Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional

Unidad Zacatenco
Departamento de Matemáticas

Explorando el problema de Isomorfismo: Algoritmo K−WL y las gráficas de fichas

Tesis que presenta

David Gopar Morales

Para obtener el grado de

Maestro en Ciencias
en la Especialidad en Matemáticas

Director de Tesis

Ruy Fabila Monroy

Ciudad de México Julio 2024

Agradecimientos

Al CINVESTAV, en especial al Departamento de Matemáticas, les extiendo mi más profundo agradecimiento por brindarme las herramientas y el entorno necesarios para prosperar académicamente. Agradezco especialmente al CONAHCYT por su apoyo financiero durante mis estudios.

Quisiera expresar mi sincera gratitud al Dr. Ruy Fabila Monroy, mi asesor y guía durante todo el proceso, a la Dra. Ana Laura Trujillo Negrete, por todo el apoyo y asesoramiento en la teoría de gráficas y realización de esta tesis.

Un especial reconocimiento a mis colegas y compañeros del Departamento de Matemáticas, por las estimulantes discusiones, y horas de trabajo dedicadas a comprender las matemáticas. Su amistad y apoyo han sido un pilar en mi viaje.

No puedo dejar de agradecer a mis padres Marcela Ivonne Morales Arango y David Gopar Jarquín, por su amor incondicional, apoyo, comprensión y empuje, gracias a ellos he logrado lo que soy, a mis hermanos y familia, así como amigos y personas que estuvieron en los momentos más desafiantes de mi carrera. Su confianza y apoyo brindado en mí ha sido una fuente constante de motivación y fuerza para el ayer, hoy y mañana.

Finalmente, a todos los que de alguna manera han sido parte de este camino, mi más sincera gratitud. Este logro no solo representa un hito académico, sino también el fruto de un esfuerzo colectivo.

Esta tesis fue elaborada como parte del proyecto Ciencia de Frontera "Funciones y estructuras en gráficas y digráficas" (39570)

Resumen

Esta tesis se enfoca en el problema del isomorfismo de gráficas, el objetivo principal es demostrar el funcionamiento de un algoritmo que facilita la identificación en ocasiones de cuándo dos gráficas son isomorfas, es decir, cuando poseen una estructura intrínseca similar a pesar de las diferencias en sus representaciones.

La importancia del problema de isomorfismo radica en sus amplias aplicaciones, desde la química teórica hasta las redes de computadoras. Un enfoque tradicional para abordar este problema es mediante el análisis del espectro de la matriz de adyacencia de las gráficas. Sin embargo, esta metodología tiene limitaciones, ya que la información obtenida del espectro puede no ser suficiente para determinar el isomorfismo de manera efectiva en todos los casos.

En esta tesis, se revisa un algoritmo que se prueba es más fuerte que la información del espectro, el cual emplea gráficas de fichas, lo cual mejora significativamente la capacidad de discernir el isomorfismo entre gráficas.

La demostración de la efectividad del algoritmo se realiza a través de esta tesis exponiendo parte de la teoría necesaria para la comprensión de este, así cómo un primer enfoque sin fichas que nos ayuda a tener una idea de como funciona ya aplicado a fichas el cual es el algoritmo WL.

También este nos motiva a la posible extensión de comparar este algoritmo ahora con el espectro Laplaciano, para comparar la información que podemos obtener.

Abstract

This thesis focuses on the graph isomorphism problem, aiming to demonstrate the functioning of an algorithm that facilitates identifying when two graphs are isomorphic, meaning they possess a similar intrinsic structure despite differences in their representations.

The importance of the isomorphism problem lies in its broad applications, from theoretical chemistry to computer networks. A traditional approach to tackle this problem involves analyzing the spectrum of the adjacency matrix of the graphs. However, this methodology has limitations as the information obtained from the spectrum may not be sufficient to effectively determine isomorphism in all cases.

In this thesis, an algorithm is reviewed that is proven to be stronger than spectrum information, utilizing graph labeling which significantly enhances the ability to discern isomorphism between graphs.

The effectiveness of the algorithm is demonstrated throughout this thesis, presenting the necessary theory for understanding it, as well as an initial approach without graph labeling that provides insight into its functioning when applied with graph labeling, specifically the WL algorithm.

Additionally, this motivates us to explore the comparison of this algorithm with Laplacian spectrum, to evaluate the information that can be obtained.

Índice general

Capítulo 1

Introdución

1.1 Preliminares

A continuación presentaremos algunas de las definiciones que seutilizarán a lo largo de esta tesis, para lo cual nos apoyaremos principalmente del libro *Graph Theory* de Diestel [6]. Primeramente, dado un conjunto A denotaremos por $[A]^k$ como el conjunto de todos los subconjuntos de k-elementos de A.

Una *gráfica* es un par $G = (V, E)$ de conjuntos tales que $E \subseteq [V]^2$, por lo que los elementos de E son 2-subconjuntos de V. Los elementos de V serán los *vértices* de nuestra gráfica G, mientras que los elementos de E son las *aristas* de esta. Dada una gráfica G, se denota por $|G|$ al número de vértices de la gráfica, el cual diremos que es el orden de la gráfica, mientras que su cantidad de aristas será denotada por $||G||$. Una gráfica se dice que es finita, infinita, contable de acuerdo al orden de esta. En la presente tesis consideraremos solamente gráficas finitas.
Dada una gráfica G se denotará a su conjunto de vértices por $V(G)$ y a su conjunto de aristas por $E(G)$, es decir, si tenemos otra gŕafica $H = (W, F)$, tendremos que su conjunto de vértices es $W = V(H)$ y suconjunto de aristas es $F = E(H)$.
Decimos que dos vértices x e y de G son adyacentes o vecinos si $\{x, y\}$, es una arista de G. Ocasionalmente podemos escribir xy para representar a una arista $\{x, y\}$ de G. Dos aristas de G son adyacentes si tienen un vértice en común. Se define como vecindad de un vértice v en una gráfica G, denotada por $N_G(v)$ a el conjunto de vértices que son adyacentes a este y el grado de un vértice v denotado por $d(v)$ será la cantidad de vértices adyacentes. Si todos los vértices de una gráfica G son adyacentes a pares se dice que G es una gráfica completa. Una gráfica completa de n vértices se denota por K^n. Un par de vértices o aristas se dice que son independientes si no son adyacentes. Más aún, un conjunto de aristas o vértices se

dice independiente si ningún par de elementos es adyacente.

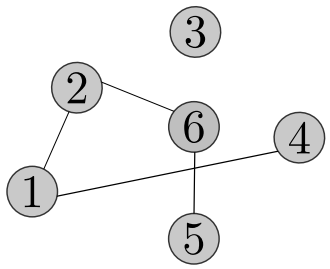

Figura 1.1: Gráfica G, donde $V = \{1, 2, 3, 4, 5, 6\}$
y $E = \{\{1, 2\}, \{2, 6\}, \{6, 5\}, \{1, 4\}\}$

Sean G y H dos gráficas, decimos que G y H son isomorfas y escribimos $G \simeq H$ si existe una biyección $\varphi : V(G) \mapsto V(H)$ tal que $(x, y) \in E(G) \iff (\varphi(x), \varphi(y)) \in E(H)$. A la función φ se le llamará *isomorfismo*. Además, en el caso que $G = H$ diremos que φ es un automorfismo de G. Usualmente no se distingue entre gráficas isomorfas por lo cual se escribe $G = H$ en lugar de $G \simeq H$, esto resalta la importancia de poder determinar si dos gráficas son isomorfas o no lo son.

Con lo anterior podemos decir que un automofismo es una permutación de los veŕtices de la gráfica que preserva incidencias. Consideremos el conjunto de todos los automorfismos de una gráfica G, es claro que la permutación identidad es un automorfismo, además si g es un automorfismo de G entonces también lo es su inversa g^{-1}, y si h es un automorfismo distinto, entonces la composición $g \circ h$ también será un automorfismo de G. Estas observaciones nos permiten ver que el conjunto de los automorfismos de G forman un grupo bajo la composición de funciones, el cual es llamado *el grupo de automorfismos de* G y se denota por $\mathrm{Aut}(G)$. El *grupo simétrico de* V $\mathrm{Sym}(V)$, es el grupo de todas las permutaciones del conjunto de vértices V, así que $\mathrm{Aut}(G)$ es subgrupo de $\mathrm{Sym}(V)$. En general, no es fácil determinar cuando dos gráficas dadas son isomorfas, o si una gráfica tiene un autorfismo diferente al automorfismo identidad. En algunos casos estas propiedades resultan ser obvias, como es el caso de la gráfica completa K_n, en donde cualquier permutación de sus vértices corresponde a un automorfismo, así que $\mathrm{Aut}(K_n) \simeq \mathrm{Sym}(K_n)$.

Se dice que una afirmación es *propiedad de gráficas* si esta se preserva bajo isomorfismos. Por ejemplo, "contener un triángulo" es una propiedad de gráficas, es decir, si G contiene 3 vértices adyacentes a pares, entonces cualquier gráfica isomorfa a G también los contiene. Una función η definida en la familia de gráficas es llamada *invariente de gráficas* si $\eta(G) = \eta(H)$ siempre que $G \simeq H$, es decir, asigna valores iguales a gráficas isomorfas.
Por ejemplo, la cantidad de aristas y la cantidad de vértices son invariantes de gráficas.
Si consideramos dos gráficas $G = (V, E)$ y $G' = (V', E')$, definimos la unión e intersección de gráficas de la siguiente manera $G \cup G' := (V \cup V', E \cup E')$ y $G \cap G' := (V \cap V', E \cap E')$.

Si se cumple que $G \cap G' = \emptyset$ entonces decimos que G y G' son disjuntas. Decimos que

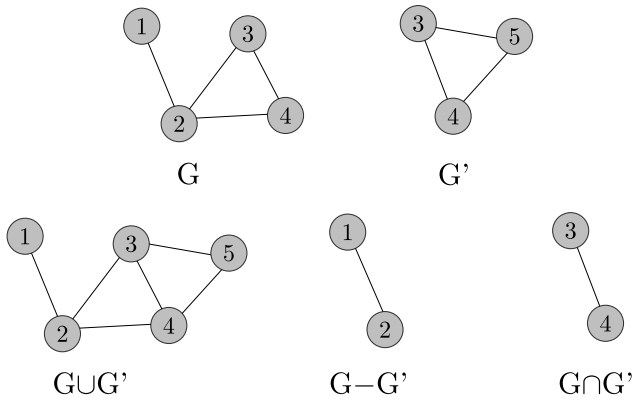

Figura 1.2: Se muestra la unión, diferencia e intersección de las gráficas G y G'.

G' es subgráficas de G (y G supergráfica de G') si se cumple que $V' \subseteq V$ y $E' \subseteq E$ y escribiremos $G' \subseteq G$. De manera menos formal diremos que G contiene a G', si además de tener que $G' \subseteq G$ se tiene que $G' \neq G$ entonces diremos que G' es una subgráfica propia de G. Si $G' \subseteq G$ y tenemos que G' tiene todas las aristas de G para sus vértices en V', decimos que G' es una gráfica inducida de G, diremos que V' induce G' en G, y escribimos $G[V'] := G'$.

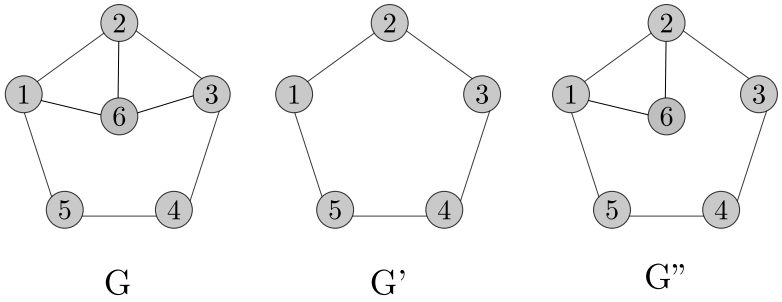

Figura 1.3: Gráficas G, G' y G', donde G es supergráfica de G' y G'', y además, G' es subgráfica inducida de G, mientras que G'' no lo es.

Si $U \subseteq V$ es cualquier conjunto de vértices entonces $G[U]$ denota la gráfica en U cuyas aristas son precisamente las aristas de G que tienen ambos vértices que la componenen U. Si U es cualquier conjunto de vértices (usualmente de G), escribimos $G-U$ para denotar la gráfica $G[V-U]$, es decir, $G-U$ se obtiene de a partir de G eliminando todos los vértices en $V \cap U$ así como las aristas que sean incidentes a estos vértices. El *complemento* $\overline{G}$ de G, es la gráfica con vértices V cuyas aristas están dadas por $[V]^2 - E$.

A continuación presentamos algunas definiciones.

Trayectoria: Una *trayectoria* es una gráfica no vacía $P = (V, E)$ tal que

$$V = \{x_0, \ldots, x_k\} \quad E = \{\{x_0, x_1\}, \ldots, \{x_{k-1}, x_k\}\}$$

donde los x_i son distintos. La cantidad de aristas es la *longitud* de P.

Camino: Definimos un camino de longitud l en G de un vértice v a un vértice w, como una secuencia finita de vértices en G

$$v = u_0, u_1, \ldots, u_l = w,$$

tal que u_{i-1} y u_i son adyacentes para $1 \leq i \leq l$.

Conexidad: Una gráfica G se dice que es *conexa*, si es no vacía y cualquier par de vértices están conectados por una trayectoria en G, en otro caso decimos que es *disconexa*.

Matriz de incidencia: La *matriz de incidencia* $B = (b_{ij})_{n\times m}$ de una gráfica G con $V = \{v_1, \ldots, v_n\}$ y $E = \{e_1, \ldots, e_m\}$ está dada por

$$b_{ij} := \begin{cases} 1 & \text{si } v_i \in e_j \\ 0 & \text{en caso contrario} \end{cases}$$

Matriz de Adyacencia: La *matriz de adyacencia* $A = (a_{ij})_{n\times n}$ de una gráfica G, donde $n = |V|$, está dada por.

$$a_{ij} := \begin{cases} 1 & \text{si } (v_i, v_j) \in E \\ 0 & \text{en caso contrario.} \end{cases}$$

Denotamos a la matriz de adyacencia de una gráfica G como A_G.

Dadas dos gráficas G y H definimos el *producto cartesiano* $G \times H$ de la siguiente manera:

$$A_{G\times H}(i_1 i_2, j_1 j_2) = \begin{cases} 1 & \text{si } A_G(i_1, j_1) = 1 \text{ e } i_2 = j_2, \text{ o bien, } A_G(i_2, j_2) = 1 \text{ e } i_1 = j_1 \\ 0 & \text{en caso contrario} \end{cases}$$

Matriz de grados: La matriz de grados $D = (d_{ij})_{n\times n}$ de una gráfica G donde $n = |V|$ está dada por:

$$m_{ij} := \begin{cases} d(v_i) & \text{si } i = j \\ 0 & \text{en caso contrario.} \end{cases}$$

Matriz Laplaciana: La *matriz Laplaciana* se define como $L = D - A$

Notemos que para estas matrices sus columnas y filas corresponden a ciertas etiquetas elegidas para los vértices de G, por lo cual nos interesaremos principalmente en las propiedades de la matriz de adyacencia que son invariantes bajo permutaciones de sus filas

y columnas. Dentro de esas propiedades se encuentran las propiedades espectrales de la matríz de de adyacencia A.

Espectro de gráficas: A continuación esbozamos algunas nociones básicas de la teoría espectral de gráficas que nos serán de utilidad a lo largo de esta tesis. Para esto consideraremos a A como una matriz sobre los números complejos y supongamos que λ es un eigenvalor de A, como A es real y simétrica se tiene que λ es real, y la multiplicidad de λ como raíz de la ecuación $\det(\lambda I - A) = 0$, es igual a la dimensión del espacio de eigenvectores que le corresponden al eigenvalor λ.

El *espectro de una gráfica* G es el conjunto de eigenvalores de la matriz de adyacencia A de G, junto con sus multiplicidades. Así, si los distintos eigenavalores de G son $\lambda_0 > \ldots > \lambda_{s-1}$ con sus multiplicidades $m(\lambda_0), \ldots, m(\lambda_{s-1})$ respectivamente, escribimos

$$\text{Spec } G = \begin{pmatrix} \lambda_0 & \ldots & \lambda_{s-1} \\ m(\lambda_0) & \ldots & m(\lambda_{s-1}) \end{pmatrix}$$

Dos gráficas con el mismo espectro se les llama coespectrales.

El siguiente lema es un resultado conocido de la teoría espectral de gráficas y será utilizado más adelante.

Lema 1.1. *[3] La cantidad de caminos de longitud l en una gráfica G del vértice v_i al v_j es la entrada en la posición (i, j) de la matríz A^l (el producto de A consigo mismo l-veces).*

Demostración. Procederemos por inducción sobre l. Primeramente, para $l = 1$ es claro pues $A^1 = A$ la entrada $(A)_{ij}$ nos dice si los vértices v_i, v_j son adyacentes o no, por lo cual, determina la cantidad de caminos de longitud 1 que existen entre estos vértices. Supongamos que el lema es cierto para $l = L$ fijo, vamos a demostrarlo para $l = L + 1$. Tenemos que la cantidad de caminos de longitud $L + 1$ del vértice v_i al v_j tiene una correspondencia biyectiva con la cantidad de caminos de longitud L del vértice v_i a los vértices v_h tales que son vecinos de v_j. Así que la cantidad de esos caminos es

$$\sum_{(v_h, v_j) \in E(G)} (A^L)_{ih} = \sum_{h=1}^{n} (A^L)_{ih} A_{hj} = (A^{L+1})_{ij}.$$

Por lo tanto la cantidad de caminos de longitud $L + 1$ del vértice v_i al vértice v_j está dada por $(A^{l+1})_{ij}$. Luego tenemos lo deseado. □

A continuación presentamos las k-potencias simétricas de una gráfica. Para esto, daremos algunas definiciones que se encuentran en el artículo "Spectra of symmetric powers of graphs and the weisfeiler-lehman refinements" de autores Alzaga, Iglesias y Pignol[1].

Una k-tupla de vértices $(i_1 \cdots i_k)$ es una función de $\{1, \cdots, k\}$ en V. Sea $\mathcal{U}_k$ el conjunto de las k-tuplas y sea $\mathcal{D}_k \subset \mathcal{U}_k$ el subconjunto de las k-tuplas tales que los vértices son distintos a pares. Tenemos que el grupo simétrico S_k actúa naturalmente en $\mathcal{D}_k$ de la siguiente manera $\sigma(i_1 \cdots i_k) = (i_{\sigma(1)} \cdots i_{\sigma(k)})$, para $\sigma \in S_k$.

La k-potencia simétrica de G, denotada por $G^{\{k\}}$, tiene como vértices los k-subconjuntos de V, donde dos k-subconjuntos son adyacentes si su diferencia simétrica es una arista.

La *k-potencia* G^k de una gráfica G está definida como el producto cartesiano iterado de G consigo mismo k-veces. Su conjunto de vértices es $\mathcal{U}_k$ y su matriz de adyacencia A_{G^k} está dada por

$$A_{G^k}(i_1 i_2 \cdots i_k, j_1 j_2 \cdots j_k) = \begin{cases} 1 & \text{si existe } u \in \{1, \cdots, k\} \text{ tal que} \\ & A_G(i_u, j_u) = 1 \text{ y } i_l = j_l \text{ para } l \neq u \\ 0 & \text{en caso contrario} \end{cases}$$

Función generadora de caminos: Dada una gráfica G se define la *función generadora de caminos* como la siguiente serie de potencias

$$\sum_{r=0}^{\infty} t^r (A_G)^r$$

Por lo anterior sabemos que el coeficiente de t^r en la entrada (i, j) determina la cantidad de caminos de longitud r que hay del vértice i al vértice j. Además tenemos que la traza de la función generadora de caminos es una invariante de gráficas que denotaremos por

$$F(G, t) = \ Tr\left(\sum_{r=0}^{\infty} t^r (A_G)^r\right)$$

Debido a que el espectro de dos matrices A y B coincide si y solo si $Tr(A^r) = Tr(B^r)$ para todo r, se sigue que dos gráficas G y H son coespectrales si y solo si $F(G, t) = F(H, t)$. Donde la traza es la suma de los elementos de la diagonal de la matriz.

Observación 1.2. *En particular, las gráficas no son distinguibles por el espectro de su k-potencia simétrica si y solo si $F(G^{\{k\}}, t) = F(H^{\{k\}}, t)$.*

Gráficas cocientes: También podemos construir $G^{\{k\}}$ a partir de G^k de la siguiente manera. Tomamos G^k y eliminamos todos los vértices que no pertenecen a $\mathcal{D}_k$, de esta manera obtenemos lo que denotaremos como la k-potencia restringida $G^{(k)}$, definida como la subgráfica de $G^{\{k\}}$ cuyos vértices son las k-tuplas en $\mathcal{D}_k$. Luego para obtener a $G^{\{k\}}$,

tomaremos cociente bajo la acción natural de S_k sobre la k-potencia restringida $G^{(k)}$. Así pues demos una definición general de lo que es una gráfica cociente. Dada una gráfica X y un grupo Γ actuando sobre X bajo automorfismos, el cociente X/Γ es una gráfica dirigida, en general con posibles aristas mútiples y lazos; y se define como sigue: los vértices de X/Γ son las órbitas de los vértices de X, y dadas dos órbitas U y W, existen tantas aristas dirigidas de U hacia W como aristas en X conectando un elemento fijo $u \in U$ con vértices de W. Nuestro interés está en el caso en que no existen aristas multiples y no hay lazos. Este caso se define a continuación. Se dice que un cociente X/Γ es *simplemente ligado* si:

1. $\{u, v\} \in E$ implica que u y v no están en la misma órbita;
2. $\{u, v\} \in E$ y $\{u, w\} \in E$ implica que v y w no están en la misma órbita.

Si X/Γ es simplemente ligada, la consideramos una gráfica simple, donde (U, W) es una arista de esta si y solo si existe una arista dirigida en X/Γ que las conecta. En el caso de *simplemente enlazada*, cualquier trayectoria en X/Γ puede ser transportada a un único camino en X. Este hecho simplifica la tarea de contar trayectorias y nos ayuda a derivar una fórmula simple para la función generadora de caminos de una gráfica cociente. Aplicaremos esto a la potencia simétrica $G^{\{k\}}$ para obtener una formula que será de utilidad después.

Proposición 1.3. *Sea X una gráfica, X/Γ un cociente simplemente enlazado, y sean U y W dos órbitas. Luego, la potencia r-ésima de la matriz de adyacencia de X/Γ está dada por*

$$A^r_{X/\Gamma}(U, W) = \frac{1}{|U|} \sum_{u \in U} \sum_{w \in W} A^r_X(u, w)$$

Demostración. La entrada $A^r_{X/\Gamma}(U, W)$ corresponde a la cantidad de caminos de longitud r de U a W en X/Γ. Fijemos un elemento $u_0 \in U$ y sea $V_0, V_1, \cdots, V_r$ una trayectoria de longitud r en X/Γ, con $V_0 = U$ y $V_r = W$. Por definición hay a lo más una arista en X, conectando un vértice de una órbita a un vértice en una órbita diferente, por lo que existe una única trayectoria $v_0, v_1, \cdots, v_r$ en G tal que $v_0 = u_0$ y $v_j \in V_j$ para $0 \leq j \leq r$. Luego tenemos que

$$A^r_{X/\Gamma}(U, W) = \sum_{w \in W} A^r_X(u_0, w)$$

El conjunto de trayectorías de longitud r de u_0 a W es llevado, biyectivamente, a un conjunto de trayectorias de cualquier $u \in U$ a W via un automorfismo en Γ, luego la suma

$$\sum_{w \in W} A^r_X(u, w)$$

no depende de u, por lo cual tenemos lo deseado. □

Notemos que, si además de que X/Γ sea simplemente enlazado también es conexa, por la fórmula anterior, tenemos que todas las orbitas tienen necesariamente el mismo tamaño.

Denotamos por $M_{X/\Gamma}$ a la matriz cuyas columnas y renglones están indexados por los vértices de X, cuyas entradas están dadas por

$$M_{X/\Gamma}(u,w) = \begin{cases} |U| & \text{si } u \text{ y } w \text{ están en la misma órbita } U \\ 0 & \text{en caso contrario} \end{cases}$$

Claramente, la proposición anterior implica el siguiente resultado.

Proposición 1.4. *Sean X/Γ un cociente simplemente enlazado y $M_{X/\Gamma}$ la matriz definida anteriormente. Luego*

$$Tr(A^r_{X/\Gamma}) = Tr(A^r_X M_{X/\Gamma})$$

Ahora trasladaremos estos resultados a los temas de nuestro interés. Sea $X = G^{(k)}$ y $\Gamma = S_k$, actuando de manera natural sobre $G^{(k)}$. El cociente $G^{(k)}/S_k$ es isomorfo a $G^{\{k\}}$, y además, es fácil ver que es un cociente simplemente enlazado. En este caso la matriz $M_{X/\Gamma}$ es la matriz M_k cuyos renglones y columnas están indexados por las k-tuplas en $\mathcal{D}_k$, donde sus entradas están dadas por:

$$M_k(i_1 \cdots i_k, j_1 \cdots j_k) = \begin{cases} k! & \text{si } \{i_1 \cdots i_k\} = \{j_1 \cdots j_k\} \\ 0 & \text{en caso contrario} \end{cases}$$

Como consecuencia de la proposición anterior, obtenemos el siguiente resultado.

Proposición 1.5. *Sean $G^{(k)}$ y $G^{\{k\}}$ la k-potencia restringida y k-potencia simétrica de la gŕafica G, respectivamente. Sea M_k la matriz definida anteriormente, tenemos que*

$$Tr(A^r_{G^{\{k\}}}) = Tr(A^r_{G^{(k)}} M_k)$$

1.1.1 Gráficas de fichas.

A continuación definimos las gráficas de fichas como las presentaron Ruy Fabila-Monroy, David Flores-Peñaloza, Clemens Huemer, Ferran Hurtado, Jorge Urrutia, David R. Wood en [7].

Sea G una gráfica y $k \geq 1$ un entero, definimos $F_k(G)$ como la gráfica cuyo conjunto de vértices está dado por los k-subconjuntos de vértices $V(G)$, es decir, $\binom{V(G)}{k}$ es su conjunto de vértices, donde dos vértices A y B de $F_k(G)$ son adyacentes si su diferencia simétrica es un par $\{a,b\}$ tal que $a \in V$, $b \in W$ y $ab \in E(G)$. Así pues, podemos pensar en los vértices de $F_k(G)$ como configuraciones de k fichas indistinguibles colocadas en distintos vértices de G, donde dos configuraciones serán adyacentes si es posible obtener una configuración a partir de la otra, deslizando una ficha, a través de una arista, de su posición inicial a

un vértice que no esté ocupado por una ficha. Así pues, llamaremos a $F_k(G)$ la gráfica de k-fichas de G. Como podemos notar, esta definición es equivalente a la definición de las k-potencias simétricas de una gráfica mencionada anteriormente, pero desde un enfoque diferente.

Propiedades básicas.
A continuación presentamos algunas propiedades básicas sobre las gráficas de fichas, las cuales fueron presentadas en el artículo "*Graphs and Combinatorics*" [7]. Primeramente, sea G una gráfica con n vértices y k un entero positivo, vamos a suponer que $n \geq k+1$ para evitar los casos triviales. Tenemos que el orden de $F_k(G)$ es

$$|V(F_k(G))| = \binom{n}{k}$$

Ahora, para cada arista XY en $F_k(G)$ asignémosle la única arista xy en G para la cual la diferencia simétrica de X y Y es $\{x, y\}$, luego la cantidad de aristas que le corresponde a $\{x, y\}$ es $\binom{n-2}{k-1}$ por lo cual tenemos que

$$|E(F_k(G)| = \binom{n-2}{k-1} |E(G)|.$$

La vecindad de cada vértice A en $F_k(G)$ es

$$\{A - \{x\} \cup \{y\} : x \in A, y \in V(G) - A, xy \in E(G)\}.$$

Por lo cual el grado de A en $F_k(G)$ es igual a la cantidad de aristas entre A y $V(G) - A$.

Para $k = 1$ la gráfica resultante es isomorfa a G, es decir, $F_1(G) \simeq G$. Notemos que dos vértices X e Y son adyacentes en $F_k(G)$ si y solo si $V(G) - X$ y $V(G) - Y$ son adyacentes en $F_{n-k}(G)$, por lo que tenemos

$$F_k(G) \simeq F_{n-k}(G)$$

Esta observación nos permite asumir que $k \leq \frac{n}{2}$.

Uno de los objetivos principales de esta tesis es estudiar el problema de isomorfismo entre gráficas. Se sabe que existen gráficas no isomorfas pero que son coespectrales donde usualmente se ocupa la matriz de adyacencia de las gráficas para determinar esto, sin embargo al utilizar el espectro de adyacencia de las 2-fichas T. Rudolph [10] demostró que pueden ser distinguidas algunas gráficas que son coespectrales pero no isomorfas. Barghi y Ponomarenko[2], y de manera independiente Alzaga, Iglesias, y Pignol[1] probaron que para cada k existen pares infinitos de gráficas no isomorfas con gráficas de k-fichas coespectrales. En "Constructing physically intuitive graph invariants"[10], Rudolph exhibió gráficas tales que el espectro de la matriz de adyacencias no está contenido en el espectro de la matriz de adyacencia de $F_k(G)$. Sin embargo, este no es el caso del espectro Laplaciano, como se

puede observar en el siguiente resultado demostrado en "*On the laplacian spectra of token graphs. Linear Algebra and its Applications*"[5].

Teorema 1.6. *Sea G una gráfica y $F_k(G)$ su gráfica de fichas, entonces el espectro Laplaciano de G está contenido en el espectro Laplaciano de $F_k(G)$.*

De manera más general, en [5], los autores demostraron lo siguiente.

Teorema 1.7. *Sea G una gráfica de n vértices, sean h, k enteros tales que $1 \leq h \leq k \leq n/2$. Si λ es un eigenvalor de $F_h(G)$, entonces λ es un eigenvalor de $F_k(G)$.*

Debido a estas propiedades especiales del espectro Laplaciano de las gráficas de k-fichas, es que nos interesaría poder extrapolar resultados que tenemos sobre el espectro de las matrices de Adyacencia al espectro Laplaciano. Esto nos da indicios de que, al considerar las gráficas de k-fichas, la información obtenida del espectro Laplaciano puede ser más fuerte que la información que se obtiene del espectro de las matriz de adyacencia.

1.1.2 Algoritmos

En esta sección nos apoyamos del libro *Introduction to Algorithms*[4], de Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein.

Informalmente, un algoritmo es un conjunto finito de reglas o pasos a seguir, que convierten información de entrada (input), en una salida (output). La finalidad de los algoritmos es tratar de resolver problemas.

Nos interesa saber dado una entrada, cual sería el tiempo que se tardaría un algoritmo en ser ejecutado, para así tener una idea de la eficiencia de este. El *tiempo de ejecución* de un algoritmo en una entrada es la cantidad de pasos que tendrá que ejecutar este, donde cada paso i tendrá su propio tiempo de ejecución que será una constante c_i, y además, determina la cantidad de veces que se ejecutará cada paso, el cual depende del tamaño de la entrada que le proporcionemos al algoritmo. Es posible que este tiempo de ejecución dependa también del tipo de entrada que le ingresamos, es decir, puede ocurrir que para ciertas entradas sea más eficiente que para otras, por lo cual es usual enfocarnos en el peor caso para así determinar el tiempo de ejecución de un algoritmo. Para darnos una idea, puede ser que el tiempo de ejecución de un algoritmo, para una entrada de tamaño n esté dado por la función $c_1n+c_2+c_3n^2+c_5+n+c_7$ donde los c_i son constantes, pero lo podemos simplificar y expresarlo como an^2+bn+c, donde a, b, c son constantes, lo que se hace para comprender mejor es que ignoraremos no solo los costos de estas nuevas constantes, sino los costos abstractos de las c_i.

Con la finalidad de estudiar el tiempo de ejecución de un algoritmo, introducimos la noción de *orden de crecimiento*. Para tener un mejor entendimiento de este concepto, veámos

primero un ejemplo. Para esto pongamos como ejemplo el tiempo de ejecución mencionado anteriormente, para este vamos a considerar solo el término líder, an^2, pues para valores muy grandes de n, los otros términos no son significativos, más aún, podemos ignorar también la constante de este término líder, pues nuevamente, para valores suficientemente grandes, esto es insignificante, por lo cual, podemos decir, que para el peor caso como el que indicamos anteriormente, el tiempo de ejecución será $\Theta(n^2)$ (theta de $n-$cuadrada). Algunas veces es posible determinar con exactitud el tiempo de ejecución de los algoritmos, pero no siempre vale la pena calcular el valor exacto. Como mencionamos anteriormente, para n suficientemente grande, las constantes multiplicativas y términos menores son dominados por el término líder.

Así pues, nuestro interés cae en la eficiencia *asintótica* de los algoritmos, es decir, nos importa saber cómo incrementa el tiempo de ejecución de un algoritmo en relación al tamaño de la *entrada* que le damos *en el límite*, conforme el tamaño de la entrada crece. Usualmente, un algoritmo que sea asintóticamente mas eficiente, será mejor para todas las entradas, salvo las pequeñas.

Usaremos la notación asintótica para describir los tiempos de ejecución de los algoritmos, de manera similar a como lo hicimos con $\Theta(n^2)$.

Notación Θ. Vamos a definir que significa la notación que usamos anteriormente. Sea $g(n)$ una función, denotamos por $\Theta(g(n))$ al conjunto de funciones

$$\begin{aligned}\Theta(g(n)) = \{f(n) : \exists c_1 > 0, c_2 > 0, n_0 > 0 \text{ tales que} \\ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ para todo } n \geq n_0\}\end{aligned}$$

Podemos decir que es una manera de acotar a la función por arriba y por abajo para un valor suficientemente grande de n. Si $f(n) \in \Theta(g(n))$ escrbiremos usualmente $f(n) = \Theta(n)$ en su lugar por abuso de notación y porque tiene su utilidad. Así que si $f(n) = \Theta(g(n))$ existe n_0 tal que para todo $n \geq n_0$, la función $f(n)$ es casi igual a $g(n)$ salvo un factor constante. En este caso diremos que $g(n)$ es una *cota asintótica justa* para $f(n)$. Usualmente diremos $f(n) = \Theta(g(n))$ en lugar de $f(n) \in \Theta(g(n))$.

Notación O. En el caso cuando sólo tenemos una *cota asintótica superior*, usamos la notación O. Dada una función $g(n)$ definimos

$$\begin{aligned}O(g(n)) = \{f(n) : \text{ existen constantes positivas } c, n_0 \text{ tales que} \\ 0 \leq f(n) \leq cg(n) \text{ para todo } n \geq n_0\}\end{aligned}$$

Claramente, tenemos que $\Theta(g(n)) \subseteq O(g(n))$. Dado que la notación O representa una cota superior, usualmente se utiliza para acotar el peor caso de ejecución de un algoritmo.

Notación Ω . Esta notación nos da una *cota inferior asintóntica*. Dada una función

$g(n)$ definimos

$$\Omega(g(n)) = \{f(n) : \text{ existen constantes positivas } c, n_0 \text{ tales que } 0 \leq cg(n) \leq f(n) \text{ para todo } n \geq n_0\}$$

El siguiente resultado nos da una relación entre las notaciones asintóticas

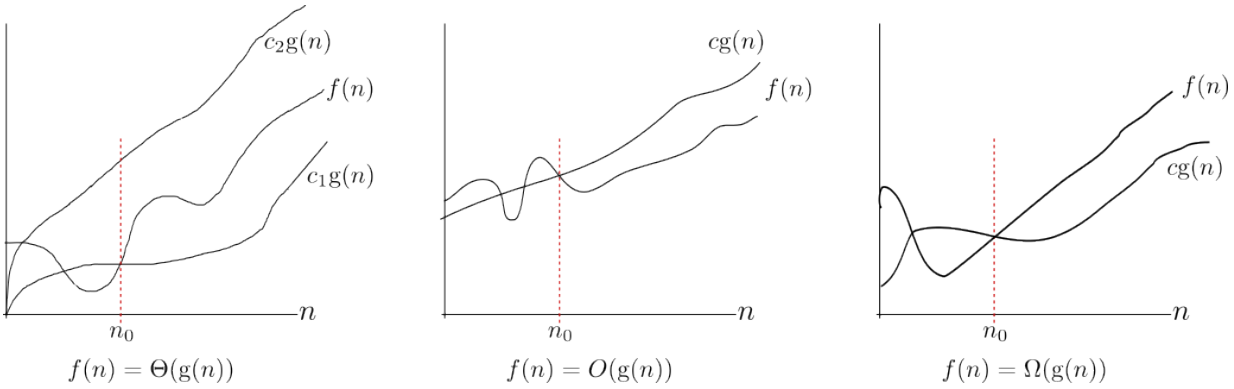

Figura 1.4: Representación de las notaciones asintóticas Θ, O y Ω .

Teorema 1.8. *Para cualquier par de funciones $f(n)$ y $g(n)$, tenemos que $f(n) = \Theta(g(n))$ si y solo si $f(n) = O(g(n))$ y $f(n) = \Omega(g(n))$.*

Notación o. En el caso de la notación O, la cota puede o no ser justa, por lo cual definimos la notación para cotas que no son asintóticamente justas. Así pues dada una función $g(n)$ definimos

$$o(g(n)) = \{f(n) : \text{ para cualquier constante } c > 0, \text{existe } n_0 > 0 \text{ tal que } 0 \leq f(n) < cg(n) \text{ para todo } n \geq n_0\}$$

Así pues, notemos que si $f(n) = o(g(n))$, entonces $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$. Como ejemplo, tenemos que $2n = o(n^2)$. Por otro lado, tenemos $2n^2 \neq o(n^2)$, pero $2n = O(n^2)$ y $2n^2 = O(n^2)$.

Notación ω. Analogamente utilizaremos la notación ω para cotas inferiores que no sean asintoticamente justas. Dada una función $g(n)$ definimos

$$\omega(g(n)) = \{f(n) : \text{ para cualquier constante positiva } c > 0, \text{existe una constante } n_0 > 0 \text{ tal que } 0 \leq cg(n) < f(n) \text{ para todo } n \geq n_0\}$$

Notemos que también podemos definirla como

$$f(n) \in \omega(g(n)) \text{ si y solo si } g(n) \in o(f(n)).$$

Así pues $f(n) = \omega(g(n))$ implica que $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, si el límite existe.

Como ejemplo tenemos que $n^2/2 = \omega(n)$, pero $n^2/2 \neq \omega(n^2)$.

Decimos que un algoritmo es eficiente si su tiempo de ejecución es polinomial, es decir, si es $O(n^c)$ para alguna constante $c > 0$.

Para las siguientes definiciones usaremos nociones intuitivas; para una definición formal se hace del uso de las máquinas de Turing.

Problemas de clase P. Diremos que un problema es de clase P si existe un algoritmo que puede resolverlo en tiempo polinomial.

Problemas de clase NP. Son problemas que tienen un certificado para verificar la solución en tiempo polinomial, es decir, aquellos que a través de un proceso se puede verificar las soluciones en tiempo polinomial.

Los problemas NP-duros son aquellos que si existiera un algoritmo para resolveros en tiempo polinomial entonces podríamos resolver todos los problemas en NP en tiempo polinomial. Los problemas NP-completos son aquellos que están en NP y son NP-duros.

Hasta el momento se desconoce si el problema de isomorfismo de gráficas está en la clase P o en la clase NP-completo. Por otro lado, es fácil ver que está en NP, pues el isomorfismo es un certificado. En caso de poder probar que cae en medio de esas clases se podría probar que P $\neq$ NP. Cabe mencionar que el problema P vs NP es uno de los Problemas del Milenio. El algoritmo presentado a continuación pareciera que nos acerca el problema de isomorfismo en una de estas categorías, se sabe de la existencia de familias infinitas de gráficas que no son distinguibles bajo este algoritmo como se muestra en [9], por lo cual continúa la intriga de a que clase pertenece el problema de isomorfismo entre gráficas.

Dadas las definiciones anteriores, nace la pregunta si el espectro de la matriz de adyacencia de la k-potencia o de la k-potencia simétrica es un invariante de gráficas, la cual se responde de manera negativa en [1] y aquí dónde extenderemos este mismo análisis a las k-potencias, pues se prueba que si el agoritmo $2k$-Weisfeiler-Lehman falla en distinguir dos gráficas, entonces sus correspondientes k-potencias y k-potencias simétricas son coespectrales, y por lo mencionado antes, no puede ser un invariante de gráficas.

Por otro lado, la importancia del problema de isomofismo en gráficas también cae en es sus aplicaciones tanto a química como al diseño de circuitos. Los circuitos se pueden representar como gráficas al representar a los componentes como vértices y a las conexiones como aristas, con esto podríamos determinar cuándo dos circuitos que se ven diferentes son, en efecto, iguales; mientras que en química, las moléculas se pueden representar como gráficas siendo los vértices los átomos y las aristas los enlaces que existen entre estos, así pues el problema de isomorfismo ayudaría a determinar cuándo dos estructuras son, en efecto, la misma, lo que ayudaría en la creación de moléculas que permitan combatir enfermedades.

Capítulo 2

Algoritmo

A continuación expondremos el algoritmo Weisfeiler-Lehman, un algoritmo que ataca el problema de isomorfismo en gráficas, es decir, el problema de distinguir cuándo dos gráficas son isomorfas. La idea básica del algoritmo Weisfeiler-Lehman es asignarles *colores* a los vértices de la gráfica de manera iterativa, de tal modo que la coloración final, a la cual llamaremos la *forma canónica* de la gráfica, nos puede ayudar a distinguir cuándo dos gráficas son no-isomorfas, que es precisamente cuando sus formas canónicas no son equivalentes. Uno de los principales atributos de este algoritmo es que la forma canónica de una gráfica puede calcularse en tiempo polinomial, y de la misma manera, comparar si dos gráficas tienen formas canónicas equivalentes, puede hacerse en tiempo polinomial, por lo que este algoritmo es polinomial. Sin embargo, este algoritmo no es completo, es decir, existen pares de gráficas no-isomorfas que tienen formas canónicas equivalentes, por lo que el algoritmo Weisfeiler-Lehman falla en distinguir dichos pares de gráficas.

El algoritmo que consideraremos en esta tesis es el algoritmo k-Weisfeiler-Lehman al cual denotaremos por k-WL y considera k-tuplas de los vértices de la gráfica. Este algoritmo es una generalización del algoritmo Weisfeiler-Lehman. La idea básica de este algoritmo es colorear las k-tuplas de los vértices de la gráfica, tomando como coloración inicial una coloración dada por una relación de equivalencia sobre las k-tuplas. Luego, mediante un proceso iterativo, refinaremos cada coloración hasta llegar a la forma canónica. En el proceso de refinar las coloraciones, podemos ir añadiendo nuevos colores. El algoritmo termina con la forma canónica de la gráfica, puesto que en esta coloración final podemos tener cada k-tupla de un color, además, el proceso iterativo termina cuando ya no es posible refinar, es decir, de un paso al siguiente, las clases cromáticas no cambian.

Con la finalidad de lograr un mejor entendimiento del algoritmo k-WL, comenzaremos presentando el algoritmo Weisfeiler-Lehman (WL) acompañado de un par de ejemplos. Después, trasladaremos la esencia del algoritmo WL al algoritmo k-WL.

2.0.1 Algoritmo Weisfeiler-Lehman

En esta subsección presentamos, de manera resumida, el algoritmo Weisfeiler-Lehman como una antesala al algoritmo de interés en esta tesis, el algoritmo k-Weisfeiler-Lehman. Además, ejemplificaremos este algoritmo en dos situaciones: (1) para un par de gráficas (no-isomorfas) que pueden ser distinguidas por el algoritmo, y mostrando con esto el hecho de que son no isomorfas, y, (2) para un par de gráficas (también no-isomorfas) que son indistinguibles para el algoritmo, por lo que el algoritmo no permite determinar si las gráficas son isomorfas o no. Dividiremos el algoritmo WL en dos partes, en la primera se hace referencia al proceso para calcular la forma canónica de una gráfica, y en la segunda, a la comparación de las formas canónicas de dos gráficas dadas para determinar si son equivalentes o no.

Algoritmo Weisfeiler-Lehman para calcular la forma canónica de una gráfica:

Entrada: Una gráfica G de orden n.
Salida: La forma canónica $c(G)$ de la gráfica, que corresponde a una coloración de los vértices de G.

Paso 1. Colorear todos los vértices de G con un mismo color, digamos $c_1(v) := 1$, para todo vértice v de G.
Paso 2. Sea v un vértice de G. De manera iterativa, en la coloración $(i+1)$-ésima le asignamos a v el color $c_{i+1}(v) := (c_i(v), \{c_i(w) : w \text{ es un vecino de } v\})$, donde $c_{i+1}(v)$ es una tupla y el segundo elemento de dicha tupla es un multiconjunto.
Paso 3. Si el número de clases cromáticas en c_{i+1} es igual al número de clases cromáticas en c_i, el algoritmo termina y llamamos a $c(G) := c_i$ la forma canónica de G.

Algoritmo Weisfeiler-Lehman para determinar si dos gráficas son no-isomorfas:

Entrada: Dos gráficas G y H de orden n.
Salida: Determinar, si es posible, si las gráficas son no-isomorfas.

Paso 1: Calcular la forma canónica $c(G)$ y $c(H)$ de G y H, respectivamente.
Paso 2: Determinar si existe una biyección f entre las clases cromáticas en $c(G)$ y las clases cromáticas en $c(H)$ que respete cardinalidades.
Paso 3: Si f no existe, imprimir "Las gráficas G y H son no-isomorfas". Si f existe, imprimir "El algoritmo no puede determinar si las gráficas G y H son isomorfas o no.

El primer paso del algoritmo WL para calcular la forma canónica de G es asignar una coloración c_1, que en este caso es el mismo color para todos los vértices de la gráfica G. Luego, de manera iterativa, definimos el color c_{i+1} en función de la coloración anterior c_i. Notemos que el color $c_{i+1}(v)$ es una tupla en donde el primer elemento es el color del vértice en la i-ésima iteración, mientras que el segundo elemento de la tupla es un multiconjunto

que contiene los colores en la i-ésima iteración de los vecinos de v en G. Así pues, dados dos vértices v y w de G, tenemos que $c_{i+1}(v) = c_{i+1}(w)$ siempre que las siguientes dos propiedades se cumplen:

- $c_i(v) = c_i(w)$, y por tanto pertenecen a la misma clase cromática, y
- el multiconjunto de colores (en c_i) del conjunto de vecinos de v es igual al multiconjunto de colores (en c_i) del conjunto de vecinos de w.

Para evitar notación engorrosa, podemos reetiquetar los colores utilizados en c_{i+1} con los primeros enteros positivos no utilizados en c_i. Notemos que este algoritmo es, en realidad, un refinamiento, puesto que cada clase cromática en la iteración i-ésima puede, o bien partirse en dos o más clases cromáticas en c_{i+1}, o permanecer sin cambios.

Como mencionamos anteriormente, el algoritmo WL ha sido utilizado para estudiar el Problema de Isomorfismo en gráficas. Notemos que si G y H son isomorfas, entonces, para cada $i \geq 1$, existe un mapeo biyectivo entre $c_i(G)$ y $c_i(H)$ (el natural inducido por el isomorfismo entre las dos gráficas), en particular existe un mapeo biyectivo entre $c(G)$ y $c(H)$ que preserva el orden de las clases cromáticas. El algoritmo WL se basa en la contraposición de este hecho. La esencia del algoritmo WL es la siguiente: Dadas dos gráficas G y H, calculamos sus formas canónicas, si éstas son no-equivalentes, es decir, si no existe un mapeo biyectivo entre sus clases cromáticas que respete el orden, entonces podemos concluir que G y H son no-isomorfas; si por el contrario, las formas canónicas de G y H son equivalentes, entonces el algoritmo WL no puede determinar si las gráficas son isomorfas o no.

En los Ejemplos 2.1 y 2.2 mostramos las dos posibles situaciones: en el Ejemplo 2.1 consideraremos dos gráficas con formas canónicas no-equivalentes, y que por tanto las gráficas son no-isomorfas, y en el Ejemplo 2.2 examinaremos dos gráficas con formas canónicas equivalentes, por lo que el algoritmo no puede determinar si las gráficas son isomorfas o no.

Ejemplo 2.1. *Consideremos las gráficas G y H que se representan en la figura 2.1. Para cada una de estas gráficas vamos a calcular sus respectivas formas canónicas mediante el algoritmo WL, y finalmente veremos si dichas formas canónicas son equivalentes o no. Asumiendo que en cada iteración vamos a reetiquetar las clases cromáticas para facilitar el proceso, para las clases cromáticas utilizaremos las etiquetas $1, 2, 3, \ldots$. Como podemos ver, las formas canónicas de G y H son no-equivalentes, puesto que no existe un biyección entre sus clases cromáticas que respete cardinalidad, y por tanto, el algoritmo WL nos garantiza que las gráficas subyacentes son no-isomorfas.*

Notemos que se hace una reasignación de colores para simplificar la notación. En cierto punto, con la gráfica G ya no se continua con el algoritmo pues la cantidad de colores no

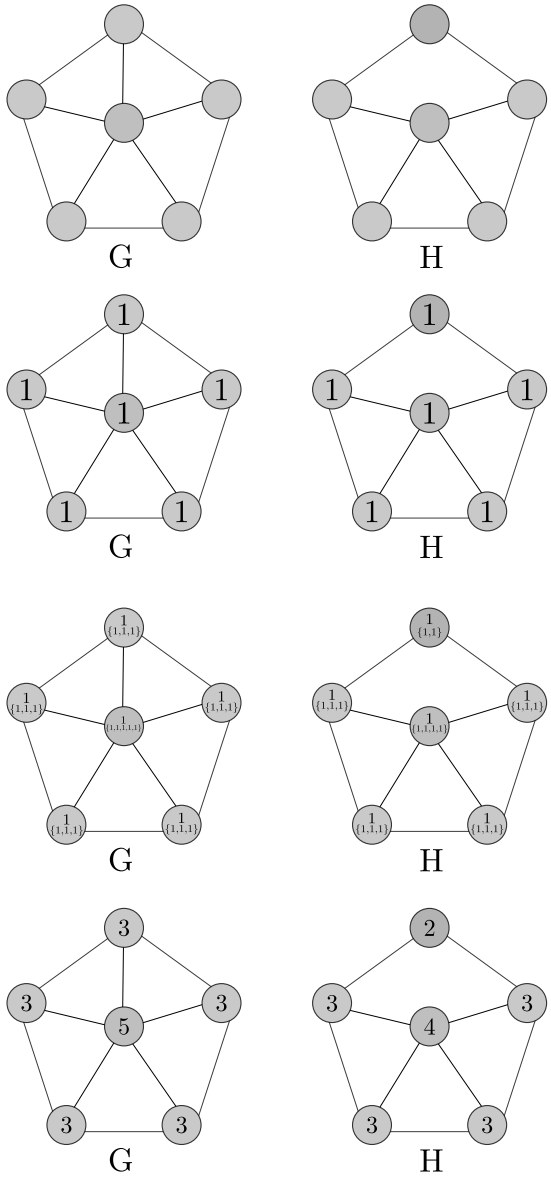

Figura 2.1: Representación del algoritmo WL para dos gráficas cuyas formas canónicas son no-equivalentes.

aumentó respecto al paso anterior, y podríamos concluir desde ahí, que la gráfica H no es isomorfa a G, pues su cantidad de colores ya era mayor a la de G, y lo único que podía ocurrir es que la cantidad de colores de H continuara siendo la misma o aumentara, por lo que serían no-isomorfas. Aún así, continuamos aplicando el algoritmo a la gráfica H para ejemplificar el algoritmo.

Así pues, tenemos un ejemplo de cómo el algoritmo WL distingue dos gráficas que nos

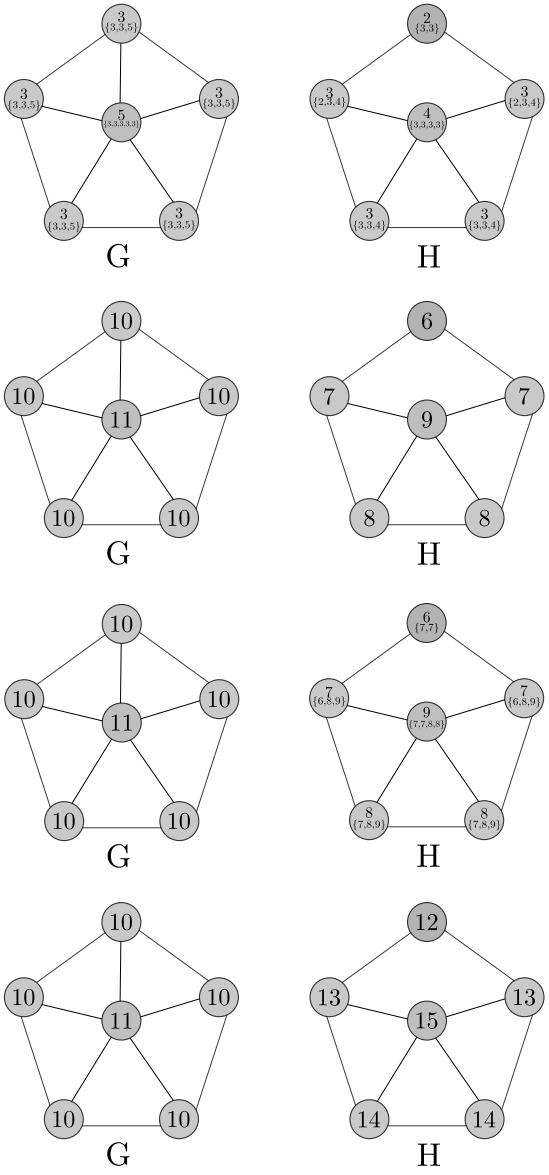

Figura 2.2: Representación del algoritmo WL para dos gráficas cuyas formas canónicas son no-equivalentes.

son isomorfas de una manera sencilla.

Ejemplo 2.2. *Consideremos las gráficas G y H de la Figura 2.3. Aplicando el algoritmo WL tenemos que las formas canónicas $c(G)$ y $c(H)$ son equivalentes, puesto que existe una biyección entre sus clases cromáticas que respeta cardinalidad. En este caso, el algoritmo WL falla en distinguir si las gráficas son isomorfas o no.*

Notemos que, tanto para G como para H, el número de clases cromáticas en c_2 no

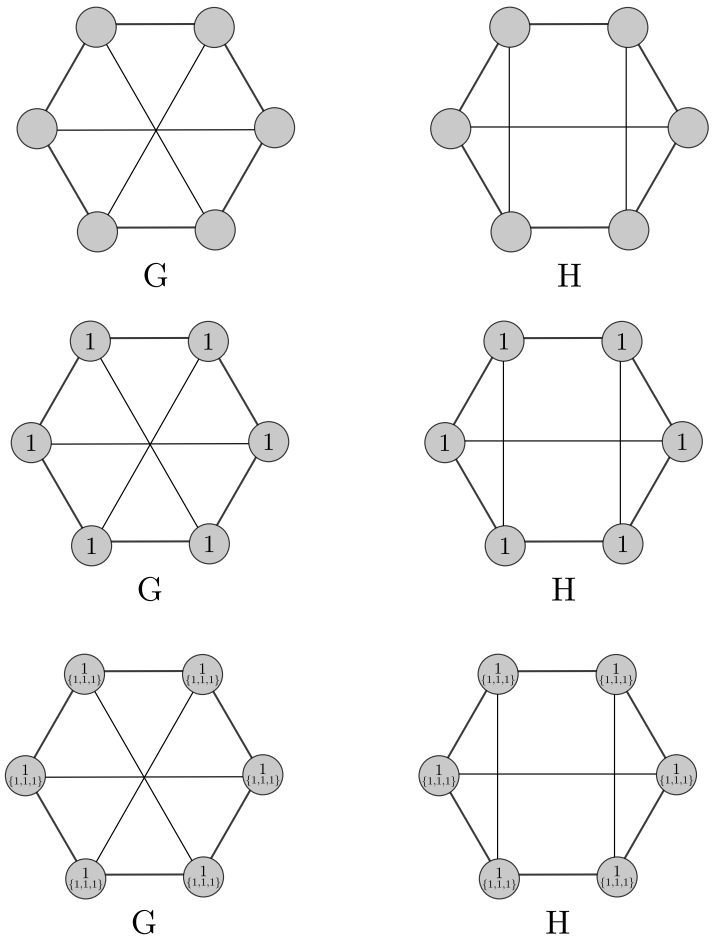

Figura 2.3: Gráficas a aplicar el algoritmo WL.

aumenta con respecto a c_1, por lo que el algoritmo termina y las formas canónicas de G y H están dadas por la coloración c_1. Este ejemplo se puede generalizar para cualquier par de gráficas G y H, ambas r-regulares para $r \geq 1$. Particularmente, esto nos da un número infinito de pares de gráficas que no son distinguibles por el algoritmo WL, mostrando así la no-completitud de este algoritmo.

Como observación, tenemos que este algoritmo es diferente que el algoritmo $k-$WL, pero nos ayuda a comprender de dónde se inspira.

2.1 Algortimo k-WL

En esta subsección, presentaremos el algoritmo k-WL, el cual, como mencionamos anteriormente, es una generalización el algoritmo WL, y además es el algoritmo estudiado en la presente tesis.

Lo mostrado a continuación viene en el artículo de Alzaga, Iglesias y Pignol[1]. Sea $G = (V, E)$ una gráfica de orden n. Denotaremos por $\mathcal{U}_G$ al conjunto de k-tuplas de los vértices de G. Vamos a definir una relación de equivalencia sobre $\mathcal{U}_G$. Diremos que dos k-tuplas $(i_1 \dots i_k)$ y $(j_1 \dots j_k)$ están relacionadas si se cumplen las siguientes condiciones:

1. $i_l = i_{l'}$ si y solo si $j_l = j_{l'}$, y
2. $(i_l, i_{l'}) \in E$ si y solo si $(j_l, j_{l'}) \in E$.

Sea el *tipo* $tp(i_1...i_k)$ de una k-tupla $(i_1...i_k)$ su clase de equivalencia. Sea S_1 el conjunto de los difrentes tipos de las k-tuplas. Recordemos la esencia del algoritmo WL para definir la forma canónica de una gráfica: En la primer coloración se colorean todos los vértices de un mismo color, y después, de manera iterativa, en c_{r+1} se colorean los vértices de acuerdo a su color en c_r y al color de sus vecinos en c_r. Vamos a trasladar esta esencia del algoritmo WL al algoritmo k-WL, a grandes rasgos, de la siguiente manera: el color en la $(r+1)$-iteración de una k-tupla $(i_1 i_2 \dots i_k)$ consistirá del color en la r-ésima iteración de esa misma k-tupla, además de un multiconjunto que contiene la información de los colores en la r-ésima iteración de ciertas k-tuplas *cercanas* a $(i_1 i_2 \dots i_k)$.

Sea $r \geq 1$ un entero. Denotaremos por $W^r_{G,k}$ a la r-ésima coloración de las k-tuplas de $\mathcal{U}_{G,k}$. La coloración inicial $W^1_{G,k}$ estará determinada por los tipos de las tuplas, es decir,

$$W^1_{G,k}(i_1 \dots i_k) := tp(i_1 \dots i_k).$$

Luego, de manera iterativa, definimos $W^{r+1}_{G,k}$ en términos de $W^r_{G,k}$ de la siguiente manera:

$$W^{r+1}_{G,k}(i_1 \dots i_k) := (W^r_{G,k}(i_1 \dots i_k), \sum_{m \in V} S^r_{G,k}(i_1 i_2 \dots i_k, m)),$$

donde

$$S^r_{G,k}(i_1 i_2 \dots i_k, m) := (W^r_{G,k}(m i_2 \dots i_k), W^r_{G,k}(i_1 m \dots i_k\}, \dots, W^r_{G,k}(i_1 i_2 \dots m)),$$

es decir, $S^r_{G,k}(i_1 i_2 \dots i_k, m)$ es la k-tupla de los colores en $W^r_{G,k}$ de las k-tuplas en $\mathcal{U}_{G,k}$ que resultan al reemplazar cada i_j por el vértice m, para $j \in \{1, 2, \dots, k\}$.

Como hicimos anteriormente, podemos reetiquetar las colores en $W^r_{G,k}$ por enteros positivos pequeños, asumiendo que guardamos esta información de reetiquetaciones.

Sea

$$S := \bigcup_{k=1}^{\infty} S_k,$$

donde los elementos de S_{r+1}, para $r \geq 1$, son multiconjuntos finitos de elementos que se encuentran en $S = \bigcup_{k=1}^{r} S_k$. Esto solo es para garantizar que tendremos colores suficientes, al igual que el algoritmo anterior, basta con renombrar los colores después de cada iteración.

Diremos que el esquema de colores se *estabiliza* en la $r-$ésima iteración si la cantidad de colores utilizados no aumenta en la $(r+1)-$ésima iteración, es decir, si $|S_{i+1}| = |S_i|$.

Para cada iteración definimos el multiconjunto

$$M^r_{G,k} = \sum_{(i_1 \ldots i_k) \in \mathcal{U}_k} W^r_{G,k}(i_1 \ldots i_k) \tag{1}$$

Notemos que $M^r_{G,k}$ captura todas los clases cromáticas de la r-ésima iteración, además del número de k-tuplas que hay en cada una de estas clases.

Para comparar el invariante $F(G^k, t)$ con refinamiento de k-WL, definimos el invariante $I_{G,k}$ que captura el resultado de la coloración por el k-WL, y al mismo tiempo es un análogo combinatorio de $F(G^k, t)$. Así pues, definimos la serie de potencias formales

$$I_{G,k}(t) = \sum_{r=0}^{\infty} t^r M^r_{G,k} \tag{2}$$

Observación 2.1. *Del algoritmo k-WL, tenemos que el color de una k-tupla en $W^{r+1}_{G,k}$ está en estrecha relación con su color en $W^r_{G,k}$, de modo que si dos tuplas tienen el mismo color en $W^{r+1}_{G,k}$, necesariamente tienen el mismo color en $W^r_{G,k}$, más aún, tienen el mismo color en $W^l_{G,k}$, para todo $l \in \{1, 2, \ldots, r\}$. Esto implica que el algoritmo k-WL define un refinamiento en el conjunto de k-tuplas $\mathcal{U}_{G,k}$.*

Observación 2.2. *Notemos que si el esquema de colores se estabiliza en la r-ésima iteración, entonces las clases cromáticas son iguales para todas las iteraciones siguientes. Además, el esquema de colores siempre se estabiliza, pues a lo más puede haber tantos colores como número de k-tuplas, n^k.*

Observación 2.3. *Dado que el algoritmo define un refinamiento de las k-tuplas, tenemos que si dos k-tuplas tienen distinto color en la r-ésima iteración, entonces no es posible que en una iteración consecuente se les asigne el mismo color.*

Proposición 2.4. *Sean G y H dos gráficas con n vértices. Entonces $I_{G,k}(t) = I_{H,k}(t)$ si y solo si existe una permutación $\sigma : \mathcal{U}_{G,k} \to \mathcal{U}_{H,k}$ tal que $W^r_{G,k}(i_1 \ldots i_k) = W^r_{H,k}(\sigma(i_1 \ldots i_k))$ para todo $r \geq 1$. En particular se tiene que $tp(i_1 \ldots i_k) = tp(\sigma(i_1 \ldots i_k))$.*

Demostración. Por la definición de $I_{G,k}(t)$ e $I_{H,k}(t)$, es claro que si $W^r_{G,k}(i_1...i_k) = W^r_{H,k}(i_1...i_k)$ para todo $r \geq 1$, entonces $I_{G,k}(t) = I_{H,k}(t)$. Ahora, supongamos que $I_{G,k}(t) = I_{H,k}(t)$. Consideremos el coeficiente de t^r cuando $r = n^k$, nuestra suposición implica una igualdad de multiconjuntos, es decir, deben ser iguales elemento a elemento, por lo cual basta fijarnos en los colores que son iguales, ver las tuplas que les corresponda, y a partir de esta, crear la biyección que nos garantiza que

$$W^{n^k}_{G,k}(i_1...i_k) = W^{n^k}_{H,k}(i_1...i_k). \tag{3}$$

Por la definición del algoritmo k-WL, se tiene que si la ecuación anterior se cumple para un determinado r_0, se cumple necesariamente para todo $1 \leq r \leq r_0$, por la observación 2.1. Por lo cual tenemos que

$$W^r_{G,k}(i_1...i_k) = W^r_{H,k}(i_1...i_k) \tag{4}$$

para todo $1 \leq r \leq n^k$. Además, como mencionamos anteriormente, el algorimto k-WL se estabilza, necesariamente, al menos, en la iteración n^k, por lo que la ecuación anterior también se cumple para todo $r \geq n^k$, así pues tenemos lo deseado. Finalmente basta considerar $r = 1$ para obtener que $tp(i_1...i_k) = tp(\sigma(i_1...i_k))$. □

Capítulo 3

Espectro de las k potencias

Como primer resultado, veremos que el espectro de las gráficas es un invariante más débil que el algortimo 2-WL. Haremos distinción de algunas observaciones que nos serán de utilidad para la prueba de un k arbitrario.

Teorema 3.1. *Sean G y H dos gráficas, y sean A_G y A_H sus respectivas matrices de adyacencia. Si $W^r_{G,2}(i,j) = W^r_{H,2}(p,q)$ para un $r \geq 1$ entonces $A^r_G(i,j) = A^r_H(p,q)$.*

Demostración. Procederemos por inducción sobre r, la cantidad de iteraciones del algoritmo. El caso $r = 1$ es trivial, pues si $W^1_{G,2}(i,j) = W^1_{H,2}(p,q)$, entonces por definición se tiene que $tp(i,j) = tp(p,q)$, y por la definición de la clase de equivalencia, se tiene que $(i,j) \in E$ si y solo si $(p,q) \in E$, es decir, $A_G(i,j) = A_H(p,q)$. Luego, supongamos que el teorema es válido para la iteración r-ésima, y probémoslo para la iteración $r+1$.

Supongamos que

$$W^{r+1}_{G,2}(i,j) = W^{r+1}_{H,2}(p,q)$$

Por definición del algoritmo WL, tenemos lo siguiente:

$$\sum_{m \in V} (W^r_{G,2}(i,j), W^r_{G,2}(i,m), W^r_{G,2}(m,j)) = \sum_{m \in V} (W^r_{G,2}(p,q), W^r_{G,2}(p,m), W^r_{G,2}(m,q)).$$

Esta es una igualdad de multiconjuntos, por lo cual existe una permutación σ del conjunto de vértices $\{1, ..., n\}$ tal que,

$$(W^r_{G,2}(i,j), W^r_{G,2}(i,m), W^r_{G,2}(m,j)) = (W^r_{G,2}(p,q), W^r_{G,2}(p,\sigma(m)), W^r_{G,2}(\sigma(m),q))$$

En particular, tenemos

$$W^r_{G,2}(i,m) = W^r_{H,2}(p,\sigma(m))$$

$$W^r_{G,2}(m,j) = W^r_{H,2}(\sigma(m),q))$$

Luego, por hipótesis de inducción, se sigue

$$A^r_G(i,m) = A^r_H(p,\sigma(m)),$$

$$A^r_G(m,j) = A^r_H(\sigma(m),q).$$

Además, recordemos que por la observación 2.1, $W^r_{G,2}(i,m) = W^r_{H,2}(p,\sigma(m))$ es equivalente a que $W^l_{G,2}(i,m) = W^l_{H,2}(p,\sigma(m))$ para todo $1 \leq l \leq r$, en especial para $l = 1$, así que también tenemos

$$A_G(i,m) = A_H(p,\sigma(m)),$$

y de manera análoga

$$A_G(m,j) = A_H(\sigma(m),q).$$

Luego

$$A_G(i,m)A^r_G(m,j) = A_H(p,m)A^r_H(m,q)$$

Así pues, sumando sobre m obtenemos que

$$\sum_m A_G(i,m)A^r_G(m,j) = \sum_m A_H(p,m)A^r_H(m,q),$$

Es decir, $A^{r+1}_G(i,j) = A^{r+1}_H(p,q)$. □

Teorema 3.2. *Sean G y H dos gráficas. Si $I_{G,2}(t) = I_{H,2}(t)$, entonces G y H son coespectrales.*

Demostración. Supongamos que $I_{G,2}(t) = I_{H,2}(t)$. Por la proposición 2.4, tenemos que existe una permutación σ de las 2-tuplas tal que para cada 2-tupla ij

$$W^r_{G,2}(ij) = W^r_{H,2}(\sigma(ij)),$$

para cada $r \geq 1$. En especial, para $r = 1$, tenemos que

$$tp(ij) = tp(\sigma(ij)).$$

Luego, por la definición del tipo inciso (i), tenemos que σ manda la diagonal de $W^r_{G,2}$ en la diagonal de $W^r_{H,2}$, luego

$$\sigma(ii) = pp$$

para algún p, es decir, tenemos la igualdad

$$W^r_{G,2}(ii) = W^r_{H,2}(\sigma(ii)),$$

luego, por el teorema 3.1, es equivalente a

$$A^r_G(i.i) = A^r_H(\sigma(ii)).$$

Así pues tenemos que

$$\sum_i A^r_G(ii) = \sum_i A^r_H(\sigma(ii))$$

es decir, $Tr(A^r_G) = Tr(A^r_H)$ para todo $r \geq 1$, lo cual es equivalente $F(G,t) = F(H,t)$, por lo que podemos concluir que G y H son coespectrales.

□

Para cada iteración r-ésima, pensaremos el $2k$-WL como una matriz de colores, las columnas y renglones están indexadas por las k tuplas, con lo cual se tiene el color $W^r_{G,k}(i_1...i_k j_1...j_k)$ en la entrada $(i_1...i_k, j_1...j_k)$.

Teorema 3.3. *Sean G^k y H^k las k-potencias de las gráficas G y H respectivamente. Sean $A^r_{G^k}$ y $A^r_{H^k}$ las r-ésimas potencias de sus matrices de adyacencia.*

Si

$$W^r_{G,2k}(i_1...i_k j_1...j_k) = W^r_{H,2k}(p_1...p_k q_1...q_k),$$

entonces

$$A^r_{G^k}(i_1...i_k, j_1...j_k) = A^r_{H^k}(p_1...p_k, q_1...q_k).$$

Demostración. La prueba es análoga al teorema 3.1. Procederemos por inducción sobre r. Para $r = 1$, sabemos que $A_{G^k}(i_1...i_k, j_1...j_k)$ solo puede tener dos posibles valores, 0 o 1, así que vamos por casos.
Supongamos que

$$A_{G^k}(i_1...i_k, j_1...j_k) = 1$$

Luego, por definición esto quiere decir que $i_l = j_l$ para todo l excepto para un único valor l_0 para el cuál se cumple que $A_G(i_{l_0}, j_{l_0}) = 1$.
Por hipótesis,

$$W^1_{G,2k}(i_1...i_k j_1...j_k) = W^1_{H,2k}(p_1...p_k q_1...q_k),$$

es decir,

$$tp(i_1...i_k j_1...j_k) = tp(p_1...p_k q_1...q_k).$$

Por definición del tipo tenemos que, $p_l = q_l$ para todo $l \neq l_0$ y $A_H(p_0, q_0) = 1$, así que $A_{H^k}(p_1...p_k, q_1...q_k) = 1$. Por lo cual tenemos la igualdad deseada.

Ahora consideremos el caso en que

$$A_{G^k}(i_1...i_k, j_1...j_k) = 0.$$

Por definición tenemos que no existe un l_0 tal que $A_G(i_{l_0}, j_{l_0}) = 1$ y $i_l = j_l$ para todo $l \neq l_0$.
Por hipótesis tenemos que

$$tp(i_1...i_k j_1...j_k) = tp(p_1...p_k q_1...q_k).$$

Por definición del tipo tenemos que no existe un l_0 tal que $A_H(p_{l_0}, q_{l_0}) = 1$ y $p_l = q_l$ para todo $l \neq l_0$. Luego necesariamente $A_{H^k}(p_1...p_k, q_1...q_k) = 0$, por lo que

$$A_{G^k}(i_1...i_k, j_1...j_k) = A_{H^k}(p_1...p_k, q_1...q_k)$$

Esto concluye la demostración para el caso $r = 1$.
Ahora supongámos que el teorema es cierto para r fijo y vamos a mostrarlo para $r + 1$, es decir, queremos mostrar que

$$W^{r+1}_{G,2k}(i_1...i_k j_1...j_k) = W^{r+1}_{H,2k}(p_1...p_k q_1...q_k)$$

Por definición del algoritmo WL tenemos que,

$$(W^r_{G,2k}(i_1...i_k j_1...j_k), \sum_{m\in V} S^r_{G,2k}(i_1...i_k j_1...j_k m)) = (W^r_{H,2k}(p_1...p_k q_1...q_k), \sum_{m\in V} S^r_{H,2k}(p_1...p_k q_1...q_k m))$$

Luego,

$$\sum_{m\in V} S^r_{G,2k}(i_1...i_k j_1...j_k m) = \sum_{m\in V} S^r_{H,2k}(p_1...p_k q_1...q_k m)$$

Luego, por definición de igualdad en multiconjuntos debe existir una permutación σ de $\{1, ..., n\}$ tal que

$$W^r_{G,2k}(i_1...i_k j_1...j_{k-1} m) = W^r_{H,2k}(p_1...p_k q_1...q_{k-1}\sigma(m)),$$

$$\vdots$$

$$W^r_{G,2k}(m i_2...i_k j_1...j_k) = W^r_{H,2k}(\sigma(m) p_2...p_k q_1...q_k).$$

Además, sabemos que si $W^r_{G,2k}(i_1...i_k j_1...j_{k-1} m) = W^r_{H,2k}(p_1...p_k q_1...q_{k-1}\sigma(m))$, entonces $W^l_{G,2k}(i_1...i_k j_1...j_{k-1} m) = W^l_{H,2k}(p_1...p_k q_1...q_{k-1}\sigma(m))$ para todo $1 \leq l \leq r$, entonces tenemos que en particular $tp(i_1...i_k j_1...j_{k1} m) = tp(p_1...p_k q_1...q_{k-1}\sigma(m))$. Así que en general tenemos que

$$tp(i_1...i_k j_1...j_{k-1} m) = tp(p_1...p_k q_1...q_{k-1}\sigma(m)),$$

$$\vdots$$

$$tp(m i_2...i_k j_1...j_k) = tp(\sigma(m) p_2...p_k q_1...q_k).$$

Luego, por el punto (2) en la definición de tp, tenemos que

$$A_G(i_t, m) = A_H(p_t, \sigma(m)) \quad \forall t \in \{1, ..., k\}$$

Además, por hipótesis de inducción tenemos que

$$A^r_{G^k}(i_1...i_k j_1...j_{k-1} m) = A^r_{H^k}(p_1...p_k q_1...q_{k-1}\sigma(m)),$$

$$\vdots$$

$$A^r_{G^k}(mi_2...i_kj_1...j_k) = A^r_{H^k}(\sigma(m)p_2...p_kq_1...q_k).$$

Falta ver que

$$A^{r+1}(i_1 \dots i_k, j_1 \dots j_k) = A^{r+1}(p_1 \dots p_k, q_1 \dots q_k).$$

Recordemos que

$$A^{r+1}(i_1 \dots i_k, j_1 \dots j_k) = \sum_{s_1 \dots s_k} A_{G^k}(i_1 \dots i_k, s_1 \dots s_k) A^r_{G^k}(s_1 \dots s_k, j_1 \dots j_k)$$

Notemos que los únicos términos que nos importan en la suma son aquellos en los que $A_{G^k}(i_1 \dots i_k, s_1 \dots s_k) \neq 0$, lo cual ocurre solamente cuando existe un índice t tal que $A_G(i_t, s_t) = 1$ y $i_l = s_l$ para todo $l \neq t$, por lo cual $A^r_{G^k}(s_1 \dots s_k, j_1 \dots j_k) = A^r_{G^k}(i_1 \dots s_t \dots i_k, j_1 \dots j_k)$. Luego tenemos

$$\begin{aligned} A^{r+1}(i_1 \dots i_k, j_1 \dots j_k) &= \sum_{s_1 \dots s_k} A_{G^k}(i_1 \dots i_k, s_1 \dots s_k) A^r_{G^k}(s_1 \dots s_k, j_1 \dots j_k) \\ &= \sum_{m_t \in V} \sum_{t=1}^{k} A_G(i_t, m_t) A^r_{G^k}(i_1 \dots m_t \dots i_k, j_1 \dots j_k) \end{aligned}$$

donde el subíndice t en m_t índica la posición en la tupla en la cual se varían los vértices en V. De manera análoga tenemos que

$$A^{r+1}_{H^k} = \sum_{m_t \in V} \sum_{t=1}^{k} A_H(p_t, \sigma(m_t)) A^r_{H^k}(p_1 \dots \sigma(m_t) \dots p_k, q_1 \dots q_k)$$

Por las igualdades anteriores, es tenemos que

$$A_G(i_t, m_t) A^r_{G^k}(i_1 \dots m_t \dots i_k, j_1 \dots j_k) = A_H(p_t, \sigma(m_t)) A^r_{H^k}(p_1 \dots \sigma(m_t) \dots p_k, q_1 \dots q_k)$$

Implicando que

$$\begin{aligned} &\sum_{m_t \in V} \sum_{t=1}^{k} A_G(i_t, m_t) A^r_{G^k}(i_1 \dots m_t \dots i_k, j_1 \dots j_k) \\ = &\sum_{m_t \in V} \sum_{t=1}^{k} A_H(p_t, \sigma(m_t)) A^r_{H^k}(p_1 \dots \sigma(m_t) \dots p_k, q_1 \dots q_k). \end{aligned}$$

Luego,

$$A^{r+1}(i_1 \dots i_k, j_1 \dots j_k) = A^{r+1}(p_1 \dots p_k, q_1 \dots q_k)$$

por lo cual tenemos lo deseado. □

Teorema 3.4. *Sean G y H dos gráficas. Si $I_{G,2k}(t) = I_{H,2k}(t)$, entonces*

$$F(G^k, t) = F(H^k, t).$$

Es decir, si el refinamiento $2k-WL$ no puede distinguir G de H entonces sus $k-$potencias son coespectrales.

Demostración. Supongamos que $I_{G,2k}(t) = I_{H,2k}(t)$. Por la Proposición 2.4, existe una permutación $\sigma : \mathcal{U}_{G,2k} \mapsto \mathcal{U}_{H,2k}$, es decir del conjunto de las $2k-$tuplas, tal que para todo $2k-$tupla $i_1 \dots i_k j_1 \dots j_k$,

$$W^r_{G,2k}(i_1 \dots i_k j_1 \dots j_k) = W^r_{H,2k}(\sigma(i_1 \dots i_k j_1 \dots j_k))$$

para todo $r \geq 1$. En particular, para $r = 1$ tenemos

$$tp(i_1 \dots i_k j_1 \dots j_k) = tp(\sigma(i_1 \dots i_k j_1 \dots j_k)$$

luego, por la definición de tipo, en particular tenemos que σ manda la diagonal de $W^r_{G,2k}$ en la diagonal de $W^r_{H,2k}$, es decir,

$$\sigma(i_1 \dots i_k i_1 \dots i_k) = p_1 \dots p_k p_1 \dots p_k$$

para alguna $k-$tupla $p_1 \dots p_k$. Por lo que tenemos

$$W^r_{G,2k}(i_1 \dots i_k i_1 \dots i_k) = W^r_{H,2k}(\sigma(i_1 \dots i_k i_1 \dots i_k))$$

Luego, por el Teorema 3.3,

$$A^r_{G^k}(i_1 \dots i_k, i_1 \dots i_k) = A^r_{H^k}(\sigma(i_1 \dots i_k, i_1 \dots i_k))$$

así que

$$\sum_{i_1 \dots i_k} A^r_{G^k}(i_1 \dots i_k i_1 \dots i_k) = \sum_{i_1 \dots i_k} A^r_{H^k}(\sigma(i_1 \dots i_k i_1 \dots i_k))$$

es decir, $TrA^r_{G^k} = TrA^r_{H^k}$ para $r \geq 1$. Luego $F(G^k, t) = F(H^k, t)$ por lo que tenemos lo deseado. □

Recordemos que nuestro es demostrar el análogo del Teorema 3.4 para las k-potencias simétricas. Antes de eso probaremos los análogos del Teorema 3.3 y del Teorema 3.4 para las $k-$potencias restringidas.

Teorema 3.5. *Sean $G^{(k)}$ y $H^{(k)}$ las $k-$potencias restringidas de dos gráficas G y H respectivamente. Sean $A^r_{G^{(k)}}$ y $A^r_{H^{(k)}}$ las $r-$potencias de sus respectivas matrices de adyacencia, y sean $i_1 \dots i_k$, $j_1 \dots j_k$, $p_1 \dots p_k$ y $q_1 \dots q_k$ $k-$tuplas en $\mathcal{D}_k$. Si*

$$W^r_{G,2k}(i_1 ... i_k j_1 ... j_k) = W^r_{H,2k}(p_1 ... p_k q_1 ... q_k),$$

entonces

$$A^r_{G^{(k)}}(i_1 ... i_k, j_1 ... j_k) = A^r_{H^{(k)}}(p_1 ... p_k, q_1 ... q_k).$$

Demostración. Esta prueba es análoga a la del Teorema 3.3, es decir, procedemos por inducción sobre r. Tenemos que para el caso $r = 1$ el argumento es el mismo que en el Teorema 3.3, pues el hecho que las tuplas se encuentren en $\mathcal{D}_k$ no afecta la demostración, sólo son un caso especial de tuplas. Así que asumamos que se cumple para r y vamos a mostrar que

$$W^{r+1}_{G,2k}(i_1 ... i_k j_1 ... j_k) = W^{r+1}_{H,2k}(p_1 ... p_k q_1 ... q_k).$$

Por definición del algoritmo $2k$-WL tenemos que,

$$(W^r_{G,2k}(i_1...i_kj_1...j_k), \sum_{m\in V} S^r_{G,2k}(i_1...i_kj_1...j_km)) = (W^r_{H,2k}(p_1...p_kq_1...q_k), \sum_{m\in V} S^r_{H,2k}(p_1...p_kq_1...q_km)).$$

Luego,

$$\sum_{m\in V} S^r_{G,2k}(i_1...i_kj_1...j_km) = \sum_{m\in V} S^r_{H,2k}(p_1...p_kq_1...q_km).$$

Por definición de igualdad en multiconjuntos, debe existir una permutación σ de $\{1, ..., n\}$ tal que

$$W^r_{G,2k}(i_1...i_kj_1...j_{k-1}m) = W^r_{H,2k}(p_1...p_kq_1...q_{k-1}\sigma(m)),$$

$$\vdots$$

$$W^r_{G,2k}(mi_2...i_kj_1...j_k) = W^r_{H,2k}(\sigma(m)p_2...p_kq_1...q_k).$$

Luego, se sigue que

$$tp(i_1...i_kj_1...j_{k-1}m) = tp(p_1...p_kq_1...q_{k-1}\sigma(m)),$$

$$\vdots$$

$$tp(mi_2...i_kj_1...j_k) = tp(\sigma(m)p_2...p_kq_1...q_k).$$

De estas ecuaciones notemos que si m está en la posición t_0 y $m = i_t$ para un $t \neq t_0$ entonces por la definición de *tipo* necesariamente se tiene que $\sigma(m) = p_t$, es decir, es fácil ver que $(i_1 \dots i_{l-1}mi_{l+1} \dots i_k)$ está em $\mathcal{D}_k$ si y solo si $(p_1 \dots p_{l-1}mp_{l+1} \dots p_k)$ está en $\mathcal{D}_k$. De la observación anterior se sigue que si se cumple que $m \neq i_t$ para todo $t \in \{1, \dots, k\}$ podemos aplicar la hipótesis de inducción para así tener

$$A_G(i_t, m) = A_H(p_t, \sigma(m)) \quad \forall t \in \{1, ..., k\}$$

Por tanto,

$$A^r_{G^{(k)}}(i_1...i_kj_1...j_{k-1}m) = A^r_{H^{(k)}}(p_1...p_kq_1...q_{k-1}\sigma(m)),$$

$$\vdots$$

$$A^r_{G^{(k)}}(mi_2...i_kj_1...j_k) = A^r_{H^{(k)}}(\sigma(m)p_2...p_kq_1...q_k).$$

Luego, como

$$\begin{aligned} A^{r+1}_{G^{(k)}}(i_1 \dots i_k, j_1 \dots j_k) &= \sum_{(s_1...s_k)\in\mathcal{D}_k} A_{G^{(k)}}(i_1 \dots i_k, s_1 \dots s_k) A^r_{G^{(k)}}(s_1 \dots s_k, j_1 \dots j_k) \\ &= \sum_{m_t\notin\{i_1,...,i_k\}} \sum_{t=1}^{k} A_G(i_t, m_t) A^r_{G^{(k)}}(i_1 \dots m_t \dots i_k, j_1 \dots j_k) \end{aligned}$$

Esto debido a los argumentos mencionados en el Teorema 3.3, donde nuevamente $m_t \in V$ y el subíndice t nos índica la posición en la tupla en la que se encuentra m_t.
Por las ecuaciones anteriores, tenemos la siguiente igualdad con la ecuación anterior:

$$= \sum_{\sigma(m_t)\notin\{p_1,\dots,p_k\}} \sum_{t=1}^{k} A_H(p_t,\sigma(m_t)) A^r_{H^{(k)}}(p_1\dots\sigma(m_t)\dots p_k, q_1\dots q_k)$$

$$= A^{r+1}_{H^{(k)}}(p_1\dots p_k, q_1\dots q_k)$$

Con lo cual tenemos lo deseado. □

Teorema 3.6. *Si el refinamiento $2k-WL$ no puede distinguir una gráfica G de una gráfica H, entonces sus respectivas $k-$potencias restringidas son coespectrales.*

Demostración. Esta demostración es análoga a la del Teorema 3.4. Sean G, H dos gráficas, supongamos que el refinamiento $2k-$WL no las puede distinguir, es decir, tenemos que $I_{G,2k}(t) = I_{H,2k}(t)$. Por la Proposición 2.4 tenemos que existe una permutación $\sigma : \mathcal{U}_{G,2k} \mapsto \mathcal{U}_{G,2k}$ tal que $W^r_{G,2k}(i_1\dots i_k) = W^r_{H,2k}(\sigma(i_1\dots i_k))$, para todo $r \geq 1$. Luego, por el caso $r = 1$, tenemos que σ preserva los *tipos* de las $2k-$tuplas, se tiene que si $i_1\dots i_k$ está en $\mathcal{D}_k$, entonces $\sigma(i_1\dots i_k i_1\dots i_k) = p_1\dots p_k p_1\dots p_k$, para alguna $k-$tupla $p_1\dots p_k$ en $\mathcal{D}_k$. Luego tenemos que

$$\sum_{(i_1\dots i_k)\in\mathcal{D}_k} W^r_{G,2k}(i_1\dots i_k i_1\dots i_k) = \sum_{(i_1\dots i_k)\in\mathcal{D}_k} W^r_{H,2k}(\sigma(i_1\dots i_k i_1\dots i_k)).$$

Por el Teorema 3.5 se sigue que

$$\sum_{(i_1\dots i_k)\in\mathcal{D}_k} A^r_{G^{(k)}}(i_1\dots i_k i_1\dots i_k) = \sum_{(i_1\dots i_k)\in\mathcal{D}_k} A^r_{H^{(k)}}(\sigma(i_1\dots i_k i_1\dots i_k))$$

Esto implica que $Tr\ A^r_{G^{(k)}} = Tr\ A^r_{H^{(k)}}$ para $r \geq 1$, luego se sigue que $F(G^{(k)}, t) = F(H^{(k)}, t)$. Es decir, sus respectivas $k-$potencias restringidas son coespectrales.

□

Ahora procederemos a probar el teorema principal el cual lo podemos plantear de la siguiente manera.

Teorema 3.7. *Sean G y H dos gráficas, si se cumple que $I_{G,2k}(t) = I_{H,2k}(t)$ entonces $F(G^{\{k\}}, t) = F(H^{\{k\}}, t)$, es decir,si el algoritmo $2k$-WL falla en distinguir las gráficas, entonces sus correspondientes gráficas de fichas son coespectrales.*

Demostración. Supongamos que $I_{G,2k}(t) = I_{H,2k}(t)$. Luego, por la proposición 2.4 existe una permutación σ tal que

$$W^r_{G,2k}(i_1\dots i_{2k}) = W^r_{H,2k}(\sigma(i_1\dots i_{2k})),$$

para todo $r \geq 1$, en particular se tiene que

$$tp(i_1 \dots i_{2k}) = tp(\sigma(i_1 \dots i_{2k})).$$

Ahora, si θ es una permutación en S_k, denotaremos por $\theta(i_1 \dots i_k)$ a la k-tupla $(i_{\theta(1)} \dots i_{\theta(k)})$, además, podemos representar a las $2k$-tuplas: parejas de k-tuplas, como $(i_1 \dots i_k, j_1 \dots j_k)$. Notemos que si una $2k$-tupla es de la forma

$$(i_1 \dots i_k, \theta(i_1 \dots i_k)),$$

con $(i_1 \dots i_k) \in \mathcal{D}_k$ y $\theta \in S_k$, entonces, debido a que σ debe preservar el tipo necesariamente manda a esta $2k$-tupla a una de la forma $(j_1 \dots j_k, \theta(j_1 \dots j_k))$, para alguna k-tupla $(j_1 \dots j_k) \in \mathcal{D}_k$, por lo cual existe una permutación ω del conjunto $\mathcal{D}_k$ tal que para cada $(i_1 \dots i_k) \in \mathcal{D}_k$

$$W^r_{G,2k}(i_1 \dots i_k, \theta(i_1 \dots i_k)) = W^r_{H,2k}(\omega(i_1 \dots i_k), \theta(\omega(i_1 \dots i_k))).$$

Luego, por el teorema 3.5 se sigue que

$$A^r_{G^{(k)}}(i_1 \dots i_k, \theta(i_1 \dots i_k)) = A^r_{H^{(k)}}(\omega(i_1 \dots i_k), \theta(\omega(i_1 \dots i_k))).$$

Como esto se cumple para $(i_1 \dots i_k) \in \mathcal{D}_k$ y para $\theta \in S_k$ se sigue que

$$\sum_{(i_1 \dots i_k) \in \mathcal{D}_k} \sum_{\theta \in S_k} A^r_{G^{(k)}}(i_1 \dots i_k, \theta(i_1 \dots i_k)) = \sum_{(i_1 \dots i_k) \in \mathcal{D}_k} \sum_{\theta \in S_k} A^r_{H^{(k)}}(\omega(i_1 \dots i_k), \theta(\omega(i_1 \dots i_k)))$$

Más aún, como ω es una biyección, el lado derecho de la ecuación anterior puede ser reescrito para obtener

$$\sum_{(i_1 \dots i_k) \in \mathcal{D}_k} \sum_{\theta \in S_k} A^r_{G^{(k)}}(i_1 \dots i_k, \theta(i_1 \dots i_k)) = \sum_{(i_1 \dots i_k) \in \mathcal{D}_k} \sum_{\theta \in S_k} A^r_{H^{(k)}}(i_1 \dots i_k, \theta(i_1 \dots i_k)).$$

Ahora, siendo M^k la matriz de la proposición 1.4, si multiplicamos ambos lados de la ecuación anterior por $k!$, lo anterior sería equivalente a tener que

$$Tr(A^r_{G^{(k)}} M_k) = Tr(A^r_{H^{(k)}} M_k)$$

Luego, por la proposición 1.4, lo anterior es equivalente a que

$$Tr(A^r_{G^{\{k\}}}) = Tr(A^r_{H^{\{k\}}}).$$

Luego, como lo anterior es cierto para todo r, se sigue que $F(G^{\{k\}}, t) = F(H^{\{k\}})$, así pues tenemos lo deseado. □

Bibliografía

[1] A. Alzaga, R. Iglesias, and R. Pignol. Spectra of symmetric powers of graphs and the weisfeiler-lehman refinements, 2008. URL https://arxiv.org/pdf/0801.2322.pdf.

[2] A. R. Barghi and I. Ponomarenko. Non-isomorphic graphs with cospectral symmetric powers. *the electronic journal of combinatorics*, pages R120–R120, 2009.

[3] N. Biggs. *Algebraic graph theory*. Number 67. Cambridge university press, 1993.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2009.

[5] C. Dalfó, F. Duque, R. Fabila-Monroy, M. Fiol, C. Huemer, A. Trujillo-Negrete, and F. Zaragoza Martínez. On the laplacian spectra of token graphs. *Linear Algebra and its Applications*, 625:322–348, 2021. ISSN 0024-3795. doi: https://doi.org/10.1016/j.laa.2021.05.005. URL https://www.sciencedirect.com/science/article/pii/S002437952100207X.

[6] R. Diestel. *Graph theory*. New York: Springer, 2000.

[7] R. Fabila-Monroy, D. Flores-Peñaloza, C. Huemer, F. Hurtado, J. Urrutia, and D. R. Wood. Token graphs. *Graphs and Combinatorics*, 28:365–380, 2012. URL https://arxiv.org/pdf/0910.4774.pdf.

[8] C. Godsil and G. F. Royle. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2001.

[9] N. I. J-Y. Cai, M. F¨urer. *An optimal lower bound on the number of variables for graph identification*, volume 12. Combinatorica, 1992.

[10] T. Rudolph. Constructing physically intuitive graph invariants, 2002.