Some Geometrical and Vertex-Partitioning Techniques for Graph Isomorphism

By

Gaurav Rattan

MATH10201005005

The Institute of Mathematical Sciences, Chennai

A thesis submitted to the

Board of Studies in Physical Sciences

In partial fulfillment of requirements

For the Degree of

DOCTOR OF PHILOSOPHY

of

HOMI BHABHA NATIONAL INSTITUTE

July, 2016

Homi Bhabha National Institute

Recommendations of the Viva Voce Board

As members of the Viva Voce Board, we certify that we have read the dissertation prepared by Gaurav Rattan entitled "Some Geometrical and Vertex-Partitioning Techniques for Graph Isomorphism" and recommend that it maybe accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

_____ Date: _____

Chair - Meena Mahajan

_____ Date: _____

Guide/Convener - V. Arvind

_____ Date: _____

Member 1 - Saket Saurabh

_____ Date: _____

Member 2 - Venkatesh Raman

_____ Date: _____

Examiner - Shashank K. Mehta

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to HBNI.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it may be accepted as fulfilling the dissertation requirement.

Date:

Place: V. Arvind

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at Homi Bhabha National Institute (HBNI) and is deposited in the Library to be made available to borrowers under rules of the HBNI.

Brief quotations from this dissertation are allowable without special permisiion, provided that accurate acknowledgement of source is made. Requests for permisiion for extended quotation from or reproduction of this manuscriptin whole or in part may be granted by the Competent Authority of HBNI when in his or her judgement the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

Gaurav Rattan

DECLARATION

I, hereby declare that the investigtion presented in the thesis has been carried out by me. The work is original and has notbeen submitted eralier as a whole or in part for a degree / diploma at this or any other Institution / University.

Gaurav Rattan

ACKNOWLEDGEMENTS

Firstly, I wish to thank my advisor V. Arvind for his constant encouragement and guidance. I am always inspired by his passion and enthusiasm for research. He introduced me to many exciting topics in mathematics and computation. Thanks to him, I have realized the importance of clarity and coherence in my research and writing. I have been very fortunate to be advised by him for the past five years.

I also wish to thank my collaborators for my formative research experiences. I would like to sincerely thank Johannes Köbler for hosting my stays at the Humboldt University, Berlin. I enjoyed many interesting discussions with him regarding the Graph Isomorphism problem. I would also like to thank Oleg Verbitsky, Sebastian Kuhnert, Pushkar Joglekar, Yadu Vasudev and Frank Fuhlbrück for our successful collaborative efforts.

I would like to thank the CS group at IMSc for creating a wonderful learning environment. I sincerely thank the faculty members for their patience and encouragement during my coursework. The group seminars and discussions opened up a lot of new directions in research. I would also like to thank the administrative staff at IMSc for their unrelenting support.

Thanks to my friends and colleagues, my stay at IMSc was truly memorable. Thanks to them, I re-discovered the joys of outdoors and cuisine. The cheerful presence of my friends has been a constant source of happiness and satisfaction.

Finally, I wish to thank my parents and my sister Anisha for their unwavering support and presence. They always give me the courage and freedom to pursue my dreams and goals.

Contents

Synopsis

The Graph Isomorphism Problem consists of the following natural question: given two graphs, are they isomorphic? In other words, does there exist an adjacency-preserving bijection between the vertex sets of the two graphs? A multitude of theoretical ideas and techniques have found application to the Graph Isomorphism Problem. Many of these seemingly disparate techniques have been found to be connected with each other in interesting ways, and often equivalent in their power and scope. These techniques include group theory, descriptive complexity, convex optimization and geometry. In this thesis, we explore some non-group-theoretic approaches to Graph Isomorphism. We evaluate the power and limitations of these approaches towards isomorphism testing.

Given two sets A and B of n points in a k-dimensional Euclidean space, does there exist a distance-preserving bijection between them? Formally called *Geometric Graph Isomorphism* (GEOM-GI), this problem is a geometric analogue of Graph Isomorphism (GI). In our study, we consider point-sets with *rational* entries. Using techniques from geometry and lattices, we obtain a $\tilde{O}(2^{O(k^2)})$ time FPT algorithm for this problem (the $\tilde{O}$ notation hides factors which are polynomial in input size). Here, the dimension k of the underlying space is the parameter of interest.

We then consider the problem of computing *canonical forms* for geometric point-sets in $\mathbb{Q}^k$. We obtain a $\tilde{O}(k^{O(k)})$ time procedure for *geometric graph canonization*. The canonization procedure immediately implies a faster $\tilde{O}(k^{O(k)})$ running time FPT algorithm for GEOM-GI. We also briefly consider the isomorphism problem for other l_p metrics (the case $p = 2$ is Euclidean).

The one-dimensional version of the Weisfeiler-Leman algorithm is commonly known as *Color-Refinement*, or naive vertex-classification. Color-Refinement is a classical procedure used to distinguish non-isomorphic graphs. Although it works incorrectly on certain

input instances, it works well in practice. This motivates the following natural question: what is the exact scope of applicability of color-refinement? We call a graph *G amenable* if color-refinement successfully distinguishes G from any other non-isomorphic graph H. Therefore, the graph class Amenable represents the limits of applicability of color-refinement. In our work, we give a structural characterization for this class. Consequently, we obtain a polynomial time algorithm for testing whether a given graph is amenable. In fact, we show that this problem is complete for the complexity class P, under logspace reductions.

Continuing this theme, we examine the power and limitations of convex optimization techniques for GI. In this context, there exists a very natural class of graphs called *compact* graphs. The compactness property is based on the properties of the convex polytope arising from Tinhofer's linear program. The isomorphism problem for compact graphs can be efficiently solved using linear programming methods. Therefore, compact graphs represent the limits of applicability of Tinhofer's LP approach to GI. In this context, we revisit the following question: can we give a characterization for the class of compact graphs? In our work, we show that the class of compact graphs contains the class of amenable graphs. In other words, the range of applicability of Tinhofer's LP approach is at least as large as color-refinement. Exploring this connection further, we study a strict hierarchy of graph classes based on structural, group-theoretic and algorithmic properties of graphs with respect to isomorphism testing.

List of Figures

Chapter 1

Introduction

The Graph Isomorphism Problem consists of the following natural question: given two graphs, are they isomorphic? In other words, does there exist an adjacency-preserving bijection between the vertex sets of the two graphs? Apart from being a fundamental algorithmic question of interest, this problem possesses a unique status in the world of computational complexity in the following sense. The problem has resisted attempts to show that it is polynomial-time solvable, yet there is substantial evidence that this problem is not NP-complete. The other notable feature of this problem is the diverse range of theoretical perspectives which have been found to be useful. In the remainder of this section, we will briefly overview these approaches and their remarkable connections. This overview will establish the setting for describing the main results of this thesis.

Let $X = (V, E)$ be a simple undirected graph on a vertex set V of size n. The set of all automorphisms of X forms a group, denoted by $Aut(X)$. We can view $Aut(X)$ as a permutation group acting on the vertex set V. For computational purposes, a permutation group is usually represented by a generating set. In particular, a permutation group acting on a set of size n can always be represented by a generating set of size at most n^2. The framework of computational group theory addresses various algorithmic questions regarding permutation groups (see e.g. [29] for a comprehensive treatment). Indeed, this framework turns

out to be very useful for graph isomorphism due to the following fact. Given a graph X, let GRAPH-AUT denote the problem of computing a generating set for $Aut(X)$.

Theorem 1. *The search problem* GRAPH-AUT *is polynomial-time equivalent to the Graph Isomorphism problem.*

The first successful application of this framework was the polynomial-time algorithm for testing isomorphism of bounded color-class-size graphs, due to Babai [8]. A breakthrough result in this direction was the polynomial-time algorithm for testing isomorphism of bounded degree graphs, due to Luks [24]. For graphs of degree at most d, this algorithm runs in time $n^{O(d)}$ and heavily relies on permutation group-theoretic machinery. Augmented by a combinatorial partitioning technique of Zemlyachenko (e.g. [35]), this algorithm yielded a $exp(\sqrt{n \log n})$ running-time procedure for Graph Isomorphism. Recently, in Nov. 2015, Babai gave a breakthrough $exp((\log n)^{O(1)})$ running-time algorithm for general Graph Isomorphism.

Theorem 2 ([7], Corollary 1.1.2)**.** *The Graph Isomorphism Problem can be solved in quasipolynomial time.*

This algorithm builds on the previous work, using new combinatorial partitioning and group-theoretic tools to achieve the improved time complexity. Currently, group isomorphism presents the next barrier to an improved algorithm for Graph Isomorphism: although a $n^{O(\log n)}$ running-time algorithm exists for this problem, any substantial improvements (in the exponent) have remained elusive over the decades.

On the other hand, it is natural to ask whether there exists a *simple combinatorial* algorithm for this problem, which avoids the sophisticated group-theoretic machinery. The *k-dimensional Weisfeiler-Leman procedure*, denoted by k-WL, is one of the earliest attempts in this direction. The initial objective behind this procedure was to associate a unique object with a graph, called a cellular algebra (the survey [34] contains a comprehensive treatment). The one-dimensional version of this procedure is commonly known

as *color-refinement* (CR). Given a graph G, CR iteratively colors the vertices as follows.

Initial Step: Every vertex u gets the *same* color: $C^0(u) := 1$.

Iterative Step: Let u and v be vertices of the same color. If u and v have a different number of neighbors of some color, then give different colors to u and v:

$$C^{i+1}(u) = \left(C^i(u), \left\{C^i(a) : a \in N(u)\right\}\right).$$

After each round, sort the colors and replace them by integers.

Termination: If no color class gets further refined, the procedure stops and outputs the coloring.

Color-refinement is a classical tool, used for distinguishing non-isomorphic graphs. Although it fails on certain input instances, it works very well in practice [9]. When color-refinement (k-WL) fails to distinguish two graphs, we call them 1-WL-*indistinguishable* (k-WL-indistinguishable).

The Weisfeiler-Leman procedure assumed a greater theoretical significance due to the seminal work of Immerman and Lander [21]. They used the descriptive complexity paradigm to study the isomorphism and canonization problem for graphs. The usual first-order language of graph theory is built up from the variables $x_1, x_2, \ldots$, the relations symbols E and $=$, the logical connectives $\wedge, \vee, \sim$, and the quantifiers, $\forall$ and $\exists$. In their paper, they proposed two modifications to this language. First, we are allowed to use only a bounded number of variables, say k. Secondly, we add the ability to count using *counting* quantifiers of the form $(\exists_i x)$. For example, the meaning of $\exists_5 x\ \varphi(x)$ is: there exist at least 5 vertices in the graph which satisfy $\varphi(x)$. We use C_k to denote the new language obtained in this fashion. In general, two graphs G and H are said to be C_k-*equivalent* if they agree on the same set of sentences in C_k. In this context, the following theorem of Immerman and Lander connects the k-WL procedure and the C_k language.

Theorem 3 ([21], restated)**.** *Two graphs G and H are k-WL-indistinguishable if and only if they are C_{k+1}-equivalent.*

Indeed, it was conjectured that k-WL method with a slowly growing value of k (e.g. $k = O(\log n)$ or even $k = O(1)$) would solve the Graph Isomorphism problem. However, this conjecture was disproved in a paper of Cai, Furer and Immerman [13].

Theorem 4 ([13], restated)**.** *There exists a sequence of pairs of graphs $\{G_n, H_n\}$, $n \in \mathcal{N}$ on $O(n)$ vertices such that G_n and H_n are non-isomorphic, but C_n-equivalent.*

Nevertheless, the k-WL method has been shown to efficiently solve Graph Isomorphism for many interesting graph classes (e.g. planar graphs [16], graphs of bounded genus [18] and graphs with excluded minors [17]).

Convex optimization is a fundamental tool for designing new algorithms for hard problems. In a very interesting paper [32], Tinhofer developed this approach by proposing an integer linear program for GI. The integral solutions to this program are permutation matrices, corresponding to isomorphisms between graphs. The fractional solutions to this linear program are *doubly-stochastic* matrices, and are called *fractional* isomorphisms. We call two graphs to be fractionally isomorphic if there exists a fractional isomorphism between them. A surprising result of Ramana, Scheinerman and Ullman showed the following connection between color-refinement and linear programming.

Theorem 5 ([28], restated)**.** *Two graphs are fractionally isomorphic if and only if they are* 1*-WL-indistinguishable.*

Therefore, Tinhofer's convex optimization approach is equivalent to the color-refinement approach. It is natural to ask whether we can strengthen Tinhofer's linear programming framework for isomorphism testing. A systematic method for adding additional constraints to a linear program is the standard 'lift-and-project' LP hierarchies, such as Lovasz-Schrijver hierarchy [23] and the Sherali-Adams LP relaxation hierarchy [30]. In

particular, Asterias and Maneva studied the Sherali-Adams relaxation hierarchy for Tinhofer's linear program [6]. They showed that the levels of Sherali-Adams hierarchy interleave in power with the levels (i.e. dimensions) of Weisfeiler-Leman algorithm.

Nevertheless, it remains an interesting research direction to further develop the convex optimization approach to Graph Isomorphism. The possibility of obtaining a non-group theoretic algorithm for Graph Isomorphism, which is better than the trivial $n!$, perhaps lies there.

Another non-group-theoretic approach to Graph Isomorphism which has proved to be useful is to analyze the spectrum of the adjacency matrices of graphs. An important parameter of interest is the eigenvalue multiplicity of a graph, defined as the maximum dimension of an eigenspace of its adjacency matrix. Babai, Grigoriev and Mount [10] showed a polynomial-time algorithm for isomorphism testing of graphs with bounded eigenvalue multiplicity. Their algorithm runs in time $O(n^{O(b)})$ where b is the bound on the eigenvalue multiplicity. Subsequently, Evdokimov and Ponomarenko [15] presented a FPT algorithm for this problem, where b is the parameter of interest. The algorithm runs in time $\tilde{O}(b^{O(b)})$ (where $\tilde{O}$ hides factors polynomial in input size).

Roughly, the idea behind these algorithms is to project the problem onto each eigenspace, solve for each eigenspace independently, and finally merge the partial solutions to obtain a valid isomorphism. The sub-problem obtained for an eigenspace turns out to be an instance of an isomorphism problem, known as the *Geometric Graph Isomorphism* (GEOM-GI). A geometric analogue of GI, this problem asks whether there is a distance-preserving bijection between two sets of points in a Euclidean vector-space. The dimension k of the underlying space is an important parameter of interest. Evdokimov and Ponomarenko have shown a $\tilde{O}(2^{O(k^4)})$ running time FPT algorithm for this problem [14]. In general, the spectral and geometric methods constitute a promising set of tools for studying Graph Isomorphism.

The structural complexity of Graph Isomorphism has been a subject of active research,

considering its unique complexity-theoretic status. In particular, it is known to be in the complexity class co-AM, a randomized analogue of co-NP. As a consequence, GI is unlikely to be NP-hard, as evident from the following theorem.

Theorem 6 ([11], restated)**.** GI *is not* NP*-complete, unless the polynomial hierarchy collapses to its second level.*

Hence, there is reasonable complexity-theoretic evidence that GI is not NP-complete. This belief was further strengthened using the counting properties of Graph Isomorphism, in a series of work culminating in [5]. The reader is refered to the survey of Köbler et al. [22] for a concise exposition of the research in this direction. On the other end, the P-hardness of GI remains open. Currently, the best known hardness result for GI is due to Toran [33]: GI is hard under AC_0 many-one reductions for the complexity class NL, logspace counting classes Mod_kL for all $k \geq 2$, and the class DET of problems NC_0 reducible to the determinant.

Summarizing, a multitude of theoretical ideas and techniques have found application to the Graph Isomorphism Problem. Many of these seemingly disparate techniques have been found to be connected with each other in interesting ways, and often equivalent in their power and scope. In the next two sections, we describe the contents of this thesis and their relevance to these results.

1.1 Thesis Outline

In this thesis, we explore some non-group-theoretic approaches to Graph Isomorphism. Our objective will be to evaluate the power and limitations of these approaches towards isomorphism testing. In this section, we briefly introduce the main results of the thesis. In the next section, we elaborate on these results and describe the organization of the thesis in detail.

Given two sets A and B of n points in a k-dimensional Euclidean space, does there exist a distance-preserving bijection between them? Formally called *Geometric Graph Isomorphism* (GEOM-GI), this problem is a geometric analogue of GI. In our study, we consider point-sets with *rational* entries. Using techniques from geometry and lattices, we obtain a $\tilde{O}(2^{O(k^4)})$ time FPT algorithm for this problem. Here, the dimension k of the underlying space is the parameter of interest.

We then consider the problem of computing *canonical forms* for geometric point-sets in $\mathbb{Q}^k$. We obtain a $\tilde{O}(k^{O(k)})$ time procedure for *geometric graph canonization*. The canonization procedure immediately implies a faster $\tilde{O}(k^{O(k)})$ running time FPT algorithm for GEOM-GI. We also briefly consider the isomorphism problem for other l_p metrics (the case $p = 2$ is Euclidean).

The one-dimensional version of the Weisfeiler-Leman algorithm is commonly known as *Color-Refinement*, or naive vertex-classification. Color-Refinement is a classical procedure used to distinguish non-isomorphic graphs. Although it works incorrectly on certain input instances, it works well in practice. This motivates the following natural question: what is the exact scope of applicability of color-refinement? We call a graph G *amenable* if color-refinement successfully distinguishes G from any other non-isomorphic graph H. Therefore, the graph class Amenable represents the limits of applicability of color-refinement. In our work, we give a structural characterization for this class. Consequently, we obtain a polynomial time algorithm for testing whether a given graph is amenable. We also show a matching P-hardness for amenability testing.

Continuing this theme, we examine the power and limitations of convex optimization techniques for GI. In this context, Tinhofer [31] defined a very natural class of graphs called *compact* graphs. The compactness property is based on the properties of the convex polytope arising from Tinhofer's linear program. The isomorphism problem for compact graphs can be efficiently solved using linear programming methods. Therefore, compact graphs represent the limits of applicability of Tinhofer's LP approach to GI. In this

context, we revisit the following question: can we give a characterization for the class of compact graphs? In our work, we show that the class of compact graphs contains the class of amenable graphs. In other words, the range of applicability of Tinhofer's LP approach is at least as large as color-refinement. Exploring this connection further, we study a strict hierarchy of graph classes based on structural, group-theoretic and algorithmic properties of graphs with respect to isomorphism testing.

1.2 Results and Thesis Organization

In this section, we describe the main results of each of the chapters of the thesis in detail. The technical contents of the thesis are organized in the following four chapters.

1.2.1 Geometric Graph Isomorphism

In general, the isomorphism problem has been studied for a variety of discrete mathematical structures other than graphs, e.g. groups [26], semigroups [12], rings [12], lattices [20] etc. The particular structure of these objects can either yield faster algorithms for isomorphism testing, or could be sufficient to encode Graph Isomorphism already.

Given two point-sets A and B of size n in a k-dimensional Euclidean space, does there exist a distance-preserving bijection between them? This problem is formally known as the *Geometric Graph Isomorphism* (GEOM-GI) problem. This problem is, in fact, computationally equivalent to GI for unbounded k. Indeed, it is possible to encode a graph with n vertices and m edges as a set of $n + m$ points in $\mathbb{R}^n$ with the following property. Two graphs G and H are isomorphic if and only if the corresponding point-sets are geometrically isomorphic.

Since the problem is GI-hard in the regime $k = O(n)$, it is interesting to ask whether there are efficient algorithms for this problem when the dimension k of the underlying

space is small. Indeed, for bounded dimension, there is a straight-forward polynomial-time algorithm for this problem. The algorithm runs in time $n^{O(k)}$ and its correctness can be proved using elementary linear algebra. Therefore, we can ask: is there a fast fixed-parameter tractable algorithm for this problem, with the dimension k as a parameter? Indeed, the problem is known to be fixed-parameter tractable with the dimension k as a parameter [14]. Given two point sets with real coordinates, the FPT algorithm runs in time $\tilde{O}(2^{O(k^4)})$, where $\tilde{O}$ notation hides factors which are polynomial in the input size.

In our work, we consider point-sets with *rational* coordinates. We show a faster FPT algorithm for GEOM-GI, where the dimension k of the underlying space is the parameter. Formally, we show the following

Theorem 7. *There is a deterministic algorithm for* GEOM-GI *which runs in time* $2^{O(k^2)}$. *Here, the* $\tilde{O}$ *notation hides factors which are polynomial in the input size.*

The proof of the main theorem has three main ingredients. First, we show that any distance-preserving bijection naturally extends to a linear isometry of the underlying space. Second, we appeal to the lattices generated by the two point-sets and compute the set of shortest vectors for each point-set. Since the dimension of the underlying space is bounded, these sets must be small. Finally, we reconstruct the linear isometry, and hence the distance-preserving bijection, using these shortest vector sets. Putting these three steps together, we obtain our FPT algorithm. The results of this chapter are reported in [1].

1.2.2 Geometric Graph Canonization

Continuing our study of geometrical point-sets, we consider the problem of computing *canonical forms* for point-sets. I.e., given a point-set A, the algorithm should compute a point-set $f(A)$ with the following properties: (a) $f(A)$ is isomorphic to A, and (b) if a point set B is isomorphic to A, then $f(B) = f(A)$.

We obtain a fast canonization procedure for point-sets, which runs in time $\tilde{O}(k^{O(k)})$. As a consequence, the GEOM-GI problem for point sets in $\mathbb{Q}^k$ has a deterministic $\tilde{O}(k^{O(k)})$ time algorithm.

Theorem 8 ([1]). *Given a finite point set $A \subset \mathbb{Q}^k$ of size n as input, there is a deterministic $\tilde{O}(k^{O(k)})$ time algorithm that computes a canonizing function $f(A)$.*

Our procedure is similar to the FPT algorithm of the previous section. To obtain a faster running time of $\tilde{O}(k^{O(k)})$, the procedure uses a recent result of [20] regarding shortest vector sets in integer lattices.

We also briefly study the isomorphism problem for non-Euclidean metrics. This case is considerably harder because of the lack of linear-algebraic structure present in the Euclidean case. Even a $n^{O(k)}$-time algorithm for this case is not clear for a general l_p metric, $p \neq 2$. However, we show a polynomial time algorithm for the two-dimensional case using color-refinement and geometrical considerations.

Theorem 9 ([1]). *Given point-sets A and B in $\mathbb{Q}^2$ as input, for any l_p metric, there is a deterministic polynomial-time algorithm for checking whether A and B are isomorphic.*

The results of this chapter are reported in [1].

1.2.3 On the Power of Color Refinement

The well-known *color-refinement* procedure underlies a simple yet fundamental approach to the problem of distinguisuhing two non-isomorphic graphs. Given a graph G, this procedure computes a colored partition of the vertex-set as follows. Initially, we assign a uniform color to every vertex of G. Iteratively, we classify the vertices according to their degrees into each color-class. Finally, we terminate the procedure if the vertex coloring cannot be refined further. Since this procedure can run for at most n steps, it terminates in a finite number of steps. The resulting partition of the vertex-set is known as the *stable*

color partition of the graph. Each color-class in the stable color partition is called a *cell* of the graph. A graph is called *discrete* if the stable color partition consists of singleton cells.

This procedure can be used to create a simple isomorphism test as follows. Given graphs G and H, we run color-refinement on their disjoint union $G \cup H$. We declare G and H to be isomorphic if they contribute the same number of vertices to each cell of $G \cup H$. Otherwise, we declare them to be non-isomorphic.

In general, this algorithm can be incorrect on certain pairs of graphs. The standard example is that of a pair of regular non-isomorphic graphs of same degree. But in practice, this algorithm is correct for almost all graphs [9]. We call a graph *amenable* if color-refinement correctly distinguishes it from any other non-isomorphic graph. This leads us to a natural question: can we characterize amenable graphs? Such a characterization would establish the exact scope of the applicability of color-refinement. Computationally, we ask for an efficient algorithm for the following problem. Given a graph G, is G amenable?

In our work, we give a polynomial time algorithm for testing whether a graph is amenable. The algorithm is based on our structural characterization for the class Amenable. In fact, we show the following result.

Theorem 10 ([3])**.** *The class of amenable graphs is recognizable in time $O((n+m)\log n)$, where n and m denote the number of vertices and edges of the input graph.*

We also show P-hardness for this problem. This establishes a complete complexity classification for the problem of amenability recognition. As noted in the introduction, a graph is amenable if and only if it is definable in the counting logic C_2. As an immediate corollary of our result, the problem of recognizing graphs which are definable in C_2 is P-complete. As an application, we give an alternative proof for the amenability of trees and forests.

Our characterization is based on the analysis of the stable color partition of the graph. We propose a series of necessary conditions on the cells of an amenable graph, which are of two types. The *local* conditions specify the kind of graphs which can be induced on each cell, and between a pair of cells. The *global* conditions, roughly speaking, forbid connections among a set of cells. Together, these conditions imply a 'tree-like' decomposition for an amenable graph. We then show that these conditions are also sufficient, which gives us the desired characterization. We show that these conditions can be efficiently checked, which shows the P-membership of the amenability recognition problem. The results of this chapter are reported in [3].

1.2.4 On Tinhofer's LP Approach to GI

It is possible to formulate Graph Isomorphism as an integer linear program (ILP) using the following observation [32]. If two graphs G and H are isomorphic via a permutation π, then the respective adjacency matrices A and B must satisfy the linear equations $AX = XB$. Here, X is the permutation matrix corresponding to the permutation π. Since a permutation matrix can be characterized as a 0-1 matrix with every row-sum and column-sum equal to 1, we immediately obtain an integer linear program for GI.

The linear programming relaxation relaxes the condition $X_{ij} \in \{0, 1\}$ to $X_{ij} \in [0, 1]$. A feasible solution to the relaxed LP is called a *fractional isomorphism* between G and H. In the special case $G = H$, it is called a *fractional automorphism*. It is easy to check that a feasible solution to the LP is indeed a doubly-stochastic matrix. As mentioned in the introduction, we have the following remarkable theorem [28]: two graphs have a fractional isomorphism if and only if they are indistinguishable by color-refinement.

In this context, Tinhofer defined an interesting class of graphs as follows. Call a graph G *compact* if the convex polytope of fractional automorphisms of G has only integral extreme points [31]. In other words, every fractional automorphism is a convex combi-

nation of the automorphisms. Given a compact graph G and a graph H, it is possible to efficiently decide whether they are isomorphic using Tinhofer's linear program. Since compact graphs represent the known limits of Tinhofer's LP approach, it is natural to ask whether we can characterize and recognize compact graphs.

In our main theorem, we show that the class of compact graphs strictly contains the class of amenable graphs. Therefore, the applicability of Tinhofer's LP approach is at least as large as that of color-refinement.

Theorem 11 ([4])**.** *All amenable graphs are compact.*

In general, Tinhofer's approach is strictly more powerful than color refinement because it is known that the class of compact graphs contains many regular graphs (for example, all cycles [32]) which are not amenable.

We look at the relationship between the concepts of compactness and color refinement also from the other side. Let us call a graph G *refinable* if the color partition produced by color refinement coincides with the orbit partition of the automorphism group of G. It is known that compact graphs are refinable [32]. Taking a finer look at the inclusion Compact $\subset$ Refinable, we discuss algorithmic and algebraic graph properties that were introduced by Tinhofer [31] and Godsil [19]. We note that, along with the other graph classes under consideration, the corresponding classes Tinhofer and Godsil form a hierarchy under inclusion:

$$\mathsf{Discrete} \subset \mathsf{Amenable} \subset \mathsf{Compact} \subset \mathsf{Godsil} \subset \mathsf{Tinhofer} \subset \mathsf{Refinable}.$$

We also show the following results on these graph classes.

Theorem 12 ([4])**.** *The above hierarchy is strict. Moreover, testing membership in any of these graph classes is* P*-hard.*

The results in this section are reported in [4].

Chapter 2

Geometric Graph Isomorphism

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 for this metric space is to decide if A and B are isomorphic. A well-studied version of this general problem is the standard k-dimensional *euclidean space* $(\mathbb{R}^k, l_2)$ equipped with the l_2 metric, where $\mathbb{R}$ denotes the set of real numbers. We denote this version of the problem by GEOM-GI. This problem is also known as the *Point Set Congruence* problem in the computational geometry literature [37, 38, 39]. It is called "Geometric Graph Isomorphism" by Evdokimov and Ponomarenko in [36], which we find more suitable as the problem is closely related to Graph Isomorphism.

When k is constant, there is an easy polynomial-time algorithm for the problem [39]. When $k = n$, Papadimitriou and Safra [40] note that the problem is polynomial-time equivalent to the standard Graph Isomorphism problem. The interesting case is when the dimension k is much smaller than n. In the computational geometry literature, a randomized algorithm running in time $O(n^{\frac{k-1}{2}} \cdot \log n)$ was given in [37]. This was improved to an $O(n^{\lceil \frac{k}{3} \rceil} \cdot \log n)$ algorithm in [38]. We note that both these results are for a random access model of computation, used in computational geometry, which allows for arbitrary precision real arithmetic.

For point sets A, B with real coordinates, when the dimension k treated as a fixed parameter, an FPT algorithm for the GEOM-GI problem, running in time $(2^{k^4}nM)^{O(1)}$, was given by Evdokimov and Ponomarenko [36], where M upper bounds the binary encodings of the rational numbers in the input point sets. Their algorithm uses concepts from cellular algebras and is technically nontrivial.

Our Results

In this chapter, we describe an FPT algorithm for GEOM-GI, when the input point sets have *rational* co-ordinates. Here, the parameter of interest is the dimension k of the underlying space. Our algorithm follows a novel approach using integer lattices for this problem. In Section 2.1, we introduce necessary definitions and some well-known results to set the context for presenting our results. In Section 2.2, we describe a XP algorithm for this problem. Here, XP denotes the class of parameterized problems that can be solved in time $n^{f(k)}$ for some computable function f. This simple algorithm serves as a template for faster algorithms. In Section 2.3, we describe our FPT algorithm for this problem. Theorem 16 is the main result of this chapter. The algorithm uses ideas from geometry and lattices to obtain a better running time complexity. We conclude with a brief discussion in Section 2.4.

2.1 Preliminaries

2.1.1 Linear Algebra

We assume familiarity with the basic notions of vector spaces, linear transformations and matrices. We recall some important concepts from linear algebra. Every vector space contains the zero vector, denoted by $\mathbf{0}$. A set of vectors $\{u_1, \ldots, u_n\}$ is *linearly dependent* if there exist corresponding scalars $\alpha_1, \ldots, \alpha_n$, not all of them zero, such that the *linear*

combination

$$\sum_{i=1}^{n} \alpha_i x_i = \mathbf{0}.$$

Otherwise, the set is *linearly independent*. Given a set of vectors S, the vector space *spanned* by the set S, denoted by $\mathsf{Span}(S)$, is the set of all linear combinations of vectors in S. If the field of scalars associated with the vector space is $\mathbb{F}$, it is also called the $\mathbb{F}$-*linear span of* S. A vector space V is *spanned* by a set S of vectors if $V = \mathsf{Span}(S)$. A *basis* of a vector space V is a linearly independent set of vectors which spans V. The *dimension* of a vector space is the cardinality of a basis of a vector space. A *subspace* of a vector space is a subset which is closed under vector space operations. A *linear functional* ℓ on a vector space V over a field $\mathbb{F}$ is a function from V to $\mathbb{F}$ which respects linearity. I.e., for any vectors $u, v \in V$ and scalars $\alpha, \beta \in \mathbb{F}$,

$$\ell(\alpha u + \beta v) = \alpha \ell(u) + \beta \ell(v).$$

The set of all linear functionals on a vector space V is itself a vector space over $\mathbb{F}$, and is called the *dual space* of V. We denote the dual space of V by V'.

The *external direct sum* of two vector spaces U and V, denoted by $U \oplus V$, is the set $U \times V$ where the vector space operations are defined in a component-wise manner. Given a vector space W with subspaces U and V, we call W to be the *internal direct sum* of U and V if $U \cap V = \{\bar{0}\}$ and $U \cup V$ spans W. These definitions are known to be equivalent by the following well-known characterization.

Theorem 13. *The following are equivalent.*

- *W is the external direct sum of U and V.*
- *W is the internal direct sum of U and V.*
- *Every vector $w \in W$ can be uniquely decomposed as a sum $w = u + v$ where $u \in U$ and $v \in V$.*

Given a subspace U of a vector space W, there exists a subspace $\tilde{U}$ such that $W = U \oplus \tilde{U}$. The above theorem ensures that every vector $w \in W$ can be uniquely decomposed as $u + \tilde{u}$ where $u \in U$ and $\tilde{u} \in \tilde{U}$. Given a vector $w \in W$, the vector u is the *projection* of w onto the subspace U, denoted by $pr(w, U)$.

We now turn our attention to the special case of *real* vector spaces. The *dot product* of two vectors u and v in $\mathbb{R}^k$ is defined to be

$$u \cdot v = \sum_{i=0}^{k} u_i v_i.$$

Two vectors are said to be *orthogonal* if $u \cdot v = 0$. The Euclidean norm of a vector v, denoted by $\|v\|$, is $\sqrt{v \cdot v}$. The *distance* between two vectors u and v is $\|u - v\|$.

In general, for $p \geq 1$, the p-norm of a vector $x = (x_1, \ldots, x_k)$ is $\|x\|_p = (|x_1|^p + \cdots + |x_k|^p)^{1/p}$, and the ∞-norm $\|x\|_\infty$ is $max\{|x_1|, \ldots, |x_k|\}$. The Euclidean norm is the 2-norm.

Given real vector spaces U, V, a *linear isometry* is a bijective linear map $T : U \mapsto W$ which is distance-preserving. I.e., $\|u - v\| = \|T(u) - T(v)\|$ for every $u, v \in U$.

We focus our attention to the k-dimensional rational vector space, denoted by $\mathbb{Q}^k$. Given a vector $v \in \mathbb{Q}^k$ and a basis $J = \{u_1, \ldots, u_l\}$ of a l-dimensional subspace U, we can compute the projection $Pr(v, U)$ as follows. Let the projection $Pr(v, U)$ be a linear combination $\alpha_1 u_1 + \cdots + \alpha_l u_l$ for some unknown coefficients $\alpha_1, \ldots, \alpha_l \in \mathbb{Q}$. Observe that $(v - Pr(v, U)) \cdot u_j$ is zero for all $u_j \in J$. Simplifying, we obtain a linear equation $\alpha_1 u_1 \cdot u_j + \alpha_2 u_2 \cdot u_j + \cdots + \alpha_l u_l \cdot u_j = v \cdot u_j$ for every $u_j \in J$. Using matrix notation, we can rewrite this system as $Gx = c$ where G is a $l \times l$ matrix with entries $G_{ij} = u_i \cdot u_j$, x is a l-dimensional vector of unknowns $[\alpha_1 \ldots \alpha_l]^T$ and c is a l-dimensional vector $[v \cdot u_1 \ldots v \cdot u_l]^T$. Solving this system of linear equations, we obtain the coefficients $\alpha_1, \ldots, \alpha_l$ and hence, the projection $Pr(v, U)$. We summarize the above discussion as the following claim.

Claim 1. *Given a vector $v \in \mathbb{Q}^k$ and a basis $J = \{u_1, \ldots, u_l\}$ of a l-dimensional subspace U, let the projection $Pr(v, U)$ be the linear combination $\alpha_1 u_1 + \cdots + \alpha_l u_l$. Then, the vector*

$x = [\alpha_1 \dots \alpha_l]^T$ can be obtained as the solution to the system of linear equations of the form $Gx = c$ as defined above.

2.1.2 Integer Lattices

We now recall some definitions and properties of integer lattices [45]. Given a set of vectors $B = \{b_1, \dots, b_m\}$, an integer linear combination of B is a vector of the form $\alpha_1 b_1 + \cdots + \alpha_m b_m$ for some $\alpha_1, \dots, \alpha_m \in \mathbb{Z}$.

Definition 1 (Integer Lattices)**.** *A* lattice *is a* discrete *additive* subgroup *of $\mathbb{R}^n$. I.e., it is a set $\mathcal{L}$ with the following two properties.*

- *$\mathcal{L}$ is closed under addition and subtraction*
- *$\mathcal{L}$ is discrete, i.e. there exists an $\epsilon > 0$ such that any two lattice points x, y are at distance at least ϵ from each other.*

An equivalent definition of a lattice is the following.

Definition 2 (Integer Lattices)**.** *Let B be a set of k linearly independent vectors in $\mathbb{R}^n$. The lattice generated by set B, denoted by $\mathcal{L}_B$, is the set of all integer linear combinations of B. The number n is called the* dimension *of the lattice.*

Examples of lattices include (a) $\mathbb{Z}^n$ and (b) set of all integer linear combinations of a set B of vectors in $\mathbb{Q}^k$. In general, if a set B is linearly dependent and has irrational entries, it is possible that the set of all integer linear combinations of B may not form a discrete lattice. For example, the set $\{1, \sqrt{2}\}$ does not generate a lattice for the following reason. For every $\epsilon > 0$, we can find integers l, m such that $|l + \sqrt{2}m| \leq \epsilon$. The existence of such integers can be deduced from the fact that there exists a sufficiently large n for which the number $(\sqrt{2} - 1)^n < \epsilon$.

Regarding the vector set B, we will assume that the vectors b_i have rational entries with standard binary encodings (bounded by M). Then, we can compute a linearly independent basis of $r \leq k$ vectors for the lattice in time polynomial in k, M and m [45]. The number r is 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 an important algorithmic problem known to be NP-hard. There are several algorithms for exactly computing shortest vectors and for approximating them in the literature [45]. We recall an important and relatively recent result due to Micciancio and Voulgaris [41] for enumerating all the shortest vectors in a given lattice.

Theorem 14 ([41], Corollary 5.8)**.** *There is a deterministic algorithm that takes as input a basis of some lattice $\Lambda \subset \mathbb{Q}^k$, a target vector $\boldsymbol{t} \in \mathbb{Q}^k$, and an integer $p \geq 2$, and in time $\tilde{O}((4p)^k) \cdot poly(M)$ it outputs all vectors in Λ within distance $p\lambda_1(\Lambda)$ from $\boldsymbol{t}$. (The $\tilde{O}(\cdot)$ notation suppresses polylogarithmic factors). Here, M is the bound on the representation size of the input vectors.*

An immediate corollary of the above theorem is the following result.

Corollary 1. *There is a deterministic algorithm that takes as input a basis of some lattice $\Lambda \subset \mathbb{Q}^k$, and in time $\tilde{O}(4^k) \cdot poly(M)$, it outputs all shortest vectors in Λ.*

We also recall a well-known bound on the number of short vectors in a lattice (see [41]).

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 $(2p+1)^k$.*

2.2 XP algorithm for GEOM-GI

In this section, we describe an XP algorithm for GEOM-GI, parameterized by dimension k. This algorithm will serve as a template for faster GEOM-GI algorithms, which run in

FPT time.

We first define the problem GEOM-GI. Given two point sets A and B of size n in $\mathbb{Q}^k$, does there exist a bijection $\pi : A \to B$ such that for all $x, y \in A$, $\|x - y\| = \|\pi(x) - \pi(y)\|$? We study the fixed-parameter tractability of this problem and prove the following

Theorem 15. *There is an XP algorithm for* GEOM-GI*, parameterized by dimension k, which runs in time* $n^{O(k)}$.

The crux of the algorithm is the following lemma about isomorphic point sets. Let A and B be two point sets in $\mathbb{Q}^k$. Assume without loss of generality that the centroids of A and B are located at origin (we will shortly justify this assumption).

Lemma 2. *Two point sets A and B are geometrically isomorphic if and only if there exists a linear isometry between vector spaces* Span(A) *and* Span(B)*, which in addition, maps A to B. Here,* Span(A) *(and* Span(B)*) refers to the* $\mathbb{Q}$*-linear span of the set A (and set B).*

In other words, a geometric isomorphism between two point sets A and B in $\mathbb{Q}^k$ naturally *extends* to a linear isometry between vector spaces Span(A) and Span(B). We briefly justify our assumption above regarding the centroids of point sets. Given two point sets in $\mathbb{Q}^k$, we can always translate them such that their centroids are located at origin. Since the centroid of a point-set in $\mathbb{Q}^k$ has rational coordinates, the translated point-set has rational coordinates as well.

In the remainder of this section, we build towards Theorem 15, proving Lemma 2 in the process. We reiterate our assumption that the point sets A and B have origin as their centroid. Let $\pi : A \mapsto B$ be a geometric isomorphism between A and B. The image of a point $u \in A$ under the mapping π is denoted u^π. We remark that the points in these sets are vectors in $\mathbb{Q}^k$ and we will use these notions interchangeably in what follows.

We first claim that the mapping $\pi : A \mapsto B$ preserves lengths.

Claim 2. $\|u\| = \|u^\pi\|$ *for all* $u \in A$.

Proof. Since the centroid of set A is origin,

$$\begin{aligned}
\|u\|^2 &= \|u - \mathbf{0}\|^2 \\
&= \|u - \frac{1}{n}\sum_{j=1}^{n} u_j\|^2 \\
&= \frac{1}{n^2}\|\sum_{j=1}^{n}(u - u_j)\|^2 \\
&= \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n}(u - u_j)\cdot(u - u_k) \\
&= \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n}\frac{1}{2}(\|u - u_j\|^2 + \|u - u_k\|^2 - \|u_j - u_k\|^2) \\
&= \frac{1}{2n^2}\left[n\left(\sum_{j=1}^{n}\|u - u_j\|^2\right) + n\left(\sum_{k=1}^{n}\|u - u_k\|^2\right) - \left(\sum_{i\in[n],j\in[n]}\|u_j - u_k\|^2\right)\right].
\end{aligned}$$

Similarly, for $v = u^\pi$ in set B,

$$\|v\|^2 = \frac{1}{2n^2}\left[n\left(\sum_{j=1}^{n}\|v - v_j\|^2\right) + n\left(\sum_{k=1}^{n}\|v - v_k\|^2\right) - \left(\sum_{j\in[n],k\in[n]}\|v_j - v_k\|^2\right)\right].$$

Since π is a geometric isomorphism which maps u to v, we can rewrite

$$\begin{aligned}
\|v\|^2 &= \frac{1}{2n^2}\left[n\left(\sum_{j=1}^{n}\|u - u_j\|^2\right) + n\left(\sum_{k=1}^{n}\|u - u_k\|^2\right) + \left(\sum_{j\in[n],k\in[n]}\|u_j - u_k\|^2\right)\right] \\
&= \|u\|^2
\end{aligned}$$

which proves the claim. □

We claim that the mapping $\pi : A \mapsto B$ preserves dot products.

Claim 3. *$u_i \cdot u_j = u_i^\pi \cdot u_j^\pi$ for all $u_i, u_j \in A$.*

Proof. The dot product $u_i \cdot u_j$ can be rewritten using Claim 2 as follows.

$$\begin{aligned} u_i \cdot u_j &= \frac{1}{2}\left(\|u_i\|^2 + \|u_j\|^2 - \|u_i - u_j\|^2\right) \\ &= \frac{1}{2}\left(\|u_i^\pi\|^2 + \|u_j^\pi\|^2 - \|u_i^\pi - u_j^\pi\|^2\right) \\ &= u_i^\pi \cdot u_j^\pi \end{aligned}$$

which proves the claim. □

We claim that the mapping $\pi : A \mapsto B$ respects linear dependencies. Let $U = \{u_1, \ldots, u_m\}$ be a subset of A, and $\alpha_1, \ldots, \alpha_m$ be corresponding scalar elements.

Claim 4. *$\alpha_1 u_1 + \cdots + \alpha_m u_m = \boldsymbol{0}$ if and only if $\alpha_1 u_1^\pi + \cdots + \alpha_m u_m^\pi = \boldsymbol{0}$.*

Proof. We show the forward direction (the reverse direction follows by symmetry). Suppose $\alpha_1 u_1 + \cdots + \alpha_m u_m = \mathbf{0}$. Let $v = \alpha_1 u_1^\pi + \cdots + \alpha_m u_m^\pi$. We will show that $v = \mathbf{0}$.

Let v_j be a vector in B. Then, there exists an index j' such that $\pi(u_{j'}) = v_j$, for some $u_{j'} \in A$. The dot product $v \cdot v_j$ can be rewritten using Claim 3 as follows.

$$\begin{aligned} v \cdot v_j &= (\alpha_1 u_1^\pi + \cdots + \alpha_m u_m^\pi) \cdot v_j \\ &= (\alpha_1 u_1^\pi + \cdots + \alpha_m u_m^\pi) \cdot u_{j'}^\pi \\ &= (\alpha_1 u_1 + \cdots + \alpha_m u_m) \cdot u_{j'} \\ &= 0. \end{aligned}$$

Hence, $v \cdot v_j$ is zero for every $v_j \in B$. Since $v \in \mathsf{Span}(B)$, $v = \mathbf{0}$. This proves the claim. □

We state a useful corollary of the above claim. Let U be a subset of A, and U^π be the image of U under the mapping $\pi : A \mapsto B$.

Corollary 2. *The following holds.*

(a) U is linearly independent if and only if U^π is linearly independent.

(b) A vector $u \in A$ is a linear combination of vectors in U if and only if the vector $u^\pi \in B$ is a linear combination of vectors in U^π.

Proof. (a) Follows from Claim 4.

(b) If $u = \alpha_1 u_1 + \cdots + \alpha_m u_m$, then the linear combination $\alpha_1 u_1 + \cdots + \alpha_m u_m + (-1)u = \mathbf{0}$. Claim 4 implies that $\alpha_1 u_1^\pi + \cdots + \alpha_m u_m^\pi + (-1)u^\pi = \mathbf{0}$. Therefore, $u^\pi = \alpha_1 u_1^\pi + \cdots + \alpha_m u_m^\pi$.

□

The following claim summarizes the above series of claims. Let U be a subset of A.

Claim 5. *U is a basis for* $\mathsf{Span}(A)$ *if and only if U^π is a basis for* $\mathsf{Span}(B)$.

Proof. We show the forward direction (the reverse direction follows by symmetry). Let U be a basis for $\mathsf{Span}(A)$. Since U is linearly independent, U^π is also linear independent (by Corollary 2 (a)). Since U spans the vector space $\mathsf{Span}(A)$, every $u \in A$ can be expressed as a linear combination of vectors in U. By Corollary 2 (b), every $v \in B$ must be expressible as a linear combination of vectors in U^π. Consequently, U^π spans the vector space $\mathsf{Span}(B)$. Therefore, U^π is a basis for the vector space $\mathsf{Span}(B)$. □

We are ready to associate a linear map $\mu : \mathsf{Span}(A) \mapsto \mathsf{Span}(B)$ with the geometric isomorphism $\pi : A \mapsto B$.

Lemma 3. *Suppose π is a geometric isomorphism between A and B. There exists a linear isometry $\mu : \mathsf{Span}(A) \to \mathsf{Span}(B)$ such that μ agrees with π on A. Consequently, the point-sets A and B are geometrically isomorphic if and only if there exists an orthonormal transformation O such that $B = O \cdot A$.*

Proof. Let $J = \{u_1, \ldots, u_m\}$ be a subset of A such that J is a basis for $\mathsf{Span}(A)$. By Claim 5, J^π is a basis for $\mathsf{Span}(B)$. Define the linear transformation $\mu : \mathsf{Span}(A) \mapsto \mathsf{Span}(B)$

as follows. The map μ maps the basis J of $\mathsf{Span}(A)$ to the basis J^π of $\mathsf{Span}(B)$ (in the correct order according to π). The action of μ on $\mathsf{Span}(A)$ is then defined by linearity, since J is a basis.

It remains to show that (a) μ is bijective, (b) $\mu(x) = x^\pi$ for all vectors $x \in A$, and (c) μ is isometric.

To show that μ is bijective, it suffices to observe that J^π is a basis for $\mathsf{Span}(B)$.

To show that $\mu(x) = \pi(x)$ for all vectors $x \in A$, consider a vector $x \in A$. Since $x = \alpha_1 u_1 + \cdots + \alpha_m u_m$ for some scalars $\alpha_1, \ldots, \alpha_m$, Corollary 2 (b) implies that $x^\pi = \alpha_1 u_1^\pi + \cdots + \alpha_m u_m^\pi$. By linearity of μ, $\mu(x) = \alpha_1 \mu(u_1) + \cdots + \alpha_k \mu(u_k)$. Since μ agrees with π on J, $\mu(x) = \alpha_1 u_1^\pi + \cdots + \alpha_k u_k^\pi$. Hence, $\pi(x) = \mu(x)$ for every $x \in A$.

To show that μ is an isometry, it suffices to show that μ preserves dot products. I.e., for every $x, y \in \mathsf{Span}(A)$, $x \cdot y = \mu(x) \cdot \mu(y)$. By Claim 3, it follows that π preserves dot products among vectors in J. Since μ agrees with π on the set J, μ also preserves dot products among vectors in J. By bilinearity of dot product, μ preserves dot products among vectors in $\mathsf{Span}(A)$ as well and hence, μ is an isometry. □

The proof of Lemma 2 immediately follows from the above lemma. We end with the proof of Theorem 15.

Proof. Algorithm 2.2 describes the XP algorithm for GEOM-GI. We first show the correctness of the algorithm. It suffices to show that if the input point sets are geometrically isomorphic, the algorithm accepts. Let $\pi : A \mapsto B$ be a geometric isomorphism. By Lemma 2, there is a natural isometry μ associated with π, which extends π. On one of the branches of computation in Step 2, we will discover the linear transformation μ. Hence, the algorithm will accept the input.

We show that the running time cost of the algorithm is indeed bounded by $n^{O(k)}$. The size of set J in Step 1 is bounded by k. The branching factor in Step 2 is bounded by n^k

Input: Two point sets A and B in $\mathbb{Q}^k$, each of size n.

1. Pick a maximal linearly independent (ordered) set J in A.
2. For every (ordered) subset J' of B of size $|J|$,
   - Let T be the linear transformation defined by mapping J to J' (in an ordered manner).
   - Let T_A be the restriction of T to the set A.
   - Check if T_A maps A to B, in a distance-preserving manner. If yes, **accept**. Otherwise, try the next subset J'.
3. If none of the above branches accept, **reject**.

Figure 2.1. Algorithm 2.2

(since we pick an ordered image of the basis J). Other steps are routine polynomial time procedures. Therefore, the overall running time is bounded by $n^{O(k)}$. □

2.3 Lattice-based FPT algorithm for GEOM-GI

In the previous section, we mentioned that the XP algorithm for GEOM-GI serves as a template for faster FPT algorithms for GEOM-GI. In this section, we describe such an algorithm for GEOM-GI. In the following result, the usual $\tilde{O}(\cdot)$ notation hides multiplicative factors which are polynomial in input size.

Theorem 16. *Given two sets A, B in $\mathbb{Q}^k$, there is an FPT algorithm, parameterized by dimension k, which correctly decides whether they are geometrically isomorphic. The algorithm has running time complexity $\tilde{O}(2^{O(k^2)})$.*

Our FPT algorithm employs a novel use of integer lattices for this problem. Before proceeding with the proof of Theorem 16, we give a brief overview of the algorithm. Given two point sets A and B, consider the integer lattices $\mathcal{L}_A$ and $\mathcal{L}_B$ generated by them. Let S_A and S_B be the sets of shortest vectors in $\mathcal{L}_A$ and $\mathcal{L}_B$ respectively. Suppose the sets A and B are geometrically isomorphic. By Lemma 2, there exists a linear isometry

$\mu : \mathsf{Span}(A) \mapsto \mathsf{Span}(B)$, which in addition, sends A to B. By linearity, μ also maps $\mathcal{L}_A$ to $\mathcal{L}_B$. Since μ is distance-preserving, μ maps the set S_A to S_B. This observation facilitates the use of sets S_A and S_B (instead of A and B) for discovering the isometry μ. The improvement in running time stems from the fact that the sets S_A and S_B are bounded in size by 3^k. Hence, the branching factor in Step 2 of Algorithm 2.2 can be reduced from n^k to $\binom{3^k}{k}k!$. This is the crux behind our FPT algorithm.

However, there is a caveat: the subspace $\mathsf{Span}(S_A)$ can be a proper subspace of $\mathsf{Span}(A)$. In this case, the algorithm has only discovered the *restriction* of μ to the subspace $\mathsf{Span}(S_A)$, denoted by $\mu_1 : \mathsf{Span}(S_A) \mapsto \mathsf{Span}(S_B)$. To discover μ entirely, the algorithm employs a project-and-recurse approach as follows. We project out the point sets A and B onto the perpendicular subspaces $\mathsf{Span}(S_A)^\perp$ and $\mathsf{Span}(S_B)^\perp$ respectively. Thus, we obtain point sets A' and B' in spaces of strictly lower dimension. Recursively, we compute the restriction of μ to the subspace $\mathsf{Span}(S_A)^\perp$ using sets A' and B'. Therefore, we recover the isometry $\mu : \mathsf{Span}(A) \mapsto \mathsf{Span}(B)$ entirely. We are ready to proceed with the formal proof of Theorem 16.

Proof. Algorithm 2.3 describes the steps of our FPT algorithm. We first prove the correctness of the algorithm.

Claim 6. *Algorithm 2.3 accepts if and only if point sets A and B are geometrically isomorphic.*

Proof. It suffices to show that if there exists a geometric isomorphism π between A and B, the algorithm accepts. In this case, Lemma 2 implies that there exists a linear isometry $\mu : \mathsf{Span}(A) \mapsto \mathsf{Span}(B)$ which extends $\pi : A \mapsto B$. The algorithm discovers this isometry μ, and hence the isomorphism π, as follows.

Firstly, the isometry μ maps the set $S \subseteq S_A$ fixed in Step 2 to some set S' such that $S' \subseteq S_B$. Therefore, there exists a branch of computation in Step 2 such that the linear map μ_1 agrees with μ on S and hence, on $\mathsf{Span}(S)$ by linearity. Secondly, we claim that

Input: Two point sets A and B in $\mathbb{Q}^k$, each of size n.

1. Compute the shortest vector sets S_A and S_B of both $\mathcal{L}_A$ and $\mathcal{L}_B$ respectively.
2. Fix a maximal linearly independent set of shortest vectors S inside the set S_A. Let $|S|$ be k'. Branch on all ordered subsets S' of S_B of size k'. If S' is not a maximal linearly independent set in S_B, **reject**. Otherwise, let μ_1 denote the linear mapping which sends the set S to S' and therefore, extends to $\mathsf{Span}(S)$ by linearity.
3. Next, on each branch the algorithm projects the set A to the orthogonal complement $\mathsf{Span}(S)^\perp$ of the subspace spanned by S and projects B to the orthogonal complement $\mathsf{Span}(S')^\perp$ of the subspace spanned by S'.
4. Recursively continue to compute a linear map μ_2 using the above projected point sets that are in subspaces of strictly smaller dimension. Let μ_2 be the linear map thus obtained which maps $\mathsf{Span}(S)^\perp$ to $\mathsf{Span}(S')^\perp$.
5. Let $\mu : \mathsf{Span}(A) \to \mathsf{Span}(B)$ be the map which acts on $\mathsf{Span}(S)$ as μ_1 and on $\mathsf{Span}(S)^\perp$ as μ_2, and extends to $\mathsf{Span}(A)$ by linearity. Check if μ maps A to B in a distance-preserving manner. If this is the case, **accept**; otherwise reject this branch. If all branches reject, **reject**.

Figure 2.2. Algorithm 2.3

μ maps the set A' (obtained by projecting A onto $\mathsf{Span}(S)^\perp$) to the set B' (obtained by projecting B onto $\mathsf{Span}(S')^\perp$). To see this, let $u \in A$ be a point which is mapped by μ to a point $v \in B$. Since μ respects projections, $\mu(Pr(a, \mathsf{Span}(S))) = Pr(b, \mathsf{Span}(S'))$ and $\mu(Pr(a, \mathsf{Span}(S)^\perp)) = Pr(b, \mathsf{Span}(S')^\perp)$. Therefore, μ must map A' to B'.

As a result, we have point sets A' and B' which lie in subspaces $\mathsf{Span}(S)^\perp$ and $\mathsf{Span}(S')^\perp$ of strictly smaller dimension. Moreover, the restriction $\mu_2 : \mathsf{Span}(S)^\perp \mapsto \mathsf{Span}(S')^\perp$ of the isometry μ sends A' to B'. The algorithm will discover the restriction μ_2 recursively in Step 4. Since the isometry μ is completely described by the restrictions μ_1 and μ_2, we will discover μ in Step 5. Therefore, the algorithm discovers the isomorphism π and accepts. □

It remains to bound the running time required by the algorithm.

Claim 7. *The running time complexity of Algorithm 2.3 is bounded by* $\tilde{O}(2^{O(k^2)})$.

Proof. Observe that Step 2 of the algorithm involves branching on all subsets S' of S_B. We proceed by bounding the recursive branching that occurs during the execution of the algorithm. Assume that the given point sets A and B span subspaces of dimension p. We claim that the overall branching during the execution of the algorithm on A, B is bounded by $2^{O(k)\cdot p}$. Given any input point sets A and B in $\mathbb{Q}^k$, the overall branching during the execution of the algorithm therefore must be bounded by $2^{O(k^2)}$ (since $p \leq k$).

We prove the above claim by induction. Since any lattice in $\mathbb{Q}^k$ has at most $2^{O(k)}$ many shortest vectors (by Lemma 1), the initial branching in Step 2 is bounded by $\left(2^{O(k)}\right)^{k'}$. Here, k' is the number of linearly independent vectors inside set S_A. In Step 3, we project the point set A (and point set B) onto the orthogonal subspace $\mathsf{Span}(S)^\perp$ (and $\mathsf{Span}((S')^\perp)$) of dimension $p - k'$. Hence, the input point sets to the recursive call in Step 4 lie in subspaces of dimension $p - k'$. By our inductive assumption, the recursive branching incurred in Step 4 must be bounded by $2^{O(k)\cdot(p-k')}$. Therefore, the overall branching during the execution of the algorithm is bounded by $\left(2^{O(k)}\right)^{k'} \cdot \left(2^{O(k)\cdot(p-k')}\right)$, which is at most $2^{O(k)\cdot p}$.

We have shown that the overall branching during the execution of the algorithm must be bounded by $2^{O(k^2)}$. It remains to show that the computation along any branch of the algorithm takes at most $\tilde{O}(2^{O(k^2)})$ time. The computation of shortest vectors in Step 1 takes $\tilde{O}(k^{O(k)})$ time by Corollary 1. Claim 1 shows that projection operations in Step 3 are routine linear-algebraic procedures and can therefore be performed in polynomial time in input size. It remains to prove that the increase in the bit-complexity of entries in vectors of sets A and B is bounded by a suitable multiplicative factor after each projection operation of Claim 1. Claim 8 below shows that the bit-size can increase by at most a multiplicative factor of $O(k^3)$. Since we can do the projection operation at most k times, this implies a tolerable upper bound of $O(k^{3k})$ on bit-complexity of entries. □

Claim 8. *Let $A \in Q^{k\times k}$, $b \in \mathbb{Q}^k$ be such that each entry in A and b is represented by at most M bits. Suppose the system of linear equations $Ax = b$ has a unique solution $x_0 \in \mathbb{Q}^k$.*

Then, any entry in x_0 can be represented by at most $O(k^3)M$ bits.

Proof. Since the entries in A and b are rational numbers with bit-size of numerator and denominator bounded by M, we can clear all the denominators in A and b to obtain A' and b' with integral entries. In the worst case, the cost of this operation will be a multiplicative factor of $O(k^2)$ in bit complexity (since we have $O(k^2)$ entries in A and b). Let $M' = O(k^2)M$ denote the bit complexity of the integral entries in A and b after this step. Given the system $A'x = b'$, we note that any entry in x is a ratio of determinants of sutiable matrices, given by Cramer's rule. For a $k \times k$ matrix Q with entries of size $2^{M'}$, we have the upper bound $Det(Q) \leq k!(2^{M'})^k$. Hence, we need $O(k \log k) + kM'$ bits to represent the entries in x. Simplifying, we obtain a bound of $O(k^3)M$ on the bit complexity of entries in x_0. □

This finishes the proof of Theorem 16. □

2.4 Discussion

We showed that a lattice approach to GEOM-GI yields faster FPT algorithms for this problem. Can we further improve our FPT algorithm? We observe that these algorithms are sub-optimal in the following sense. In the branching step of either algorithm, we fix a set J and try every possible subset J' of size $|J|$ as its image under the isometry μ. Since μ preserves the geometry of J, it will be more efficient to try the subsets J' with the same geometry as the basis J. For example, if the set J is orthogonal, we need to try only orthogonal sets J' as the images.

In this next chapter, we employ a recent result of [20] which defines a certain class of bases, called *chain bases*. These bases have very useful properties in the following sense: (a) they always exist, (b) they are small in number, at most $\tilde{O}(k^{O(k)})$ and (c) they are

efficiently computable. Using these bases, we obtain a faster FPT algorithm for GEOM-GI, which runs in time $\tilde{O}(k^{O(k)})$.

Chapter 3

Geometric Graph Canonization

Computing canonical forms for structures is a fundamental algorithmic problem. *Graph Canonization*, which is the problem of computing canonical forms for graphs, is closely connected to Graph Isomorphism. For a class $\mathcal{K}$ of graphs, 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 for general graphs. Whether there is *some* polynomial-time computable canonizing function for graphs is open. It is also open whether graph canonization is polynomial-time equivalent to graph isomorphism. Graph classes with efficient isomorphism tests are often known to have canonization algorithms [43] of comparable complexity.

Analogously, corresponding to geometric isomorphism, we can define *canonical forms* and the *canonization problem* for an n-point sets $A \subset \mathbb{Q}^k$. Given $A \subset \mathbb{Q}^k$ as input, a *canonizing function* $f : A \mapsto f(A)$ outputs an isomorphic point set $f(A)$ such that $f(A) = f(B)$ if and only if A and B are isomorphic point sets.

Our Results

In this chapter, we describe an FPT algorithm for GEOM-GC, when the input sets have *rational* coordinates. Here, the parameter of interest is the dimension k of the underlying space. In Section 3.1, we introduce necessary definitions and some results to set the context for presenting our algorithm. In Section 3.2, we describe our FPT algorithm for this problem. In Section 3.3, we consider the GEOM-GI problem in non-Euclidean metrics and present our results. We finally conclude this chapter with a brief discussion in Section 3.4. In all the results here, we crucially use that A, B are rational point sets.

3.1 Preliminaries

In this section, we restate some important concepts and results regarding integer lattices. Haviv and Regev, in [42], study the lattice isomorphism problem under *orthogonal transformations*. In the process, they develop a general *Isolation Lemma* which they apply to the lattice isomorphism problem, and give a $\tilde{O}(k^{O(k)})$ time algorithm for checking whether two rank-k lattices are isomorphic under orthogonal transformations. They introduce the notion of a *linearly independent chain* in a given set of vectors. We recall the definition as we will require it to describe our canonical forms algorithm for point sets in $\mathbb{Q}^k$.

Definition 3. [42] *For a finite set of vectors $A \subseteq \mathbb{Q}^k$ and a vector $v \in \mathbb{Q}^k$, we say that v* uniquely defines a linearly independent chain of length *m in A if there are m vectors $x_1, \ldots, x_m \in A$ such that for every $1 \leq j \leq m$, the minimum inner product of v with vectors in $A \backslash \mathsf{Span}(x_1, \ldots, x_{j-1})$ is uniquely achieved by x_j.*

Given a lattice $\mathcal{L}$, its *dual lattice* $\mathcal{L}^*$ is defined as the set of vectors in $\mathsf{Span}(\mathcal{L})$ that have an integer inner product with every vector in $\mathcal{L}$. More precisely,

$$\mathcal{L}^* = \{u \in \mathsf{Span}(\mathcal{L}) \mid (u) \cdot (v) \in \mathbb{Z} \text{ for all } v \in \mathcal{L}\}.$$

The following theorem of Haviv and Regev [42] shows for any given lattice $\mathcal{L}$ that there exists a suitably short vector v in the dual lattice $\mathcal{L}^*$, which uniquely defines a linearly independent chain in the set of shortest vectors of the given lattice $\mathcal{L}$.

Theorem 17 ([42], Theorem 4.2)**.** *Let $\mathcal{L}$ be a lattice of rank k. Let S be the set of shortest vectors in $\mathcal{L}$. If the dimension of $\mathsf{Span}(S)$ is k, there exists a vector $v \in \mathcal{L}^*$ that uniquely defines a linearly independent chain of length k in S and satisfies $\|v\| \leq 5k^{17/2} \cdot \lambda_1(\mathcal{L}^*)$.*

3.2 FPT algorithm for GEOM-GC

In this section, we prove the main result of this chapter.

Theorem 18. *Given a finite point set $A \subset \mathbb{Q}^k$ of size n as input, there is a deterministic $\tilde{O}(k^{O(k)})$ time algorithm that computes a canonizing function $f(A)$. As a consequence, the* GEOM-GI *problem for point sets in $\mathbb{Q}^k$ has a deterministic $\tilde{O}(k^{O(k)})$ time algorithm.*

We begin with an overview of our algorithm. Given a point set $A \subset \mathbb{Q}^k$, we first compute the shortest vector set $\mathcal{S}_A$ of the lattice $\mathcal{L}_A$. For the overview description, we assume for simplicity that $\mathcal{S}_A$ spans $\mathsf{Span}(A)$. When that is not the case, the actual algorithm proceeds by projecting $\mathsf{Span}(A)$ to the orthogonal complement of $\mathsf{Span}(\mathcal{S}_A)$, similar to the Algorithm 2.3 of previous chapter. For a given point set $A \subset \mathbb{Q}^k$ let short_A denote the following set of short vectors in the dual lattice $\mathcal{L}_A^*$:

$$\text{short}_A = \{v \in \mathcal{L}_A^* \mid \|v\| \leq 5k^{17/2} \cdot \lambda_1(\mathcal{L}_A^*)\}.$$

Then, we proceed as follows.

1. We apply the Haviv-Regev algorithm [42] (Theorem 17) to pick the set short_A of short vectors in the dual lattice $\mathcal{L}_A^*$ which yield a unique linearly independent chain, of length k, in the set of shortest vectors $\mathcal{S}_A$ of $\mathcal{L}_A$. As shown in [42] such vectors

in the dual lattice exist and by Theorem 14 and Lemma 1 the set short_A is of size bounded by $k^{O(k)}$ and can be listed in $\tilde{O}(k^{O(k)})$ time.

2. Corresponding to each $v \in \text{short}_A$ the linearly independent chain of length k in $\mathcal{S}_A$, given by Theorem 17, yields a basis for $\mathsf{Span}(A)$. In all we obtain $k^{O(k)}$ such bases for $\mathsf{Span}(A)$.

3. For each such basis J we generate a description of the set A as follows: We first compute the Gram matrix $G(J)$ for J. Then for each vector $u_i \in A$, we compute the k-tuple Γ_i of the coordinates of u_i in basis J. The description of A obtained from basis J is the tuple $(G(J), \Gamma_1, \ldots, \Gamma_n)$.

We now briefly explain how these descriptions can be used to compute a canonical form for the point set A.

Suppose A and B are isomorphic point sets in $\mathbb{Q}^k$ and $\mu : \mathsf{Span}(A) \to \mathsf{Span}(B)$ is the corresponding linear isometry. Then μ is an isometric map between the lattices $\mathcal{L}_A$ and $\mathcal{L}_B$ as also between the dual lattices $\mathcal{L}_A^*$ and $\mathcal{L}_B^*$. Furthermore, μ maps short_A to short_B. More precisely, if $v \in \text{short}_A$ gives rise to a unique linearly independent chain J in $\mathcal{S}_A$ then $\mu(v) \in \text{short}_B$ gives rise to a unique linearly independent chain in $\mathcal{S}_B$ (which is in fact $\mu(J)$).

Now, crucially, we note that the description for A $(G(J), \Gamma_1, \ldots, \Gamma_n)$ generated using the chain J is *identical* to the description for B generated for $\mu(J)$. This is because the Gram matrices $G(J)$ and $G(\mu(J))$ are equal.

This suggests that the lexicographically least description is a canonical representation for the input point set A, and can be used to generate a canonical form for it. I.e. for each $v \in \text{short}_A$ satisfying the condition of Theorem 17, compute the description $(G(J), \Gamma_1, \ldots, \Gamma_n)$ using the corresponding linearly independent chain J. Among these descriptions pick the lexicographically least one $(G(J), \Gamma_1, \ldots, \Gamma_n)$ from which we will recover a canonical form for A.

3.2.1 Proof of Theorem 18

In this subsection, we give a proof of our main theorem. Algorithm 3.2.1 is a formal description of our canonization algorithm. We first prove two lemmas which show that the algorithm indeed outputs a canonical form C_A for a given point set A in $\mathbb{Q}^k$.

Lemma 4. *The point sets A and C_A are geometrically isomorphic.*

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 $J = \{r_1, \ldots, r_l\}$. By construction, a vector $u_i \in A$ is a Γ_i-linear-combination of J. Similarly, a vector $v_i \in C_A$ is Γ_i-linear-combination of the set $J_0 = \{l_1, \ldots, l_k\}$, the rows of the unique matrix L obtained in Step 4 (a). Since the sets J_0 and J have the same Gram matrix, there exists an orthogonal matrix O such that $r_i = O(l_i)$ for all $i \in [k]$. By linearity, vector $u_i = O(v_i)$ 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 point sets A and B in $\mathbb{Q}^k$ are geometrically isomorphic if and only if $C_A = C_B$.*

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 $O : \mathsf{Span}(A) \to \mathsf{Span}(B)$ 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 J_1 be the basis discovered in Step 2(a). We claim that $(O(w_1), \ldots, O(w_l))$ will be processed in the computation of the canonical form of B. This is true for $O(w_1)$ because (a) $\mathcal{L}_B = O\mathcal{L}_A$ and therefore, (b) for any $v \in \mathcal{L}_B$, $(O(w_1)) \cdot v = (O(w_1)) \cdot (O(u)) \in \mathbb{Z}$ for some u in $\mathcal{L}_A$. Also $\|w_1\| = \|O(w_1)\|$. Therefore,

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 $dim(\mathsf{Span}(A)) \neq 0$
   (a) Compute the set S_A of shortest vectors in $\mathcal{L}_A$ using Theorem 14.
   (b) Define the lattice $\Lambda_1 = \mathcal{L}_A \cap \mathsf{Span}(S_A)$.
   (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 14 and 17.
   (d) For each vector w in W, check whether 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 $\mathsf{Span}(S_A)$. I.e. replace every $u \in A$ by $u - u_{S_A}$.
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) Let $C_i = (s_1^i, \ldots, s_{k_i}^i)$ be the ordered chain of vectors corresponding to vector w_i. Here, the length of chain C_i is denoted by k_i.
   (b) Define an ordered basis J to be the set of the vectors s_f^i, $i \in [l]$, $f \in [k_i]$, ordered lexicographically by the tuple (i, f). We represent J as $(r_1, \ldots, r_k)$.
   (c) Compute the Gram matrix $G(J)$ for the ordered set J.
   (d) For each u_i in the input set A, compute the tuple $\Gamma_i = (\gamma_1, \ldots, \gamma_k)$ such that $u_i = \sum_{j=1}^{k} \gamma_j r_j$.
   (e) Lexicographically order the set $\{\Gamma_1, \ldots, \Gamma_n\}$ to obtain the string $lex(\{\Gamma_1, \ldots, \Gamma_n\})$.
   (f) Define the string σ for the tuple $(w_1, \ldots, w_l)$ to be $(G(J), lex(\{\Gamma_1, \ldots, \Gamma_n\})$
3. Let Σ be the set of all strings generated in the previous step. Determine the lexicographically least string σ_0 in Σ.
4. Given the string $\sigma_0 = (G, (\Gamma_1, \ldots, \Gamma_n))$,
   (a) Let $G = LL^T$ be the unique Cholesky decomposition of the postive semi-definite matrix G. Here, L is a lower triangular matrix.
   (b) Let J_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 J_0.
5. Output C_A as the canonical form for the set A.

Figure 3.1. Algorithm 3.2.1

$O(w_1)$ is a vector in the dual lattice $\mathcal{L}_B^*$ and is short enough to be discovered. Moreover, for any r_j in the chain generated by w_1, $w_1 \cdot r_j = (O(w_1)) \cdot (O(r_j))$ which implies that $O(r_j)$ is in the chain generated by $O(w_1)$. 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 computation for sets A and B proceeds by projecting the point sets A and B out of the subspaces spanned by shortest vectors. Since O maps $\mathsf{Span}(S_A)$ to $\mathsf{Span}(S_B)$ 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 $(O(w_1), \ldots, O(w_l))$ will be discovered in the computation for set B. Moreover, the basis J_B obtained for this tuple must be OJ_A. 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 J_A, then $O(u_i) \in B$ must be the Γ_i-linear-combination of $J_B = OJ_A$ 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. □

We are now ready to finish the proof of Theorem 18.

Proof of Theorem 18. It follows from Lemma 4 and Lemma 5 that Algorithm 3.2.1 correctly computes a canonical form for any input point set $A \subset \mathbb{Q}^k$. It remains to prove the running time bound. We first bound the time taken in Step 1 which can execute for at most k iterations. Computing sets S_A and W takes time $O^*(k^{O(k)}) \cdot poly(M)$, as a consequence of Theorem 14 and Theorem 17. Updating set S in Step 1(e) requires projection operations. We already know from the analysis of Algorithm 2.3 that projection operations can

be performed by usual polynomial-time linear-algebraic procedures. Moreover, Claim 8 shows that this can blow-up the bit-complexity of the entries by a multiplicative factor of at most $O(k^3)$. Since there can be at most k iterations of Step 1, the bit-complexity can increase at most by a tolerable multiplicative factor of $O(k^{3k})$.

Next, we bound the running time of Step 2. Recall that the set W_i computed in the i^{th} iteration of Step 1 is the set of all vectors in the dual lattice Λ_1^*, such that their length is bounded by $5k^{17/2} \cdot \lambda_1(\Lambda_1^*)$. In the i^{th} iteration of Step 1, let k_i be the rank of the lattice Λ_1 obtained in Step 1 (b). Since vectors in W_i are lattice vectors of bounded length, we can use Lemma 1 to show that $|W_i|$ is at most $(2 \cdot 5k_i^{17/2} + 1)^{k_i} = {k_i}^{O(k_i)}$. The number of tuples thus examined is at most $|W_1| \cdot \cdots \cdot |W_l|$ which is bounded by $k_1^{O(k_1)} \cdot \cdots \cdot k_l^{O(k_l)} \leq k^{O(k)}$. Other operations in Step 2 are polynomial-time computations. Hence, Step 2 takes $\tilde{O}(k^{O(k)})$ time. Steps 3, 4, and 5 all take time bounded by a polynomial in the input size. This completes the running time analysis. □

Remark 1. *An $\tilde{O}(k^{O(k)})$ time FPT algorithm for the isomorphism problem* GEOM-GI *follows from the above theorem: we compute the canonical forms for the input point sets A and B, and accept if and only if $C_A = C_B$.*

3.3 Geometric Isomorphism in other l_p metrics

In this section, we examine the GEOM-GI problem for other l_p metrics. It is important to observe that non-Euclidean metrics are not amenable to the usual linear-algebraic tools. For example, we do not have an analog of Lemma 2 for a general l_p metric. As a result, even a simple XP algorithm is not known for this problem, where the dimension k of the underlying space is our parameter of interest.

We make partial progress towards this problem by considering restricted dimensions. For the case $k = 2$, we show the following

Theorem 19. *Given subsets A and B of $\mathbb{Q}^2$ as input, for any l_p metric, there is a deterministic polynomial-time algorithm for checking whether A and B are isomorphic in that metric.*

Algorithm 3.3 is a formal description of our algorithm. We give a brief overview of our algorithm. 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 if and only if the graphs G and H are isomorphic via a color-preserving isomorphism. This computation can be performed in polynomial time. The isomorphism problem for color class size two graphs, denoted by BCGI_2 is in polynomial time [44] which yields the result.

In the remainder of this section, we give a formal proof of Theorem 19.

Proof. We will describe the algorithm for the l_∞ case and indicate how it can be easily adapted for other l_p metrics.

It is easy to verify that the algorithm works correctly for the case when the point 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 $\mathbb{Q}^2$ centered at x. It has sides of length

Input: Point sets A and B of size n in $\mathbb{Q}^2$ (the l_∞ case).
Output: **Accept** if A and B are isomorphic, and **reject** otherwise.

1. Decide whether sets A and B are collinear by iterating over all triples and checking whether the three points are collinear. If they are not collinear: pick the first three non-collinear points $\{a, b, c\}$. Otherwise,

   - Construct two colored graphs G and H: The graph G is $(A, \emptyset)$. The color of vertex u_i is the unordered pair $\{d_1, d_2\}$ 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 algorithm of [44].

2. Otherwise, w.l.o.g we have $a, b, c \in A$ (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 A, we color a point $u \in A$ by the ordered triple $(d_{u,a}, d_{u,b}, d_{u,c})$ of its distances from a, b, c.

4. Second, we can refine these colorings and ensure that each color class is of size two as follows:

   - If some set of vertices form a color class of size more than two, they will lie on a line segment parallel to x-axis or y-axis (proof of correctness explains). Each such color class has two extreme points.
   - For each vertex $u \in A, B$, decide 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, and have color classes of size at most two. The coloring of the vertices have been already computed in Step 4. Every edge $\{u, v\}$ in G' or H' is labeled with the weight d_{uv}, the distance between points u and v.

6. Using standard gadgets (details are given in the proof), we can remove the distance edge-labels introduced in Step 5. As a result, we obtain vertex-colored graphs G and H with color class size two, with unlabeled edges.

7. Test whether G is isomorphic to H using the algorithm of [44]. If the answer is yes **accept**, else move to the next branch in Step 2. If all branches are exhausted, return **reject**.

Figure 3.2. Algorithm 3.3

$2r$ 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.

It remains to show that using standard constructions, we can transform graphs G' and H' into graphs G and H such that (a) we preserve isomorphisms and (b) G, H are vertex-colored graphs with color class size two and unlabeled edges. Then, the algorithm will clearly accept the input in Step 7. To see the construction, notice that any pair of color classes C_i, C_j (in G' or H') have at most 4 edges between them (since $|C_i|, |C_j| \leq 2$). If all of them share the same distance label, it suffices to remove this label. If any three of them share the same distance label D_1 and the fourth edge has the distance label D_2, we remove the label D_1 and add a path of length 1 in place of the edge labeled D_2. We color the vertex on this path by a new color (D_2, i, j). The only remaining case is when at most two edges can share a common distance label. In this case, we replace each labeled edge by a path of length 1. The vertex on the path is given a new color (D, i, j) where D was the corresponding edge label. It is easy to verify that in all these cases, we obtain graphs with color class size at most two, with unlabeled edges. This suffices to finish the proof.

Other l_p metrics. We now briefly explain how the above algorithm can be adapted to solve the 2-dimensional GEOM-GI problem for other l_p-metrics.

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 $\mathbb{Q}^2$, this implies that any two l_p circles can intersect in at most two points ([46], Theorem 1). Therefore, any color class can be of size two and therefore, a similar algorithm which reduces the problem to BCGI_2 works and we can apply the algorithm of [44]. □

3.4 Discussion

In this chapter, we gave an $\tilde{O}(k^{O(k)})$ time FPT algorithm for Geometric Graph Canonization in the l_2 metric, which is asymptotically faster than previous algorithms for this problem. A natural open question is to improve the running time. From the point of view of the general Graph Isomorphism problem it would be very interesting to obtain a "geometric" algorithm of running time $\tilde{O}(2^{O(k)})$, since the well-known algorithms for this problem are group-theoretic.

As observed in the introduction, Graph Isomorphism (for n-vertex graphs) is polynomial-time reducible to GEOM-GI, where, in the reduced instance, the output point sets are contained in $\mathbb{Q}^n$. We note a similar reduction even for hypergraph isomorphism. There is a standard reduction that reduces hypergraph isomorphism for n-vertex and m-edge hypergraphs to bipartite graph isomorphism on $n + m$ vertices. However, the point sets thus obtained will be in $\mathbb{Q}^{n+m}$ and m could be much larger than n. The aim is to obtain point sets in as low a dimension as possible. 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.

The current best algorithm for Hypergraph Isomorphism [47] crucially uses a group-theoretic algorithm (for Coset-Intersection of permutation groups) and has running time $O^*(2^{O(n)})$ for n-vertex hypergraphs. However, the only known algorithm for computing

canonical forms of hypergraphs is the trivial $O^*(n!)$ time algorithm which picks the lexicographically least hypergraph isomorphic to the input hypergraph. Obtaining an $O^*(2^{O(k)})$ algorithm for computing canonical forms of point sets in $\mathbb{Q}^k$ would imply an $O^*(2^{O(n)})$ time (non-group-theoretic) canonization algorithm for hypergraphs, which is a long standing open problem.

Finally, we note that the complexity of GEOM-GI for point sets in $\mathbb{Q}^k$ in other l_p metrics is open. We do not know if the problem is FPT with k as parameter. One approach to solving GEOM-GI for a metric space (X, d) is to try and efficiently embed the given points sets A and B *isometrically* into a different metric space (X', d') for which we already know an efficient algorithm. For instance, known results about embedding metric spaces imply that there is a reduction from l_1^k-GEOM-GI to $l_\infty^{2^k}$-GEOM-GI in time $2^k \cdot poly(k, n, M)$, where l_p^k denotes the l_p metric on $\mathbb{Q}^k$. We do not know of a reduction that avoids the blow-up from k to 2^k in dimension.

Chapter 4

The Power of Color-Refinement

The well-known *color refinement* (also known as *naive vertex classification*) procedure for Graph Isomorphism works as follows: it begins with a uniform coloring of the vertices of two graphs G and H and refines the vertex coloring step by step. In a refinement step, if two vertices have identical colors but differently colored neighborhoods (with the multiplicities of colors counted), then these vertices get new different colors. The procedure terminates when no further refinement of the vertex color classes is possible.

Upon termination, if the multisets of vertex colors in G and H are different, we can correctly conclude that they are not isomorphic. However, color refinement sometimes fails to distinguish non-isomorphic graphs. The simplest example is given by any two non-isomorphic regular graphs of the same degree with the same number of vertices. Nevertheless, color refinement turns out to be a useful tool not only in isomorphism testing but also in a number of other areas; see [67, 71, 80] and references there.

For which pairs of graphs G and H does the color refinement procedure succeed in solving Graph Isomorphism? We say that color refinement *succeeds* on a graph G if it distinguishes G from any non-isomorphic H. A graph on which color refinement succeeds is called *amenable*. There are interesting classes of amenable graphs:

1. An obvious class of amenable graphs is formed by *unigraphs*. Unigraphs are graphs that are determined up to isomorphism by their degree sequences.

2. Trees are known to be amenable (Edmonds [58, 86]).

3. It is easy to see that all graphs for which the color refinement procedure terminates with singleton color classes (i.e. with the discrete partition of the vertex set) are amenable. We call such graphs *discrete*. Babai, Erdös, and Selkow [52] have shown that a random graph $G_{n,1/2}$ is discrete with high probability (moreover, the discrete partition of $G_{n,1/2}$ is reached within at most two refinement steps). Thus, almost all graphs are amenable, which makes graph isomorphism efficiently solvable in the average case (see also [53]).

Which graphs are amenable? In this chapter, our aim is to determine the exact range of applicability of color refinement. We find an efficient characterization of the entire class of amenable graphs, which allows for a quasilinear-time test whether or not color refinement succeeds on a given graph. Formally, the main result of this chapter is the following

Theorem 20. *The class of amenable graphs is recognizable in time $O((n + m) \log n)$, where n and m denote the number of vertices and edges of the input graph.*

We note that a weak *a priori* upper bound for the complexity of recognizing amenable graphs is $\mathsf{coNP}^{\mathsf{GI}[1]}$, where the superscript means the one-query access to an oracle solving the Graph Isomorphism problem. To the best of our knowledge, no better upper bound was known before.

We also state the logical aspects of our result. A *counting quantifier* $\exists_m$ opens a sentence saying that there are at least m elements satisfying some property. Immerman and Lander [69] discovered an intimate connection between color refinement and 2-variable first-order logic with counting quantifiers. This connection implies that amenability of a graph

is equivalent to its definability in this logic, yielding the following corollary of Theorem 20.

Corollary 3. *The class of graphs definable in 2-variable first-order logic with counting quantifiers can be recognized in polynomial time.*

This chapter is organized as follows. In Section 4.1, we introduce necessary notation and definitions. In Sections 4.2 and 4.3, we develop a set of necessary conditions for a graph to be amenable. In Section 4.4, we show that these conditions are indeed sufficient for a graph to be amenable. This gives us a characterization of graphs which are amenable. In Section 4.5, we complete the proof of Theorem 20. We also mention some important consequences of our result.

Related work. A characterization of amenable graphs similar to that in the present chapter has been suggested independently by Kiefer, Schweitzer, and Selman [72]. Moreover, they obtain a generalization of this result to arbitrary relational structures (which includes, in particular, directed graphs, while our treatment is focused on undirected graphs).

4.1 Preliminaries

The vertex set of a graph G is denoted by $V(G)$. The vertices adjacent to a vertex $u \in V(G)$ form its neighborhood $N(u)$. A set of vertices $X \subseteq V(G)$ induces a subgraph of G, that is denoted by $G[X]$. For two disjoint subsets of $V(G)$, X and Y, $G[X, Y]$ denotes the bipartite subgraphs of G on vertex sets X and Y where edge set is $\{\{x, y\} \in E(G) \mid x \in X, y \in Y\}$. The vertex-disjoint union of graphs G and H will be denoted by $G + H$. Furthermore, we write mG for the disjoint union of m copies of G. Recall that the *complement* of a graph G has the same vertex set and exactly those edges that are absent in G. The *bipartite complement* of a bipartite graph G with vertex classes X and Y is the bipartite graph G'

with the same vertex classes such that $\{x, y\}$ with $x \in X$ and $y \in Y$ is an edge in G' if and only if it is not an edge in G. We use the standard notation K_n for the complete graph on n vertices, $K_{s,t}$ for the complete bipartite graph whose vertex classes have s and t vertices, and C_n for the cycle on n vertices.

For technical convenience, we will consider graphs to be vertex-colored. A *vertex-colored graph* is an undirected simple graph G endowed with a vertex coloring $c : V(G) \to \{1, \ldots, k\}$. Automorphisms of a vertex-colored graph and isomorphisms between vertex-colored graphs are required to preserve vertex colors.

Given a graph G, the *color-refinement* algorithm (to be abbreviated as *CR*) iteratively computes a sequence of colorings C^i of $V(G)$. The initial coloring C^0 is the vertex coloring of G, i.e., $C^0(u) = c(u)$. Then,

$$C^{i+1}(u) = \left(C^i(u), \left\{C^i(a) : a \in N(u)\right\}\right), \tag{4.1}$$

where $\{\ldots\}$ denotes a multiset.

The partition $\mathcal{P}^{i+1}$ of $V(G)$ into the color classes of C^{i+1} is a refinement of the partition $\mathcal{P}^i$ corresponding to C^i. It follows that, eventually, $\mathcal{P}^{s+1} = \mathcal{P}^s$ for some s; hence, $\mathcal{P}^i = \mathcal{P}^s$ for all $i \geq s$. The partition $\mathcal{P}^s$ is called the *stable partition* of G and denoted by $\mathcal{P}_G$.

Given a partition $\mathcal{P}$ of the vertex set of a graph G, we call its elements *cells*. We call $\mathcal{P}$ *equitable* if:

(i) Each cell $X \in \mathcal{P}$ is monochromatic, i.e., all vertices $u, v \in X$ have the same color $c(u) = c(v)$.

(ii) For any cell $X \in \mathcal{P}$ the graph $G[X]$ induced by X is *regular*, that is, all vertices in $G[X]$ have equal degrees.

(iii) For any two cells $X, Y \in \mathcal{P}$ the bipartite graph $G[X, Y]$ induced by X and Y is *biregular*, that is, all vertices in X have equally many neighbors in Y and vice versa.

It is easy to see that the stable partition of G is equitable; our analysis in the next section will make use of this fact.

A straightforward inductive argument shows that the colorings C^i are preserved under isomorphisms.

Lemma 6. *If ϕ is an isomorphism from G to H, then $C^i(u) = C^i(\phi(u))$ for any vertex u of G.*

Lemma 6 readily implies that, if graphs G and H are isomorphic, then

$$\left\{C^i(u) : u \in V(G)\right\} = \left\{C^i(v) : v \in V(H)\right\} \tag{4.2}$$

for all $i \geq 0$.

When used for isomorphism testing, the CR algorithm accepts two graphs G and H as isomorphic exactly when the above condition is met on input $G + H$. Note that this condition is actually finitary: If Equality (4.2) is false for some i, it must be false for some $i < 2n$, where n denotes the number of vertices in each of the graphs. This follows from the observation that the partition $\mathcal{P}^{2n-1}$ induced by the coloring C^{2n-1} must be the stable partition of the disjoint union of G and H. In fact, Equality (4.2) holds true for all i if it is true for $i = n$; see, e.g., [75]. Thus, it is enough that CR verifies (4.2) for $i = n$.

Note that computing the vertex colors literally according to (4.1) would lead to an exponential growth of the lengths of color names. This can be avoided by renaming the colors after each refinement step. Then CR never needs more than n color names (appearance of more than n colors is an indication that the graphs are non-isomorphic).

Definition 4. *We call a graph G* amenable *if for every graph H, procedure CR works correctly on the input pair G and H. That is, Equality (4.2) is false for $i = n$ whenever $H \ncong G$.*

4.2 Local Structure of Amenable Graphs

In this section, we state and prove certain necessary conditions for a graph to be amenable. These conditions will unravel the *local* structure of an amenable graph in the following sense. Consider the stable partition $\mathcal{P}_G$ of an amenable graph G. Our conditions will restrict the possible regular and biregular graphs that can occur, respectively, as $G[X]$ and $G[X, Y]$ for cells X, Y of $\mathcal{P}_G$. Formally, the local conditions are stated in the following

Lemma 7. *The stable partition $\mathcal{P}_G$ of an amenable graph G fulfills the following properties:*

(A) *For any cell $X \in \mathcal{P}_G$, $G[X]$ is an empty graph, a complete graph, a matching graph mK_2, the complement of a matching graph, or the 5-cycle;*

(B) *For any two cells $X, Y \in \mathcal{P}_G$, $G[X, Y]$ is an empty graph, a complete bipartite graph, a disjoint union of $|X| = s$ stars $K_{1,t}$ where the s centers constitute X and st leaves constitute Y, or the bipartite complement of the last graph.*

The proof of Lemma 7 is based on the following facts. We will require the following definition. Graphs which are determined by their degree sequence are called *unigraphs*.

Lemma 8 (Johnson [70])**.** *A regular graph of degree d with n vertices is a unigraph if and only if $d \in \{0, 1, n-2, n-1\}$ or $d = 2$ and $n = 5$.*[1]

Lemma 9 (Koren [74])**.** *A bipartite graph is determined up to isomorphism by the conditions that each of the m vertices in one part has degree c and each of the n vertices in the other part has degree d if and only if $c \in \{0, 1, n-1, n\}$ or $d \in \{0, 1, m-1, m\}$.*

If G contains a subgraph $G[X]$ or $G[X, Y]$ that is induced by some $X, Y \in \mathcal{P}_G$ but not listed in Lemma 7, then Lemmas 8 and 9 imply that this subgraph can be replaced by a non-isomorphic regular or biregular graph with the same parameters. Hence, in order to

[1] The last case, in which the graph is the 5-cycle, is missing from the statement of this result in [70, Theorem 2.12]. The proof in [70] tacitly considers only graphs with at least 6 vertices.

prove Lemma 7 it suffices to show that the resulting graph H is indistinguishable from G by color refinement. The graphs G and H in the following lemma have the same vertex set. Given a vertex u, we distinguish its neighborhoods $N_G(u)$ and $N_H(u)$ and its colors $C^i_G(u)$ and $C^i_H(u)$ in the two graphs.

Lemma 10. *Let X and Y be cells of the stable partition of a graph G.*

(i) If H is obtained from G by replacing the edges of the subgraph $G[X]$ with the edges of an arbitrary regular graph of the same degree on the same vertex set X, then $C^i_G(u) = C^i_H(u)$ for any $u \in V(G)$ and any i.

(ii) If H is obtained from G by replacing the edges of the subgraph $G[X, Y]$ with the edges of an arbitrary biregular graph with the same vertex partition such that the vertex degrees remain unchanged, then $C^i_G(u) = C^i_H(u)$ for any $u \in V(G)$ and any i.

Proof. We proceed by induction on i. In the base case of $i = 0$ the claim is trivially true. Assume that $C^i_G(a) = C^i_H(a)$ for all $a \in V(G)$. We consider an arbitrary vertex u and prove that

$$C^{i+1}_G(u) = C^{i+1}_H(u). \tag{4.3}$$

From now on we treat Parts (i) and (ii) separately.

(i) Suppose first that $u \notin X$. Since the transformation of G into H does not affect the edges emanating from u, we have $N_G(u) = N_H(u)$. Looking at the definition (4.1), we immediately derive (4.3) from the induction assumption.

If $u \in X$, we only have the equality $N_G(u) \setminus X = N_H(u) \setminus X$, implying that

$$\left\{C^i_G(a) : a \in N_G(u) \setminus X\right\} = \left\{C^i_H(a) : a \in N_H(u) \setminus X\right\}. \tag{4.4}$$

The equality $N_G(u) \cap X = N_H(u) \cap X$ is not necessarily true. However, u has equally many neighbors from X in G and in H. Furthermore, for any two vertices a and a' in X we have $C^i_G(a) = C^i_G(a')$ because X is a cell of G, and $C^i_H(a) = C^i_G(a) = C^i_G(a') = C^i_H(a')$ by the

induction assumption. That is, all vertices in X have the same C^i-color both in G and in H. It follows that

$$\left\{C^i_G(a) : a \in N_G(u) \cap X\right\} = \left\{C^i_H(a) : a \in N_H(u) \cap X\right\}. \tag{4.5}$$

Combining (4.4) and (4.5), we conclude that (4.3) holds in any case.

(ii) If $u \notin X \cup Y$, we have $N_G(u) = N_H(u)$ and Equality (4.3) readily follows from the induction assumption.

Suppose that $u \in Y$. In this case we still have (4.4) and, exactly as in Part (i), we also derive (4.5). Equality (4.3) follows. The case of $u \in X$ is symmetric. □

We are now ready to finish the proof of Lemma 7.

Proof of Lemma 7. **(A)** If $G[X]$ is a graph not from the list, by Lemma 8, it is not a unigraph. Hence, we can modify G locally on X by replacing $G[X]$ with a non-isomorphic regular graph with the same parameters. Part (i) of Lemma 10 implies that the resulting graph H satisfies Equality (4.2) for any i, implying that CR does not distinguish between G and H. The graphs G and H are non-isomorphic because, by Part (i) of Lemma 10 and by Lemma 6, an isomorphism from G to H would induce an isomorphism from $G[X]$ to $H[X]$. This shows that G is not amenable.

(B) This condition follows, similarly to Condition **A**, from Lemma 9 and Part (ii) of Lemma 10.

□

4.3 Global Structure of Amenable Graphs

In this section, we continue to state and prove certain necessary conditions for a graph to be amenable. These conditions will unravel the *global* structure of an amenable graph in the following sense. From the previous section, we already have certain local conditions on the cell graph of an amenable graph. Let us proceed with a formal description of our conditions.

Consider the stable partition $\mathcal{P}_G$ of an amenable graph G. Recall that $\mathcal{P}_G$ is the stable partition of the vertex set of a graph G, and that elements of $\mathcal{P}_G$ are called cells. We define the auxiliary *cell graph* $C(G)$ of G to be the complete graph on the vertex set $\mathcal{P}_G$ with the following labeling of vertices and edges. A vertex X of $C(G)$ is designated *homogeneous* if the graph $G[X]$ is complete or empty and *heterogeneous* otherwise. An edge $\{X, Y\}$ of $C(G)$ is designated *isotropic* if the bipartite graph $G[X, Y]$ is either complete or empty and *anisotropic* otherwise. A path $X_1X_2 \dots X_l$ in $C(G)$ where every edge $\{X_i, X_{i+1}\}$ is anisotropic will be referred to as an *anisotropic path*. If also $\{X_l, X_1\}$ is an anisotropic edge, we speak of an *anisotropic cycle*. In the case that $|X_1| = |X_2| = \dots = |X_l|$, such a path (or cycle) is called *uniform*.

For graphs fulfilling Conditions **A** and **B** of Lemma 7 we refine the labeling of the vertices and edges of $C(G)$ as follows. A heterogeneous cell $X \in \mathcal{P}_G$ is called *matching*, *co-matching*, or *pentagonal* depending on the type of $G[X]$. Note that a matching or co-matching cell X always consists of at least 4 vertices. Further, an anisotropic edge $\{X, Y\}$ is called *constellation* if $G[X, Y]$ is a disjoint union of stars, and *co-constellation* otherwise (in the latter case, the bipartite complement of $G[X, Y]$ is a disjoint union of stars). Likewise, homogeneous cells X (and isotropic edges $\{X, Y\}$) are called *empty* if the graph $G[X]$ (resp. $G[X, Y]$) is empty, and *complete* otherwise.

Note that if an edge $\{X, Y\}$ of a uniform path or cycle is a constellation, then $G[X, Y]$ is a matching graph.

The following lemma states some necessary conditions on the cell graph of an amenable graph. Figure 4.1 gives a schematic illustration of these necessary conditions.

Lemma 11. *The cell graph $C(G)$ of an amenable graph G has the following properties:*

(C) *$C(G)$ contains no uniform anisotropic path connecting two heterogeneous cells;*

(D) *$C(G)$ contains no uniform anisotropic cycle;*

(E) *$C(G)$ contains neither an anisotropic path $XY_1 \dots Y_l Z$ such that $|X| < |Y_1| = \dots = |Y_l| > |Z|$ nor an anisotropic cycle $XY_1 \dots Y_l X$ such that $|X| < |Y_1| = \dots = |Y_l|$;*

(F) *$C(G)$ contains no anisotropic path $XY_1 \dots Y_l$ such that $|X| < |Y_1| = \dots = |Y_l|$ and the cell Y_l is heterogeneous.*

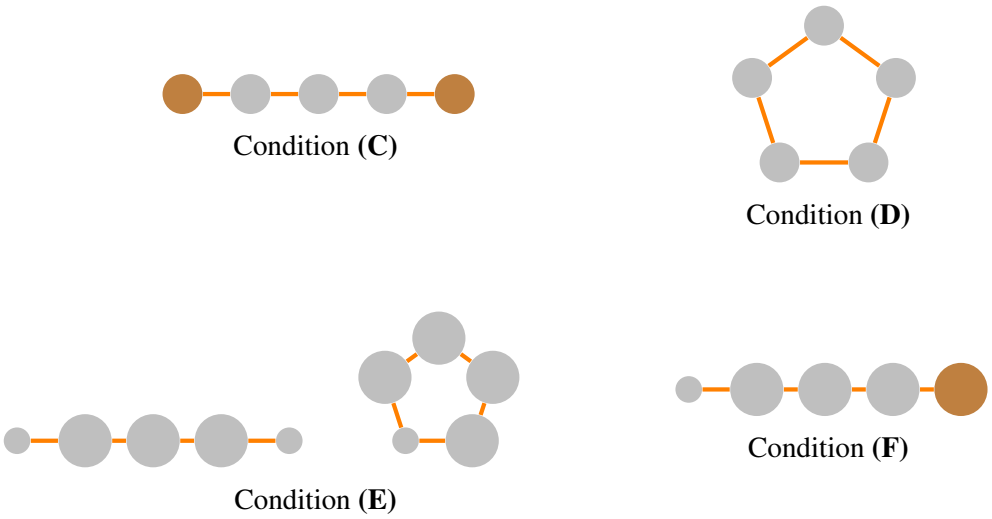

Figure 4.1. Conditions **(C)-(F)** on the cells (homogeneous ● and heterogeneous ●) and the anisotropic edges ——.

Proof. **(C)** Figure 4.2 gives a graphical description of this condition. Suppose that P is a uniform anisotropic path in $C(G)$ connecting two heterogeneous cells X and Y. Let $k = |X| = |Y|$. Complementing $G[A, B]$ for each co-constellation edge $\{A, B\}$ of P, in G we obtain k vertex-disjoint paths connecting X and Y. These paths determine a one-to-one correspondence between X and Y. Given $v \in X$, denote its mate in Y by v^*. Call P

conducting if this correspondence is an isomorphism between $G[X]$ and $G[Y]$, that is, two vertices u and v in X are adjacent exactly when their mates u^* and v^* are adjacent. We also call P *conducting* if one of X and Y is a matching and the other is a co-matching, and additionally the correspondence is an isomorphism between $G[X]$ and the complement of $G[Y]$.

We construct a non-isomorphic graph H such that CR does not distinguish between G and H. Since X and Y are heterogeneous, we can replace the edges of the subgraph $G[X]$ with the edges of an isomorphic but different graph on the same vertex set X such that P is a conducting path in the resulting graph H if and only if P is a non-conducting path in G. Now, Part (i) of Lemma 10 implies that CR computes the same coloring for G and H and does not distinguish between them. On the other hand, Lemma 6 implies that any isomorphism ϕ between G and H must map each cell to itself. Since $\phi(v^*) = \phi(v)^*$, ϕ must also preserve the conducting property along the path P. It follows that G and H are not isomorphic. Hence, G is not amenable.

(D) Suppose that $C(G)$ contains a uniform anisotropic cycle Q of length m. All cells in Q have the same cardinality, say k. Complementing $G[A, B]$ for each co-constellation edge $\{A, B\}$ of Q, in G we obtain the vertex-disjoint union of cycles whose lengths are multiples of m. As two extreme cases, we can have k cycles of length m each or we can have a single cycle of length km. Denote the isomorphism type of this union of cycles by $\tau(Q)$. Note that this type is isomorphism invariant: For an isomorphism ϕ from G to

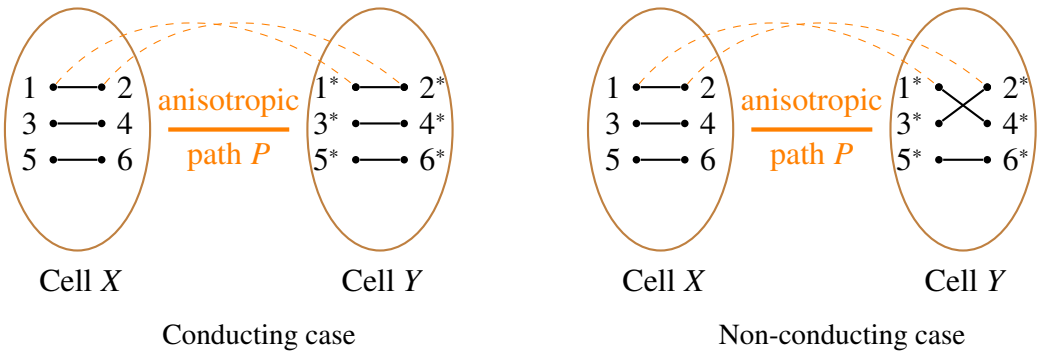

Figure 4.2. Conducting and non-conducting anisotropic paths

another graph H, $\tau(\phi'(Q)) = \tau(Q)$ for the induced isomorphism ϕ' from $C(G)$ to $C(H)$.

Let X and Y be two consecutive cells in Q. We can replace the subgraph $G[X, Y]$ with an isomorphic but different bipartite graph so that in the resulting graph H, $\tau(Q)$ becomes either kC_m or C_{km}, whatever we wish. In particular, we can replace the subgraph $G[X, Y]$ in such a way that $\tau(Q)$ is changed.

Similarly as for Condition **C**, we use Part (ii) of Lemma 10 to argue that CR does not distinguish between G and H. Furthermore, $G \ncong H$ because the types $\tau(Q)$ in G and H are different. Therefore, G is not amenable.

(E) Suppose that $C(G)$ contains an anisotropic path $P = XY_1 \dots Y_l Z$ such that $|X| < |Y_1| = \dots = |Y_l| > |Z|$ (for the case of a cycle, where $Z = X$, the argument is virtually the same). Let $G[X, Y_1] = sK_{1,t}$ and $G[Z, Y_l] = aK_{1,b}$, where $s, a, t, b \geq 2$ (if any of these subgraphs is a co-constellation, we consider its complement). Thus, $|X| = s$, $|Z| = a$, and $|Y_1| = |Y_l| = st = ab$.

Like in the proof of Condition **C**, the uniform anisotropic path $Y_1 \dots Y_l$ determines a one-to-one correspondence between the cells Y_1 and Y_l that can be used to make the identification $Y_1 = Y_l = \{1, 2, \dots, st\} = Y$. For each $x \in X$, let Y_x denote the set of vertices in Y adjacent to x. The set Y_z is defined similarly for each $z \in Z$. Note that for any $x \neq x'$ in X and $z \neq z'$ in Z,

$$|Y_x| = t, \;\; |Y_z| = b, \;\; Y_x \cap Y_{x'} = \emptyset, \;\; \text{and} \;\; Y_z \cap Y_{z'} = \emptyset.$$

We regard $\mathcal{Y}_G = \{Y_x\}_{x \in X} \cup \{Y_z\}_{z \in Z}$ as a hypergraph on the vertex set Y. Note that $\mathcal{Y}_G$ has multiple hyperedges if $Y_x = Y_z$ for some x and z. Without loss of generality, we can assume that the hyperedges Y_z, $z \in Z$, form consecutive intervals in Y. We call the anisotropic path P *flat*, if there exists no pair $(x, z) \in X \times Z$ such that one of the two hyperedges Y_x and Y_z is contained in the other.

We construct a non-isomorphic graph H such that CR does not distinguish between G

and H. If P is flat in G, we replace the edges of the subgraph $G[X, Y_1]$ by the edges of an isomorphic but different biregular graph such that P becomes non-flat in the resulting graph H. More precisely, we replace the edges in such a way that all hyperedges of $\mathcal{Y}_H$ form consecutive intervals in Y by letting $\mathcal{Y}_H = \{Y_i\}_{i\in[s]} \cup \{Y_z\}_{z\in Z}$, where $Y_i = \{(i-1)t + 1, \ldots, it\}$. Likewise, if P is non-flat in G, we replace the edges of $G[X, Y_1]$ such that P becomes flat in H by letting $Y_i = \{i, i + s, \ldots, i + (t-1)s\}$.

Now, Part (i) of Lemma 10 implies that CR computes the same coloring for G and H and does not distinguish between them. On the other hand, Lemma 6 implies that any isomorphism ϕ between G and H must map each cell to itself. As ϕ must also preserve the flatness property of the path P, it follows that G and H are not isomorphic. Hence, G is not amenable.

(F) Suppose that $C(G)$ contains an anisotropic path $XY_1 \ldots Y_l$ where $|X| < |Y_1| = \ldots = |Y_l|$ and Y_l is heterogeneous. Let $G[X, Y_1] = sK_{1,t}$ (in the case of a co-constellation, we consider the complement). Since $s, t \geq 2$ and $|Y_1| = st$, the cell Y_l cannot be pentagonal. Considering the complement if needed, we can assume without loss of generality that Y_l is matching. Like in the proof of Condition **E**, the uniform anisotropic path $Y_1 \ldots Y_l$ determines a one-to-one correspondence between the cells Y_1 and Y_l that can be used to make the identification $Y_1 = Y_l = \{1, 2, \ldots, st\} = Y$. Consider the hypergraph $\mathcal{Y}_G = \{Y_x\}_{x\in X} \cup \mathcal{E}$, where $Y_x = N(x) \cap Y_1$ and $\mathcal{E}$ consists of the pairs of adjacent vertices in $G[Y_l]$. Now, exactly as in the proof of Condition **E**, we can change the isomorphism type of $\mathcal{Y}_G$ by replacing the edges of the subgraph $G[X, Y_1]$ by the edges of an isomorphic biregular graph. This yields a non-isomorphic graph H that is indistinguishable from G by CR. □

4.4 Proof of Theorem 20

In this section, we finish the proof of Theorem 20. First, we establish that Conditions **A–F** are not only necessary for amenability (as shown in Lemmas 7 and 11) but also sufficient. Second, we use this characterization to obtain an efficient algorithm for amenability testing.

As a preparation we first prove the following Lemma 12 that reveals a tree-like structure of amenable graphs. By an *anisotropic component* of the cell graph $C(G)$ we mean a maximal connected subgraph of $C(G)$ whose edges are all anisotropic. Note that if a vertex of $C(G)$ has no incident anisotropic edges, it forms a single-vertex anisotropic component.

Lemma 12. *Suppose that a graph G satisfies Conditions* **A–F**. *The following conditions hold.*

(G) *For any anisotropic component A of $C(G)$, A is a tree with the following monotonicity property. Let R be a cell in A of minimum cardinality and let A_R be the rooted directed tree obtained from A by rooting A at R. Then $|X| \le |Y|$ for any directed edge (X, Y) of A_R.*

(H) *For any anisotropic component A of $C(G)$, A contains at most one heterogeneous vertex. If R is such a vertex, it has minimum cardinality among the cells of A.*

Proof. **(G)** A cannot contain any uniform cycle by Condition **D** and any other cycle by Condition **E**. The monotonicity property follows from Condition **E**.

(H) Assume that A contains more than one heterogeneous cell. Consider two such cells S and T. Let $S = Z_1, Z_2, \ldots, Z_l = T$ be the path from S to T in A. The monotonicity property stated in Condition **G** implies that there is j (possibly $j = 1, l$) such that $|Z_1| \ge \ldots \ge |Z_j| \le \ldots \le |Z_l|$. Since the path cannot be uniform by Condition **C**, at least one of the inequalities is strict. However, this contradicts Condition **F**.

Suppose that R is a heterogeneous cell in A. Consider now a path $R = Z_1, Z_2, \ldots, Z_l = S$ in A where S is a cell with the smallest cardinality. By the monotonicity property and Condition **F**, this path must be uniform, proving that $|R| = |S|$. □

In combination with Conditions **A** and **B**, Conditions **G** and **H** on anisotropic components give a very stringent characterization of amenability.

Theorem 21. *For a graph G the following conditions are equivalent:*

(i) *G is amenable.*

(ii) *G satisfies Conditions* **A–F**.

(iii) *G satisfies Conditions* **A, B, G** *and* ***H***.

Proof. It only remains to show that any graph G fulfilling the Conditions **A, B, G** and **H** is amenable. Let H be a graph indistinguishable from G by CR. Then we have to show that G and H are isomorphic.

Consider the coloring C^s corresponding to the stable partition $\mathcal{P}^s$ of the disjoint union $G + H$. Since G and H satisfy Equality (4.2) for $i = s$, there is a bijection $f : \mathcal{P}_G \to \mathcal{P}_H$ matching each cell X of the stable partition of G to the cell $f(X) \in \mathcal{P}_H$ such that the vertices in X and $f(X)$ have the same C^s-color. Moreover, Equality (4.2) implies that $|X| = |f(X)|$. We claim that for any cells X and Y of G,

(a) $G[X] \cong H[f(X)]$ and

(b) $G[X, Y] \cong H[f(X), f(Y)]$,

implying that f is an isomorphism from $C(G)$ to $C(H)$.

Indeed, since X and $f(X)$ are cells of the stable partitions $\mathcal{P}_G$ and $\mathcal{P}_H$, both $G[X]$ and $H[f(X)]$ are regular. Since $X \cup f(X)$ is a cell of the stable partition $\mathcal{P}^s$ of $G + H$, the

graphs $G[X]$ and $H[f(X)]$ have the same degree. By Condition **A**, $G[X]$ is a unigraph, implying Property (a). Property (b) follows from Condition **B** by a similar argument.

We now construct an isomorphism ϕ from G to H. By Lemma 6, we should have $\phi(X) = f(X)$ for each cell X. Therefore, we have to define the map $\phi : X \to f(X)$ on each X.

By Condition **H**, an anisotropic component A of the cell graph $C(G)$ contains at most one heterogeneous cell. Denote it by R_A if it exists. Otherwise fix R_A to be an arbitrary cell of the minimum cardinality in A.

For each A, define ϕ on $R = R_A$ to be an arbitrary isomorphism from $G[R]$ to $H[f(R)]$, which exists according to (a). After this, propagate ϕ to any other cell in A as follows. By Condition **G**, A is a tree. Let A_R be the directed rooted tree obtained from A by rooting it at R. Suppose that ϕ is already defined on X and (X, Y) is an edge in A. By the monotonicity property in Condition **G** and our choice of R, we can assume that $|X| \le |Y|$. Then ϕ can be extended to Y so that this is an isomorphism from $G[X, Y]$ to $H[f(X), f(Y)]$. This is possible by (b) due to the fact that all vertices in Y have degree 1 in $G[X, Y]$ or its bipartite complement (and the same holds for all vertices in $f(Y)$ in the graph $H[f(X), f(Y)]$).

It remains to argue that the map ϕ obtained in this way is indeed an isomorphism from G to H. It suffices to show that ϕ is an isomorphism between $G[X]$ and $H[f(X)]$ for each cell X of G and between $G[X, Y]$ and $H[f(X), f(Y)]$ for each pair of cells X and Y.

If X is homogeneous, $f(X)$ is homogeneous of the same type, complete or empty, according to (a). In this case, any ϕ is an isomorphism from $G[X]$ to $H[f(X)]$. If X is heterogeneous, the assumption of the lemma says that it belongs to a unique anisotropic component A (and $X = R_A$). Then ϕ is an isomorphism from $G[X]$ to $H[f(X)]$ by construction.

If $\{X, Y\}$ is an isotropic edge of $C(G)$, then (b) implies that $\{f(X), f(Y)\}$ is an isotropic edge of $C(H)$ of the same type, complete or empty. In this case, ϕ is an isomorphism from $G[X, Y]$ to $H[f(X), f(Y)]$, no matter how it is defined. If $\{X, Y\}$ is anisotropic, it belongs to

some anisotropic component A, and ϕ is an isomorphism from $G[X, Y]$ to $H[f(X), f(Y)]$ by construction. □

We are now ready to finish the proof of Theorem 20. Our characterization of amenable graphs via Conditions **A**, **B**, **G** and **H** leads to an efficient test for amenability of a given graph, that has the same time complexity as CR. It is known (Cardon and Crochemore [60]; see also [54]) that the stable partition of a given graph G can be computed in time $O((n + m) \log n)$. It is supposed that G is presented by its adjacency list.

Proof of Theorem 20. Using known algorithms, we first compute the stable partition $\mathcal{P}_G = \{X_1, \ldots, X_k\}$ of the input graph G. Let $C^*(G)$ be the version of the cell graph $C(G)$ where all isotropic edges are removed. We check if the graph G satisfies Conditions **A**,**B**, **G** and **H** as follows.

- Condition **A, B**: Using the representations of G and $\mathcal{P}_G$, compute the adjacency list representation of $C^*(G)$ as follows.
  - Given a vertex X_i of $C^*(G)$, pick a vertex $u \in X_i$ of G.
  - Traverse the adjacency list of u and list all the cells X_j such that there is a vertex $v \in X_j$ adjacent to u. Also, count the number d_{ij} of vertices in X_j which occur in the adjacency list of u.

  Knowing the numbers $|X_i|$ and d_{ij} for every edge of $C^*(G)$, we can check Conditions **A** and **B** of Lemma 7.

- Condition **G,H**: Do a BFS on $C^*(G)$.
  - if there are cycles in $C^*(G)$, output non-amenable.
  - Otherwise, for each acyclic component of $C^*(G)$, find the number N_A of heterogeneous cells.
    - If $N_A > 1$, output the graph to be non-amenable.

* If $N_A = 1$, check if it has minimum cardinality. If not, output non-amenable. Otherwise, restart the BFS search in $C^*(G)$ from this cell. Check if the monotonicity property of Condition **G** is fulfilled for each forward edge of the resulting search tree.

* If $N_A = 0$, restart the BFS search in $C^*(G)$ from an arbitrary cell in A having minimum cardinality. Check if the monotonicity property of Condition **G** is fulfilled for each forward edge of the resulting search tree.

This clearly suffices to check Conditions **G, H**.

□

4.5 Examples and Applications

Theorem 21 is a convenient tool for verifying amenability. For example, amenability of discrete graphs is a well-known fact. Recall that those are graphs whose stable partitions consist of singletons. Since the cell graph has no anisotropic edge in this case, any anisotropic component of a discrete graph consists of a single cell. Hence, Conditions **A** and **B** as well as Conditions **G** and **H** on anisotropic components are fulfilled by trivial reasons.

Checking these four conditions, we can also reprove the amenability of trees. Moreover, our argument extends to the class of forests. The amenability of forests follows also from [78, Theorem 2.5]; here we give an alternative proof to illustrate applicability of our criterion. Note that the extension to forests is not a straightforward fact because the class of amenable graphs is not closed under disjoint unions (for example, $C_3 + C_4$ is indistinguishable by CR from C_7 and, hence, is not amenable).

Corollary 4. *All forests are amenable.*

Proof. A regular acyclic graph is either an empty or a matching graph. This implies Condition **A**. Condition **B** follows from the observation that biregular acyclic graphs are either empty or disjoint unions of stars.

Let $C^*(G)$ be the version of the cell graph $C(G)$ where all empty edges are removed. If $C^*(G)$ contains a cycle, G must contain a cycle as well. Therefore, if G is acyclic, then $C^*(G)$ is acyclic too, and any anisotropic component of $C(G)$ must be a tree. To prove the monotonicity property in Condition **G**, it suffices to show that $C(G)$ cannot contain an anisotropic path $XY_1 \dots Y_l Z$ with $|X| < |Y_1| = \dots = |Y_l| > |Z|$. But this easily follows since in this case each vertex of the induced subgraph $G[X \cup Y_1 \cup \dots \cup Y_l \cup Z]$ has degree at least 2 in G, contradicting the acyclicity of G.

To prove Condition **H**, suppose that $C(G)$ contains an anisotropic path $X_0, X_1, \dots, X_l$ connecting two heterogeneous cells X_0 and X_l. Then each vertex of the induced subgraph $G[X_0 \cup X_1 \cup \dots \cup X_{l-1} \cup X_l]$ has degree at least 2 in G, a contradiction. The same contradiction arises if such a path connects a heterogeneous cell X_0 with an arbitrary cell X_l, where $|X_l| < |X_{l-1}|$. Hence, X_0 must have minimum cardinality among all cells belonging to the same anisotropic component. □

The closure properties of amenable graphs under disjoint unions admit a combinatorial characterization in terms of covering graphs. The relationship of the last concept to CR was first noticed by Angluin in [48], where it was used in the area of distributed computations. A surjective homomorphism from a graph K to a graph G is a *covering map* if its restriction to the neighborhood of each vertex in K is bijective. If there is a covering map from K to G (in other terms, K *covers* G), then K is called a *covering graph* of G. Restricting these notions to connected graphs, we say that a graph U is a *universal cover* of a graph G if U covers every covering graph of G. A universal cover $U = U_G$ of G is unique up to isomorphism. Alternatively, U_G can be defined as a tree that covers G. If G is itself a tree, then $U_G \cong G$; otherwise the tree U_G is infinite. Two graphs G and H have a common covering graph if and only if $U_G \cong U_H$.

A straightforward inductive argument shows that a covering map α from K to G preserves the coloring produced by CR, that is, $C^i(u) = C^i(\alpha(u))$ for all i, where C^i is defined by (4.1). This generalizes Lemma 6. It follows that, if G and H have a common covering graph, then

$$\left\{C^i(u) : u \in V(G)\right\} = \left\{C^i(v) : v \in V(H)\right\} \text{ for all } i. \tag{4.6}$$

On the other hand, Equality (4.6) is false for all $i \geq n$ if G and H have no common cover and each of the graphs has at most n vertices. This justifies the use of CR by Angluin [48] for deciding if two given graphs have a common cover. Moreover, in the last case we have (see [75])

$$\left\{C^{2n}(u) : u \in V(G)\right\} \cap \left\{C^{2n}(v) : v \in V(H)\right\} = \emptyset. \tag{4.7}$$

Corollary 5. *(i) Suppose that G and H are connected amenable graphs. Then $G + H$ is amenable if and only if G is a tree or $U_G \ncong U_H$.*

(ii) Suppose, in addition, that G and H have an equal number of vertices. Then $G + H$ is amenable if and only if G is a tree or $G \ncong H$.

Proof. (i) We split the proof into two cases depending on whether or not $U_G \cong U_H$, that is, whether or not G and H have a common covering graph.

Suppose first that $U_G \ncong U_H$. Equality (4.7) implies that $\mathcal{P}_{G+H} = \mathcal{P}_G \cup \mathcal{P}_H$, that is, the stable partition of $G + H$ consists of the cells of the stable partitions of G and H. Since G and H are amenable, their cells fulfil Condition **A**. Therefore, Condition **A** holds true also for $G + H$. Since the cell graph $C(G + H)$ is obtained from $C(G)$ and $C(H)$ by joining each cell of G with each cell of H by an anisotropic (empty) edge, also Condition **B** is fulfilled for $G + H$. Moreover, every anisotropic component of $C(G + H)$ is an anisotropic component either of $C(G)$ or of $C(H)$. It follows that $G + H$ also satisfies Conditions **G** and **H** and, therefore, $G + H$ is amenable by Theorem 21.

Consider now the case that $U_G \cong U_H$. If G is a tree, then $H \cong G$, and $G + H$ is amenable by Corollary 4. Suppose that G is not a tree. We have to show that $G + H$ is not amenable.

Equality (4.6) implies that there is a one-to-one correspondence between the cells in $\mathcal{P}_G$ and $\mathcal{P}_H$ such that the vertices in the corresponding cells always have the same colors in the course of the CR procedure. It follows that $\mathcal{P}_{G+H} = \{X \cup X' : X \in \mathcal{P}_G\}$, where $X' \in \mathcal{P}_H$ denotes the counterpart of a cell $X \in \mathcal{P}_G$.

Using the existence of a cycle in G, we show that $G + H$ is not amenable by constructing a graph F such that F is connected (hence, non-isomorphic to $G + H$) and indistinguishable from $G + H$ by CR. Our construction of F is based on two pairs of vertices $u, v \in V(G)$ and $u', v' \in V(H)$ satisfying the following conditions:

(a) u and v are adjacent in G, and u' and v' are adjacent in H.

(b) The edge $\{u, v\}$ belongs to a cycle C in G.

(c) Let X and Y be the cells of $\mathcal{P}_G$ containing the vertices u and v respectively, and let X' and Y' be their counterparts in $\mathcal{P}_H$. Then $u' \in X'$ and $v' \in Y'$.

We obtain F from $G + H$ by switching the edges $\{u, v\}$ and $\{u', v'\}$ to $\{u, v'\}$ and $\{u', v\}$.

By Condition (b), the connectivity of the original connected component G is not broken. Any path in H via the missing edge $\{u', v'\}$ can be rerouted via new edges and a part of the cycle C. Since the two components become connected by new edges, the whole graph F is connected.

If $X = Y$, then u, v, u', and v' are in the same cell $X \cup X'$ of $\mathcal{P}_{G+H}$, and F is indistinguishable from $G + H$ by Part (i) of Lemma 10 (applied to the regular subgraph of $G + H$ induced by the cell $X \cup X'$). If $X \neq Y$, then u and u' are in the cell $X \cup X'$, and v and v' are in the cell $Y \cup Y'$ of $\mathcal{P}_{G+H}$. In this case, F is indistinguishable from $G + H$ by Part (ii) of Lemma 10 (applied to the biregular bipartite subgraph of $G + H$ induced by the cells $X \cup X'$ and $Y \cup Y'$).

To complete the proof, we have to secure u, v, u', and v' with the properties (a)–(c) above. Choose an edge $\{u, v\}$ of an arbitrary cycle in G. Let K be a common covering graph of G

and H, α be a covering map from K to G, and β be a covering map from K to H. Choose u'' to be an arbitrary vertex of K such that $\alpha(u'') = u$. Let v'' be the vertex determined by the conditions that $v'' \in N(u'')$ and $\alpha(v'') = v$. Finally, set $u' = \beta(u'')$ and $v' = \beta(v'')$. These vertices are adjacent because they are images of adjacent vertices u'' and v'' under a homomorphism. Since covering maps α and β preserve the colorings produced by CR, we have $C^i(u) = C^i(u'') = C^i(u')$ and $C^i(v) = C^i(v'') = C^i(v')$ for any i. This implies Condition (c).

(ii) We split the proof into two cases depending on whether or not $G \cong H$. Suppose first that $G \ncong H$. Since G and H are amenable, they are distinguishable by CR. We use the following fact [76]: Two graphs G and H with the same number of vertices are distinguishable by CR if and only if they have no common covering graph, i.e., $U_G \ncong U_H$. Now, the amenability of $G + H$ readily follows from Part (i).

If $G \cong H$, then $U_G \cong U_H$. It follows directly from Part (i) that $G + H$ is amenable if G is a tree and not amenable otherwise. □

Corollary 5 easily extends to disjoint unions of any number of connected amenable graphs $G_1, \ldots, G_k$. If there are two non-tree graphs G_i and G_j sharing a common cover, Part (i) readily implies that $G_1 + \cdots + G_k$ is not amenable. If there is no such pair G_i, G_j, then the argument for Part (i) shows that $G_1 + \cdots + G_k$ is amenable (here we also need the fact that, by Corollary 4, the forest part of $G_1 + \cdots + G_k$ is amenable).

Chapter 5

The Power of LP Approach to Graph Isomorphism

A well-known approach to tackling intractable optimization problems is to consider an appropriate linear programming relaxation. Consider a natural linear algebra formulation of Graph Isomorphism. Let G and H be two graphs on n vertices with adjacency matrices A and B, respectively. Then G and H are isomorphic if and only if there is an $n \times n$ permutation matrix X such that $AX = XB$.

A linear programming relaxation of this system of equations is to allow X to be a doubly stochastic matrix. If such an X exists, it is called a *fractional isomorphism* from G to H, and these graphs are said to be *fractionally isomorphic*. The following theorem of Ramana, Scheinerman and Ullman [78] shows surprisingly that this approach is equivalent to color-refinement approach.

Theorem 22 ([78])**.** *Two graphs are indistinguishable by color refinement if and only if they are fractionally isomorphic.*

The concept of a fractional isomorphism was used by Tinhofer in [83] as a basis for yet another linear-programming approach to isomorphism testing. Tinhofer calls a graph G

compact if the polytope of all its fractional automorphisms is integral; more precisely, if A is the adjacency matrix of G, then the polytope in $\mathbb{R}^{n^2}$ consisting of the doubly stochastic matrices X such that $AX = XA$ has only integral extreme points (i.e. all coordinates of these points are integers).

If a compact graph G is isomorphic to another graph H, then the polytope of fractional isomorphisms from G to H is also integral. If G is not isomorphic to H, then this polytope has *no* integral extreme point (and in fact no integral point at all). Thus, isomorphism testing for a compact graph G and an arbitrary graph H can be done in polynomial time by using linear programming to compute an extreme point of the polytope and testing if it is integral. Before testing isomorphism in this way, we need to know that G is compact. Unfortunately, no efficient characterization of compact graphs is currently known.

In this context, the main result of this chapter is the following

Theorem 23. *All amenable graphs are compact.*

In other words, amenbility is a sufficient condition for compactness. Moreover, this implies that Tinhofer's approach to Graph Isomorphism [83] has at least as large an applicability range as color refinement. More precisely, whenever the restriction of Graph Isomorphism to input graphs G and H such that G belongs to a class C is solvable by the latter approach, then it is also solvable by the former approach. In general, Tinhofer's approach is even more powerful than color refinement because it is known that the class of compact graphs contains many regular graphs (for example, all cycles [81]), for which color refinement cannot refine even the initial coloring.

In the second part of this chapter, we take a closer look at the relationship between the concepts of compactness and color refinement. Let us call a graph G *refinable* if the color partition produced by color refinement coincides with the orbit partition of the automorphism group of G. It is interesting to note that the color refinement procedure gives an efficient algorithm to check if a given refinable graph has a nontrivial automorphism.

It follows from the results in [83] that all compact graphs are refinable. The inclusion Amenable $\subset$ Compact, therefore, implies that all amenable graphs are refinable as well. The last result is independently obtained in [72] by a different argument.

Taking a finer look at the inclusion Compact $\subset$ Refinable, we discuss algorithmic and algebraic graph properties that were introduced by Tinhofer [83] and Godsil [64]. Along with the other graph classes under consideration, we show that the corresponding classes Tinhofer and Godsil form a hierarchy under inclusion. This is the second main result of this chapter.

Theorem 24. *The classes of graphs under consideration form the inclusion chain*

$$\mathsf{Discrete} \subset \mathsf{Amenable} \subset \mathsf{Compact} \subset \mathsf{Godsil} \subset \mathsf{Tinhofer} \subset \mathsf{Refinable}. \tag{5.1}$$

Moreover, all of the inclusions are strict.

Finally, we show that testing membership in each of the classes in the color-refinement hierarchy is P-hard.

Theorem 25. *The recognition problem of each of the classes in the hierarchy (5.1) is* P*-hard under uniform* AC^0 *many-one reductions.*

We prove the hardness of membership testing by giving a suitable uniform AC^0 many-one reduction from the P-complete monotone boolean circuit-value problem MCVP. More precisely, for a given MCVP instance (C, x) our reduction outputs a graph $G_{C,x}$ such that if $C(x) = 1$ then $G_{C,x}$ is discrete and if $C(x) = 0$ then $G_{C,x}$ is not refinable. In particular, the graph classes Discrete and Amenable are P-complete. We note that Grohe [66] established, for each $k \geq 2$, the P-completeness of the equivalence problem for first-order k-variable logic with counting quantifiers; according to [69], this implies the P-completeness of indistinguishability of two input graphs by color refinement. We adapt the gadget construction in [66], that goes back to Cai, Fürer, and Immerman [59], to show our P-hardness results.

This chapter is organized as follows. Section 5.1 contains the necessary definitions and results relevant to this chapter. In Section 5.2, we give a proof of Theorem 23. In Section 5.3, we give a proof of Theorem 24. In Section 5.4, we give a proof of Theorem 25. We conclude with a brief discussion about further directions in Section 5.5.

5.1 Preliminaries

An $n \times n$ real matrix X is *doubly stochastic* if its elements are nonnegative and all its rows and columns sum up to 1. Doubly stochastic matrices are closed under products and convex combinations. The set of all $n \times n$ doubly stochastic matrices forms the *Birkhoff polytope* $B_n \subset \mathbb{R}^{n^2}$. *Permutation matrices* are exactly 0-1 doubly stochastic matrices. By Birkhoff's Theorem (see, e.g. [57]), the $n!$ permutation matrices form precisely the set of all extreme points of B_n. Equivalently, every doubly stochastic matrix is a convex combination of permutation matrices.

Let G and H be graphs with vertex set $\{1, \ldots, n\}$. An isomorphism π from G to H can be represented by the permutation matrix $P_\pi = (p_{ij})$ such that $p_{ij} = 1$ if and only if $\pi(i) = j$. Denote the set of matrices P_π for all isomorphisms π by $\mathrm{Iso}(G, H)$, and let $\mathrm{Aut}(G) = \mathrm{Iso}(G, G)$.

Let A and B be the adjacency matrices of graphs G and H respectively. If the graphs are uncolored, a permutation matrix X is in $\mathrm{Iso}(G, H)$ if and only if $AX = XB$. For vertex-colored graphs, X must additionally satisfy the condition $X[u, v] = 0$ for all pairs of differently colored u and v, i.e., this matrix must be block-diagonal with respect to the color classes. We say that (vertex-colored) graphs G and H are *fractionally isomorphic* if $AX = XB$ for a doubly stochastic matrix X, where $X[u, v] = 0$ if u and v are of different colors. The matrix X is called a *fractional isomorphism*.

Denote the set of all fractional isomorphisms from G to H by $S(G, H)$ and note that it forms a polytope in $\mathbb{R}^{n^2}$. The set of isomorphisms $\mathrm{Iso}(G, H)$ is contained in $\mathrm{Ext}(S(G, H))$,

where Ext(Z) denotes the set of all extreme points of a set Z. Indeed, Iso(G, H) is the set of integral extreme points of $S(G, H)$.

The set $S(G) = S(G, G)$ is the polytope of *fractional automorphisms* of G.

Definition 5 ([81])**.** *A graph G is called* compact *if $S(G)$ has no other extreme points than* Aut(G)*, i.e.,* Ext($S(G)$) = Aut(G).

Compactness of G can equivalently be defined by any of the following two conditions:

- The polytope $S(G)$ is integral;
- Every fractional automorphism of G is a convex combination of automorphisms of G, i.e., $S(G) = \langle \text{Aut}(G) \rangle$, where $\langle Z \rangle$ denotes the convex hull of a set Z.

We recall some well-known examples of compact graphs. Complete graphs are compact. This can be seen as an immediate consequence of Birkhoff's theorem. The compactness of trees and cycles is established in [81]. Matching graphs mK_2 are also compact. This is a particular instance of a much more general result by Tinhofer [83]: If a connected graph G is compact, then mG is compact for any m. Tinhofer [83] also observes that compact graphs are closed under complement.

For a negative example, note that the graph $C_3 + C_4$ is not compact. This follows from a general result in [83]: All regular compact graphs must be vertex-transitive (and $C_3 + C_4$ is not).

We will need a known fact on the structure of fractional automorphisms. For a partition $V_1, \ldots, V_m$ of $\{1, \ldots, n\}$ let $X_1, \ldots, X_m$ be matrices, where the rows and columns of X_i are indexed by elements of V_i. Then we denote the block-diagonal matrix with blocks $X_1, \ldots, X_m$ by $X_1 \oplus \cdots \oplus X_m$. The following result shows that the fractional automorphisms "respect " the stable color-partition.

Lemma 13 (Ramana et al. [78]). *Let G be a (vertex-colored) graph on vertex set $\{1, \ldots, n\}$ and assume that the elements $V_1, \ldots, V_m$ of the stable partition $\mathcal{P}_G$ of G are intervals of consecutive integers. Then any fractional automorphism X of G has the form $X = X_1 \oplus \cdots \oplus X_m$.*

Note that the assumption of the lemma can be ensured for any graph by appropriately renaming its vertices. An immediate consequence of Lemma 13 is that a graph G is compact if and only if it is compact with respect to its stable coloring.

The following definitions and results will be useful in Section 5.3 and Section 5.4. Let $u \in V(G)$ and $v \in V(H)$ be vertices of two graphs G and H. By *individualization* of u and v we mean assigning the same *new color* to u and v, which makes them distinguished from the remaining vertices of G and H. Tinhofer [83] has shown that, if G is compact, then the following polynomial-time algorithm correctly decides if G and H are isomorphic.

1. Run CR on G and H until the coloring of $V(G) \cup V(H)$ stabilizes.
2. If the multisets of colors in G and H are different, then output "non-isomorphic" and stop. Otherwise,
   (a) if all color classes are singletons in G and H, then if the map $u \mapsto v$ matching each vertex $u \in V(G)$ to the vertex $v \in V(H)$ of the same color is an isomorphism, output "isomorphic" and stop. Else output "non-isomorphic" and stop.
   (b) pick any color class with at least two vertices in both G and H, select an arbitrary $u \in V(G)$ and $v \in V(H)$ in this color class and individualize them. Goto Step 1.

If G and H are any two non-isomorphic graphs, then Tinhofer's algorithm will always output "non-isomorphic". However, it can fail for isomorphic input graphs, in general. We call G a *Tinhofer graph* if the algorithm works correctly on G and every H for all

choices of vertex pairs to be individualized (as specified in Step 2(b)). Thus, the result of [83] can be stated as the inclusion Compact $\subseteq$ Tinhofer.

Let A be a subgroup of the automorphism group $Aut(G)$ of a graph G. Then the partition of $V(G)$ into the A-orbits is called an *orbit partition* of G. Any orbit partition of G is equitable, but the converse is not true, in general. However, Godsil [64, Corollary 1.3] has shown that the converse holds for compact graphs. We define *Godsil graphs* as the graphs for which the two notions of an equitable and an orbit partition coincide, that is, every equitable partition is the orbit partition of some subgroup A of $Aut(G)$. Thus, the result of [64] can be stated as the inclusion Compact $\subseteq$ Godsil.

5.2 Proof of Theorem 23

In this section, we prove Theorem 23. Given an amenable graph G and a fractional automorphism X of G, we have to express X as a convex combination of permutation matrices in Aut(G). Our proof strategy consists in exploiting the structure of amenable graphs as described by Theorem 21. Given an anisotropic component A of the cell graph $C(G)$, we define the *anisotropic component* G_A *of* G as the subgraph of G induced by the union of all cells belonging to A. Our overall idea is to prove the claim separately for each anisotropic component G_A, applying an inductive argument on the number of cells in A. A key role will be played by the fact that, according to Theorem 21, A is a tree with at most one heterogeneous cell.

We can assume that G is colored by the stable coloring because, by Lemma 13, the colored version has the same polytope of fractional automorphisms. We first consider the case when G consists of a single anisotropic component A. By Theorem 21, the corresponding cell graph $C(G)$ has at most one heterogeneous vertex, and A forms a spanning tree of $C(G)$. Without loss of generality, we can number the cells $V_1, \ldots, V_m$ of G so that V_1 is the unique heterogeneous cell if it exists; otherwise V_1 is chosen among the cells of

minimum cardinality. Moreover, we can suppose that, for each $i \leq m$, the cells $V_1, \ldots, V_i$ induce a connected subgraph in the tree A.

We will prove by induction on $i = 1, \ldots, m$ that the graphs $G_i = G[V_1 \cup \cdots \cup V_i]$ are compact. In the base case of $i = 1$, the graph $G_1 = G[V_1]$ is one of the graphs listed in Condition **A** of Theorem 21. All of them are known to be compact; see [81, 83]. As induction hypothesis, assume that the graph G_{i-1} is compact. For the induction step, we have to show that also G_i is compact.

Denote $D = V_i$. Since G has no more than one heterogeneous cell, $G[D]$ is complete or empty. It will be instructive to think of D as a "leaf" cell having a unique anisotropic link to the remaining part G_{i-1} of G_i. Let $C \in \{V_1, \ldots, V_{i-1}\}$ be the unique cell such that $\{C, D\}$ is an anisotropic edge of $C(G_i)$. To be specific, suppose that $G[C, D] \cong sK_{1,t}$. If $G[C, D]$ is the bipartite complement of $sK_{1,t}$, we can consider the complement of G_i, using the fact that the polytope of fractional automorphisms is the same for a graph and its complement. By the monotonicity property stated in Condition **C** of Theorem 21, $|C| = s$ and $|D| = st$. Let $C = \{c_1, c_2, \ldots, c_s\}$ and, for each j, $N(c_j) \subseteq D$ be the neighborhood of c_j in $G[C, D]$. Thus, $D = \bigcup_{j=1}^{s} N(c_j)$.

Let X be a fractional automorphism of G_i. It is convenient to break it up into three blocks $X = X' \oplus Y \oplus Z$, where Y and Z correspond to C and D respectively, and X' is the rest. By induction hypothesis we have the convex combination

$$X' \oplus Y = \sum_{P' \oplus P \in \mathrm{Aut}(G_{i-1})} \alpha_{P',P}\, P' \oplus P, \tag{5.2}$$

where $P' \oplus P$ are permutation matrices corresponding to automorphisms π of the graph G_{i-1}, such that the permutation matrix block P denotes the action of π on the color class C and P' the action on the remaining color classes of G_{i-1}.

We need to show that X is a convex combination of automorphisms of G_i. Let A denote the adjacency matrix of G_i, and $A_{S,T}$ denote the submatrix of A row-indexed by $S \subset V(G_i)$

and column-indexed by $T \subset V(G_i)$. Since X is a fractional automorphism of G_i, we have $XA = AX$. Recall that Y and Z are blocks of X corresponding to color classes C and D. Looking at the corner fragments of the matrices XA and AX, we get

$$\begin{pmatrix} Y & 0 \\ 0 & Z \end{pmatrix}\begin{pmatrix} A_{C,C} & A_{C,D} \\ A_{D,C} & A_{D,D} \end{pmatrix} = \begin{pmatrix} A_{C,C} & A_{C,D} \\ A_{D,C} & A_{D,D} \end{pmatrix}\begin{pmatrix} Y & 0 \\ 0 & Z \end{pmatrix},$$

which implies

$$YA_{C,D} = A_{C,D}Z, \tag{5.3}$$

$$A_{D,C}Y = ZA_{D,C}. \tag{5.4}$$

Consider Z as an $st \times st$ matrix whose rows and columns are indexed by the elements of sets $N(c_1), N(c_2), \ldots, N(c_r)$ in that order. We can thus think of Z as an $s \times s$ block matrix of $t \times t$ matrix blocks $Z^{(k,\ell)}$, $1 \leq k, \ell \leq s$. The next claim is a consequence of Equations (5.3) and (5.4).

Claim 9. *Each block $Z^{(k,\ell)}$ in Z is of the form*

$$Z^{(k,\ell)} = y_{k,\ell} W^{(k,\ell)}, \tag{5.5}$$

where $y_{k,\ell}$ is the $(k,\ell)^{th}$ entry of Y, and $W^{(k,\ell)}$ is a doubly stochastic matrix.

Proof. We first note from Equation (5.3) that the $(k, j)^{th}$ entry of the $s \times st$ matrix $YA_{C,D} = A_{C,D}Z$ can be computed in two different ways. In the left hand side matrix, it is $y_{k,\ell}$ for each $j \in N(c_\ell)$. On the other hand, the right hand side matrix implies that the same $(k, j)^{th}$ entry is also the sum of the j^{th} column of the $N(c_k) \times N(c_\ell)$ block $Z^{(k,\ell)}$ of the matrix Z.

We conclude, for $1 \leq k, \ell \leq s$, that each column in $Z^{(k,\ell)}$ adds up to $y_{k,\ell}$. By a similar argument, applied to Equation (5.4) this time, it follows, for each $1 \leq k, \ell \leq s$, that each *row* of any block $Z^{(k,\ell)}$ of Z adds up to $y_{k,\ell}$.

We conclude that, if $y_{k,\ell} \neq 0$, then the matrix $W^{(k,\ell)} = \frac{1}{y_{k,\ell}} Z^{(k,\ell)}$ is doubly stochastic. If $y_{k,\ell} = 0$, then (5.5) is true for any choice of $W^{(k,\ell)}$. □

Since each $W^{(k,\ell)}$ is a doubly stochastic matrix, by Birkhoff's theorem we can write it as a convex combination of $t \times t$ permutation matrices $Q_{j,k,\ell}$, whose rows are indexed by elements of $N(c_k)$ and columns by elements of $N(c_\ell)$:

$$W^{(k,\ell)} = \sum_{j=1}^{t!} \beta_{j,k,\ell} \, Q_{j,k,\ell}. \tag{5.6}$$

For every $P = (p_{k\ell})$ appearing in an automorphism $P' \oplus P$ of G_{i-1} (see Equation (5.2)), we define the $st \times st$ doubly stochastic matrix W_P by its $t \times t$ blocks indexed by $1 \leq k, \ell \leq s$ as follows:

$$W_P^{(k,\ell)} = \begin{cases} W^{(k,\ell)} & \text{if } p_{k\ell} = 1, \\ 0 & \text{if } p_{k\ell} = 0. \end{cases} \tag{5.7}$$

Substituting Equation (5.6) in Equation (5.7), we can express W_P as a convex combination of permutation matrices $W_P = \sum_Q \delta_{Q,P} \, Q$ where Q runs over all $st \times st$ permutation matrices indexed by the vertices in color class D. Notice that the permutation matrices Q participating in this decomposition satisfy

$$Q^{(k,\ell)} = \begin{cases} Q_{j,k,\ell} & \text{if } p_{k\ell} = 1, \\ 0 & \text{if } p_{k\ell} = 0. \end{cases} \tag{5.8}$$

for some $j \in [t!]$.

We claim that for each such Q, the $(s + st) \times (s + st)$ permutation matrix $P \oplus Q$ is an automorphism of the subgraph $G_i[C, D] = sK_{1,t}$. This holds because Q maps $N(c_k)$ to $N(c_\ell)$ whenever P maps c_k to c_ℓ, according to Equation (5.8). Since $P \in \mathrm{Aut}(G_i[C])$ and D is a homogeneous set in G_i, we conclude that, $P \oplus Q$ is an automorphism of the subgraph $G_i[C \cup D]$.

Equations (5.2) and (5.5) imply that

$$X = X' \oplus Y \oplus Z = \sum_{P' \oplus P \in \mathrm{Aut}(G_{i-1})} \alpha_{P',P}\, P' \oplus P \oplus W_P. \tag{5.9}$$

In order to see this, on the left hand side consider the $(k,\ell)^{th}$ block $Z^{(k,\ell)}$ of Z. On the right hand side, note that the corresponding block in each $P' \oplus P \oplus W_P$ is the matrix $W^{(k,\ell)}$. Clearly, the overall coefficient for this block equals the sum of $\alpha_{P',P}$ over all P' and P such that $p_{k,\ell} = 1$, which is precisely $y_{k,\ell}$ by Equation (5.2).

Now, if we plug the expression of W_P as a convex combination $W_P = \sum_Q \delta_{Q,P}\, Q$ in Equation (5.9), we will finally obtain the desired convex combination

$$X = \sum_{P',P,Q} \gamma_{P',P,Q}\, P' \oplus P \oplus Q.$$

It remains to argue that every $P' \oplus P \oplus Q$ occurring in this sum is an automorphism of G_i. Recall that a pair P', P can appear here only if $P' \oplus P \in \mathrm{Aut}(G_{i-1})$. Moreover, if such a pair is extended to a matrix $P' \oplus P \oplus Q$, then $P \oplus Q \in \mathrm{Aut}(G_i[C \cup D])$ as argued before. Since $G_i[B, D]$ is isotropic for every color class $B \neq D$ of G_i, we conclude that $P' \oplus P \oplus Q \in \mathrm{Aut}(G_i)$. This completes the induction step and finishes the case when G has one anisotropic component.

Next, we consider the case when $C(G)$ has several anisotropic components $T_1, \ldots, T_k$, $k \geq 2$. Let $G_1, \ldots, G_k$, where $G_i = G[\bigcup_{U \in V(T_i)} U]$, be the corresponding anisotropic components of G. By the proof of the previous case we already know that G_i is compact for each i.

Claim 10. *The automorphism group* $\mathrm{Aut}(G)$ *of* G *is the product of the automorphism groups* $\mathrm{Aut}(G_i)$, $1 \leq i \leq k$.

Proof. Recall that any automorphism of G must map each color class of G, which is a cell of the underlying amenable graph, onto itself. Thus, any automorphism π of G is

of the form $(\pi_1, \ldots, \pi_k)$, where π_i is an automorphism of the subgraph G_i. Now, for any two subgraphs G_i and G_j, we examine the edges between $V(G_i)$ and $V(G_j)$. For any color classes $U \subseteq V(G_i)$ and $U' \subseteq V(G_j)$, the edge $\{U, U'\}$ is isotropic because it is not contained in any anisotropic component of $C(G)$. Therefore, the bipartite graph $G[U, U']$ is either complete or empty. It follows that for any automorphisms π_i of G_i, $1 \leq i \leq k$, the permutation $\pi = (\pi_1, \ldots, \pi_k)$ is an automorphism of the graph G. □

As follows from Lemma 13, any fractional automorphism X of G is of the form $X = X_1 \oplus \cdots \oplus X_k$, where X_i is a fractional automorphism of G_i for each i. As each G_i is compact we can write each X_i as a convex combination

$$X_i = \sum_{\pi \in \mathrm{Aut}(G_i)} \alpha_{i,\pi} P_\pi.$$

This implies

$$I \oplus \cdots \oplus I \oplus X_i \oplus I \oplus \cdots \oplus I = \sum_{\pi \in \mathrm{Aut}(G_i)} \alpha_{i,\pi} I \oplus \cdots \oplus I \oplus P_\pi \oplus I \oplus \cdots \oplus I, \tag{5.10}$$

where block diagonal matrices in the above expression have X_i and P_π respectively in the i^{th} block (indexed by elements of $V(G_i)$) and identity matrices as the remaining blocks.

We now decompose the fractional automorphism X as a matrix product of fractional automorphisms of G

$$X = X_1 \oplus \cdots \oplus X_k = (X_1 \oplus I \oplus \cdots \oplus I) \cdot (I \oplus X_2 \oplus \cdots \oplus I) \cdot \cdots \cdot (I \oplus \cdots \oplus I \oplus X_k).$$

Substituting for $I \oplus \cdots \oplus I \oplus X_i \oplus I \oplus \cdots \oplus I$ from Equation (5.10) in the above expression and writing the product of sums as a sum of products, we see that X is a convex combination of permutation matrices of the form $P_{\pi_1} \oplus \cdots \oplus P_{\pi_k}$ where $\pi_i \in \mathrm{Aut}(G_i)$ for each i. By Claim 10, all the terms $P_{\pi_1} \oplus \cdots \oplus P_{\pi_k}$ correspond to automorphisms of G. Hence, G is compact, completing the proof of Theorem 23.

5.3 A Color-Refinement Based Hierarchy of Graphs

In this section, we prove Theorem 24. We proceed with the following lemmata.

Lemma 14. *Any Godsil graph is a Tinhofer graph.*

Proof. Assume that G is a Godsil graph. It suffices to show that Tinhofer's algorithm is correct whenever G and H are isomorphic. Let ϕ be an isomorphism from G to H. We will prove that, after the i-th refinement step made by the algorithm, there exists an isomorphism ϕ_i from G to H that preserves colors of the vertices. If this is true for each i, the algorithm terminates only if the discrete partition (i.e., the finest partition into singletons) is reached. Suppose that this happens in the k-th step. Then ϕ_k ensures that the algorithm decides isomorphism.

We prove the claim by induction on i. At the beginning, $\phi_1 = \phi$. Assume that an isomorphism ϕ_i exists and the partition is still not discrete. Suppose that now the algorithm individualizes $u \in V(G)$ and $v \in V(H)$. If $v = \phi_i(u)$, then $\phi_{i+1} = \phi_i$. Otherwise, consider the vertices u and $\phi_i^{-1}(v)$, which are in the same monochromatic class of G. Note that the partition of G produced in each refinement step is equitable. Since G is Godsil, there is an automorphism α preserving the partition such that $\alpha(u) = \phi_i^{-1}(v)$. We can, therefore, take $\phi_{i+1} = \phi_i \circ \alpha$. □

The orbit partition of G with respect to $Aut(G)$ is always a refinement of the stable partition $\mathcal{P}_G$ of G. We call G *refinable* if $\mathcal{P}_G$ is exactly the orbit partition of $Aut(G)$. Any Godsil graph is refinable by definition. It is easy to show that Tinhofer graphs are also refinable.

Lemma 15. *Any Tinhofer graph is refinable.*

Proof. Suppose that G is not refinable. Then G has vertices u and v that are in different orbits but not separated by the stable partition $\mathcal{P}_G$. This means that individualization of u and v in isomorphic copies G' and G'' of G gives non-isomorphic results. Therefore, if

Tinhofer's algorithm is run on G' and G'' and individualizes u and v, it eventually decides that G' and G'' are non-isomorphic. □

We are now ready to prove Theorem 24.

Proof. Summarizing Theorem 23, Lemmas 14 and 15, and [64, Corollary 1.3], we obtain the inclusion hierarchy. It remains to show that the inclusions are strict.

Separation of Discrete *and* Amenable*:* For $n \geq 2$, the complete graph K_n is amenable but not discrete.

Separation of Amenable *and* Compact*:* For $n \geq 6$, the cycles C_n are not amenable, while they are known to be compact graphs [81, Theorem 2]. For another family of separating examples, consider the disjoint union $2G$ of two copies of an arbitrary connected amenable graph G that is not a tree. By Part (ii) of Corollary 5, $2G$ is not amenable. On the other hand, G is compact by Theorem 23, and $2G$ is compact as well by the closure property of compact graphs established in [83].

Separation of Compact *and* Godsil*:* These classes are separated by the Petersen graph. Evdokimov, Karpinski, and Ponomarenko [62, Corollary 5.4] prove that the Petersen graph is not compact. It remains to show that the Petersen graph belongs to the class Godsil. This problem is solvable by modern computer algebra tools; see [89] where equitable and orbit partitions are counted for various strongly regular graphs, including the Petersen graph. We give a non-computer-assisted proof in Subsection 5.3.2.

Separation of Godsil *and* Tinhofer*:* These classes are separated by the Johnson graphs $J(n, 2)$ for $n \geq 7$. The *Johnson graph* $J(n, k)$ has the k-element subsets of $[n] = \{1, \ldots, n\}$ as vertices; any two of them are adjacent if their intersection consists of $k - 1$ elements. Note that $J(n, 1) = K_n$. Furthermore, the graph $J(n, 2)$ is the line graph of K_n: It has all 2-element subsets of $[n]$ as vertices and any two of them are adjacent if their intersection is non-empty. It is noticed in [61] that $J(n, 2)$ is not Godsil for $n \geq 7$. For establishing the

separation, we prove that $J(n, 2)$ is indeed Tinhofer. The proof of Theorem 26 is contained in Subsection 5.3.1 below.

Separation of Tinhofer *and* Refinable*:* Consider the gadget CFI(P_1, P_2, P_3) depicted in Figure 5.1, with two input pairs P_1 and P_2 and one output pair P_3. This gadget [59] has the property that any automorphism of it must flip an even number of the three pairs P_1, P_2, and P_3. We can combine CFI(P_1, P_2, P_3) with a second gadget CFI(P_1, P_2, P_4) with the same input pairs and a fresh output pair P_4. We assume that the four pairs P_1, P_2, P_3, and P_4 and the intermediate sets of four connecting vertices, are all different color classes.

This defines a refinable graph G, also depicted in Figure 5.1, with four color classes P_1, P_2, P_3, and P_4 of size 2, and two color classes F and F' of size 4 corresponding to the orbit partition of G. The graph G has the property that any automorphism of it must flip either both pairs P_3 and P_4 or none of them. Now, if we run the Tinhofer procedure on two identical copies G' and G'' of G, it might individualize color class P_3 in the first round and color class P_4 in the second round in such a way that the resulting graphs are not isomorphic, since the partial isomorphism flips exactly one of the two color classes.

Note that the vertex colors of G can be removed if we connect the four vertices in F by six edges and the two vertices in P_1 by one edge. □

5.3.1 Johnson Graph $J(n, 2)$ is Tinhofer

In this subsection, we give the proof of the following

Theorem 26. *$J(n, 2)$ is a Tinhofer graph for all n.*

This concludes the proof of the separation between the classes Godsil and Tinhofer.

Proof. We begin with some necessary definitions. Let G be a graph and denote the automorphism group of G by A. For $v \in V(G)$, by A_v we denote the stabilizer subgroup of A

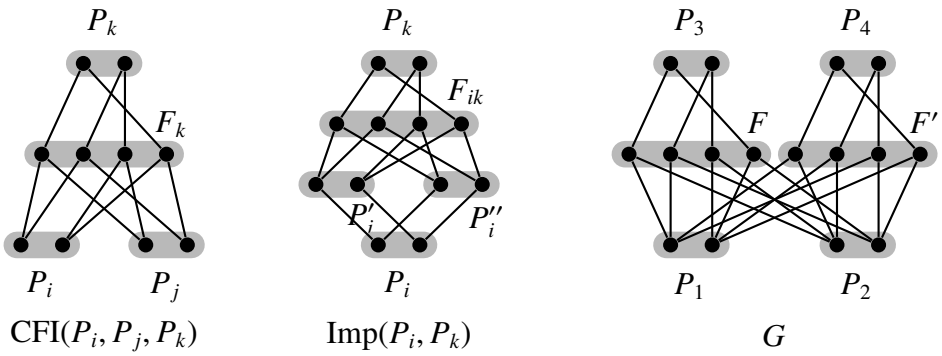

Figure 5.1. The CFI(P_i, P_j, P_k)- and Imp(P_i, P_k)-gadgets and a graph G separating Refinable from Tinhofer

that fixes the vertex v. Furthermore, for a subset $F \subset V(G)$, let $A_F = \bigcap_{v \in F} A_v$. Let $\mathcal{P}_F$ denote the stable partition of the colored version of G where each vertex in F is individualized. Then the orbit partition of A_F is a subpartition of $\mathcal{P}_F$. Note that G is Tinhofer if and only if, for every F, the orbit partition of A_F coincides with $\mathcal{P}_F$.

One way to prove that the two partitions coincide is to show that each orbit O of A_F is definable in terms of F in two-variable first-order logic. Here, "in terms of F" means that a defining formula $\Phi_O(x)$ can use constant symbols (names) for each vertex in F. Furthermore, $\Phi_O(x)$ contains occurrences of only two variables, x and y. At least one occurrence of x is free. $\Phi_O(x)$ uses two binary relation symbols $\sim$ and $=$ for adjacency and equality of vertices. This formula is true on G for $x = v$ exactly when $v \in O$.

Once $\Phi_O(x)$ is found for each O, the equality of the partitions follows by a similar argument as in [69, Theorem 1.8.1] or directly from the definitions of orbits, as those will imply that any two orbits are separated by color refinement starting from the individualization of F. The number of refinement steps sufficient to separate O from any other orbit can be only one greater than the quantifier depth of $\Phi_O(x)$.

In order to implement this scenario for $G = J(n, 2)$, it will be convenient to assume that $V(G) = \binom{[n]}{2}$ (note, however, that the formulas $\Phi_O(x)$ do not involve variables over $[n]$). Given $\alpha \in S_n$, by $\ell(\alpha)$ we denote the corresponding permutation of $\binom{[n]}{2}$. Obviously, every $\ell(\alpha)$ is an automorphism of G, and the automorphism group A contains nothing else by

the Whitney theorem [88].

Before designing the definitions $\Phi_O(x)$, we will need to make two preliminary steps: Describe A_F and, then, describe the orbits of A_F (first irrespectively of any logical formalism; expressing these descriptions in two-variable first-order logic will be the next task).

We now proceed to the detailed proof. Note that $J(2,2) = K_1$, $J(3,2) = K_3$, and $J(4,2)$ is the octahedral graph, whose complement is $K(4,2) = 3K_2$. Thus, these three graphs are amenable and, hence, Tinhofer. We can, therefore, assume that $n \geq 5$.

Call a fixed vertex $p \in F$ *isolated* if F contains no vertex adjacent to p. Let $F = F_1 \cup F_2$ be the partition of F into non-isolated and isolated vertices. Furthermore, we define the partition

$$[n] = W_1 \cup W_2 \cup W_3$$

as follows: W_1 is the union of all non-isolated pairs p (i.e., all p in F_1), and W_2 is the union of all isolated pairs p (i.e., all p in F_2). Thus, W_3 consists of the points of $[n]$ that are not included in any fixed pair.

Note now that $\ell(\alpha) \in A_F$ if and only if α either fixes or transposes the two points in each fixed pair. It follows that $\ell(\alpha) \in A_F$ exactly when

- $\alpha(w) = w$ for every $w \in W_1$ and
- $\alpha(p) = p$ for every $p \in F_2$.

Given a vertex $u = \{a, b\}$ of G, let $O(u)$ denote its orbit with respect to A_F. There are six kinds of orbits. Below we describe all of them along with providing suitable formal definitions $\Phi_{O(u)}(x)$.

Case 1: $\{a, b\} \subseteq W_1$. Then $O(u) = \{u\}$. *Formal definition:* $x = u$.

Case 2: $\{a, b\} \subseteq W_2$. Here we have two subcases. If $u \in F_2$, then $O(u) = \{u\}$ again.

Otherwise, F_2 contains two pairs $p_1 = \{a, a'\}$ and $p_2 = \{b, b'\}$. In this subcase,

$$O(u) = \{\{a, b\}, \{a', b\}, \{a, b'\}, \{a', b'\}\},$$

which is exactly the common neighborhood of p_1 and p_2. *Formal definition:* $x \sim p_1 \wedge x \sim p_2$.

Case 3: $\{a, b\} \subseteq W_3$. Now $O(u) = \binom{W_3}{2}$, which are exactly the non-fixed vertices with no neighbor in F. *Formal definition:* $\bigwedge_{p \in F}(x \neq p \wedge x \nsim p)$.

Case 4: $a \in W_1$, $b \in W_2$. Let $p = \{b, b'\}$ be the pair in F_2 containing b. Then,

$$O(u) = \{\{a, b\}, \{a, b'\}\}.$$

To give a formal definition of $O(u)$, we consider two subcases.

(i) a belongs to two adjacent vertices $q_1 = \{a, a_1\}$ and $q_2 = \{a, a_2\}$ in F_1.

Formal definition: $x \sim p \wedge x \sim q_1 \wedge x \sim q_2$. Indeed, the condition $x \sim p$ forces x to contain either b or b'. This excludes the possibility that $x = \{a_1, a_2\}$ and, therefore, x is forced to contain a by the adjacency to q_1 and q_2.

(ii) a belongs to a single vertex $q_1 = \{a, a'\}$ in F_1. By definition, F_1 contains also a vertex $q_2 = \{a', a''\}$. *Formal definition:* $x \sim p \wedge x \sim q_1 \wedge x \nsim q_2$.

Case 5: $a \in W_1$, $b \in W_3$. Then

$$O(u) = \{\{a, b'\} : b' \in W_3\}.$$

Similarly to the preceding case, we distinguish two subcases.

(i) a belongs to two adjacent vertices $q_1 = \{a, a_1\}$ and $q_2 = \{a, a_2\}$ in F_1.

Formal definition: First of all, we say that $x \sim q_1 \wedge x \sim q_2$. It remains to exclude the possibility that $x \subseteq W_1 \cup W_2$ (in particular, this will exclude $x = \{a_1, a_2\}$ and force x to

contain a). We do this by adding the following expression

$$\bigwedge_{p\in F} x \neq p \wedge \bigwedge_{p,q\in F, p\neq q} \neg(x \sim p \wedge x \sim q)$$
$$\wedge \bigwedge_{p,q\in F_1, p\sim q} (x \sim p \wedge x \sim q \rightarrow \exists y\,(y \sim x \wedge y \sim p \wedge y \sim q)). \quad (5.11)$$

The first conjunctive term prevents x to be one of the pairs in F. The second term excludes the case that x is covered by two disjoint pairs p and q in F. The third term excludes the case that x is covered by two intersecting pairs p and q in F or, equivalently, the case where x, p, and q form a triangle. It would be not enough just to forbid x, p, and q from forming a clique because this could also exclude a permissible case where x, p, and q form a star (which is captured by the subformula beginning with $\exists y$). Note, that we need the assumption $n \geq 5$ in this place.

(ii) a belongs to a single vertex $q_1 = \{a, a'\}$ in F_1, and $q_2 = \{a', a''\}$ is another vertex in F_1. *Formal definition:* $x \sim q_1 \wedge x \nsim q_2 \wedge x \notin W_1 \cup W_2$, the last being expressed by the formula (5.11).

Case 6: $a \in W_2$, $b \in W_3$. In this case, F_2 contains a pair $p = \{a, a'\}$ and

$$O(u) = \{\{a, b'\} : b' \in W_3\} \cup \{\{a', b'\} : b' \in W_3\}.$$

Formal definition: $x \sim p \wedge x \notin W_1 \cup W_2$, the latter being expressed by (5.11).

The proof is complete. □

5.3.2 The Petersen Graph is in Godsil

In this subsection, we prove that the Petersen Graph is in the class Godsil. This concludes the proof of the separation of the classes Compact and Godsil. We proceed with the proof.

It is well-known that the Petersen graph, denoted by P, is isomorphic to the Kneser graph $K(5,2)$. The *Kneser graph* $K(n,k)$ has the k-element subsets of $[n] = \{1,\dots,n\}$ as vertices and any two of them are adjacent if they are disjoint. An important fact about $K(5,2)$ is that its automorphism group is isomorphic to the symmetric group S_5 acting on the set $\{1,\dots,5\}$. In fact, any automorphism of $K(5,2)$ can be realized by extending the action of a permutation $\pi \in S_5$ to the vertex set of $K(5,2)$ [88].

First, we state some useful facts about the Petersen graph.

Lemma 16. *The Petersen graph has the following properties:*

(i) There are no cycles of length 3, 4 or 7.

(ii) There are no independent sets of size greater than 4*.*

(iii) Any two adjacent vertices have no common neighbors and any two non-adjacent vertices have a unique common neighbor.

We will need some definitions regarding partitions of the vertex set of a graph $G = (V,E)$. Given a partition $\Sigma = \{S_1,\dots,S_k\}$ of V, we refer to the sets $S_1,\dots,S_k$ as the *cells* of Σ. If the size of a cell is k, we call it a *k-cell*. Two cells S and S' are said to be *compatible* if the induced bipartite graph $P[S,S']$ is biregular (it can be empty). Otherwise, we say they are *incompatible*. Recall that any cell S of an equitable partition induces a regular graph $G[S]$. Moreover, in that case, any two cells S, S' are compatible and the number of edges in the biregular graph $G[S,S']$ is a common multiple of $|S|$ and $|S'|$.

Now we are ready to prove the following theorem.

Theorem 27. *The Petersen graph P is a Godsil graph.*

Proof. To prove the theorem, we will enumerate all equitable partitions of P. For each such partition Σ, we describe a subgroup of $Aut(P)$ such that its orbit partition coincides with Σ. We represent the vertices of P by the two-element subsets of the set

$\Omega = \{a, b, c, d, e\}$, where two vertices are adjacent if they are disjoint. This representation allows us to describe any subgroup of $Aut(P)$ as a subgroup of the permutation group S_Ω on Ω.

The two trivial partitions of $V(P)$ into one set and into ten singleton sets are clearly orbit partitions, since the Petersen graph is vertex-transitive. For our case analysis, we classify the remaining non-trivial equitable partitions of P by the minimum size δ of the cells in the partition. Clearly, $\delta \leq 5$. In the following claims we show for each $k \in \{1, 2, 3, 4, 5\}$ that any equitable partition of P with $\delta = k$ is an orbit partition of P.

Claim 11. *P does not have any equitable partition with $\delta = 3$.*

Proof. Suppose that there is an equitable partition Σ with $\delta = 3$ and let S be a 3-cell in it. Then Σ either has the form $\Sigma = \{S, T\}$, where $|T| = 7$, or the form $\Sigma = \{S, U, V\}$ where $|U| = 3$ and $|V| = 4$. The first case is ruled out since $P[T]$ can never be regular (P has neither independent sets of size 7 nor cycles of size 7). Suppose the second case is possible. Then $P[S]$ and $P[U]$ must be empty (since P has no triangles). Furthermore, the bipartite graphs $P[S, V]$ and $P[U, V]$ must be both biregular. The graph $P[S, V]$ (likewise, $P[U, V]$) is empty or it has 12 edges. It is not possible that $P[S, V]$ has 12 edges because then $P[V]$ has only 3 edges and cannot be regular. If both $P[S, V]$ and $P[U, V]$ are empty then V is disconnected from the rest of the graph, which is a contradiction. □

Claim 12. *All equitable partitions of P with $\delta = 4$ are orbit partitions.*

Proof. We first show that any equitable partition Σ with $\delta = 4$ has one 4-cell S and one 6-cell T, where $P[S]$ is empty and $P[T]$ is a 3-matching (a matching with 3 edges). Clearly, Σ must be of the form $\{S, T\}$, where $|S| = 4$ and $|T| = 6$. Moreover, $P[S]$ must be empty (0-regular) or 2-matching (1-regular) since it cannot be a 4-cycle (2-regular). In fact, the case of 2-matching can also be ruled out by counting the number of edges as follows. For S and T to be compatible, there must be 12 edges in the graph $P[S, T]$. Then there is exactly one edge left in the induced graph $P[T]$ which is impossible. Therefore, $P[S]$

must be empty. This also implies that the graph $P[S,T]$ has $4 \times 3 = 12$ edges. Hence, $P[T]$ must be a 3-matching.

Now observe that any independent set S of size 4 in P must be of the kind $S = \{ab, ac, ad, ae\}$ (up to automorphisms), implying that $T = \{bc, bd, be, cd, ce, de\}$. The partition $\{S, T\}$ can be easily verified to be equitable and that it is the orbit partition of the subgroup $S_{\{b,c,d,e\}}$. □

Claim 13. *All equitable partitions of P with $\delta = 5$ are orbit partitions.*

Proof. In this case Σ must have the form $\Sigma = \{S, T\}$ where $|S| = |T| = 5$. Moreover, since P does not have independent sets of size 5, $P[S]$ and $P[T]$ must be 5-cycles. Clearly, such partitions exist, and any such partition has a matching between sets S and T.

It remains to show that $\Sigma = \{S, T\}$ is indeed an orbit partition of some subgroup of $Aut(P)$. Denote the 5-cycle in S by 1-2-3-4-5. Let $1'$ be the matching partner of 1 in T and so on. Now, $1'$ and $2'$ cannot be adjacent, else there is a 4-cycle in P. The unique common neighbor of $1'$ and $2'$ must be $4'$, otherwise it is easy to verify that we will have a 4-cycle in P. The partners $3'$ and $5'$ can also be uniquely determined in T. The permutation $\pi = (12345)(1'2'3'4'5')$ can be verified to be an automorphism of P and the orbits of the subgroup generated by π are precisely $\{S, T\}$. □

Claim 14. *All equitable partitions of P with $\delta = 2$ are orbit partitions.*

Proof. Let Σ be an equitable partition of P with $\delta = 2$ and let $S = \{u, v\}$ be a 2-cell in it. We first show that uv must be an edge. This holds because any two non-adjacent vertices have a unique common neighbor x. The cell containing x can only be a singleton set, which contradicts $\delta = 2$.

Next we show that the neighborhood $N(S) = \bigcup_{x \in S} N(x) \setminus S$ of S is also a cell of Σ (see Figure 5.2). Since uv is an edge, there are no common neighbors of u and v. Therefore, $|N(S)| = 4$. Moreover, $N(S)$ is an independent set since any edge among vertices in $N(S)$

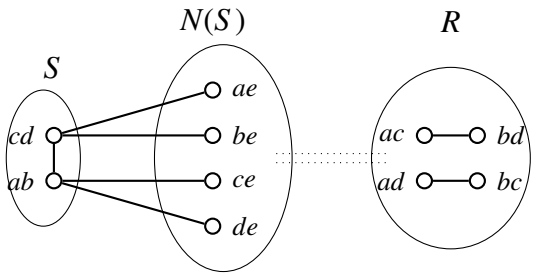

Figure 5.2. The case $\delta = 2$.

can be used to construct a cycle of length 3 or 4 passing through the edge uv. This is not possible by Lemma 16. Now let $R = V(P)\backslash(S \cup N(S))$ be the set of the four remaining vertices as shown in Figure 5.2. Observe that no cell can contain vertices from both $N(S)$ and R, since then it would be incompatible with S. Since $N(S)$ is an independent set, there cannot be a 2-cell inside $N(S)$. Clearly, there cannot be 1-cells and hence 3-cells inside $N(S)$. Therefore, $N(S)$ must indeed be a cell.

By accounting for edges of S and $N(S)$, it is easy to verify that R has exactly two edges, and hence $P[R]$ must be a 2-matching. Since $\delta = 2$, R does not contain any 1-cell and hence, any 3-cells. This leaves us with only two cases.

Case 1: R is a cell. We characterize all such partitions by naming a typical case. W.l.o.g, let $S = \{ab, cd\}$ since S is an edge. Then $N(S)$ must be $\{ae, be, ce, de\}$ and R must be $\{ac, ad, bc, bd\}$. The partition $\{ab, cd\}, \{ae, be, ce, de\}, \{ac, ad, bc, bd\}$ can be easily verified to be equitable. Moreover, it is easy to check that it is the orbit partition of the subgroup of all permutations in S_Ω which preserve the Ω-partition $\{ab\}, \{cd\}, \{e\}$. This is also the subgroup generated by the automorphisms $(ab), (cd), (ac)(bd)$.

Case 2: Σ partitions R in two sets A and B where $|A| = |B| = 2$. Since each 2-cell has to be an edge (see above), the sets A and B must be $\{ac, bd\}$ and $\{bc, ad\}$. The partition $\{ab, cd\}, \{ae, be, ce, de\}, \{ac, bd\}, \{ad, bc\}$ can be easily verified to be equitable. Moreover, it is easy to check that it is the orbit partition of the subgroup of all permutations in S_Ω which preserve the Ω-partition $\{ab\}, \{cd\}, \{e\}$ and additionally, stabilize the sets $\{ac, bd\}$ and $\{ad, bc\}$. This is also the subgroup generated by the automorphisms

$(ac)(bd), (ad)(bc), (ab)(cd)$.

The proof of Claim 14 is complete. □

Claim 15. *All equitable partitions of P with $\delta = 1$ are orbit partitions.*

Proof. Let S be a singleton set in such an equitable partition. Similar to a previous argument, a cell cannot have vertices from both $N(S)$ and $V \backslash N(S)$. Therefore, any equitable partition refines the partition $S, N(S), R$ (see Figure 5.3). Observe that $N(S)$ must be an independent set (otherwise there is a 3-cycle). Moreover, if we assume that $S = \{ab\}$, $N(S)$ must be $\{ce, de, cd\}$ and therefore, $R = \{ae, be, ac, bc, ad, bd\}$ forms a 6-cycle, as shown in the figure. We proceed by further classifying equitable partitions on the basis of the partition induced by them inside $N(S)$. Since $|N(S)| = 3$, we have three possible cases. Either $N(S)$ is a cell, or it contains three 1-cells, or it contains one singleton and one 2-cell.

Case 1: $N(S)$ is a cell. We further classify the equitable partitions in this case on the basis of the partition induced on the set R. First, we examine the possible cells X in R which are compatible with $N(S)$. X cannot be of size 1 or 2, otherwise $P[N(S), X]$ has at most two edges. Also, X cannot be of size 4 or 5 since this would imply a cell of size 1 or 2 in R. Therefore, either R is a cell, or there are two 3-cells in R.

(a) R is a cell. The partition $\{ab\}, \{de, cd, ce\}, \{ac, ad, ae, bc, bd, be\}$ can be verified to be an equitable partition. Moreover, it is easy to check that it is the orbit partition of the subgroup $S_{\{c,d,e\}} \times S_{\{a,b\}}$.

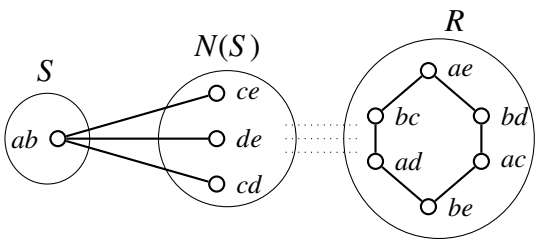

Figure 5.3. The case $\delta = 1$.

(b) The partition induced on R is of the form $\{A, B\}$, where $|A| = |B| = 3$. Because of regularity, the only possible 3-cells in R are the independent sets $\{ad, ac, ae\}$ and $\{bc, bd, be\}$. The partition $\{ab\}$, $\{de, cd, ce\}$, $\{ad, ac, ae\}$, $\{bc, bd, be\}$ is clearly equitable. Moreover, it is easy to check that this partition is the orbit partition of the subgroup $S_{\{c,d,e\}}$.

Case 2: $N(S)$ contains three 1-cells. Again, we classify the equitable partitions on the basis of the partition induced on the set R. We can check that a cell of size more than two in R will have at least one edge to some singleton in $N(S)$, and will be incompatible with that singleton. Therefore, cells in R must have size at most 2. Moreover, any 2-cell must be of the form $\{ax, bx\}$ for some $x \in \{d, c, e\}$ since all other 2-cells can be seen to be incompatible with some singleton cell in $N(S)$. Finally, it can be seen that every possible 1-cell is incompatible with these three 2-cells. Hence, R must consist of three cells of size 2, namely $\{ad, bd\}$, $\{ac, bc\}$, $\{ae, be\}$. The partition $\{ab\}$, $\{cd\}$, $\{ce\}$, $\{de\}$, $\{ad, bd\}$, $\{ac, bc\}$, $\{ae, be\}$ can be easily seen to be equitable. Moreover, it is easy to check that it is the orbit partition of the subgroup $S_{\{a,b\}}$.

Case 3: $N(S)$ contains a 2-cell $U = \{ce, de\}$ and a 1-cell $V = \{cd\}$. Again, we need to classify the equitable partitions on the basis of the partition induced on the set R. First, we examine the possible cells X in R which are compatible with U and V. Clearly, X cannot be a 5-cell since $P[X]$ cannot be regular. It cannot be a 3-cell as well since the two candidate 3-cells are the independent sets $\{ad, ac, ae\}$ and $\{bc, bd, be\}$. Neither of them can be compatible with the singleton set V. Also, R cannot be a cell since it is incompatible with the singleton set V. Moreover, the only possible 4-cell is the neighborhood of the set U, i.e. $\{ac, bd, ad, bc\}$. Any other 4-cell is incompatible with U. Overall, we have no cells of size 3, 5, or 6 in R. Therefore, we have only the following four remaining subcases.

(a) R consists of one 4-cell and two 1-cells. This case is not possible since a 1-cell cannot be compatible with a 4-cell.

(b) R consists of one 4-cell and one 2-cell. The cells are $\{ac, bd, ad, bc\}$ and $\{ae, be\}$. The partition $\{ab\}$, $\{cd\}$, $\{ce, de\}$, $\{ae, be\}$, $\{ac, bd, ad, bc\}$ can be verified to be an equitable partition. Moreover, it is easy to check that it is the orbit partition of the subgroup $S_{\{a,b\}} \times S_{\{c,d\}}$

(c) R consists of three 2-cells. First, ae and be must be in the same 2-cell, otherwise the cell containing any of them would be incompatible with V. For the remaining vertices ac, ad, bc, bd, we can pair them up in three ways: (i) ac, ad and bc, bd, (ii) ac, bc and ad, bd, or (iii) ac, bd and ad, bc The first case is not possible since $\{ae, be\}$ and $\{ac, ad\}$ are not compatible. The second case is not possible because $\{ac, bc\}$ and $U = \{ce, de\}$ are not compatible. The third case gives an equitable partition $\{ab\}$, $\{cd\}$, $\{ce, de\}$, $\{ae, be\}$, $\{ac, bd\}$, $\{ad, bc\}$. Moreover, it is easy to check that it is the orbit partition of the subgroup generated by $(ab)(cd)$.

(d) R consists of a bunch of 1-cells and 2-cells. Clearly, the vertices ac, ad, bc, bd cannot form a singleton cell, since such a 1-cell will not be compatible with U. Therefore, $\{ae\}$ and $\{be\}$ are the only possible singleton cells. Neither of them can pair up with one of ac, ad, bc, bd since that cell would be incompatible with V. Therefore, they are forced to be singleton cells. It remains to partition ac, ad, bc, bd into two 2-cells. The vertex ac cannot be paired up with bd or bc since it will be incompatible with be. Therefore, the only possible case is to have 2-cells $\{ac, ad\}$ and $\{bc, bd\}$. The partition $\{ab\}$, $\{cd\}$, $\{ce, de\}$, $\{ae\}$, $\{be\}$, $\{ac, ad\}$, $\{bc, bd\}$ can be verified to be equitable. Moreover, it is easy to check that it is the orbit partition of the subgroup $S_{\{c,d\}}$. (This case is identical to Case 2).

The proof of Claim 15 is complete. □

□

5.4 P-Hardness Results

In this section, we prove Theorem 25.

Proof. We show a uniform AC^0 many-one reduction from the *monotone circuit-value problem (MCVP)*, whose P-completeness is established in [65]. An instance of MCVP consists of a monotone boolean circuit C with and- and or-gates and constant input gates, and one has to decide if C evaluates to 1. Given such a circuit C, we construct a graph G as follows:

- For each gate g_k of C, G contains a pair $P_k = \{a_k, b_k\}$ of vertices.
- If g_k is a constant input gate with value 1, then a_k and b_k get different colors (i.e., they form singleton color classes); otherwise a_k and b_k both get the same color (i.e., they form a color class of size 2).
- For each and-gate g_k with input gates g_i and g_j, G additionally contains a color class F_k of size 4 that together with the two input pairs P_i and P_j as well as the output pair P_k forms a CFI(P_i, P_j, P_k)-gadget; see Figure 5.1.
- For each or-gate g_k with input gates g_i and g_j, G additionally contains two color classes F_{ik} and F_{jk} of size four, and four color classes P'_i, P''_i, P'_j, P''_j of size 2. The color classes P'_i, P''_i, P_k and F_{ik} form a CFI(P'_i, P''_i, P_k)-gadget and each of the pairs P'_i and P''_i is linked to P_i by two parallel edges. Henceforth, we denote this gadget by Imp(P_i, P_k); see Figure 5.1. Likewise, the color classes P'_j, P''_j and F_{jk} are used to form an Imp(P_j, P_k)-gadget.

A straightforward induction on the height of the and- and or-gates in C shows that CR on input G refines a color class P_k if and only if the corresponding gate g_k outputs value 1. This follows from the following observations.

- If g_k is an and-gate with input gates g_i and g_j, then the vertices in P_k get different C^{r+2} colors if and only if the vertices in P_i as well as the vertices in P_j have different C^r colors.

- If g_k is an or-gate with input gates g_i and g_j, then the vertices in P_k get different C^{r+3} colors if and only if either the vertices in P_i or the vertices in P_j have different C^r colors.

Now let G' be the graph that results from G by connecting the vertex pair P_l corresponding to the output gate g_l by two parallel edges with each pair P_k corresponding to a constant 0 input gate g_k. Then C evaluates to 1 if and only if G' is discrete (i.e., CR on input G' individualizes all vertices of G').

Moreover, if we connect the output pair P_l via an additional $\mathrm{Imp}(P_l, P_{l+1})$-gadget to a new vertex pair P_{l+1}, then the resulting graph G'' is not even refinable if C evaluates to 0. The reason is that no automorphism of G'' flips the pair P_{l+1}, but CR only refines the color class P_{l+1} if C evaluates to 1.

Hence, the mapping $C \mapsto G''$ simultaneously reduces MCVP to each of the graph classes in the hierarchy (5.1). □

We observe that the graph G'' used in the proof of the hardness results can be easily replaced by an uncolored graph. In fact, the vertex colors can be substituted by suitable graph gadgets in such a way that the automorphism group as well as the stable partition remain essentially unchanged (up to the addition of several singleton cells). Hence, the hardness results are also valid for the restricted versions of the classes in the hierarchy (5.1) where we consider only uncolored graphs.

5.5 Discussion

Theorem 23 unifies and extends several earlier results providing examples of compact graphs. In particular, it gives another proof of the fact that almost all graphs are compact, which also follows from a result of Godsil [64, Corollary 1.6]. This follows since Babai, Erdös, and Selkow [52] proved that almost all graphs are discrete, and hence amenable.

Furthermore, Theorem 23 subsumes Tinhofer's result that trees are compact. We note that the proof of Theorem 23 uses only compactness of complete graphs, matching graphs, and the 5-cycle. By Corollary 4, we can extend this result to forests. This extension is not straightforward as compact graphs are not closed under disjoint union. In [82], Tinhofer proves compactness for the class of *strong tree-cographs*, which includes forests only with pairwise non-isomorphic connected components. To the best of our knowledge, compactness of unigraphs, which also follows from Theorem 23, has not been observed earlier. Summarizing, we note the following result.

Corollary 6. *Discrete graphs, forests, and unigraphs are compact.*

Our Theorem 21 gives an efficiently checkable criterion for a graph G that ensures the correctness of the CR algorithm on (G, H) for any graph H. Theorem 23 implies that Tinhofer's approach to isomorphism testing based on the compactness concept has strictly larger potential of applicability. The most important question, that still remains open, is whether the applicability range of this approach admits an efficient characteriation (like in the case of CR). In other terms, what is the complexity of recognizing compact graphs? We proved in Theorem 25 that this problem is P-hard. The only known complexity upper bound, noted in [83], is coNP, because testing if every vertex of a given polytope is integral is in coNP.

Another intriguing problem is the complexity of recognizing refinable graphs. Using an AND-function for Graph Isomorphism (see, e.g., [73]), it is easy to show that this recognition problem is polynomial-time many-one reducible to Graph Isomorphism. On the

other hand, recognition of Refinable is at least as hard as recognition of vertex-transitive graphs (which, in its turn, is at least as hard as testing isomorphism of two vertex-transitive graphs).

Applicability of Tinhofer's isomorphism algorithm described in Section 5.1 seems to be a subtle issue. Recall that it works correctly on an input pair (G, H) whenever G is compact [83]. By our Theorem 24, the applicability range of this algorithm is even larger. It would be interesting to analyze the correctness of the algorithm on various classes of vertex-transitive graphs, for example, on classes of Cayley graphs. Note that compactness of Cayley graphs is studied in [79, 87].

The CR procedure can be regarded as the one-dimensional version of the more general k-dimensional Weisfeiler-Leman (k-WL) algorithm. It would, therefore, be natural to try to obtain k-dimensional analogs of our results. More specifically, let us fix the parameter $k \geq 2$ and consider the class of graphs distinguishable from any non-isomorphic graph by the k-WL algorithm. Equivalently, this is the class of graphs definable in $(k+1)$-variable first-order logic with counting quantifiers. Is it recognizable in polynomial time? Note that an affirmative answer for $k = 2$ would imply a possibility to efficiently decide if a set of parameters determines a strongly regular graph uniquely up to isomorphism (as 2-WL fails to distinguish non-isomorphic strongly regular graphs with the same parameters). Note also that, as it is shown in [72], the 2-dimensional analog of the inclusion Amenable $\subseteq$ Refinable is false.

Bibliography

[1] V. Arvind, G. Rattan. The parameterized complexity of Geometric Graph Isomorphism. *Algorithmica*, 75:2, 258–276, 2016.

[2] V. Arvind, G. Rattan. The parameterized complexity of Geometric Graph Isomorphism. In *Proceedings of 9th International Symposium on Parameterized and Exact Computation, (IPEC 2014)*, pp. 51–62, 2014.

[3] V. Arvind, J. Köbler, G. Rattan, O. Verbitsky. On the power of color refinement. In *Proceedings of the 20th International Symposium on Fundamentals of Computation Theory (FCT)*, pp. 339–350, 2015.

[4] V. Arvind, J. Köbler, G. Rattan, O. Verbitsky. On Tinhofer's linear programming approach to isomorphism testing. In *Proceedings of the 40th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pp. 26–37, 2015.

[5] V. Arvind, P. Kurur. Graph Isomorphism is in SPP. In *Proceedings of the Annual Symposium of Foundations of Computer Science*, pp. 743–750, 2002.

[6] A. Asterias, E. Maneva. Graph Isomorphism, Sherali-Adams relaxations and indistinguishability in counting logics. *SIAM Journal on Computing*, 42(1):112–137, 2013.

[7] L. Babai. Graph Isomorphism in quasipolynomial time. Accepted to the *STOC*, 2016.

[8] L. Babai. Monte Carlo algorithms in Graph Isomorphism testing. Tech. Rep. 79-10, Dep. Math. et Stat., Universite de Montreal, 1979.

[9] L. Babai, P. Erdős, S. M. Selkow. Random Graph Isomorphism. *SIAM Journal on Computing*, 9(3):628–635, 1980.

[10] L. Babai, D. Grigoryev, D. Mount. Isomorphism of graphs with bounded eigenvalue multiplicity. *In Proceedigns of the ACM STOC Conference*, pp.310–324, 1982.

[11] R. Bopanna, J. Hastad, S. Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25(2):127–132, 1987.

[12] K. Booth. Isomorphism testing for graphs,semigroups and finite automata are polynomially equivalent problems. *SIAM Journal on Computing*, 7:273–279, 1978.

[13] J.-Y. Cai, M. Fürer, N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

[14] S. Evdokimov, I. Ponomarenko. On the geometric Graph Isomorphism problem. *Pure and Applied Algebra*, 117-118:253–276, 1997.

[15] S. Evdokimov, I. Ponomarenko. Isomorphism of coloured graphs with slowly increasing multiplicity of Jordan blocks. *In Combinatorica*, 19(3): 44th, 321–333, 1999.

[16] M. Grohe. Fixed-point logics on planar graphs. In *Proceedings of the IEEE Symposium on Logic in Computer Science (LICS)*, pp. 6–15, 1998.

[17] M. Grohe. Fixed-point definability and polynomial time of graphs with excluded minors. In *Proceedings of the IEEE Symposium on Logic in Computer Science (LICS)*, pp. 179–188, 2010.

[18] M. Grohe. Isomorphism testing for embeddable graphs through definability. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pp. 63–72, 2000.

[19] C. Godsil. Compact graphs and equitable partitions. *Linear Algebra Appl.*, 255(1–3):259 – 266, 1997.

[20] I. Haviv, O. Regev. On the Lattice Isomorphism Problem. In *Proceedings of the 25th Annual Symposium on Discrete Algorithms (SODA)*, pp.391–404, 2014.

[21] N. Immerman, E. Lander. Describing graphs: A first-order approach to graph canonization. In *Complexity Theory Retrospective*, pp. 59–81. Springer, 1990.

[22] J. Kobler, U. Schoning, J. Toran. The Graph Isomorphism Problem: Its Structural Complexity. *Prog. Theoret. Comput. Sci.*, Birkhauser, 1993.

[23] L. Lovasz, A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1(2):166–190, 1991.

[24] E. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Comput. Syst. Sci.*, 25(1):42–65, 1982.

[25] E. Luks. Permutation groups and polynomial-time computation. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 11:139–175, 1993.

[26] G. Miller. Graph Isomorphism, general remarks. *Journal of Computer and System Sciences*, 18:128–142, 1979.

[27] C. Papadimitrou, S. Safra. The complexity of low-distortion embeddings between point sets. In *Proceedings of the 16th Annual Symposium on Discrete Algorithms*, pp. 112-–118, 2005

[28] M. V. Ramana, E. R. Scheinerman, D. Ullman. Fractional isomorphism of graphs. *Discrete Mathematics*, 132(1-3):247–265, 1994.

[29] A. Seress. Permutation Group Algorithms. Cambridge Univ. Press, 2003.

[30] H. Sherali, W. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM Journal on Discrete Mathematics, 3(3):411-430, 1990.

[31] G. Tinhofer. A note on compact graphs. *Discrete Applied Mathematics*, 30(2-3):253–264, 1991.

[32] G. Tinhofer. Graph Isomorphism and theorems of Birkhoff type. *Computing*, 36:285–300, 1986.

[33] J. Toran. On the hardness of Graph Isomorphism. *SIAM Journal on Computing*, 33(5):1093–1108, 2004.

[34] B. Weisfeiler. On construction and identification of graphs. Springer-Verlag, 1977.

[35] V. Zemlyachenko, N. Korneenko, R. Tyshkevich. Graph Isomorphism Problem. *Zapiski Nauchnykh Seminarov LOMI*, 118:83–158, 1982.

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

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

[38] Peter Braß and Christian Knauer. Testing the congruence of d-dimensional point sets. *International Journal of Computational Geometry and Applications*, 12:115–124, 2002.

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

[40] Christos H. Papadimitriou and Shmuel Safra. The complexity of low-distortion embeddings between point sets. In *Proceedings of the 16th Annual Symposium on Discrete Algorithms*, 112–118, 2005.

[41] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. *Society for Industrial and Applied Mathematics Journal of Computing,* 42(3):1364–1391, 2013.

[42] Ishay Haviv and Oded Regev. On the lattice isomorphism problem. In *Proceedings of the 25th Annual Symposium on Discrete Algorithms,* 391–404, 2014.

[43] László Babai and Eugene M. Luks. Canonical labeling of graphs. In *Proceedings of the 15th Annual Symposium on Theory of Computing,* 171–183, 1983.

[44] Merrick L. Furst, John E. Hopcroft, and Eugene M. Luks. Polynomial-time algorithms for permutation groups. In *Proceedings of the 21st Annual Symposium on Foundations of Computer Science Conference*, 36–41, 1980.

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

[46] A. G. Corbalan, Marisa Mazon, and Tomas Recio. About Voronoi Diagrams for Strictly Convex Distances. In *Proceedings of the 9th European Workshop on Computational Geometry*, 17–22, 1993.

[47] Eugene M. Luks, Hypergraph Isomorphism and Structural Equivalence of Boolean Functions. In *Proceedings of the 31st Annual Symposium on Theory of Computing,* 652–658, 1999.

[48] D. Angluin. Local and global properties in networks of processors. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing*, pages 82–93. ACM, 1980.

[49] V. Arvind, J. Köbler, G. Rattan, and O. Verbitsky. On the power of color refinement. In *Proceedings of the 20th International Symposium on Fundamentals of Computa-*

tion Theory (FCT), Lecture Notes in Computer Science, vol. 9210, pages 339–350. Springer, 2015.

[50] V. Arvind, J. Köbler, G. Rattan, and O. Verbitsky. On Tinhofer's linear programming approach to isomorphism testing. In *Proceedings of the 40th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, Lecture Notes in Computer Science, vol. 9235, pages 26–37. Springer, 2015.

[51] A. Atserias and E. N. Maneva. Sherali-Adams relaxations and indistinguishability in counting logics. *SIAM J. Comput.*, 42(1):112–137, 2013.

[52] L. Babai, P. Erdős, and S. M. Selkow. Random graph isomorphism. *SIAM J. Comput.*, 9(3):628–635, 1980.

[53] L. Babai and L. Kučera. Canonical labelling of graphs in linear average time. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 39–46, 1979.

[54] C. Berkholz, P. Bonsma, and M. Grohe. Tight lower and upper bounds for the complexity of canonical colour refinement. In *Proceedings of 21st Annual European Symposium on Algorithms (ESA)*, volume 8125 of *Lecture Notes in Computer Science*, pages 145–156. Springer, 2013.

[55] A. Borri, T. Calamoneri, and R. Petreschi. Recognition of unigraphs through superposition of graphs. *J. Graph Algorithms Appl.*, 15(3):323–343, 2011.

[56] R. A. Brualdi. Some applications of doubly stochastic matrices. *Linear Algebra Appl.*, 107:77–100, 1988.

[57] R. A. Brualdi. *Combinatorial matrix classes*. Cambridge University Press, 2006.

[58] R. Busacker and T. Saaty. *Finite graphs and networks: an introduction with applications*. International Series in Pure and Applied Mathematics. McGraw-Hill Book Company, New York etc., 1965.

[59] J.-Y. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

[60] A. Cardon and M. Crochemore. Partitioning a graph in $O(|A| \log_2 |V|)$. *Theor. Comput. Sci.*, 19:85–98, 1982.

[61] A. Chan and C. D. Godsil. Symmetry and eigenvectors. In *Graph symmetry*, volume 497, pages 75–106. Kluwer Acad. Publ., Dordrecht, 1997.

[62] S. Evdokimov, M. Karpinski, and I. N. Ponomarenko. Compact cellular algebras and permutation groups. *Discrete Mathematics*, 197-198:247–267, 1999.

[63] S. Evdokimov, I. N. Ponomarenko, and G. Tinhofer. Forestal algebras and algebraic forests (on a new class of weakly compact graphs). *Discrete Mathematics*, 225(1-3):149–172, 2000.

[64] C. Godsil. Compact graphs and equitable partitions. *Linear Algebra Appl.*, 255(1–3):259 – 266, 1997.

[65] L. M. Goldschlager. The monotone and planar circuit value problems are log space complete for P. *SIGACT News*, 9:25–29, 1977.

[66] M. Grohe. Equivalence in finite-variable logics is complete for polynomial time. *Combinatorica*, 19(4):507–532, 1999.

[67] M. Grohe, K. Kersting, M. Mladenov, and E. Selman. Dimension reduction via colour refinement. In *Proceeding of 22th Annual European Symposium on Algorithms (ESA)*, volume 8737 of *Lecture Notes in Computer Science*, pages 505–516. Springer, 2014.

[68] M. Grohe and M. Otto. Pebble games and linear equations. In *Computer Science Logic (CSL'12)*, volume 16 of *LIPIcs*, pages 289–304, 2012. A full version is available as an e-print http://arxiv.org/abs/1204.1990.

[69] N. Immerman and E. Lander. Describing graphs: A first-order approach to graph canonization. In *Complexity Theory Retrospective*, pages 59–81. Springer, 1990.

[70] R. Johnson. Simple separable graphs. *Pac. J. Math.*, 56:143–158, 1975.

[71] K. Kersting, M. Mladenov, R. Garnett, and M. Grohe. Power iterated color refinement. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1904–1910. AAAI Press, 2014.

[72] S. Kiefer, P. Schweitzer, and E. Selman. Graphs identified by logics with counting. In *Proceedings of the 40th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, Lecture Notes in Computer Science, vol. 9235, pages 319–330. Springer, 2015. A full version is available as an e-print http://arxiv.org/abs/1503.08792.

[73] J. Köbler, U. Schöning, and J. Torán. *The graph isomorphism problem: its structural complexity*. Boston, MA: Birkhäuser, 1993.

[74] M. Koren. Pairs of sequences with a unique realization by bipartite graphs. *Journal of Combinatorial Theory, Series B*, 21(3):224 – 234, 1976.

[75] A. Krebs and O. Verbitsky. Universal covers, color refinement, and two-variable logic with counting quantifiers: Lower bounds for the depth. In *Proceedings of the 30-th ACM/IEEE Annual Symposium on Logic in Computer Science (LICS)*, pages 689–700. IEEE Computer Society, 2015.

[76] F. T. Leighton. Finite common coverings of graphs. *J. Comb. Theory, Ser. B*, 33(3):231–238, 1982.

[77] P. N. Malkin. Sherali-adams relaxations of graph isomorphism polytopes. *Discrete Optimization*, 12:73–97, 2014.

[78] M. V. Ramana, E. R. Scheinerman, and D. Ullman. Fractional isomorphism of graphs. *Discrete Mathematics*, 132(1-3):247–265, 1994.

[79] H. Schreck and G. Tinhofer. A note on certain subpolytopes of the assignment polytope associated with circulant graphs. *Linear Algebra Appl.*, 111:125–134, 1988.

[80] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.

[81] G. Tinhofer. Graph isomorphism and theorems of Birkhoff type. *Computing*, 36:285–300, 1986.

[82] G. Tinhofer. Strong tree-cographs are Birkhoff graphs. *Discrete Applied Mathematics*, 22(3):275–288, 1989.

[83] G. Tinhofer. A note on compact graphs. *Discrete Applied Mathematics*, 30(2-3):253–264, 1991.

[84] G. Tinhofer and M. Klin. Algebraic combinatorics in mathematical chemistry. Methods and algorithms III. Graph invariants and stabilization methods. Technical Report TUM-M9902, Technische Universität München, 1999.

[85] R. Tyshkevich. Decomposition of graphical sequences and unigraphs. *Discrete Mathematics*, 220(1-3):201–238, 2000.

[86] G. Valiente. *Algorithms on Trees and Graphs*. Springer, 2002.

[87] P. Wang and J. S. Li. On compact graphs. *Acta Mathematica Sinica*, 21(5):1087–1092, 2005.

[88] H. Whitney. Congruent graphs and connectivity of graphs. *Amer. J. Math.*, 54:150–168, 1932.

[89] M. Ziv-Av. Results of computer algebra calculations for triangle free strongly regular graphs. *E-print*, http:/www.math.bgu.ac.il/~zivav/math/eqpart.pdf, 2013.