal{P} \) if and only if for some point \( p \in \mathcal{P}_\ell \) we have \( \Delta = \text{proj}_\ell^{-1}(p) \). Thus, each point in the projected set...
\( \mathcal{P}_\ell \) represents an (\( \ell \))-equivalence class. We say that permutation \( \pi \in \text{Sym}(\mathcal{P}) \) respects \( \mathcal{P}_\ell \) iff for each (\( \ell \))-equivalence class \( \Delta \subset \mathcal{P} \) the subset \( \pi(\Delta) \) is an (\( \ell \))-equivalence class. Suppose \( \pi \in \text{Sym}(\mathcal{P}) \) is a permutation that respects \( \mathcal{P}_\ell \). Then \( \pi \) induces a permutation \( \pi_\ell \) on the point set \( \mathcal{P}_\ell \) as follows: for each \( p \in \mathcal{P}_\ell \) its image is

\[ \pi_\ell(p) = \text{proj}_\ell(\pi(\text{proj}_\ell^{-1}(p))). \]

**Definition 3.** A permutation \( \pi \in \text{Sym}(\mathcal{P}) \) is said to be an induced geometric automorphism on the projection \( \mathcal{P}_\ell \subset \mathcal{W}_\ell \) if \( \pi \) respects \( \mathcal{P}_\ell \) and \( \pi_\ell \) is a geometric automorphism of the point set \( \mathcal{P}_\ell \).

**Lemma 2.** Let \((\mathcal{P}, W_1, W_2, \ldots, W_r)\) be an instance of \textsc{Geom-AUT}_k and \( \mathcal{P} \) be full dimensional in \( \mathbb{R}^n \). Let \( \pi \) be a permutation on \( \mathcal{P} \). Then \( \pi \) is a geometric automorphism of \( \mathcal{P} \) such that \( A_\pi(W_\ell) = W_\ell \) for each \( \ell \in [r] \) if and only if \( \pi \) is an induced automorphism of each \( \mathcal{P}_\ell, 1 \leq \ell \leq r \).

**Proof.** For the forward direction, suppose \( \pi \) is a geometric automorphism of \( \mathcal{P} \) such that \( A_\pi(W_\ell) = W_\ell \) for each \( W_\ell \). We claim that \( \pi \) is an induced automorphism of \( \mathcal{P}_\ell \) for each \( \ell \).

For any point \( p_i \in \mathcal{P} \) we can write

\[ p_i = \text{proj}_\ell(p_i) + u, \]

where \( u \) is a vector in \( W_\ell^\perp \). Since \( A_\pi \) stabilizes each \( W_i \), it follows by linearity that

\[ \text{proj}_\ell(A_\pi(p_i)) = A_\pi(\text{proj}_\ell(p)). \]

Hence \( A_\pi(\mathcal{P}_\ell) = \mathcal{P}_\ell \) which implies \( \pi \) is an induced automorphism of \( \mathcal{P}_\ell \) for each \( \ell \).

Conversely, suppose a permutation \( \pi \) on \( \mathcal{P} \) is an induced automorphism of each \( \mathcal{P}_\ell, 1 \leq \ell \leq r \). Since each \( \mathcal{P}_\ell \) is a full-dimensional point set in \( \mathcal{W}_\ell \), it follows that \( A_\pi(W_\ell) = W_\ell \) for each \( \ell \). To see that \( \pi \) is a geometric automorphism of \( \mathcal{P} \), let \( p_i, p_j \in \mathcal{P} \). We can write \( p_i = \sum_{\ell=1}^{r} \text{proj}_\ell(p_i) \) and \( p_j = \sum_{\ell=1}^{r} \text{proj}_\ell(p_j) \). By linearity, we have \( A_\pi(p_i) = \sum_{\ell=1}^{r} A_\pi(\text{proj}_\ell(p_i)) \) and \( A_\pi(p_j) = \sum_{\ell=1}^{r} A_\pi(\text{proj}_\ell(p_j)) \).

Hence, by Pythagoras theorem we have

\[
\| A_\pi(p_i) - A_\pi(p_j) \|^2 = \sum_{\ell=1}^{r} \| A_\pi(\text{proj}_\ell(p_i)) - A_\pi(\text{proj}_\ell(p_j)) \|^2 \\
= \sum_{\ell=1}^{r} \| \text{proj}_\ell(p_i) - \text{proj}_\ell(p_j) \|^2 \\
= \| p_i - p_j \|^2,
\]

where the third line above follows because \( \pi \) is an induced automorphism of each \( \mathcal{P}_\ell \).

\( \square \)
5 The Hypergraph Automorphism Problem

By Lemma 2 it follows that Aut(\(\mathcal{P}\)) is the group of all \(\pi \in \text{Sym}(\mathcal{P})\) such that \(\pi\) is an induced automorphism of each \(\mathcal{P}_\ell, 1 \leq \ell \leq r\). In this section we describe the algorithm for computing a generating set for Aut(\(\mathcal{P}\)) in \(O^*(k^{O(k)})\) time.

The first step is to reduce \(\text{GEOM-AUT}_k\) in \(O^*(k^{O(k)})\) time to a hypergraph automorphism problem defined below:

**Problem 2 (HYP-AUT).**

**Input:** A hypergraph \(X = (V, E)\) and a partition of the vertex set into color classes \(V = V_1 \cup V_2 \cup \cdots \cup V_r\), and subgroups \(G_i \leq \text{Sym}(V_i), 1 \leq i \leq r\), where each \(G_i\) is given as an explicit list of permutations.

**Output:** A generating set for \(\text{Aut}(X) \cap G_1 \times G_2 \times \cdots \times G_r\).

We will give a polynomial-time algorithm for this problem based on a dynamic programming strategy as used in [ADKT10]. Before that we will show that \(\text{GEOM-AUT}_k\) is reducible to HYP-AUT in \(O^*(k^{O(k)})\) time. Combining the two we will obtain the \(O^*(k^{O(k)})\) time algorithm for \(\text{GEOM-AUT}_k\).

**Theorem 1.** There is a \(O^*(k^{O(k)})\) time reduction from the \(\text{GEOM-AUT}_k\) problem to HYP-AUT.

**Proof.** Let \((\mathcal{P}, W_1, W_2, \ldots, W_r)\) be an instance of \(\text{GEOM-AUT}_k\). In order to compute \(\text{Aut}(\mathcal{P})\) we first compute each \(\mathcal{P}_\ell, \ell \in [r]\). Then, since \(W_\ell\) is \(k\)-dimensional we can compute the geometric automorphisms \(\text{Aut}(\mathcal{P}_\ell)\) in \(O^*(k^{O(k)})\) time by applying the main result of [ARL4]. Indeed, \(\text{Aut}(\mathcal{P}_\ell)\) can be explicitly listed down in \(O^*(k^{O(k)})\) time, also implying that \(|\text{Aut}(\mathcal{P}_\ell)|\) is bounded by \(O^*(k^{O(k)})\). Now, we construct a hypergraph instance \(X = (V, E)\) of HYP-AUT as follows: The vertex set \(V\) is the disjoint union \(V = \mathcal{P}_1 \cup \ldots \cup \mathcal{P}_r\), and the explicitly listed groups \(G_\ell = \text{Aut}(\mathcal{P}_\ell), \ell \in [r]\). For each point \(p_1 \in \mathcal{P}\) we include a hyperedge \(e_p \in E\), where \(e_p = \{\text{proj}_1(p_1), \text{proj}_2(p_1), \ldots, \text{proj}_r(p_1)\}\). Since the edges of \(X\) encode points in \(\mathcal{P}\), the induced action of the automorphism group \(\text{Aut}(X) \cap G_1 \times G_2 \times \cdots \times G_r\) on the edges of \(X\) is in one-to-one correspondence with \(\text{Aut}(\mathcal{P})\) by Lemma 2. Hence, we can obtain a generating set for \(\text{Aut}(\mathcal{P})\). Clearly, the reduction runs in time \(O^*(k^{O(k)})\).

In the polynomial-time algorithm for HYP-AUT we will use as subroutine a polynomial-time algorithm for the following simple coset intersection problem.

**Problem 3 (Restricted Coset Intersection).**

**Input:** Let \(V = V_1 \cup V_2 \cup \cdots \cup V_r\) be a partition of the domain into color classes and \(G_i \leq \text{Sym}(V_i)\) be an explicitly listed subgroup of permutations on \(V_i, 1 \leq i \leq r\). Let \(H\) and \(H'\) be subgroups of the product group \(G_1 \times \cdots \times G_r\), where \(H\) and \(H'\) are given by generating sets as input. Let \(\pi, \pi' \in G_1 \times \cdots \times G_r\).

**Output:** The coset intersection \(H \pi \cap H' \pi'\) which, if nonempty, is given by a generating set for \(H \cap H'\) and a coset representative \(\pi'' \in H \pi \cap H' \pi'\).

**Lemma 3.** The above restricted coset intersection problem has a polynomial-time algorithm.
Proof. We give a sketch of the algorithm which is a simple application of the classical Schreier-Sims algorithm (mentioned in Section 2): given a permutation group \( G \leq \text{Sym}(\Omega) \) by a generating set and another permutation \( \pi \in \text{Sym}(\Omega) \), for any point \( \alpha \in \Omega \) the cosubset of \( G\pi \) that fixes the point \( \alpha \) can be computed in time polynomial in \(|\Omega|\) and the size of the generating set for \( G \). See, e.g. [Ser] for details.

In order to compute the intersection \( H\pi \cap H'\pi' \), we consider the product group \( H \times H' \) acting on the set \( \Delta = \bigcup_{i=1}^r V_i \times V_i \) component-wise. The permutation pair \((\pi, \pi')\) too defines a permutation on the set \( \Delta \). We consider now the coset \((H \times H')(\pi, \pi')\) of the group \( H \times H' \). Define the diagonal sets

\[
D_i = \{ (\alpha, \alpha) \mid \alpha \in V_i \}, 1 \leq i \leq r.
\]

The following claim is immediate from the definitions.

**Claim 1.** A pair \((h, h') \in (H \times H')(\pi, \pi')\) maps each \(D_i\) to \(D_i\) if and only if \(h = h'\) and \(h \in H\pi \cap H'\pi'\).

Thus, in order to compute the coset intersection it suffices to compute the subcoset

\[
\{(h, h') \in (H \times H')(\pi, \pi') \mid (h, h')(D_i) = (D_i)1 \leq i \leq r\}
\]

of the coset \((H \times H')(\pi, \pi')\). Notice that \(D_i \subset V_i \times V_i\) and the elements of the coset \((H \times H')(\pi, \pi')\) restricted to \(V_i \times V_i\) are from the group \(G_i \times G_i\) which is polynomially bounded in input size. Let \(\Omega\) denote the entire orbit of \(D_i\) under the action of the group \(G_i \times G_i\). Clearly, \(|\Omega| \leq |G_i|^2\) and therefore is polynomially bounded in input size and can be computed. Now, \(D_i\) is just a point in the set \(\Omega\) and we can compute its pointwise stabilizer subcoset in \((H \times H')(\pi, \pi')\) by the Schreier-Sims algorithm (as outlined above) in time polynomial in \(|\Omega|\) and the generating sets sizes of \(H\) and \(H'\). Repeating this procedure for each \(D_i, 1 \leq i \leq r\) yields the subcoset that maps \(D_i\) to \(D_i\) for each \(i\). This completes the proof sketch. \(\square\)

We now describe the polynomial-time algorithm for Hyp-AUT.

**Theorem 2.** There is a polynomial-time algorithm for Hyp-AUT.

**Proof.** The algorithm is a dynamic programming strategy exactly as in [ADKT10]. But, unlike the problem considered in [ADKT10], we do not have bounded-size color classes in our hypergraph instances. Instead, we have color classes \(V_i\) and explicitly listed subgroups \(G_i \leq \text{Sym}(V_i)\) on each color class and we have to compute color-class preserving automorphisms \(\pi \in \text{Aut}(X)\) that, when restricted to each color class \(V_i\) belong to the corresponding \(G_i\). We now describe the algorithm.

The subproblems of this dynamic programming algorithm involve hypergraphs \((V, E)\) with multiple hyperedges (i.e., \(E\) is a multi-set). Thus, we may assume that the input \(X\) too is a multi-hypergraph given with the vertex set partition \(V = \biguplus_{\ell=1}^r V_\ell\), and groups \(G_\ell \leq \text{Sym}(V_\ell)\) explicitly listed as permutations. A bijection \(\varphi : V \rightarrow V\) is an automorphism of interest if \(\varphi\) maps each \(V_\ell\) to \(V_\ell\) such that:
The permutation \( \varphi \) restricted to \( V_\ell \) is an element of the group \( G_\ell \).

The map induced by \( \varphi \) on \( E \) preserves the hyperedges with their multiplicities (for each hyperedge \( e \subseteq V \), \( e \) and \( \varphi(e) \) have the same multiplicity in \( E \)).

We first introduce some notation. For \( \ell \in [r] \) and any multi-set \( D \) of hyperedges \( e \subseteq V \), let \( D[\ell] \) denote the multi-hypergraph \( (V[\ell], \{ e \cap V[\ell] \mid e \in D \}) \) on vertex set \( V[\ell] = V_1 \uplus \cdots \uplus V_\ell \). Further, let \( D_\ell \) denote the multi-hypergraph \( (V_\ell, \{ e \cap V_\ell \mid e \in D \}) \) on vertex set \( V_\ell \). For two multi-hypergraphs \( D[\ell] \) and \( D'[\ell] \) let \( \text{ISO}(D[\ell], D'[\ell]) \) denote the coset of all isomorphisms between them that belong to \( G_1 \times \cdots \times G_\ell \).

For \( \ell \in [r] \) we define an equivalence relation \( \equiv_\ell \) on the hyperedges in \( E \): for hyperedges \( e_1, e_2 \in E \) we say \( e_1 \equiv_\ell e_2 \) if

\[
e_1 \cap V_j = e_2 \cap V_j \text{ for } j = \ell + 1, \ldots, r.
\]

The equivalence classes of \( \equiv_\ell \) are called \((\ell)\)-blocks. For \( \ell \leq j \), notice that \( \equiv_\ell \) is a refinement of \( \equiv_j \). Thus, if \( e_1 \) and \( e_2 \) are in the same \((\ell)\)-block then they are in the same \((j)\)-block for all \( j \geq \ell \).

The algorithm works in stages \( \ell = 0, \ldots, r \). In stage \( \ell \), the algorithm considers the multi-hypergraphs \( A_{\ell+1} \) induced by the different \((\ell)\)-blocks \( A \) on the vertex set \( V_{\ell+1} \). For each pair of \((\ell)\)-blocks \( A, B \) the algorithm computes the cosets \( \text{ISO}(A[\ell], B[\ell]) \) (unless \( \ell = 0 \)) using the cosets of the form \( \text{ISO}(A_{\ell-1}[\ell], B_{\ell-1}[\ell]) \) computed already. Finally, for the single \((r)\)-block \( E \) the algorithm computes the coset \( \text{ISO}(E[\ell], E[\ell]) \) which is the desired group \( \text{Aut}(X) \cap G_1 \times \cdots \times G_r \).

**Stage 0:** Let \( A \) and \( B \) be \((0)\)-blocks. Then \( A \) contains a single hyperedge \( a \) with multiplicity \( |A| \), and \( B \) contains \( b \) with multiplicity \( |B| \). The coset \( \text{ISO}(A[1], B[1]) = 0 \) if \( |A| \neq |B| \) or \( |a \cap V_1| \neq |b \cap V_1| \). Otherwise, \( \text{ISO}(A[1], B[1]) \cap G_1 \) is a subcoset of all elements of \( G_1 \) that maps \( a \cap V_1 \) to \( b \cap V_1 \), which can be computed by inspecting the list of elements in \( G_1 \).

**For \( \ell := 1 \) to \( r - 1 \) do**

**Stages \( \ell \):** For each pair \((A, B)\) of \((\ell)\)-blocks compute the table entry \( T(\ell, A, B) = \text{ISO}(A[\ell], B[\ell]) \) as follows:

1. Partition the \((\ell)\)-blocks \( A \) and \( B \) into \((\ell - 1)\)-blocks \( A^1, \ldots, A^t \) and \( B^1, \ldots, B'^t \), respectively. If \( t \neq t' \) then \( \text{ISO}(A[\ell], B[\ell]) \) is empty.
2. Otherwise, \( t = t' \). Clearly, for all \( e \in A^i, e \cap V_\ell \) is identical. Let \( a_i = e \cap V_\ell, e \in A^i \) and \( b_i' = e \cap V_\ell, e \in B'^i \), for \( 1 \leq i, i' \leq t \). Let \( S_\ell \subset G_\ell \) be the subcoset of all permutations \( \tau \in G_\ell \) such that \( \tau \) (injectively) maps the set \( \{ a_1, a_2, \ldots, a_t \} \) to the set \( \{ b_1, b_2, \ldots, b_t \} \). For each \( \tau \in S_\ell \), we denote by \( \hat{\tau} \) this induced mapping that injectively maps the set \( \{ a_i \mid 1 \leq i \leq t \} \) to \( \{ b_{\hat{\tau}(i)} \mid 1 \leq i \leq t \} \).

We can compute \( S_\ell \) in polynomial time since \( G_\ell \) is given as an explicit list as part of the input.
3. For $\tau \in S_\ell$, recall that $A^i_{[\ell-1]}$ and $B^{\hat{\tau}(j)}_{[\ell-1]}$ denote the multi-hypergraphs obtained from the $(\ell - 1)$-blocks $A^j$ and $B^{\hat{\tau}(j)}$, where $j \mapsto \hat{\tau}(j)$ for $\tau \in S_\ell$ means that $\tau$ maps $a_j$ to $b_{\tau(j)}$. Then it is clear that we have

$$\text{ISO}(A_{[\ell]}, B_{[\ell]}) = \bigcup_{\tau \in S_\ell} \bigcap_{j=1}^{t} \text{ISO}(A^i_{[\ell-1]}, B^{\hat{\tau}(j)}_{[\ell-1]}) \times \{\tau\}$$

where we have already computed the coset $\text{ISO}(A^i_{[\ell-1]}, B^{\pi(j)}_{[\ell-1]})$.

4. In order to compute the coset $\text{ISO}(A_{[\ell]}, B_{[\ell]})$ from Equation 1, we cycle through the polynomially many $\tau \in S_\ell$, and compute each coset intersection $\bigcap_{j=1}^{t} \text{ISO}(A^j_{[\ell-1]}, B^{\hat{\tau}(j)}_{[\ell-1]})$ by repeated application of the restricted coset intersection algorithm of Lemma 3. We can write a generating set for the union of the cosets over all $\tau$ using Fact 2.

**Output:** In the last step, the unique $(r)$-block is the entire set of hyperedges $E$, and the table entry $T(r, E_{[r]}, E_{[r]}) = \text{ISO}(E_{[r]}, E_{[r]})$.

It is clear from the description that the running time is polynomially bounded in $|E|, |V|$ and $\max_{1 \leq \ell \leq r} |G_\ell|$.

**Acknowledgement.** We thank Saket Saurabh for suggesting the problem of obtaining a faster isomorphism algorithm for graphs of bounded eigenvalue multiplicity using the $O^*(k^{O(k)})$ time algorithm for GGI.

**References**

[AMW+88] H. Alt, K. Mehlhorn, H. Wagener, E. Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete Computational Geometry*, 3:237-256, 1988.

[Ak98] Tatsuya Akutsu. On determining the congruence of point sets in d dimensions. *Computational Geometry*, 9(4):247–256, 1998.

[BK00] Peter Brass and Christian Knauer. Testing the congruence of d-dimensional point sets. In *Symposium on Computational Geometry*, pages 310–314, 2000.

[BL83] László Babai and Eugene M. Luks. Canonical labeling of graphs. In *Proceedings of the ACM STOC Conference*, pages 171–183, 1983.

[BHZ87] Ravi B. Boppana, Johan Håstad and Stathis Zachos. Does co-NP Have Short Interactive Proofs? *Inf. Process. Lett.*, 25:2, 127-132, 1987.

[Luks80] Eugene M. Luks. Isomorphism of Graphs of Bounded Valence Can Be Tested in Polynomial Time. In *Proceedings of the IEEE FOCS Conference*, pages 42-49, 1980.
[Mil80] Gary L. Miller. Isomorphism Testing for Graphs of Bounded Genus. In *Proceedings of the ACM STOC Conference*, pages 225-235, 1980.

[GM12] Martin Grohe and Dániel Marx. Structure theorem and isomorphism test for graphs with excluded topological subgraphs. *44th ACM Symp. on Theory of Computing*, pp. 173-192, 2012.

[EP97a] S.A. Evdokimov and I.N. Ponomarenko. Isomorphism of Coloured Graphs with Slowly Increasing Multiplicity of Jordan Blocks. *Combinatorica* 19(3): 321-333 (1999).

[EP97b] S.A. Evdokimov and I.N. Ponomarenko. On the geometric graph isomorphism problem. *Pure and Applied Algebra*, 117-118:253–276, 1997.

[BGM82] László Babai, D. Yu. Grigoryev and David M. Mount. Isomorphism of Graphs with Bounded Eigenvalue Multiplicity. In *Proceedings of the ACM STOC Conference*, pages 310-324, 1982.

[Bab86] László Babai. A Las Vegas-NC Algorithm for isomorphism of graphs with bounded multiplicity of eigenvalues. In *Proceedings of IEEE FOCS Conference*, pages 303-312, 1986.

[HR14] Ishay Haviv and Oded Regev. On the lattice isomorphism problem. In *Proceedings of the 25th Annual ACM-SIAM Conference*, pages 391-404, SODA 2014.

[AR14] V. Arvind and Gaurav Rattan. The parameterized complexity of geometric graph isomorphism. In *Proceedings of IPEC Conference, 2014, to appear*.

[ADKT10] Vikraman Arvind, Bireswar Das, Johannes Köbler and Seinosuke Toda. Colored Hypergraph Isomorphism is Fixed Parameter Tractable. In *Proceedings of FSTTCS Conference*, pages 327-337, 2010.

[Ser] Á. Seress. Permutation Group Algorithms. Cambridge University Press, 2003.