Representation Theory of Graph Isomorphism

Master's Project

Jacob Urisman

Advisors: Joshua Grochow, Cameron Musco

May 16, 2025

Contents

1 Introduction

1.1 Graph Isomorphism

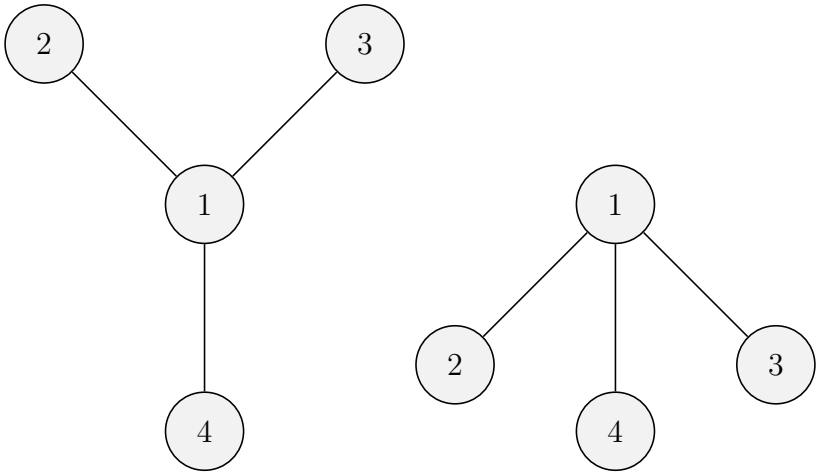
Definition 1.1 (graph)**.** A graph $G = (V, E)$ is a set of vertices V(also called nodes) and a set of edges $E \subseteq V \times V$ between the vertices.

Edges can be directed or undirected. An edge of the form (i, i) is called a loop. A graph is loopless if it has no edges (i, i) for $i \in V$.

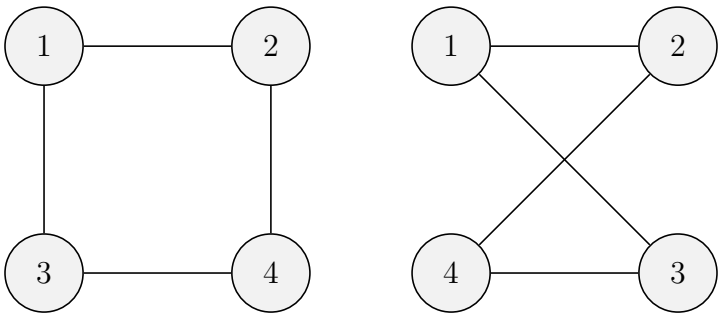
Definition 1.2 (isomorphism of graphs)**.** Two graphs G, H are said to be isomorphic if there exists a bijection $\sigma : V_G \to V_H$ such that $\forall i, j \in V_G : (i, j) \in E_G \iff (\sigma(i), \sigma(j)) \in E_H$. We sometimes refer to this function as a relabelling function. We denote graphs being isomorphic with $G \cong H$.

Definition 1.3 (Graph Isomorphism)**.** A decision problem. Given two graphs G, H decide whether $G \cong H$.

Example 1.4. The following two graphs are isomorphic. For ease of understanding, the labels have been kept the same. Note that even if the labels were shuffled, the graphs would still be isomorphic.



Example 1.5. Here the labels have again been kept the same.



Definition 1.6 (adjacency matrix)**.** Given a graph $G = (V, E)$, its adjacency matrix is a $|V| \times |V|$ matrix where the entry in row i and column j for all i, j is 1 if there is an edge from vertex i to vertex j and 0 otherwise. Without loss of generality, assume the vertices are indexed from 1 to $|V|$.

1.2 Prior results

1.2.1 Weisfeiler–Lehman

The Weisfeiler–Lehman (WL) algorithm is an algorithm for graph isomorphism first formulated in 1968 by Weisfeiler and Lehman [WL68]. Despite its age, the algorithm has important and powerful combinatorial and first order logical properties which make it relevant whenever analysing graphs for isomorphism.

The algorithm relies on color refinement, which is easiest to think of in terms of a hash function. Let f be an injective function from finite sets to real numbers. The WL algorithm iteratively assigns each node a real number value output by f, i.e. a color, based on the colors of its neighbors. Following the exposition from [HV21], the simplest version of the algorithm is stated as follows:

Algorithm 1 Weisfeiler–Lehman

Input: $G = (V, E)$

1: Let $f : S \hookrightarrow \mathbb{R}$ ▷ a hash function of sets
2: $\forall v \in V : c_v^0 = f(\deg(v))$ ▷ c_v^t is the color of vertex v at time t
3: **repeat**
4: $\quad \forall v \in V : c_v^t \leftarrow f(c_v^{t-1}, \{\!\{c_w^{t-1} \mid (v, w) \in E\}\!\})$ ▷ f takes the color at the previous iteration and the multiset of all neighboring colors
5: **until** $\forall v \in V : c_v^t = c_v^{t-1}$
6: **return** $\{\!\{c_v^t \mid v \in V\}\!\}$

G and H are distinguished by WL if and only if $WL(G) \neq WL(H)$ where $WL(G)$ is the output of algorithm 1.

This version of the WL algorithm is referred to as 1-dimensional WL. Before we continue, we must define arbitrary dimensional WL, or k-WL, explored by Cai, Fürer, and Immerman. See [CFI92] for the history.

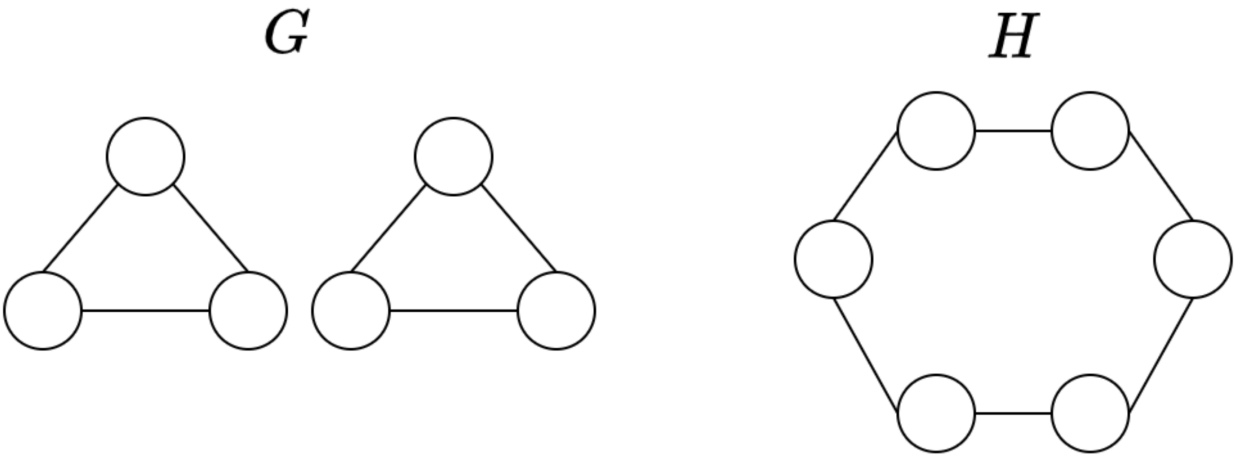
Algorithm 2 k-dimensional Weisfeiler–Lehman ($k \geq 2$)

Input: $G = (V, E)$

1: Let $f : S \hookrightarrow \mathbb{R}$
2: $\forall \vec{v} \in V^k : c_{\vec{v}}^0 = f(\text{type}(\vec{v}))$ ▷ $\vec{v}$ is a k-tuple of vertices and type is the isomorphism type of the induced subgraph of $\vec{v}$
3: **repeat**
4: $\quad \forall \vec{v} \in V^k, w \in V : c_{\vec{v},w}^t \leftarrow (c_{\vec{v}_{1/w}}^{t-1}, \ldots, c_{\vec{v}_{k/w}}^{t-1})$ ▷ $\vec{v}_{i/w}$ is $\vec{v}$ with w substituted at index i
5: $\quad \forall \vec{v} \in V^k : c_{\vec{v}}^t \leftarrow f(c_{\vec{v}}^{t-1}, \{\!\{c_{\vec{v},w}^{t-1} \mid w \in V\}\!\})$
6: **until** $\forall \vec{v} \in V : c_{\vec{v}}^t = c_{\vec{v}}^{t-1}$
7: **return** $\{\!\{c_{\vec{v}}^t \mid \vec{v} \in V\}\!\}$

We call the final value of t in algorithm 2, the number of rounds that the algorithm took.

Example 1.7 ([HV21] Fig. 2 and 3). Here is an example of 2-WL.



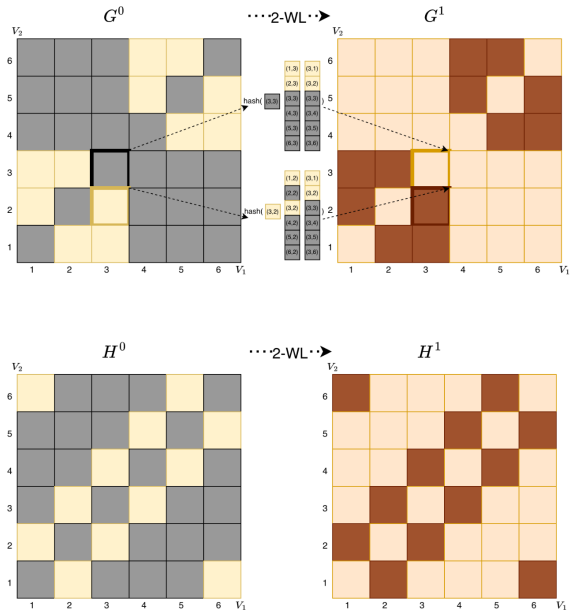


Fig. 2. 2-WL color assignment at initialization G^0, H^0 and refinement at first round G^1, H^1. The color refinement stabilizes in one step, failing to distinguish G and H

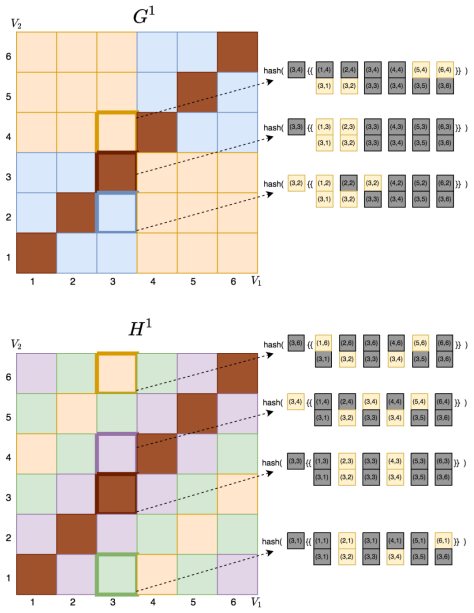


Fig. 3. 2-FWL color refinement at first round G^1, H^1 outputs different color patterns for G and H. Thus it distinguishes G and H, and terminates in one iteration.

k-WL is used in many places, most notably in Babai's quasipolynomial time graph isomorphism algorithm [Bab16]. As mentioned previously, WL has very important first order logical properties, which will become relevant later on.

Theorem 1.8 ([CFI92]). *Let φ be a first order logical statement (allowing counting) that distinguishes two graphs, i.e. is true for one and false for the other. Two graphs are distinguished by φ using $k-1$ variables and quantifier depth d if and only if k-WL distinguishes the graphs in d rounds.*

1.2.2 Polynomial Calculus

The polynomial calculus (PC) proof system, first introduced by Clegg, Edmonds, and Impagliazzo [CEI96], is an algebraic proof system. Begin with a set of polynomial axioms $F = \{f_0, \ldots, f_n\}$ over a field $\mathbb{F}$ in variables $X = \{x_0, \ldots, x_m\}$. The basic idea is to use a

sequence of derivations which should always output a contradiction if the system of polynomials has no solution over $\mathbb{F}$. The derivation rules for the system are as follows:

1. From nothing, we may derive any $f_i \in F$.
2. From nothing, we may derive $x_i^2 - x_i$ for any $x_i \in X$.
3. From any polynomial g we have derived, we may derive $x_i g$ for any $x_i \in X$.
4. From two polynomials g, h we have derived, we may derive $ag + bh$ for constants a, b.

If 1 (or indeed any constant by the fourth derivation rule) is ever derived using these rules, it means that F has no solutions. We refer to this derivation of 1 as a refutation. The degree of the refutation is the maximum degree of any polynomial derived during the process. The PC proof system is sound and complete, i.e. a refutation exists if and only if F has no solution.

Theorem 1.9 ([CEI96]). *If a PC refutation exists given $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$, a refutation of degree d can be found in time $n^{O(d)}$.*

1.2.3 Connection between WL and PC

In 2015, Berkholz and Grohe showed a connection between WL and PC [BG15]. In their approach, they introduce and utilize a restriction of PC called monomial-PC. In monomial-PC, we require that the g in the third derivation rule is either a monomial or the product of a monomial and an axiom.

Proposition 1.10 ([BG15]). *Let A, B be the adjacency matrices of two graphs G, H respectively. $G \cong H \iff \exists X : AX = XB$ and X is a permutation matrix.*

Given a matrix X let x_{ij} be a variable which takes the value at row i and column j. Consider the following sets of polynomials and let their union be called P. Let V_G, E_G (resp. V_H, E_H) be the vertex set and edge set of G (resp. H).

$$\sum_{v \in V(G)} x_{vw} - 1 = 0 \qquad \forall w \in V(H) \quad (1)$$

$$\sum_{w \in V(H)} x_{vw} - 1 = 0 \qquad \forall v \in V(G) \quad (2)$$

$$x_{vw}x_{v'w'} = 0 \quad \forall v, v' \in V(G), w, w' \in V(H) : (v, v') \in E(G) \oplus (w, w') \in E(H) \wedge v = v' \iff w = w' \quad (3)$$

$$x_{vw}^2 - x_{vw} = 0 \qquad \forall v \in V(G), w \in V(H) \quad (4)$$

The set of solutions to P are exactly the set of permutation matrices X from proposition 1.10 such that $AX = XB$. Note that the 4th set of polynomials are already axioms of PC. The first set of polynomials forces each row to have only one nonzero entry. The second set forces each column to have only one nonzero entry. The third set of polynomials forces the permutation matrix to preserve isomorphisms.

Theorem 1.11 ([BG15] Thm. 4.4). *Let $\mathbb{F}$ be a field of characteristic 0. For two graphs G, H, G and H are distinguishable by k-WL if and only if there is a degree k monomial-PC refutation of P over $\mathbb{F}$.*

1.3 Representation theory

We need to define some important representation theoretic terms and concepts that will become important later. We assume the reader is comfortable with undergraduate group theory and linear algebra. First, let us give some motivation. Consider the following alternative definition of an isomorphism of graphs:

Definition 1.12 (isomorphism)**.** Given two n-vertex graphs $G, H, G \cong H \iff \exists \sigma \in S_n : G = \sigma(H)$.

In other words, if there exists a permutation of vertices of one graph that turns it into the other, the two are isomorphic.

We can "represent" graph properties with polynomials as follows. Let A be the adjacency matrix of an arbitrary graph G on n nodes. For each row i and column j of A, let x_{ij} be a variable which takes a value for a fixed graph as in the previous section.

We can encode graph properties via sets of polynomials vanishing.

Example 1.13. $\sum_i x_{ii} = 0 \iff G$ is a loopless graph.

Example 1.14. $\forall i, j : x_{ij} - x_{ji} = 0 \iff G$ is undirected.

Now let us define what we mean by "represent."

Definition 1.15 (representation)**.** Given a group G and a vector space V over a field F, a *representation* is a homomorphism $\rho : G \to GL(V)$. If $\dim_F V = k < \infty$, we can use the alternate notation $\rho : G \to GL_k(F)$.

We sometimes refer to the vector space V as the representation if the homomorphism is obvious. Note: $G \curvearrowright V$.

Definition 1.16 (subrepresentation)**.** Given a representation (ρ, V), a *subrepresentation* is a subspace W of V that is invariant under the action of G. Specifically, $\rho(g)W \subseteq W, \forall g \in G$.

What we really mean is that if we instead took a homomorphism of group elements into just the subspace $GL(W)$ it would be equivalent to how the homomorphism of elements into $GL(V)$ acts on W.

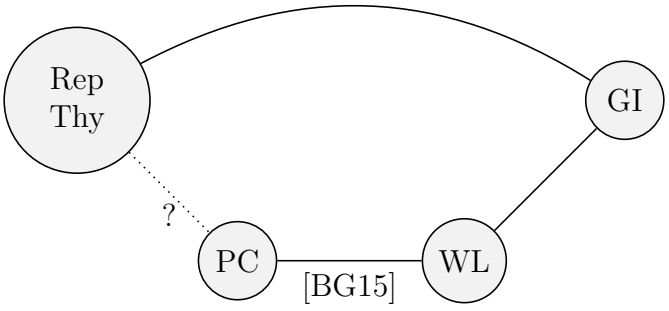
Definition 1.17 (irreducible representation)**.** An *irreducible representation* or irrep is a representation that has no nonzero proper subrepresentation.

The irreps of S_n are very well studied [Jam78][FH91][Sag01][Ste11]. In fact, we can list all irreps for any n and compute them efficiently.

Definition 1.18 (polynomial model)**.** A *polynomial model* of a G-representation V is a polynomial ring R on which G acts by ring automorphisms, together with a subrepresentation U of R, such that U is equivalent to V as a G-representation.

Remember that representations are vector spaces, meaning they have bases. Polynomial models have sets of polynomials as bases. Any polynomial in a polynomial model is a linear combination of irrep basis polynomials.

Proposition 1.19. *A polynomial vanishes on an orbit if and only if the irrep it belongs to vanishes.*



1.4 The Big picture

Berkholz and Grohe showed a connection between PC and GI through WL. There is obviously a connection between representation theory and GI, but it is not known how helpful that connection is. In our work, we explore the connection between representation theory and PC.

We classify ways to distinguish graphs via differences between slices in three ways increasing in distinguishing power.

1. Separating modules: specific polynomials that vanish on one graph but not the other.
2. Multiplicity obstructions: an isomorphism type of irreps that occurs in degree d such that more irreps of that type vanish on one graph than the other.
3. Occurrence obstructions: an isomorphism type of irreps that occurs in degree d such that at least one irrep of that type vanishes on one graph but none vanish on the other.

2 Polynomials and Weisfeiler–Lehman

Crucial to the WL algorithm is the ability to access node degrees (line 2).

Example 2.1. Consider the following polynomial model. Let $G = S_n, F = \mathbb{C}, V$ be the space of adjacency matrices of graphs with n vertices. For generality, let the entries of these adjacency matrices be $\{0, \lambda\}$ for $\lambda \in \mathbb{C}$.

$$V_d^* = \text{Span}\{\sum_j x_{ij} - dz \mid \forall i\}$$

Where, z is a variable that takes value λ and d is the node degree we are interested in. ρ maps permutations in S_n to permutation matrices. $S_n \curvearrowright V$ in the same way as in definition 1.12 (via conjugation). $\sum_j x_{ij} - dz$ will equal zero if and only if the node degree of i is equal to d. In other words, all polynomials in V_d^* vanish if and only if G is d-regular. Thus, we have degree-1 polynomials whose vanishing represent node degree.

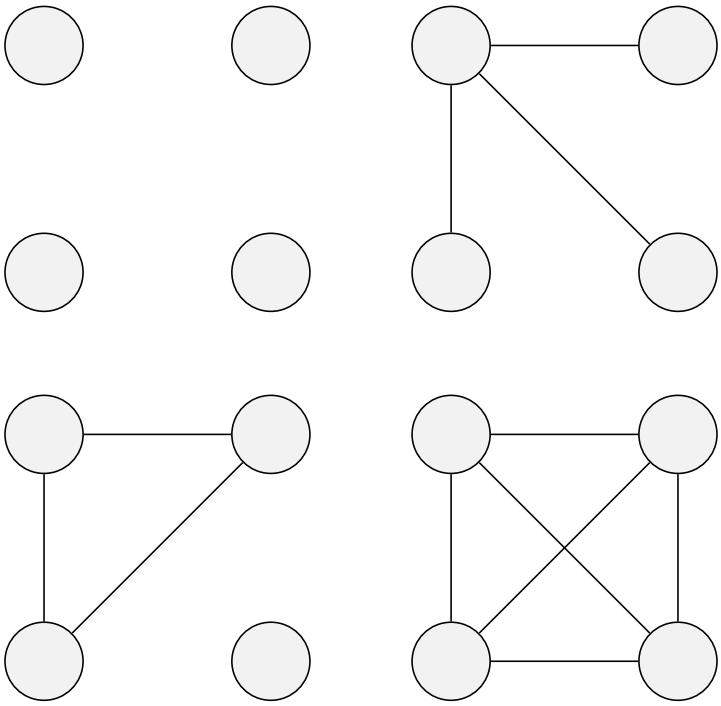
2.1 Properties detectable by degree-1 polynomials

In order to see what each irrep encodes, we must find what irreps occur in each representation.

Proposition 2.2. *The degree-1 polynomials of V_d^*, denoted $V_d^*[1]$, break into irreps as follows. There are 3 copies of the trivial representation, 3 copies of the reflection representation (also called the standard representation), and one copy each of the representations of shape $(n-2, 2)$ and $(n-2, 1, 1)$.*

We have the following:

1. *The diagonal trivial irrep vanishes if and only if there are no self loops.*
2. *The sum of all entries trivial irrep vanishes if and only if the graph has no edges.*
3. *The diagonal entry reflection irrep vanishes if and only if the number of self loops in the graph is n or 0, i.e. every node has a self loop or none do.*
4. *The row sum reflection irrep vanishes if and only if every node has the same outdegree.*
5. *The column sum reflection irrep vanishes if and only if every node has the same indegree.*
6. *The irrep of shape $(n-2,1,1)$ vanishes if and only if every induced subgraph on three vertices has the same number of directed edges going clockwise as counterclockwise. In particular, this irrep always vanishes on undirected graphs.*
7. *The irrep of shape $(n-2,2)$ vanishes if every induced subgraph on four vertices is one of the following. There is a more complex corresponding condition for directed graphs.*



Proof Sketch. The total dimension of the degree-1 polynomials is n^2+1, as the adjacency matrix is $n \times n$ and we also have z. Note that all irreps are polynomials over x_{ij}, z.

z spans a trivial representation, because S_n acts trivially on z by definition.

The sum of diagonal entries of the adjacency matrix spans a trivial representation.

The sum of all entries of the adjacency matrix spans a trivial representation.

The set of polynomials $x_{ii}-x_{jj}$ for all i,j is a basis of size $n-1$. Furthermore, note that since $\sum_{i=1}^{n} x_{ii}$ is a trivial representation, this must be orthogonal, meaning it must be a reflection representation. In other words, each polynomial vanishing implies that the diagonal entries of row i and j are equal. This set of polynomials has a basis $\{x_{11}-x_{22}, x_{22}-x_{33}, \cdots\}$ of size $n-1$, so this is a reflection representation.

The set of polynomials $\sum_j (x_{ij}-x_{i'j})$ for all i,i' is a reflection representation. This irrep vanishes when all rows of the adjacency matrix have the same sum.

Similarly, the set of polynomials $\sum_j (x_{ji} - x_{ji'})$ for all i, i' is also a reflection representation. This irrep vanishes when all columns of the adjacency matrix have the same sum.

The remaining dimension is $n^2 - 3n + 1$. This implies that there is an irrep of shape $(n-2, 2)$ and one of shape $(n-2, 1, 1)$. We check this by comparing the group action on the these representations and see that they are identical to the action on the corresponding standard Young tableaux. To describe the polynomial bases of these irreps, let C be the matrix of coefficients of all polynomials in variables x_{ij}, i.e. C_{ij} is the coefficient for x_{ij}. Since these irreps are orthogonal to all the other ones, for any polynomial basis element in either irrep, C must have zeroes on the diagonal, and all row and column sums must equal zero. For a polynomial to be a basis element of the irrep of shape $(n-2, 2)$, C must be a symmetric matrix. For a polynomial to be a basis element of the irrep of shape $(n-2, 1, 1)$, C must be a skew-symmetric matrix. □

Remark 2.3. This means that $V_d^*[1]$ is exactly the same for every degree, as the spans all overlap. Thus, from now on, we will refer to the representation as simply $V^*[1]$.

We can also look at what different multiplicities of the same irreps vanishing means. This is not interesting for the trivial representations. For the reflection representations, not only does the number of vanishing reflection representations contain information, but which ones specifically vanish contains information too.

Proposition 2.4. *For each vertex i of the graph, let i_{refl} be a 3-tuple with entries equalling the number of self-loops of i, the indegree of i, and the outdegree of i. For the whole graph, we get a point cloud in $\mathbb{R}^3$. If just one reflection representation vanishes, the point cloud lies on a plane. If two vanish, the point cloud lies on a line. If all three vanish, the point cloud is just a single point. In this case, the graph is also regular.*

Example 2.5. Consider the undirected triangle graph. On this graph, all three reflection representations vanish.

2.2 Simulating logic

Recall Theorem 1.8 linking k-WL and first order logic. Looking at our representation $V^*[1]$, we see that we can simulate quantified statements using products of polynomials. If we want to know if all polynomials vanish, we can simply take a product of them. Existential quantifiers are a bit more tricky, but can be done, as we describe below.

Immerman describes the FO language of graphs as follows, $G = (V, E)$ for universe V and binary predicate E (the edge relation), $\vee, \neg, =$, and $\exists$. Of course, we care about $\wedge, \to, \forall$, but each can be made using a double negation, which we exhibit below.

Let M be the $\{0, \lambda\}$ adjacency matrix of graph G. Let φ, ψ be logical statements and f, g be their corresponding polynomials such that:

$$f(M) = \begin{cases} 0 & G \not\models \varphi \\ \lambda^{\deg(f)} & G \models \varphi \end{cases}$$

	Polynomial	Degree
$E(i,j)$	x_{ij}	1
$\neg\varphi$	$z^{\deg(f)} - f$	$\deg(f)$
$\varphi \wedge \psi$	fg	$\deg(f) + \deg(g)$
$f\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$	$z^{\deg(f)+\deg(g)} - (z^{\deg(f)} - f)(z^{\deg(g)} - g)$	$\deg(f) + \deg(g)$
$\forall i : \varphi_i$	$\prod_i f_i$	$\sum_i \deg(f_i)$
$\exists i : \varphi_i$	$z^{\sum_i \deg(f_i)} - \prod_i (z^{\deg(f_i)} - f_i)$	$\sum_i \deg(f_i)$

Note that variables of the polynomials do not correspond one-to-one with variables of the logic of graphs. In some sense, the x_{ij}'s correspond to pairs of variables in FO logic, i.e. the universe can be thought of as $V \times V$. Furthermore, $=$ is not in this table, as any statement that requires an $=$ or an $\neq$ can do so in the product declaration of the polynomial.

It turns out, even ignoring the degree blowup for quantifiers, it seems that there is no correspondence between the polynomial degrees and WL-dimension. The degree appears to be completely dependent on the number of calls to the edge predicate in the logical formula. In polynomials with variables corresponding to edges, the degrees of polynomials relate to how many times we look at edges.

3 Simulating our approach with Polynomial Calculus

We decided to turn our attention back to Berkholz and Grohe's approach. It's important to outline the difference between our approach and their approach. Their approach sets up polynomial equations that are satisfied if there exists an isomorphism between graphs. Specifically, if there exists a permutation matrix X such that $AX = XB$. Our approach looks at the orbits of the two graphs under the action of the symmetric group. Their orbits are equivalent if and only if they are isomorphic.

Our question became: can we encode orbit equivalence with equations?

Proposition 3.1.

$$A \ncong B \iff \exists f : f(\mathcal{O}_A) = 0 \wedge f(B) = 0$$

In other words, A and B are in the same orbit if $f(X^\top AX) = f(B)$ for any polynomial f.

Proof. Let $\mathcal{O}_A$ and $\mathcal{O}_B$ be the orbits of A and B, i.e. all permutations of A and B. Let U be an irrep of polynomials in variables over $n \times n$ matrices. If U vanishes on the orbit of A, but not on the orbit of B, i.e. $U(\mathcal{O}_A) = 0$ and $U(\mathcal{O}_B) \neq 0$, then $\exists f \in U$ such that $f(A) = f(\mathcal{O}_A) = 0$ and $f(B) \neq 0$. Moreover, $f(X^\top AX) = 0$. □

Thus, $A \ncong B \iff \mathcal{O}_A \cap \mathcal{O}_B = \emptyset \iff \exists f : f(\mathcal{O}_A) = 0 \wedge f(B) \neq 0$, and in particular via polynomial calculus, $f(B) = 1$.

Berkholz and Grohe proved that we could obtain $AX - XB$. In order to show that our approach can be simulated, we need to show that we can obtain $X^\top AX - X^\top XB = X^\top AX - B$. From there, given some polynomial which vanishes on $\mathcal{O}_A$ but not on $\mathcal{O}_B$, we need to obtain $f(X^\top AX) - f(B)$. Finally, if we can obtain $f(X^\top AX)$ on its own, we can derive $f(X^\top AX) - f(B) - f(X^\top AX) = f(B)$ which will be nonzero. Note that we cannot derive $f(B)$ immediately, since B is a constant in PC. We must do all of this in monomial-PC.

3.1 Results

We can simulate our approach using Berkholz and Grohe's approach.

Theorem 3.2. *Given two graphs G, H, if there exists a representation in degree d that vanishes on $\mathcal{O}_G$ but not on $\mathcal{O}_H$, then the Berkholz-Grohe equations have a monomial-PC refutation of degree $2d$, and in particular $2d$-WL distinguishes G and H.*

First, we must derive $X^\top AX - B$. In order to do that, we must show $X^\top X$ gives the identity matrix.

Lemma 3.3. *There is a degree-2 monomial-PC derivation of $X^\top X - I$ from P.*

Proof. Starting with 4, we can derive

$$\sum_w (x_{vw}^2 - x_{vw})$$

Then subtract 2 to get

$$\sum_w (x_{vw}^2 - x_{vw}) - \sum_w x_{vw} - 1 = \sum_w x_{vw}^2 - 1$$

This is true when the diagonal entries are one. For the non-diagonals, start with 3 and derive

$$\sum_{w=w', v \neq v'} x_{vw} x_{v'w'}$$

This immediately gives us what we want. All of this is done in monomial-PC. □

Next we prove that we can derive $f(X^\top AX) - f(B)$ using 1, 2, and 4. Note that since we have derived $X^\top AX - B$, we can use $X^\top AX_{ij} - B_{ij}$ for all i, j. For the proof, we will generalize this to $x_i - y_i$.

Lemma 3.4. *Given any polynomial f, from $\{x_i^2 - x_i\} \cup \{y_i^2 - y_i\} \cup \{x_i - y_i\}$, $f(\vec{x}) - f(\vec{y})$ can be derived in monomial-PC by a derivation of degree* $\deg(f)$.

Proof. We can construct $f(X) - f(Y)$ for any f that is a multivariate monomial where each variable has degree at most 1. Without loss of generality, assume each variable in f is indexed without any missing indices and each degree is exactly 1. The general formula is

$$\sum_{i=1}^{n} (x_i - y_i) \prod_{j=i+1}^{n} x_j \prod_{k=1}^{i-1} y_k$$

This is clearly in monomial-PC.

We can further convert this to arbitrary monomials. Assume we want to increase the degree of the index 1 variable. Starting from $x_1 x_2 \cdots x_n - y_1 y_2 \cdots y_n$, we can add $(x_1^2 - x_1) x_2 \cdots x_n$. This will produce $x_1^2 x_2 \cdots x_n - y_1 y_2 \cdots y_n$. We can do the same thing for the y's. Now that we can derive $f(X) - f(Y)$ where f is an arbitrary monomial, we can combine them straightforwardly to get an arbitrary polynomial. This is all done in monomial-PC and the highest degree of any monomial derived during the process is equal to the highest degree of f. □

We now just need to derive $f(X^\top AX)$ by itself. To do that, we prove something stronger.

Lemma 3.5. *P is a Gröbner basis [CLO15].*

Proof. Fix a degree-lexicographic ordering of monomials. We need to check $2n-1$ remaninders. The first, is between 1 where $w = 1$ the first row and 2 where $v = 1$. The general formula is

$$\begin{aligned} x_{11} + x_{12} + \cdots + x_{1n} - 1 - (x_{11} + x_{21} + \cdots + x_{n1} - 1) \\ x_{11} + x_{12} + \cdots + x_{1n} - 1 - x_{11} - x_{21} - \cdots - x_{n1} + 1 \\ x_{12} + \cdots + x_{1n} - x_{21} - \cdots - x_{n1} \end{aligned}$$

We can then subtract off column sum from 1 with same starting monomial by above ordering.

$$\begin{aligned}
x_{12}+\cdots+x_{1n}-x_{21}-\cdots-x_{n1}-x_{12}-x_{22}-\cdots-x_{n2}+1 &\quad \text{Subtract}\\
x_{13}\cdots+x_{1n}-x_{21}-\cdots-x_{n1}-x_{22}-\cdots-x_{n2}+1 &\quad \text{Simplify}\\
x_{13}\cdots+x_{1n}-x_{21}-\cdots-x_{n1}-x_{22}-\cdots-x_{n2}+1-x_{13}-x_{23}-\cdots-x_{n3}+1 &\quad \text{Subtract}\\
x_{14}\cdots+x_{1n}-x_{21}-\cdots-x_{n1}-x_{22}-\cdots-x_{n2}-x_{23}-\cdots-x_{n3}+2 &\quad \text{Simplify}\\
\vdots\\
-(x_{21}+x_{22}+\cdots+x_{2n})-(x_{31}+x_{32}+\cdots+x_{3n})-\cdots-(x_{n1}+x_{n2}+\cdots+x_{nn})+n\\
-\sum_{i=2}^{n}(\sum_{j=1}^{n}x_{ij}-1)
\end{aligned}$$

The last polynomial is a sum of axioms from 2.

The remaining $2n$ remainders are between $x_{1i}+x_{2i}+\cdots+x_{ni}-1$ and $(x_{1i}^2-x_{1i})$ or $x_{i1}+x_{i2}+\cdots+x_{in}-1$ and $(x_{i1}^2-x_{i1})$. For the first case, the strategy is

$$\begin{aligned}
x_{1i}(x_{1i}+x_{2i}+\cdots+x_{ni})-1-(x_{1i}^2-x_{1i})\\
x_{1i}x_{2i}+\cdots+x_{1i}x_{ni}\\
x_{1i}x_{2i}+\cdots+x_{1i}x_{ni}-x_{2i}(x_{1i}+x_{2i}+\cdots+x_{ni}) &\quad \text{Subtract}\\
x_{1i}x_{3i}+\cdots+x_{1i}x_{ni}-x_{2i}(x_{2i}+\cdots+x_{ni}) &\quad \text{Simplify}\\
\vdots\\
-2x_{12}x_{13}-2x_{12}x_{14}-\cdots-2x_{12}x_{1n}-2x_{13}x_{14}-\cdots-2x_{13}x_{1n}-\cdots-2x_{1(n-1)}x_{1n}
\end{aligned}$$

The strategy is symmetric for the other case. In essence, if we start with a column sum, we are left with a sum of $-2x_{ij}x_{ij'}$ where j and j' are two different columns. If we start with a row sum, we are left with a sum of $-2x_{ij}x_{i'j}$ where i and i' are two different rows. These binomials are all present in 3. All of this is done in monomial-PC and uses degree twice that of any remainders. □

Since P is a Gröbner basis, we can derive $f(X^\top AX)$ using degree $2\deg(f)$. The only thing left to prove is that any such possible f is contained in the ideal generated by P if f^k is in the ideal for some k, i.e. the ideal generated by P is radical.

Proposition 3.6 (Proof due to Grochow)**.** *If $I \subseteq F[x_1,\ldots,x_n]$ contains $x_i^2-x_i$ for all i, then I is radical.*

Proof. Note that an ideal I is radical if and only if R/I has no nonzero nilpotent elements (aka R/I is reduced.)

Let $J=\langle x_i^2-x_i : i\in[n]\rangle$. Since each generator of J is on a different (set of) variable(s), we have $F[x]/J \cong \bigotimes_i F[x_i]/(x_i^2-x_i)$, which is the coordinate ring of the direct product $\prod_{i=1}^n\{0,1\}$, the Boolean cube. (It is not hard to show by direct calculation that in the univariate polynomial ring $\langle x_i^2-x_i\rangle$ is radical, since its generator is square-free $x_i(x_i-1)$.)

Now, we know that Spec$F[x]/J$ consists of 2^n isolated points, so we must have $F[x]/J \cong F\times\cdots\times F$, one copy of F for each of the 2^n points. The isomorphism is given by taking a polynomial $f\in F[x]/J$ to its 2^n-tuple of evaluations on the points of the Boolean cube. Let π be this isomorphism.

But now suppose $J\subseteq I\subseteq F[x]$. Then $F[x]/I \cong F^{2^n}/\pi(I)$. But as F is a field, the only ring quotients of $F\times\cdots\times F$ are again of the form F^k for some $k\le 2^n$, and thus $F[x]/I$ is a subring of a direct product of fields, hence cannot have nonzero nilpotent elements. □

It is also clear that this argument generalizes to any ideal of the form $\langle f_1(x_1), f_2(x_2), \ldots, f_n(x_n)\rangle$ where each f_i is a squarefree univariate polynomial, instead of the ideal of Boolean equations J.

Proof of Theorem 3.2. Suppose V is a separating module in degree d, and f is in V. Then we know that f vanishes on $\mathcal{O}_A$, so we know that some power of $f(X^\top AX)$ is in the ideal. But because the ideal is radical, in fact $f(X^\top AX)$ is in the ideal. Combining lemmas 3.3, 3.4, and 3.5, we can derive $f(X^\top AX) - f(B) - f(X^\top AX) = f(B)$. If f vanishes on $\mathcal{O}_A$ and not on $\mathcal{O}_B$, then we have a refutation in monomial-PC of degree $2d$. □

3.2 Implications

This implies that separating modules (1) can be simulated by $2d+1 = O(d)$ dimensional WL. Moreover, since the orbits of graphs are finite, thus closed, if two graphs are not isomorphic, then their orbits under the action of S_n are disjoint. This is also true in projective space, meaning that there must exist a homogeneous polynomial h that vanishes on one graph and not the other.

Lemma 3.7 ([Gro15] Lemma 3.1)**.** *If any property can be used to proved a lower bound against a G-invariant complexity class, then a G-invariant property can be used to prove the same lower bound.*

Corollary 3.8. *If we consider the S_n representation that h generates, it vanishes entirely on one graph, but not the other, which means in particular that there exists an irrep which vanishes on one graph but not the other.*

This means that separating modules are no stronger than monomial-PC and thus no stronger than WL.

4 Approach via multiplicities

We are now looking at multiplicity obstructions. Our first approach was to check all possible combinations of irrep multiplicities and see if it is even possible to distinguish all graphs. We only need to consider representations of S_n/G for subgroups $G \leq S_n$, because for any algebraic group Γ, if $\mathrm{Orb}_\Gamma(x)$ is a variety, then it is isomorphic to $\Gamma/\mathrm{Stab}_\Gamma(x)$.

4.1 Results

One natural way to relate subgroups of S_n or permutation groups, to graph isomorphism is via automorphism groups.

Definition 4.1 (graph automorphism group)**.** Consider the set of all automorphisms of a graph G, this forms a group, and in particular a permutation group.

By the above fact about algebraic groups, conjugate automorphism groups do not distinguish graphs via multiplicities.

Definition 4.2 (cycle index)**.** The cycle index of a group is the generating function for the multiset of its cycle types.

Theorem 4.3. *The multiset of irrep multiplicities of S_n/G are in bijection with cycle indices of G.*

Proof. Let M be the character table of S_n. Let u be the cycle index of G and v be the cycle index of H for G, H subgroups of S_n. The multiplicities of S_n/G is Mu by Frobenius reciprocity, and similarly for Mv.

For the first direction, if $u = v$, then the multiplicites of S_n/G equal those of S_n/H and so $S_n/G = S_n/H$. For the other direction, if $Mu = Mv$, then because M is square and invertible, then $u = v$. □

4.2 Current work

There are examples of non-conjugate automorphism groups that have the same cycle indices.

Example 4.4. $\langle (1854)(2376), (27)(36), (16)(28)(35)(47) \rangle = \mathbb{Z}_4 \circ D_4$ and $\langle (27)(36), (1746)(2835) \rangle = \mathbb{Z}_2^2 \rtimes \mathbb{Z}_4$ are two groups that are not conjugate, but have the same cycle indices.

We are currently looking at when non-conjugate automorphism groups with identical cycle indices arise.

Proposition 4.5 ([Wie94][Cam04])**.** *A permutation group $G \leq S_n$ on Ω a set of n elements is said to be 2-closed if every permutation of Ω which preserves all the G-orbits in $\Omega \times \Omega$ belongs to G. More generally, the 2-closure of G is the group of permutations which preserve all the G-orbits on $\Omega \times \Omega$.*

All automorphism groups of graphs are 2-closed.

This means that we only need to look at 2-closed permutation groups. Here is the Sagemath code we are using.

```
    def duplicate_two_closed(n): # Function takes n and returns the (order, group) of any
    ↪  2-closed subgroup of S_n whose cycle index is indistinguishable from another
        G = SymmetricGroup(n)
        subgroups = G.conjugacy_classes_subgroups()
        print("Filtering 2-closed subgroups")
        two_closed_subgroups = [sg for sg in subgroups if is_two_closed(sg)]
        orders = [sg.order() for sg in two_closed_subgroups]
        print("Checking for duplicate orders")
        duplicate_orders = [order for order in orders if orders.count(order) > 1]
        duplicate_order_subgroups = [sg for sg in two_closed_subgroups if sg.order() in
        ↪  duplicate_orders]
        cycle_indices = []
        total = len(duplicate_order_subgroups)
        print("There are "+str(total)+" duplicate order subgroups that are 2-closed")
        for sg in duplicate_order_subgroups:
            cycle_indices.append(sorted([g.cycle_type() for g in sg.list()]))
        indices = [i for i, elem in enumerate(cycle_indices) if cycle_indices.count(elem)
        ↪  > 1]
        all_duplicate_subgroups = [(duplicate_orders[i], duplicate_order_subgroups[i])
        ↪  for i in indices]
        return all_duplicate_subgroups

    def is_two_closed(G): # Checks if a group is 2-closed
        try:
            return str(gap.IsTransitive(G)) == "true" and gap.Order(gap.TwoClosure(G)) ==
            ↪  G.order()
        except:
            print(G, 'produced an error')


    def automorphism_groups_of_constructed_graphs(group,n): # Constructs graphs from
    ↪  group using algorithm from Cameron (n is S_n)
```

```
        group_orbits=list(libgap(group).Orbits(tuples([1..n], 2),libgap.OnTuples))[1:]
        orbits_powerset = group_orbits
        r = range(2, len(group_orbits) // 2 + 1)
        for i in r:
            print(i / max(r))
            orbits_powerset += [list(x) for x in itertools.combinations(group_orbits, i)]
        for subset in orbits_powerset:
            G = Graph()
            if len(subset[0]) <= 2:
                G.add_edges([tuple(x) for x in subset])
            else:
                for i in subset:
                    G.add_edges([tuple(x) for x in i])
            autG = G.automorphism_group()
            print(autG, autG.order())
```

References

Babai, László. "Graph isomorphism in quasipolynomial time". In: *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '16. Cambridge, MA, USA: Association for Computing Machinery, 2016, pp. 684–697. ISBN: 9781450341325. DOI: 10.1145/2897518.2897542. URL: https://arxiv.org/abs/1512.03547.

Berkholz, Christoph and Martin Grohe. "Limitations of Algebraic Approaches to Graph Isomorphism Testing". In: *Automata, Languages, and Programming*. Springer Berlin, Heidelberg, 2015, pp. 155–166. ISBN: 9783662476727. DOI: 10.1007/978-3-662-47672-7_13. URL: http://arxiv.org/abs/1502.05912.

Cai, Jin-Yi, Martin Fürer, and Neil Immerman. "An optimal lower bound on the number of variables for graph identification". In: *Combinatorica* 12.4 (Dec. 1992), pp. 38–410. ISSN: 1439-6912. DOI: 10.1007/bf01305232. URL: http://dx.doi.org/10.1007/BF01305232.

Cameron, Peter. "Automorphisms of Graphs". In: *Topics in Algebraic Graph Theory*. Ed. by L. W. Beineke and R. J. Wilson. Cambridge University Press, Cambrige, 2004, pp. 137–155. ISBN: 0521801974. DOI: 10.1017/CBO9780511529993.008.

Clegg, Matthew, Jeffery Edmonds, and Russell Impagliazzo. "Using the Groebner basis algorithm to find proofs of unsatisfiability". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 174–183. ISBN: 0897917855. DOI: 10.1145/237814.237860.

Cox, David A, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms*. en. 4th ed. Undergraduate texts in mathematics. Cham, Switzerland: Springer International Publishing, May 2015.

Fulton, W and J Harris. *Representation theory*. en. Graduate Texts in Mathematics. New York, NY: Springer, Oct. 1991.

Grochow, Joshua A. "Unifying Known Lower Bounds via Geometric Complexity Theory". In: *computational complexity* 24.2 (May 2015), pp. 393–475. ISSN: 1420-8954. DOI: 10.1007/s00037-015-0103-x.

Huang, Ningyuan Teresa and Soledad Villar. "A Short Tutorial on The Weisfeiler–Lehman Test And Its Variants". In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, June 2021, pp. 8533–8537. DOI: 10.1109/icassp39728.2021.9413523.

James, G D. *The representation theory of the symmetric groups*. en. 1978th ed. Lecture notes in mathematics. Berlin, Germany: Springer, Oct. 1978.

Sagan, Bruce E. *The symmetric group*. en. 2nd ed. Graduate Texts in Mathematics. New York, NY: Springer, Apr. 2001.

Steinberg, Benjamin. *Representation theory of finite groups*. en. 2012th ed. Universitext. New York, NY: Springer, Oct. 2011.

Weisfeiler, B. and A. Lehman. "A Reduction of a Graph to a Canonical Form and an Algebra Arising During This Reduction". In: *Nauchno-Technicheskaya Informatsia* Ser. 2.N9 (1968), pp. 12–16.

Wielandt, H. "Permutation groups through invariant relations and invariant functions [83]". In: *Volume 1 Group Theory*. Ed. by Bertram Huppert and Hans Schneider. Berlin, New York: De Gruyter, 1994, pp. 237–296. ISBN: 9783110863383. DOI: 10.1515/9783110863383.237.