UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
PROGRAMA DE MAESTRÍA Y DOCTORADO EN CIENCIAS MATEMÁTICAS
Y DE LA ESPECIALIZACIÓN EN ESTADÍSTICA APLICADA

Estudio de Amoebas y sus Propiedades: Detección Computacional de esta Familia de Gráficas y el Caso de los Reemplazos Raros

TESIS
QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN CIENCIAS

PRESENTA:
MARCOS EMMANUEL GONZÁLEZ LAFFITTE

DIRECTORA DE TESIS
DRA. AMANDA MONTEJANO CANTORAL
FACULTAD DE CIENCIAS, UNAM

CIUDAD DE MÉXICO, SEPTIEMBRE 2022

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Para mi mamá, mi hermano
y para toda mi familia.

Gracias a Ali por acompañarme durante
la licenciatura, la maestría y durante la
cuarentena de dos semanas que duró dos años.

Índice de Contenido

Resumen de la Tesis

Las amoebas son una familia de gráficas simples definidas originalmente por la Dra. Adriana Hansberg, el Dr. Yair Caro y la Dra. Amanda Montejano, quienes las estudiaron inicialmente en el contexto de problemas de coloraciones de tipo Ramsey-Turán [1, 2]. El estudio de estas gráficas es de interés, en particular, por su relación con el problema del isomorfismo de gráficas. Consideremos una gráfica G de orden n, una arista e de G y una arista e' del complemento de G. Si la gráfica $G - e + e'$ es isomorfa a G, decimos que el reemplazo de e por e' es un reemplazo admisible de aristas de G. Luego, si para cualesquiera dos gráficas G_1 y G_2 isomorfas a G, y encajadas en la gráfica completa K_n, es posible obtener G_2 desde G_1 por una sucesión de reemplazos admisibles de aristas, decimos que G es una amoeba local. Una definición igualmente interesante es aquella de amoeba global, que son gráficas satisfaciendo la misma propiedad que las amoebas locales, pero respecto a la gráfica completa con una cantidad de vértices mucho mayor a la de G. En esta tesis estudiaremos ambas definiciones de amoebas, y veremos que sus propiedades pueden ser modeladas por medio de Teoría de Grupos, como planteado originalmente en [1]. Más aún, este marco teórico nos permitirá desarrollar algoritmos para llevar a cabo la detección computacional de las amoebas dentro de conjuntos de gráficas disponibles en bases de datos públicas. Adicionalmente, en esta tesis estudiaremos por primera vez un tipo peculiar de reemplazo admisible de aristas con un comportamiento contraintuitivo, al que hemos llamado reemplazo raro. Como parte de los resultados se incluyen las cantidades y ejemplos de las amoebas detectadas computacionalmente, así como los algoritmos desarrollados para lograr esto. A su vez, las implementaciones de dichos algoritmos se pueden encontrar en el repositorio público: Detection of Amoeba Graphs [3], creado como parte del trabajo en esta tesis. Del mismo modo, mostraremos dos caracterizaciones para los reemplazos raros, además de definir y analizar a una familia infinita de árboles amoeba, locales y globales, que cuentan también con reemplazos raros.

Abstract of the Thesis

Amoebas are a family of simple graphs first defined by Adriana Hansberg, Yair Caro and Amanda Montejano, who initially studied them in the context of Ramsey-Turán Theory [1, 2]. The study of these graphs is of interest, in particular, due to its relation with the graph isomorphism problem. Consider a graph G of order n, an edge e of G, and an edge e' of the complement of G. If the graph $G - e + e'$ is isomorphic to G, we say that the edge-replacement of e for e' is a feasible edge-replacement of G. Then, if for any two graphs G_1 and G_2 both isomorphic to G, embedded in the complete graph K_n, it is possible to obtain G_2 from G_1 by performing a sequence of feasible edge-replacements, we say that G is a local amoeba. An equally interesting definition is that of global amoebas, graphs satisfying the same property as local amoebas, but with respect to the complete graph having a much larger number of vertices than that of G. Here we will study both definitions of amoebas, and we will see that their properties can be modeled by means of a group-theoretical approach, as originally presented in [1]. Furthermore, based on this algebraic framework we will design algorithms to computationally detect amoebas within sets of graphs available in public data bases. Additionally, we will study for the first time a peculiar type of feasible edge-replacement displaying a counterintuitive behavior, which we have called weird edge-replacement. The results achieved with this thesis include the proportions and examples of the amoebas that were computationally detected, together with the algorithms developed for this task. Moreover, the implementations of these algorithms can be found in the public repository: Detection of Amoeba Graphs [3], created as part of the work shown here. In a similar way, we present two characterizations for the weird edge-replacements, and we define and analyze an infinite family of both local and global amoeba trees, also having weird edge-replacements.

Agradecimientos

El desarrollo de esta tesis, así como el envío del manuscrito
Computational Detection of The Amoeba Graph-Family and The Case of The Weird Edge-Replacements
sometido a revisión para el congreso
15th Latin American Theoretical Informatics Symposium - LATIN 2022,
fue posible gracias a la beca que recibió Marcos Emmanuel González Laffitte,
proporcionada por el
Consejo Nacional de Ciencia y Tecnología (CONACyT) de México,
para el desarrollo de sus estudios de maestría.

Igualmente, el desarrollo de esta tesis y el envío del manuscrito
Computational Detection of The Amoeba Graph-Family and The Case of The Weird Edge-Replacements
sometido a revisión para el congreso
15th Latin American Theoretical Informatics Symposium - LATIN 2022,
fue posible gracias al apoyo proporcionado por el
Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) de la UNAM
por medio del proyecto con
clave BG100822 y no. IG100822,
con propósito de la obtención del grado de maestría.

Finalmente, el análisis computacional desarrollado como parte del trabajo mostrado en esta tesis, se llevó a cabo y completó satisfactoriamente haciendo uso de un servidor del
Laboratorio Nacional de Visualización Científica (LAVIS) de la UNAM,
disponiendo siempre del atento apoyo técnico de
Luis Aguilar, Alejandro de León y Jair García,
pertenecientes a dicha institución.

Capítulo 1

Introducción

Las amoebas son una familia de gráficas simples originalmente definidas por la Dra. Adriana Hansberg, el Dr. Yair Caro y la Dra. Amanda Montejano, quienes las estudiaron inicialmente en el contexto de problemas de coloraciones de tipo Ramsey-Turán [2]. Más aún, estas fueron estudiadas sistemáticamente por los mismos autores en [1], donde se planteó por primera vez la definición de *amoeba local*, y se construyó un marco teórico basado en grupos de permutaciones, que facilita modelar todas las propiedades con las que cuenta una amoeba.

El estudio de estas gráficas es de interés por las aplicaciones, en particular, de las *amoebas globales* en el análisis de patrones dentro de 2-coloraciones de aristas, específicas de la gráfica completa K_n [2]. Aunque en esta tesis no abordaremos este tipo de problemas, sí debemos mencionar que las propiedades que hacen relevantes a las amoebas dentro de tales aplicaciones, se derivan de la existencia de las *sucesiones de reemplazos admisibles de aristas*, relacionadas directamente con el problema del isomorfismo de gráficas, aspecto que también explica nuestro interés en las amoebas.

Ahora bien, es posible dar para todos estos conceptos definiciones basadas sólo en Teoría de Gráficas. Por ejemplo, a pesar de no ser definiciones formales, las siguientes son descripciones intuitivas que motivan el estudio de las amoebas. Específicamente, consideremos una gráfica G de orden n, una arista e de G y una arista e' del complemento de G. Si la gráfica $G - e + e'$ es isomorfa a G, el reemplazo de e por e' es llamado reemplazo admisible de aristas de G. Así, decimos que G es una amoeba local, si para cualesquiera dos gráficas G_1 y G_2 isomorfas a G, y encajadas en la gráfica completa K_n, es posible obtener G_2 desde G_1 por una sucesión de reemplazos admisibles de aristas. Similarmente, se dice que G es una amoeba global, si esta misma propiedad se mantiene respecto a la gráfica completa, pero ahora con una cantidad mucho mayor de vértices que la de G.

No obstante, una vez que hayamos formalizado todas las nociones anteriores, veremos cómo algunas de estas sucesiones de reemplazos admisibles de aristas, pueden ser representadas por composiciones de permutaciones relacionadas con los vértices de G. Esto nos ayudará a determinar de forma más sencilla si G es una amoeba o no, analizando únicamente los grupos generados por dichas permutaciones. Más aún, con base en esta nueva representación en términos de permutaciones, podremos diseñar e implementar algoritmos para detectar computacionalmente a las amoebas.

Adicionalmente, estudiaremos por primera vez un tipo peculiar de reemplazo admisible de aristas, descubierto durante el análisis de una estrategia propuesta también como parte del trabajo en esta tesis, para calcular a todos los reemplazos admisibles de G. Sin embargo, veremos que estos presentan una propiedad contraintuitiva, razón por la que los hemos llamado reemplazos raros, y por la que deberemos analizar su repercusión sobre la detección computacional de las amoebas.

Con base en esto, podemos ahora resumir puntualmente los objetivos específicos de esta tesis:

- **1)** Comprender el marco teórico basado en grupos de permutaciones que permite modelar todas las propiedades de las amoebas, para poder diseñar algoritmos cuya tarea sea determinar si una gráfica dada es amoeba o no, y su tipo de amoeba de ser el caso.
- **2)** Implementar dichos algoritmos por medio de lenguaje Python [4], y hacer públicos los programas resultantes por medio de un repositorio de Github [5].
- **3)** Llevar a cabo la detección de amoebas dentro del conjunto de todas las gráficas no isomorfas que tienen desde 1 y hasta 10 vértices, y dentro del conjunto de todos los árboles no isomorfos que tiene desde 1 hasta 22 vértices, ambos tomados de las bases de datos [6,7].
- **4)** Definir a los reemplazos raros y estudiar sus propiedades teóricas para poder analizar su repercusión en la detección de las amoebas, así como llevar a cabo la detección y análisis de las gráficas que poseen reemplazos raros, dentro del conjunto de todas las gráficas no isomorfas que tienen desde 1 y hasta 10 vértices, tomadas de [6].

Para lograr estos objetivos, hemos organizado esta tesis de la siguiente forma:

- **Capítulo 2.** Incluimos conceptos preliminares de Gráficas, Grupos y Algoritmos.
- **Capítulo 3.** Analizamos las definiciones de amoebas basadas en sucesiones de reemplazos admisibles de aristas, y mostramos cómo estas pueden ser sustituidas por definiciones en términos de grupos de permutaciones.
- **Capítulo 4.** Presentamos propiedades de las amoebas ya demostradas en la literatura [1,8], con propósito de desarrollar una discusión respecto a estas, pero teniendo ahora un contexto computacional en mente para sentar las bases de los algoritmos de detección de amoebas.
- **Capítulo 5.** Introducimos los reemplazos raros, dos caracterizaciones para ellos, y analizamos una familia infinita de árboles amoeba, locales y globales, que tienen reemplazos raros.
- **Capítulo 6.** Exhibimos los algoritmos de detección de amoebas y se demuestra su correcto funcionamiento, así como cotas inferiores para sus tiempos de ejecución.
- **Capítulo 7.** Presentamos el repositorio público de Github: **Detection of Amoeba Graphs** [3], que contiene todos los programas desarrollados como parte de esta tesis. Mostramos además todas las cantidades y ejemplos de amoebas obtenidas computacionalmente con estos programas, incluyendo un análisis de las gráficas que tienen reemplazos raros.
- **Capítulo 8.** Desarrollamos una discusión sobre los resultados obtenidos, presentando una conjetura y su consecuencia, y dando pie a posible trabajo a futuro.

Capítulo 2
Preliminares

Para poder estudiar a la familia de las amoebas es necesario contar con conocimientos de Teoría de Gráficas y de Teoría de Grupos. Más aún, para ser capaces de decidir computacionalmente si una gráfica dada es amoeba o no, también deberemos contar con algunos conceptos sobre algoritmos y sus tiempos de ejecución. El propósito de este capítulo es exponer la notación y los conceptos básicos de cada una de estas áreas, que han de usarse a lo largo de la tesis. Primero abordamos las definiciones de Teoría de Gráficas y posteriormente introduciremos las de Teoría de Grupos, prestando especial atención dentro de estas últimas a los grupos formados por permutaciones. Luego, mostraremos definiciones relacionadas con el grupo de automorfismos de una gráfica y las acciones de este sobre sus vértices y aristas, ya que estas se utilizarán en el capítulo 5 para tratar con los *reemplazos raros*. Por último estudiaremos la complejidad computacional en tiempo de los algoritmos, incluyendo una breve descripción de los algoritmos de isomorfismos -entre gráficas arbitrarias y entre árboles- usados para la detección de las amoebas.

2.1. Teoría de Gráficas

Buscando facilitar la presentación de los conceptos, aquí los agrupamos en 3 apartados. Primero mostramos algunas definiciones básicas y posteriormente incluimos definiciones sobre subgráficas e isomorfismos. Concluimos con la exposición de algunas familias de gráficas. Todo lo presentado es material usual de los cursos básicos de Teoría de Gráficas y se puede encontrar en [9–11]. Debemos resaltar que el propósito de esta sección es dar un breve repaso de conceptos básicos en Teoría de Gráficas, más no representa una recopilación propia de un curso completo en el tema.

Conceptos Básicos de Teoría de Gráficas.

Antes de comenzar debemos indicar que esta tesis trata en su totalidad con *gráficas simples*, es decir, consideraremos que entre cualesquiera dos vértices puede existir a lo más una arista y que ninguna arista puede unir a un vértice consigo mismo. Además se asumirá que toda gráfica discutida tiene un número finito de vértices y de aristas. De esto se tiene la primera definición.

Definición 2.1.1. *Una* **gráfica (simple)** G*, se define como una pareja ordenada* $G = (V(G), E(G))$*, donde* $V(G)$ *es un conjunto no vacío de elementos llamados* **vértices**, *y* $E(G)$ *es un conjunto cuyos elementos son parejas no ordenadas de vértices distintos entre sí, tomadas de* $V(G)$*, a las que se les llama* **aristas**. *Como es usual, para simplificar la notación, en ocasiones escribiremos* V *en vez de* $V(G)$*, y también* E *en lugar de* $E(G)$*, siempre que no exista ambigüedad respecto a las gráficas en contexto.*

De aquí en adelante usaremos el término *gráfica* para referirnos a toda *gráfica simple*, omitiendo el adjetivo *simple* sin volver a hacer mención de este hecho.

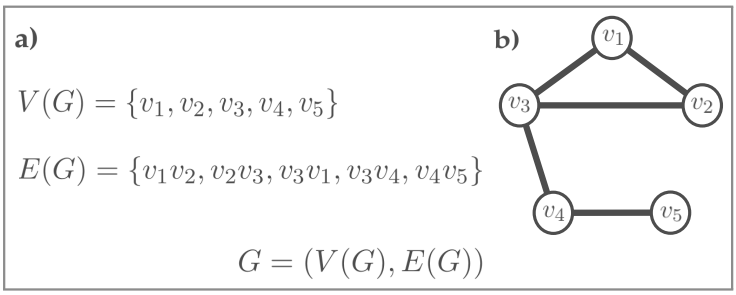

Figura 2.1: Ejemplo de una gráfica $G = (V(G), E(G))$. Se muestran **a)** los conjuntos $V(G)$ y $E(G)$ que conforman a G y **b)** una representación visual para G, como las que usaremos a lo largo de toda la tesis.

Definición 2.1.2. *Dada una gráfica $G = (V(G), E(G))$, la cardinalidad de $V(G)$ y la de $E(G)$, son llamadas* **orden** *y* **tamaño** *de G, respectivamente.*

Definición 2.1.3. *Sea G una gráfica. Dados vértices distintos u y v de G, si existe $e = \{u, v\} = \{v, u\}$ en $E(G)$, se dice que u y v son* **adyacentes**, *o bien que u es* **vecino** *de v en G. En este mismo caso se dice que e es* **incidente** *con u y con v en G, y que u y v son los* **extremos** *de e. Además, buscando simplificar la notación se escribe $e = uv = vu$.*

Definición 2.1.4. *Sea $G = (V(G), E(G))$ una gráfica. Se define como* **gráfica complemento** *de G, a la gráfica $\overline{G} = (V(\overline{G}), E(\overline{G}))$, con conjunto de vértices $V(\overline{G}) = V(G)$ y conjunto de aristas dado por $E(\overline{G}) = \{\, uv \mid u, v \in V(G) \ \text{y} \ uv \notin E(G)\}$, es decir, todos los pares de vértices que no están en $E(G)$.*

Definición 2.1.5. *La colección de todos los vecinos de un vértice v en una gráfica G es llamada* **vecindad** *de v, y la cardinalidad de dicha vecindad, o equivalentemente la cantidad de aristas incidentes con v, es llamada el* **grado** *de v en G y denotada por $d_G(v)$, o bien $d(v)$ si no existe ambigüedad. Un vértice de grado cero es llamado* **vértice aislado**. *Se denota además por $\delta(G)$ al* **grado mínimo** *de los vértices de G y por $\Delta(G)$ a su* **grado máximo**. *Adicionalmente, si G tiene vértices $v_1, v_2, ..., v_n$ ordenados por su grado de mayor a menor, la secuencia $(d(v_1), d(v_2), ..., d(v_n))$ es llamada* **secuencia de grados** *de G.*

La siguiente definición se incluye principalmente para mostrar el resultado del **Teorema 2.1.1**, coloquialmente conocido como el Lema de apretón de manos (Handshaking Lemma) [10].

Definición 2.1.6. *Sea G una gráfica con n vértices y m aristas, para $m, n \geq 1$. Se define como* **matriz de incidencia** *de G, a la matriz M_G de tamaño $n \times m$, cuyos renglones y columnas están indexados, respectivamente, por los vértices y aristas de G, y con entradas $m_{v,e}$ en $\{0, 1\}$ para $v \in V(G)$ y $e \in E(G)$, tales que $m_{v,e}$ es 1 si e es incidente con v, y 0 en otro caso.*

Teorema 2.1.1. *Para toda gráfica G se tiene* $\sum_{v \in V(G)} d_G(v) = 2\,|E(G)|$.

Demostración. Si $|E(G)| = 0$, el resultado se sigue inmediatamente. De otra forma, por doble conteo sobre las entradas de la matriz de incidencia de G, tenemos,

$$\sum_{v \in V(G)} d_G(v) = \sum_{v \in V(G)} \Big|\{e \in E(G) \mid e \text{ es incidente con } v\}\Big|$$
$$= \sum_{v \in V(G)} \left[\sum_{e \in E(G)} m_{v,e}\right] = \sum_{e \in E(G)} \left[\sum_{v \in V(G)} m_{v,e}\right] = \sum_{e \in E(G)} 2 = 2\,|E(G)|,$$

donde además se usa el hecho de que toda arista es incidente con exactamente 2 vértices. ■

Definición 2.1.7. *Sean $G = (V(G), E(G))$ y $G' = (V(G'), E(G'))$ dos gráficas. Decimos que G' es obtenida al* **agregar un vértice** *a G, si $E(G') = E(G)$ y si existe $x \in V(G')$ tal que $x \notin V(G)$ y tal que $V(G') = V(G) \cup \{x\}$. Por otro lado, dado un vértice x de G y si $|V(G)| \geq 2$, decimos que G' es obtenida al* **borrar el vértice** *x de G, si se tiene $V(G') = V(G) \setminus \{x\}$ y $E(G') = \{uv \in E(G) \,|u \neq x \text{ y } v \neq x\,\}$.*

Por último incluimos algunas definiciones que son usadas explícitamente en la motivación y estudio de las amoebas, así como en su detección computacional.

Definición 2.1.8. *Sean $G = (V(G), E(G))$ y $G' = (V(G'), E(G'))$ dos gráficas con $V(G') = V(G)$. Dada una arista uv en G, se dice que G' es obtenida al* **borrar la arista** *uv de G, si se tiene $E(G') = E(G) \setminus \{uv\}$, y en tal caso G' se denota por $G' = G - uv$. Por otro lado, de existir dos vértices $u, v \in V(G)$ que no son adyacentes en G, diremos que G' es obtenida al* **agregar la arista** *uv a G, si se cumple $E(G') = E(G) \cup \{uv\}$, y en tal caso escribimos $G' = G + uv$.*

A lo largo de la tesis escribiremos $G' = G - uv + xy$, para expresar que una gráfica G' es obtenida de otra gráfica G, al borrar en G una arista uv y luego agregar otra arista xy a la gráfica $G - uv$.

Subgráficas, Isomorfismos y Unión Disjunta.

Aquí se tratan definiciones que sirven para estudiar formalmente algunas relaciones de "contención" y de "similitud" que pueden existir entre dos gráficas dadas.

Definición 2.1.9. *Sea $G = (V(G), E(G))$ una gráfica. Dada otra gráfica $H = (V(H), E(H))$, decimos que H es una* **subgráfica** *de G y que G es una* **supergráfica** *de H, denotado por $H \subseteq G$, si se tiene $V(H) \subseteq V(G)$ y también $E(H) \subseteq E(G)$. Adicionalmente, si se cumple $V(H) \subseteq V(G)$ y $E(H) \subsetneq E(G)$, se dice que H es una* **subgráfica propia** *de G, y respectivamente G es una* **supergráfica propia** *de H. Por otro lado, si $V(H) = V(G)$ y $E(H) = E(G)$, entonces se dice que G y H son* **gráficas idénticas**, *y en cualquier otro caso decimos que estas son* **gráficas no idénticas**.

A continuación se dan dos definiciones de subgráficas que cumplen propiedades particulares respecto a los conjuntos de vértices y de aristas de la gráfica que las contiene.

Definición 2.1.10. *Sean $G = (V(G), E(G))$ y $H = (V(H), E(H))$ dos gráficas. Si se tiene $V(H) = V(G)$ y $E(H) \subseteq E(G)$, entonces decimos que H es una* **subgráfica generadora** *de G.*

Definición 2.1.11. *Sea $G = (V, E)$ una gráfica y sea X un subconjunto no vacío de V. La subgráfica $G' = (V', E')$ de G, con vértices $V' = X$ y aristas $E' = \{uv \in E \mid u, v \in X\}$, es decir, todas las aristas de G con ambos extremos en X, es llamada* **subgráfica inducida** *por X en G y se denota por $G[X]$.*

Ahora bien, es posible pensar en gráficas que tienen la misma estructura combinatoria, pero que no tienen los mismos conjuntos de vértices ni de aristas, es decir, no son gráficas idénticas. Las siguientes definiciones y propiedades permiten formalizar estas nociones.

Definición 2.1.12. *Se dice que dos gráficas $G = (V(G), E(G))$ y $H = (V(H), E(H))$, son* **isomorfas**, *si existe una biyección $\varphi : V(G) \to V(H)$ tal que $uv \in E(G)$, si y sólo si, $\varphi(u)\varphi(v) \in E(H)$. Si se cumple dicha condición, decimos que φ* **preserva adyacencia** *de G en H. En tal caso escribimos $G \simeq H$, y a φ se le llama* **isomorfismo** *de G en H. Además, dadas dos gráficas isomorfas G y G', si no es de nuestro interés resaltar aspectos particulares de G', diremos que esta es una* **copia** *de G, buscando simplificar el lenguaje.*

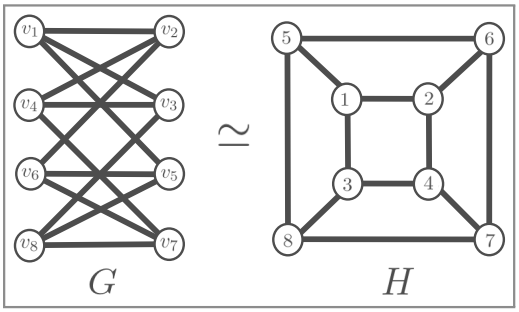

Figura 2.2: Dos gráficas G y H con conjuntos disjuntos de vértices (y aristas). Es posible verificar que la biyección $\varphi : V(G) \to V(H)$ dada por $\varphi(v_k) = k$ para todo $v_k \in V(G)$, es un isomorfismo entre ellas.

Proposición 2.1.1. *Sean $G = (V(G), E(G))$ y $H = (V(H), E(H))$ dos gráficas. Si φ es un isomorfismo de G en H, entonces la biyección inversa $\varphi^{-1} : V(H) \to V(G)$ de φ es un isomorfismo de H en G.*

Demostración. Como para todo par de vértices $u, v \in V(G)$ se tiene $uv \in E(G)$, si y sólo si, $\varphi(u)\varphi(v) \in E(H)$, y como además u y v se pueden escribir como $u = \varphi^{-1}(u')$ y $v = \varphi^{-1}(v')$ para únicos $u', v' \in V(H)$, sustituyendo en la equivalencia de antes tenemos $\varphi^{-1}(u')\varphi^{-1}(v') \in E(G)$ si y sólo si $\varphi(\varphi^{-1}(u'))\varphi(\varphi^{-1}(v')) \in E(H)$, o lo que es lo mismo, $\varphi^{-1}(u')\varphi^{-1}(v') \in E(G)$ si y sólo si $u'v' \in E(H)$, es decir, φ^{-1} es un isomorfismo de H en G. ■

Antes de ver el siguiente resultado debemos mencionar que en esta tesis, la composición de cualesquiera funciones $f : X \to Y$ y $g : Y \to Z$, se denotará por $gf(x) = g(f(x))$.

Proposición 2.1.2. *Dadas tres gráficas $G = (V(G), E(G))$, $H = (V(H), E(H))$ y $F = (V(F), E(F))$, si φ es un isomorfismo de G en H y φ' es un isomorfismo de H en F, la composición $\varphi'\varphi : V(G) \to V(F)$ es también un isomorfismo de G en F.*

Demostración. Los isomorfismos $\varphi : V(G) \to V(H)$ y $\varphi' : V(H) \to V(F)$, son biyecciones que cumplen $uv \in E(G)$ si y sólo si $\varphi(u)\varphi(v) \in E(H)$, y $xy \in E(H)$ si y sólo si $\varphi'(x)\varphi'(y) \in E(F)$, para vértices $u, v \in V(G)$ y $x, y \in V(H)$. Así, para $\varphi'\varphi : V(G) \to V(F)$, que es también una biyección, se tiene $uv \in E(G)$ si y sólo si $\varphi(u)\varphi(v) \in E(H)$, y equivalentemente $\varphi'(\varphi(u))\varphi'(\varphi(v)) \in E(F)$, por lo que $\varphi'\varphi$ es un isomorfismo de G en F. ■

Proposición 2.1.3. *La relación $\simeq$ de isomorfismo entre gráficas es una relación de equivalencia.*

Demostración. Una relación de equivalencia es aquella que es reflexiva, simétrica y transitiva. Dada una gráfica G, la reflexividad de $\simeq$ se sigue al notar que la biyección dada por $\varphi(v) = v$ para todo $v \in V(G)$, preserva adyacencia en G. Por otro lado, de tener dos gráficas isomorfas G y H, digamos con $G \simeq H$, por la **Proposición 2.1.1** se sigue que $H \simeq G$, es decir, $\simeq$ es simétrica. Finalmente, si $G \simeq H$ y $H \simeq F$ para tres gráficas G, H y F, por la **Proposición 2.1.2** se sigue que $G \simeq F$ y en consecuencia $\simeq$ es transitiva. Por lo tanto $\simeq$ es una relación de equivalencia entre gráficas. ■

Como veremos existen dos tipos de amoebas, unas llamadas *amoebas locales* y las otras *amoebas globales*. Mostraremos ahora una definición que nos permitirá hablar en particular de amoebas globales, y que incluimos aquí ya que parte de ella depende de la definición de *copia* de una gráfica.

Definición 2.1.13. *Sean $G = (V(G), E(G))$ y $H = (V(H), E(H))$ dos gráficas con conjuntos de vértices (y aristas) disjuntos. Se define como* **unión disjunta** *de G con H, a la gráfica denotada por $G \dot\cup H$, con conjunto de vértices $V(G \dot\cup H) = V(G) \dot\cup V(H)$ y conjunto de aristas $E(G \dot\cup H) = E(G) \dot\cup E(H)$, es decir, con vértices y aristas en la unión disjunta entre conjuntos de, respectivamente, los vértices y aristas de cada gráfica. Más aún, pondremos $G \dot\cup tH$ para denotar la unión disjunta de G, realizada recursivamente con $t \geq 0$ copias de H disjuntas entre sí, considerando $G \dot\cup tH = G$ para $t = 0$.*

Definición 2.1.14. *Se dice que una gráfica H está* **encajada** *en otra gráfica G, si existe una subgráfica K de G isomorfa a H, y al isomorfismo de H en K se le llama* **encaje** *de H en G. Nos referiremos a la subgráfica K de G isomorfa a H, como una copia de H encajada en G, o bien una* **copia encajada** *de H en G.*

Definición 2.1.15. *Dada una gráfica G, se llama* **automorfismo** *de G, a cualquier isomorfismo de G en sí misma, es decir, toda biyección $\varphi : V(G) \to V(G)$ tal que $uv \in E(G)$ si y sólo si $\varphi(u)\varphi(v) \in E(G)$. El conjunto de todos los automorfismos de G se denota por $Aut(G)$ y su cardinalidad por $aut(G)$.*

Como indicado en la demostración de la **Proposición 2.1.3**, respecto a cualquier gráfica G, la biyección $\varphi : V(G) \to V(G)$ dada por $\varphi(v) = v$ para todo vértice v en G, es un isomorfismo de G en si misma, es decir, un automorfismo de G. Por esta razón podemos afirmar que $Aut(G)$, para toda gráfica G, nunca es un conjunto vacío, y más adelante veremos que este tiene estructura algebraica de grupo. Por ahora concluimos este apartado con la siguiente,

Observación 2.1.1. *Sean G y H dos gráficas. Si $G \simeq H$, entonces $aut(G) = aut(H)$.* ■

Algunas Familias Relevantes de Gráficas.

Las siguientes definiciones nos facilitarán hablar sobre ejemplos de amoebas que cumplan ciertas propiedades de relevancia teórica.

Definición 2.1.16. *Sea $G = (V, E)$ una gráfica de orden n. Si cualesquiera dos vértices $u, v \in V$ distintos entre sí, son adyacentes en G, entonces decimos que G es la* **gráfica completa** *con n vértices tomados de V, y la denotamos por K_n. Por otro lado, de tener $E = \varnothing$, decimos que G es una* **gráfica vacía**.

Definición 2.1.17. *Una gráfica $G = (V, E)$, es llamada* **gráfica bipartita**, *si existen dos subconjuntos no vacíos X y Y de V, disjuntos entre sí, tales que $V = X \dot\cup Y$ y tales que para toda arista $uv \in E$ se tenga $u \in X$ y $v \in Y$, i.e., cada arista de G tiene un extremo en X y el otro en Y. Más aún, si todo vértice en X es adyacente con todo vértice en Y, se dice que G es una gráfica* **bipartita completa**. *Si en este último caso sabemos que $|X| = n$ y $|Y| = m$, denotamos a G por $K_{n,m}$. En particular, $K_{1,n}$ es llamada* **gráfica estrella**.

Definición 2.1.18. *Una* **trayectoria** *en una gráfica G de orden $n \geq k$, para algún entero $k \geq 1$, es una subgráfica P_k de G, cuyo conjunto de vértices consta de k vértices no repetidos de G, que se pueden escribir en una sucesión $\{v_1, v_2, ..., v_k\}$ tal que para todo par $v_i, v_j \in V(P_k)$, se tiene $v_iv_j \in E(P_k)$ si y sólo si $|i - j| = 1$. Los vértices v_1 y v_k son llamados, respectivamente,* **vértice inicial** *y* **vértice final** *de P_k.*

Para simplificar la notación escribiremos $P_k = \{v_1, ..., v_k\}$, y diremos que P_k es una trayectoria inducida de G si para todo par de vértices v_i y v_j en P_k, se tiene $v_iv_j \in E(G)$ si y sólo si $|i - j| = 1$. Si en este último caso se tiene además $k = n$, nos referiremos a G misma como una trayectoria.

Definición 2.1.19. *Un* **ciclo** *en una gráfica G de orden $n \geq k$, para algún entero $k \geq 3$, es una es una subgráfica C_k de G, cuyo conjunto de vértices consta de k vértices no repetidos de G, que se pueden escribir en una sucesión $\{v_1, v_2, ..., v_k\}$ cumpliendo las siguientes dos condiciones: $i)$ para todo par $v_i, v_j \in V(C_k)$ con $i, j \in \{1, ..., k-1\}$ se tiene $v_iv_j \in E(C_k)$ si y sólo si $|i - j| = 1$ y $ii)$ v_k es adyacente sólo con v_1 y v_{k-1} en C_k. Adicionalmente, si k es par (respectivamente impar), decimos que C_k es un* **ciclo par (resp. impar)**.

Más aún, si $\{v_1, ..., v_{k-1}\}$ forman una trayectoria inducida en G, y de entre los vértices de C_k sólo v_1 y v_{k-1} son adyacentes con v_k en G, entonces decimos que C_k es un ciclo inducido en G. Similarmente, si en este último caso $k = n$, entonces nos referiremos a G misma como un ciclo.

La definición de trayectoria no sólo nos permite hablar de subgráficas particulares, sino que además facilita formalizar una propiedad muy importante de las gráficas llamada *conexidad*.

Definición 2.1.20. *Sea G una gráfica. Si para todo par de vértices u y v de G, existe por lo menos una trayectoria contenida en G con u como vértice inicial y con v como vértice final, entonces decimos que G es una* **gráfica conexa**. *En caso contrario se dice que G es una* **gráfica disconexa**.

Definición 2.1.21. *Sea G una gráfica y H una subgráfica conexa inducida de G. Si no existe otra subgráfica conexa de G que sea supergráfica propia de H, entonces se dice que H es una* **componente conexa** *de G.*

Estas definiciones permiten hacer una partición de todas las gráficas en dos grandes familias, específicamente las gráficas conexas y las disconexas. Además, veremos por medio de ejemplos como es que existen amoebas de ambos tipos. Sin embargo, una subfamilia de las gráficas conexas que nos dará ejemplos relevantes para esta tesis, comprende a aquellas que no contienen ciclos.

Definición 2.1.22. *Sea T una gráfica conexa. Si adicionalmente T no contiene ningún ciclo, entonces se dice que esta es un* **árbol**. *Como es usual, en ocasiones usaremos la letra T en vez de G para hablar de árboles.*

Definición 2.1.23. *Una gráfica es llamada* **bosque**, *si todas sus componentes conexas son árboles.*

La siguiente es una propiedad ampliamente conocida de los árboles, y en realidad forma parte de una colección más amplia de definiciones equivalentes para un árbol. En este trabajo haremos mención y uso de ella, pero omitimos su demostración. Para ello referimos a [10], donde además se puede encontrar la prueba completa de dichas equivalencias.

Teorema 2.1.2. *Una gráfica conexa G de orden n, es un árbol si y sólo si tiene tamaño $n-1$.* ■

Para concluir los conceptos de Teoría de Gráficas presentamos la siguiente proposición. Esta nos será de utilidad para hablar sobre las secuencias de grados de árboles.

Proposición 2.1.4. *Sea $S = (d_1, d_2, ..., d_n)$ una secuencia de $n \geq 2$ enteros positivos. Entonces, S es la secuencia de grados de un árbol de orden n, si y sólo si, se cumple $\sum_{i=1}^{n} d_i = 2n - 2$.*

Demostración.

Si S es la secuencia de grados de un árbol T, por los **Teoremas 2.1.1** y **2.1.2** inmediatamente tenemos

$$\sum_{i=1}^{n} d_i = 2|E(T)| = 2(n-1) = 2n - 2.$$

Por otro lado supongamos que $\sum_{i=1}^{n} d_i = 2n-2$. Buscaremos probar por inducción sobre $n \geq 2$ que existe un árbol de orden n con secuencia de grados $(d_1, d_2, ..., d_n)$. Para $n = 2$ tenemos $d_1 + d_2 = 2$, pero como $d_i \geq 1$, entonces $d_1 = d_2 = 1$ y vemos que K_2 satisface el caso base.

Supóngase que la proposición es cierta para $n = k$ con $k \geq 2$, y considérese ahora una secuencia $S_0 = (d_1, d_2, ..., d_{k+1})$ con $k + 1$ enteros positivos que satisfaga lo requerido, es decir, tal que $\sum_{i=1}^{k+1} d_i = 2(k + 1) - 2 = 2k$. Adicionalmente, podemos suponer que $d_1 \geq d_2 \geq \cdots \geq d_{k+1}$.

Nótese que si $d_i = 1$ para todo $i \in [k+1]$, entonces se tendría $\sum_{i=1}^{k+1} d_i = k+1 < k+k = 2k$, por lo que debe existir por lo menos un $d_j > 1$, digamos $d_1 \geq 2$. Sin embargo, si se tiene $d_i \geq 2$ para toda i, entonces $\sum_{i=1}^{k+1} d_i \geq 2(k+1) > 2k$. Pero si sólo $d_{k+1} = 1$, entonces $\sum_{i=1}^{k+1} d_i \geq [\sum_{i=1}^{k} 2]+1 = 2k+1 > 2k$, por lo que debe haber por lo menos dos d_i igual a 1, en particular $d_{k+1} = d_k = 1$.

Ahora bien, ya que $d_1 \geq 2$ y $d_k = 1$, la secuencia $S' = (d_1 - 1, d_2, ..., d_k)$ tiene k enteros positivos. Además, ya que $d_{k+1} = 1$, podemos ver que la suma de los términos en S' es,

$$d_1-1+d_2+\cdots+d_k = [d_1-1+d_2+\cdots+d_k]+(d_{k+1}-d_{k+1}) = d_1+d_2+\cdots+d_k+(d_{k+1}-1)-1 = 2k-2.$$

De esta forma, por la hipótesis de inducción hay un árbol T' de orden k con S', o un reordenamiento de esta, como secuencia de grados.

Supongamos que T' tiene conjunto de vértices $V(T') = \{v_1, ..., v_k\}$ con $d_{T'}(v_1) = d_1 - 1$ y con $d_{T'}(v_i) = d_i$ para $i \in [2, k]$. Así, respecto a un nuevo árbol T obtenido al agregar a T' un vértice v_{k+1} y la arista v_1v_{k+1}, se cumple $d_T(v_1) = d_1$, $d_T(v_{k+1}) = 1$ y $d_T(v_i) = d_i$ con $i = 2, ..., k$. Pero entonces $S_0 = (d_1, d_2, ..., d_{k+1})$ es precisamente la secuencia de grados de T, y por lo tanto T es el árbol buscado, con lo que se concluye la inducción. ■

2.2. Teoría de Grupos

El contenido de esta sección se desarrolla en 4 apartados. Revisaremos material usual de los cursos de Álgebra Moderna y Álgebra Superior, que también se puede encontrar en [12], [13] y [14]. Primero damos algunas definiciones y propiedades básicas de grupos, para después poder hablar de grupos generados. Posteriormente tratamos con grupos de permutaciones y concluimos con la definición de la acción de un grupo sobre un conjunto y las órbitas asociadas a esta. Nuevamente debemos remarcar que el propósito de esta sección es dar un breve repaso de conceptos básicos en Teoría de Grupos, más no representa una recopilación propia de un curso completo en el área.

Conceptos Básicos de Teoría de Grupos.

Es usual encontrar en la literatura el uso de la letra G para denotar a un grupo. Sin embargo, en esta tesis reservaremos la G para hablar de gráficas, y procurando la notación escribiremos $\mathcal{G}$ para referirnos al conjunto base de un grupo arbitrario, que definimos a continuación.

Definición 2.2.1. *Un* **grupo** *se define como una pareja ordenada $(\mathcal{G}, \bullet)$, donde $\mathcal{G}$ es un conjunto no vacío y $\bullet$ es una función llamada* **operación binaria**, *que cumplen todas las siguientes propiedades,*

i) Cerradura: *$\bullet$ es una función $\bullet : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$. Adicionalmente, denotamos por $x \bullet y$ a la imagen $\bullet((x, y)) \in \mathcal{G}$, del par $(x, y) \in \mathcal{G} \times \mathcal{G}$ bajo $\bullet$, con lo que se tiene $x \bullet y \in \mathcal{G}$ para todo par $x, y \in \mathcal{G}$.*

ii) Asociatividad: *Para cualesquiera $x, y, z \in \mathcal{G}$, se tiene $(x \bullet y) \bullet z = x \bullet (y \bullet z)$.*

iii) Elemento Neutro: *Existe un elemento $e \in \mathcal{G}$, llamado* **identidad** *o* **neutro** *de $\mathcal{G}$, tal que para todo $x \in \mathcal{G}$ se tiene $e \bullet x = x \bullet e = x$.*

iv) Elemento Inverso: *Para todo $x \in \mathcal{G}$ existe un elemento de $\mathcal{G}$ llamado* **inverso** *de x, que denotaremos por $x^{-1} \in \mathcal{G}$, tal que $x \bullet x^{-1} = x^{-1} \bullet x = e$.*

Adicionalmente, si para todo par de elementos $x, y \in \mathcal{G}$ se tiene $x \bullet y = y \bullet x$, conocida como propiedad de **v) Conmutatividad**, *entonces decimos que $(\mathcal{G}, \bullet)$ es un* **grupo abeliano** *o* **grupo conmutativo**.

Para simplificar la notación en ocasiones sólo pondremos $\mathcal{G}$ para hablar de un grupo $(\mathcal{G}, \bullet)$, siempre y cuando no exista posibilidad de confusión respecto a la operación binaria usada.

Definición 2.2.2. *Dado un grupo $(\mathcal{G}, \bullet)$, la cardinalidad del conjunto $\mathcal{G}$ es llamada* **orden** *de dicho grupo.*

A continuación enunciamos una propiedad ampliamente conocida de las operaciones binarias pero omitimos su demostración.

Proposición 2.2.1. *Dado un grupo $(\mathcal{G}, \bullet)$, la expresión $x_1 \bullet x_2 \bullet \cdots \bullet x_n$, es decir, la operación iterada para n elementos de $\mathcal{G}$ manteniendo en general el orden relativo de cada término, mapea al mismo elemento $x_1 \bullet x_2 \bullet \cdots \bullet x_n = g \in \mathcal{G}$ independientemente del orden de los paréntesis que se le asignen.* ■

Definición 2.2.3. *Sea $(\mathcal{G}, \bullet)$ un grupo. Se dice que un subconjunto no vacío $\mathcal{H}$ de $\mathcal{G}$, es* **subgrupo** *de $\mathcal{G}$, denotado $\mathcal{H} \leq \mathcal{G}$, si $\mathcal{H}$ mismo cumple ser un grupo junto con la restricción de $\bullet$ a $\mathcal{H} \times \mathcal{H}$ como operación binaria. En tal caso escribimos $(\mathcal{H}, \bullet)$, o bien $\mathcal{H}$, para hablar de dicho subgrupo, y si adicionalmente $\mathcal{H} \subsetneq \mathcal{G}$, decimos que $\mathcal{H}$ es* **subgrupo propio** *de $\mathcal{G}$.*

Las siguientes son propiedades que se siguen directamente de las definiciones anteriores.

Proposición 2.2.2. *Sea $(\mathcal{G}, \bullet)$ un grupo con neutro e.*

i) El elemento e es el único elemento neutro de $\mathcal{G}$.

ii) Todo $x \in \mathcal{G}$ tiene un inverso único $x^{-1} \in \mathcal{G}$ y además $x = (x^{-1})^{-1}$. ■

Es importante notar que las propiedades anteriores se cumplen para cualquier grupo $\mathcal{G}$, incluyendo a todos los subgrupos de $\mathcal{G}$ de los que se hable, como es el caso del subgrupo indicado en la siguiente proposición. Debemos añadir que se incluye aquí la demostración de esta última, ya que será usada más adelante para deducir otros resultados.

Proposición 2.2.3. *Sea $(\mathcal{G}, \bullet)$ un grupo con neutro e. Un subconjunto no vacío $\mathcal{H}$ de $\mathcal{G}$ es un subgrupo de $\mathcal{G}$ si y sólo si se cumplen las siguientes tres condiciones: se tiene $x \bullet y \in \mathcal{H}$ para cualesquiera $x, y \in \mathcal{H}$, $e \in \mathcal{H}$, y además $x^{-1} \in \mathcal{H}$ para todo x en $\mathcal{H}$.*

Demostración. Para demostrar ambas direcciones haremos uso implícitamente de la asociatividad de $\bullet$ en $\mathcal{H}$, que en ambos casos se sigue de la asociatividad de $\bullet$ en $\mathcal{G}$.

Si $\mathcal{H} \leq \mathcal{G}$, la propiedad de cerradura en $\mathcal{H}$ se verifica por definición. Ahora bien, en este mismo caso $\mathcal{H}$ debe tener un neutro, llámese $h \in \mathcal{H}$. Luego, para todo $x \in \mathcal{H}$ tenemos $x \bullet h = x$. Pero entonces respecto al grupo $\mathcal{G}$ se cumple $x \bullet h = x = x \bullet e$, y como $x^{-1} \in \mathcal{G}$, podemos escribir $x^{-1} \bullet (x \bullet h) = x^{-1} \bullet (x \bullet e)$, de donde $h = e$. Igualmente, cada $x \in \mathcal{H}$ debe tener un inverso también en $\mathcal{H}$, digamos $z \in \mathcal{H}$. Este es un elemento que operado con x nos regresa el neutro de $\mathcal{H}$, que sabemos que es e y en consecuencia $x \bullet z = z \bullet x = e$. Luego, respecto a $\mathcal{G}$ tenemos $x \bullet z = e = x \bullet x^{-1}$ y operando ambos lados con $x^{-1} \in \mathcal{G}$, vemos que necesariamente $z = x^{-1}$.

Ahora supondremos que para el subconjunto no vacío $\mathcal{H}$ se cumplen las tres propiedades y buscaremos mostrar que este es un grupo con la restricción de $\bullet$ como operación binaria. Pero en $\mathcal{H}$ ya se cumplen los cuatro axiomas en la definición de grupo. En efecto, $\mathcal{H}$ cumple cerradura por nuestra suposición. Además $\bullet$ es, en particular asociativo en $\mathcal{H}$, ya que es asociativo para cualesquiera elementos de $\mathcal{G}$. Tiene un elemento neutro que es e, y todo $x \in \mathcal{H}$ tiene un inverso x^{-1} también en $\mathcal{H}$. Con esto se verifica que $\mathcal{H}$ es un subgrupo de $\mathcal{G}$. ■

Observación 2.2.1. *Sea $(\mathcal{G}, \bullet)$ un grupo con neutro e. El subconjunto $\{e\} \subseteq \mathcal{G}$, es decir, el que sólo contiene al elemento neutro de $\mathcal{G}$, es un subgrupo de $\mathcal{G}$ y además es el subgrupo más pequeño de $\mathcal{G}$.* ■

Definición 2.2.4. *Sea $\mathcal{G}$ un grupo con neutro e. El subgrupo $\{e\}$ de $\mathcal{G}$ es llamado* **subgrupo trivial** *de $\mathcal{G}$.*

Para concluir este apartado veremos que de forma semejante a las gráficas, existen definiciones que nos permiten estudiar relaciones de "similitud" entre grupos.

Definición 2.2.5. *Sean $(\mathcal{G}, \bullet)$ y $(\mathcal{H}, *)$ dos grupos. Se dice que una función $f : \mathcal{G} \to \mathcal{H}$ es un* **morfismo** *u* **homomorfismo** *del grupo $\mathcal{G}$ en el grupo $\mathcal{H}$, si se tiene $f(x \bullet y) = f(x) * f(y)$ para cualesquiera $x, y \in \mathcal{G}$. Si además f es biyectivo, decimos que este un* **isomorfismo** *entre estos dos grupos y escribimos $\mathcal{G} \cong \mathcal{H}$.*

Es importante remarcar que el isomorfismo de grupos, que denotamos con $\cong$, es distinto del isomorfismo de gráficas, para el que ponemos $\simeq$, ya que estos son objetos matemáticos distintos.

Grupo Generado.

Considérese un grupo $(\mathcal{G}, \bullet)$ con neutro e. Podemos ver que no todo subconjunto X de $\mathcal{G}$ conforma un subgrupo de $\mathcal{G}$, ya que incluso si $X \neq \varnothing$, basta con que $e \notin X$ para que este falle en ser subgrupo. Sin embargo, como veremos en este apartado, siempre podremos determinar el subgrupo más pequeño de $\mathcal{G}$ que contiene a X, incluso cuando $X = \varnothing$.

Proposición 2.2.4. *Sea $(\mathcal{G}, \bullet)$ un grupo y sea $\{\mathcal{H}_i\}_{i \in I}$ una familia no vacía de subgrupos de $\mathcal{G}$, indexados en algún conjunto I. Entonces el conjunto $\bigcap_{i \in I} \mathcal{H}_i$ es también un subgrupo de $\mathcal{G}$.*

Demostración. Verificaremos para dicha intersección las tres propiedades en la **Proposición 2.2.3**. Denotemos por e al neutro de $\mathcal{G}$, que sabemos es único (ver **Proposición 2.2.2**). Sabemos además que $e \in \mathcal{H}_i$ para todo $i \in I$, al ser cada $\mathcal{H}_i$ subgrupo de $\mathcal{G}$, por lo que por definición $e \in \bigcap_{i \in I} \mathcal{H}_i$. Por otro lado, considérese cualquier $x \in \bigcap_{i \in I} \mathcal{H}_i$, i.e., $x \in \mathcal{H}_i$ para toda $i \in I$. Debemos tener $x^{-1} \in \mathcal{H}_i$ para toda i, nuevamente al ser estos subgrupos de $\mathcal{G}$, por lo que $x^{-1} \in \bigcap_{i \in I} \mathcal{H}_i$. De forma similar, para cualesquiera $x, y \in \bigcap_{i \in I} \mathcal{H}_i$, como $x, y \in \mathcal{H}_i$, entonces $x \bullet y \in \mathcal{H}_i$, lo que ocurre para toda $i \in I$ y entonces $x \bullet y \in \bigcap_{i \in I} \mathcal{H}_i$. Con todas estas propiedades podemos concluir que $\bigcap_{i \in I} \mathcal{H}_i \leq \mathcal{G}$. ■

Definición 2.2.6. *Sea $(\mathcal{G}, \bullet)$ un grupo y sea X un subconjunto arbitrario de $\mathcal{G}$. Se define como* **subgrupo de $\mathcal{G}$ generado** *por X, al subgrupo $\langle X \rangle$ de $\mathcal{G}$, dado por $\langle X \rangle = \bigcap \{\mathcal{H} \leq \mathcal{G} \mid X \subseteq \mathcal{H}\}$. Si en particular se tiene $\langle X \rangle = \mathcal{G}$, decimos que X es un* **(subconjunto) generador** *de $\mathcal{G}$.*

Buscando simplificar la notación, si X es un subconjunto finito de $\mathcal{G}$, digamos con $X = \{x_1, x_2, ..., x_n\}$ para $n \geq 1$, entonces escribiremos $\langle X \rangle = \langle x_1, x_2, ..., x_n \rangle$.

Definición 2.2.7. *Sea $(\mathcal{G}, \bullet)$ un grupo de orden n. Si existe un elemento x de $\mathcal{G}$ tal que se tiene $\mathcal{G} = \langle x \rangle$, entonces diremos que $\mathcal{G}$ es un* **grupo cíclico** *y lo denotaremos por $\mathcal{C}_n$.*

Nótese que el grupo cíclico de orden n se escribe como $\mathcal{C}_n$, para diferenciarlo de la gráfica cíclica, o ciclo, C_n con n vértices (ver **Definición 2.1.19**), ya que estos son objetos distintos.

Proposición 2.2.5. *Sea $(\mathcal{G}, \bullet)$ un grupo y X un subconjunto arbitrario de $\mathcal{G}$. El subgrupo $\langle X\rangle$ de $\mathcal{G}$ generado por X, es el menor subgrupo de $\mathcal{G}$ que contiene a X, es decir, $\langle X\rangle \subseteq \mathcal{H}$ para todo subgrupo $\mathcal{H}$ de $\mathcal{G}$ que contiene a X.*

Demostración. Denótese por $\{\mathcal{H}_i\}_{i\in I}$ a la familia de subgrupos de $\mathcal{G}$ que contienen a X, indexados en algún conjunto I, y nótese que $\langle X\rangle \in \{\mathcal{H}_i\}_{i\in I}$ ya que $X \subseteq \bigcap_{i\in I}\mathcal{H}_i = \langle X\rangle$. Escójase algún subconjunto $\mathcal{M} \in \{\mathcal{H}_i\}_{i\in I}$ que sea a la vez de menor cardinalidad en dicha familia. Para todo $x \in \langle X\rangle$, se tiene $x \in \mathcal{H}_i$, por lo que $\langle X\rangle \subseteq \mathcal{H}_i$ para toda $i \in I$ y en particular $\langle X\rangle \subseteq \mathcal{M}$. Pero de esta forma, suponer $\langle X\rangle \subsetneq \mathcal{M}$, contradice la elección de $\mathcal{M}$. Por lo tanto se tiene $\langle X\rangle = \mathcal{M}$. ■

Es importante notar que de la anterior proposición y de la **Observación 2.2.1** se tiene que dado un grupo $(\mathcal{G}, \bullet)$ con neutro e y un subconjunto X de $\mathcal{G}$, si se tiene $X = \varnothing$, entonces $\langle X\rangle = \{e\}$. Por otro lado, recuérdese que dado un grupo $\mathcal{G}$, por la **Proposición 2.2.1**, la expresión $x_1 \bullet x_2 \bullet \cdots \bullet x_n$ para cualesquiera $x_1, x_2, ..., x_n \in \mathcal{G}$, corresponde siempre a un mismo elemento en $\mathcal{G}$.

Definición 2.2.8. *Sea $(\mathcal{G}, \bullet)$ un grupo con neutro e y sea X un subconjunto arbitrario de $\mathcal{G}$. Diremos que un elemento $w \in \mathcal{G}$ es una* **palabra** *sobre X, si existe una sucesión $\{x_1, x_2, ..., x_n\}$ de $n \geq 1$ elementos de X, tal que w se puede escribir como $w = x_1^{r_1} \bullet x_2^{r_2} \bullet \cdots \bullet x_n^{r_n}$ con $r_i \in \{-1, +1\}$ para toda $i = 1, ..., n$, donde $x_i^{+1} = x_i$. Se denotará por $W(X)$ al conjunto de todas las palabras sobre X, con $W(\varnothing) = \{e\}$.*

Proposición 2.2.6. *Sea $(\mathcal{G}, \bullet)$ un grupo con neutro e y sea X un subconjunto arbitrario de $\mathcal{G}$. El conjunto $W(X)$ es un subgrupo de $\mathcal{G}$.*

Demostración. Nuevamente verificamos las tres propiedades en la **Proposición 2.2.3**. Si $X = \varnothing$, entonces $W(X) = \{e\} \leq \mathcal{G}$. Supóngase $X \neq \varnothing$. Dado $x \in X$, tenemos $e = x \bullet x^{-1} \in W(X)$. Por otro lado, dada $w = x_1^{r_1} \bullet \cdots \bullet x_n^{r_n} \in W(X)$, la expresión $w' = x_n^{q_n} \bullet \cdots \bullet x_1^{q_1}$ con $q_i = -r_i$ para toda $i = 1, ..., n$, también es una palabra en $W(X)$, ya que se tiene $x_i^{q_i} = x_i^{-1}$ si $r_i = +1$, o bien $x_i^{q_i} = x_i^{+1} = x_i$ si $r_i = -1$, por lo que además $x_i^{q_i} = (x_i^{r_i})^{-1}$ para cualquier $r_i \in \{-1, +1\}$. Pero de esta forma, inductivamente tenemos

$$\begin{aligned} w \bullet w' &= (x_1^{r_1} \bullet \cdots \bullet x_n^{r_n}) \bullet (x_n^{q_n} \bullet \cdots \bullet x_1^{q_1}) \\ &= x_1^{r_1} \bullet \cdots \bullet (x_n^{r_n} \bullet x_n^{q_n}) \bullet \cdots \bullet x_1^{q_1} \\ &= x_1^{r_1} \bullet \cdots \bullet (x_n^{r_n} \bullet (x_n^{r_n})^{-1}) \bullet \cdots \bullet x_1^{q_1} \\ &= x_1^{r_1} \bullet \cdots \bullet e \bullet \cdots \bullet x_1^{q_1} \\ &\;\;\vdots \\ &= e, \end{aligned}$$

por lo que $w^{-1} = w' \in W(X)$. Finalmente, como para cualesquiera palabras $w_1 = x_{11}^{r_{11}} \bullet \cdots \bullet x_{1n}^{r_{1n}}$ y $w_2 = x_{21}^{r_{21}} \bullet \cdots \bullet x_{2n}^{r_{2n}}$ se verifica que $w_1 \bullet w_2 = x_{11}^{r_{11}} \bullet \cdots \bullet x_{1n}^{r_{1n}} \bullet x_{21}^{r_{21}} \bullet \cdots \bullet x_{2n}^{r_{2n}} \in W(X)$, podemos concluir que $W(X)$ es un subgrupo de $\mathcal{G}$. ■

Proposición 2.2.7. *Sea $(\mathcal{G}, \bullet)$ un grupo y sea X un subconjunto arbitrario de $\mathcal{G}$. Entonces $W(X) = \langle X \rangle$.*

Demostración. Todo $w \in W(X)$ se escribe como $w = x_1^{r_1} \bullet x_2^{r_2} \bullet \cdots \bullet x_n^{r_n}$, para una sucesión $\{x_1, x_2, ..., x_n\} \subseteq X$, y con $r_i \in \{-1, +1\}$, es decir que $x_i^{r_i} \in \{x_i, x_i^{-1}\}$ para toda $i \in \{1, ..., n\}$. Pero por la **Proposición 2.2.4**, tenemos que $x_i, x_i^{-1} \in \langle X \rangle$ para toda i, y en consecuencia $w = x_1^{r_1} \bullet x_2^{r_2} \bullet \cdots \bullet x_n^{r_n} \in \langle X \rangle$, al ser $\langle X \rangle$ un subgrupo de $\mathcal{G}$. Así, como w fue arbitrario, tenemos $W(X) \subseteq \langle X \rangle$. No obstante, ya que por definición toda $x \in X$ es también una palabra $x \in W(X)$, entonces $X \subseteq W(X)$. Más aún, por la **Proposición 2.2.6**, $W(X)$ es un subgrupo de X. Sin embargo, por la **Proposición 2.2.5**, $\langle X \rangle$ es el menor subgrupo de $\mathcal{G}$ que contiene a X, por lo que $\langle X \rangle \subseteq W(X)$. Como esto prueba las contenciones en ambos sentidos, podemos ver que $W(X) = \langle X \rangle$. ■

Grupos de Permutaciones.

Una *permutación* en un conjunto X es una biyección de X en sí mismo. Para comprender a las amoebas estudiaremos las permutaciones de conjuntos finitos no vacíos, que están representadas por las permutaciones del conjunto $[n] = \{1, ..., n\}$ con $n \geq 1$. Recordemos además que para cualesquiera conjuntos X, Y y Z, la composición de una biyección $f : X \to Y$, seguida por otra biyección $g : Y \to Z$, es también una biyección $gf : X \to Z$ tal que para cada $x \in X$ se tiene $gf(x) = g(f(x))$. Iniciamos este apartado viendo que la colección de las permutaciones del conjunto $\{1, ..., n\}$, junto con su composición, tiene estructura algebraica de grupo.

Proposición 2.2.8. *Sea S_n, para $n \geq 1$, el conjunto de todas las biyecciones de $[n] = \{1, ..., n\}$ en sí mismo. Entonces S_n constituye un grupo junto con la composición de biyecciones como operación binaria.*

Demostración. Debemos mostrar que las propiedades **i) - iv)** en la **Definición 2.2.1** se cumplen para S_n con la composición de biyecciones. Sabemos que para cualesquiera $\sigma, \rho \in S_n$, se tiene $\sigma\rho \in S_n$, ya que $\sigma\rho$ es también una biyección de $[n]$ en sí mismo, con lo que se verifica la **i) Cerradura** en este conjunto. De forma similar, por la composición de biyecciones se tiene $(\sigma\rho)\pi = \sigma(\rho\pi)$ para cualesquiera $\rho, \sigma, \pi \in S_n$, verificándose así la **ii) Asociatividad** de esta operación. Consideremos ahora a la biyección I_n de $[n]$ en $[n]$ dada por $I_n(i) = i$ para todo $i \in [n]$. Componiendo esta con cualquier $\sigma \in S_n$ se tiene $\sigma I_n(i) = \sigma(I_n(i)) = \sigma(i) = I_n(\sigma(i)) = I_n\sigma(i)$, por lo que I_n satisface la definición de **iii) Elemento Neutro** para S_n con esta operación. Finalmente, ya que para toda $\sigma \in S_n$ existe una biyección inversa $\sigma^{-1} \in S_n$ con la que por definición se tiene $\sigma\sigma^{-1} = I_n = \sigma^{-1}\sigma$, podemos deducir que σ^{-1} es en efecto el **iv) Elemento Inverso** de σ respecto a la composición. ■

Es común encontrar en la literatura el uso del símbolo $\circ$ para denotar la composición $\sigma \circ \rho = \sigma\rho$ de biyecciones $\sigma, \rho \in S_n$. De esta forma, podríamos haber definido la composición $\circ$ de elementos de S_n, como el mapeo con dominio en $S_n \times S_n$ dado por $\circ((\sigma, \rho)) = \sigma \circ \rho = \sigma\rho$. Así, siguiendo la notación de la **Definición 2.2.1**, lo que la **Proposición 2.2.8** indica formalmente es que el par $(S_n, \circ)$ es un grupo con la operación binaria $\circ : S_n \times S_n \to S_n$ dada.

No obstante, buscando simplificar la notación, en esta tesis expresaremos la composición entre elementos del conjunto S_n como $\sigma\rho$ omitiendo el símbolo $\circ$, conocida en álgebra como *notación multiplicativa*, y no volveremos a hacer mención explícita de este hecho.

Definición 2.2.9. *Sea n un entero positivo. Se llama* **grupo simétrico** *en n elementos, al grupo formado por el conjunto S_n de todas las biyecciones de $[n]$ en sí mismo, y con la composición de biyecciones como operación binaria. Nos referiremos como* **permutación** *en $[n]$ a cada elemento de S_n y en particular diremos que la biyección I_n dada por $I_n(i) = i$ para todo $i \in [n]$, es la* **permutación identidad** *de S_n. Similarmente, dada cualquier $\sigma \in S_n$, diremos que la biyección inversa σ^{-1} es la* **permutación inversa** *de σ en S_n.*

Nótese que por las **Proposiciones 2.2.2** y **2.2.8**, la permutación identidad I_n de S_n es el único elemento neutro de S_n, y a la vez, toda $\sigma \in S_n$ tiene una única permutación inversa $\sigma^{-1} \in S_n$.

Ahora estudiaremos dos formas distintas de denotar a las permutaciones en S_n, una como n-tuplas y la otra ampliamente conocida como *notación en ciclos*. La primera será de ayuda para determinar el orden de este grupo y se incluye además ya que juega un breve papel en la implementación computacional de la detección de amoebas. No obstante, prestaremos mayor atención a la segunda representación, ya que esta resulta más compacta que la primera y se usa para el desarrollo teórico de las ideas a lo largo de toda la tesis.

Definición 2.2.10. *Sea S_n el grupo simétrico en $n \geq 1$ elementos. Para cada $\sigma \in S_n$, llamaremos* **representación como n-tupla** *de σ, a la n-tupla $(x_1, x_2, ..., x_n)$ con $x_i \in [n]$ para toda $i \in [n]$, y con $x_i \neq x_j$ para $i \neq j$, tal que $x_i = \sigma(i)$ para $i \in [n]$.*

Proposición 2.2.9. *Para cualquier $n \geq 1$, el grupo simétrico S_n tiene orden $n!$.*

Demostración. Nótese que toda $\sigma \in S_n$ tiene una única representación como n-tupla, ya que para cualesquiera dos posibles representaciones de σ, digamos $X = (x_1, ..., x_n)$ y $Y = (y_1, ..., y_n)$, por definición se tiene $x_i = \sigma(i) = y_i$ para toda $i \in [n]$, de donde $X = Y$. Luego, dado el conjunto $\mathcal{D}$ de las n-tuplas con entradas en distintos elementos de $[n]$, podemos definir una función $\Phi : S_n \to \mathcal{D}$ que a cada σ asocia su representación $(\sigma(1), \sigma(2), ..., \sigma(n))$, y veremos que esta es una biyección. En efecto, de tener $\Phi(\sigma) = \Phi(\rho)$ para $\sigma, \rho \in S_n$, se sigue $(\sigma(1), \sigma(2), ..., \sigma(n)) = (\rho(1), \rho(2), ..., \rho(n))$, y entonces $\sigma(i) = \rho(i)$ para toda $i \in [n]$, por lo que $\sigma = \rho$, es decir, Φ es inyectiva. Por otro lado, dada cualquier $(d_1, ..., d_n) \in \mathcal{D}$, al tener $d_i \in [n]$ para toda $i \in [n]$ y $d_i \neq d_j$ si $i \neq j$, podemos definir una biyección σ de $[n]$ en sí mismo, tal que a cada $i \in [n]$ asocie su respectivo d_i, escribiendo $d_i = \sigma(i)$. Sin embargo, por definición $\sigma \in S_n$ y más aún $\Phi(\sigma) = (d_1, ..., d_n)$, por lo que Φ es sobreyectiva y en consecuencia biyectiva. Luego $|S_n| = |\mathcal{D}|$. Pero por conteo sabemos que existen $n!$ tuplas en $\mathcal{D}$, ya que existen n formas de escoger una primera entrada x_1 en cada tupla, $n-1$ formas para la segunda entrada x_2 y así sucesivamente. De esta forma concluimos que S_n mismo tiene $n!$ elementos. ■

A continuación establecemos algunas definiciones que nos facilitarán hablar sobre la notación en ciclos. De estas se siguen propiedades ampliamente conocidas de las permutaciones y de dicha notación. Sin embargo, buscando sintetizar el contenido, nuevamente omitiremos las pruebas de estas propiedades, así como las de la mayoría de los demás resultados en el resto de este apartado.

Definición 2.2.11. *Sea S_n el grupo simétrico en $n \geq 1$ elementos y sea $\sigma \in S_n$. Se define como* **soporte** *de la permutación σ, al conjunto $Sop(\sigma) = \{k \in [n] \mid \sigma(k) \neq k\}$.*

Adicionalmente, dada $\sigma \in S_n$ y dado cualquier $k \in [n]$, se dice que σ mueve a k si $\sigma(k) \neq k$. Por otro lado, si $\sigma(k) = k$, entonces decimos que σ deja fijo a dicha k.

Definición 2.2.12. *Sea S_n el grupo simétrico en $n \geq 1$ elementos. Decimos que dos permutaciones $\sigma, \rho \in S_n$ son* **disjuntas**, *si $Sop(\sigma) \cap Sop(\rho) = \varnothing$, es decir, si mueven a colecciones disjuntas de elementos.*

La siguiente es una propiedad ampliamente conocida sobre permutaciones disjuntas.

Proposición 2.2.10. *Para cualesquiera dos permutaciones disjuntas $\sigma, \rho \in S_n$, se tiene $\sigma\rho = \rho\,\sigma$.* ■

Definición 2.2.13. *Sea S_n el grupo simétrico en $n \geq 1$ elementos y sea $\sigma \in S_n$. Si el soporte de σ se puede escribir como una sucesión $Sop(\sigma) = \{k_1, k_2, ..., k_r\}$ de $r \geq 1$ elementos distintos de $[n]$, tales que $\sigma(k_1) = k_2$, $\sigma(k_2) = k_3$, ..., $\sigma(k_{r-1}) = k_r$ y $\sigma(k_r) = k_1$, entonces decimos que σ es una* **permutación cíclica** *o un* **ciclo** *de* **longitud** *r, o bien que σ es un r-***ciclo***, y la denotamos por $\sigma = (k_1\ k_2\ \cdots\ k_r)$. En particular, todo ciclo de longitud 2 es llamado* **transposición**.

Observación 2.2.2. *Sea S_n el grupo simétrico en $n \geq 1$ elementos y sea $\sigma \in S_n$. Si σ es un r-ciclo, digamos $\sigma = (k_1\ k_2\ \cdots\ k_{r-1}\ k_r)$, entonces σ se puede escribir en r formas equivalentes, dadas por,*

$$\sigma = (k_1\ k_2\ \cdots\ k_{r-1}\ k_r) = (k_2\ k_3\ \cdots\ k_r\ k_1) = \cdots = (k_r\ k_1\ \cdots\ k_{r-2}\ k_{r-1})$$ ■

Nótese además que por la **Proposición 2.2.1**, la composición $\sigma_1\sigma_2\cdots\sigma_m$ de cualesquiera m permutaciones de S_n, corresponde a una única permutación $\sigma_1\sigma_2\cdots\sigma_m = \alpha \in S_n$.

Definición 2.2.14. *Sea S_n el grupo simétrico en $n \geq 1$ elementos y sean $\sigma_1, \sigma_2, ..., \sigma_m$ cualesquiera m permutaciones de S_n. Si σ_j es un ciclo de S_n para $j = 1, ..., m$, digamos con $\sigma_j = (k_{j1} \cdots k_{jr_j})$ para $r_j \geq 1$, entonces denotaremos la composición $\sigma_1\sigma_2\cdots\sigma_m$ por*

$$\sigma_1\sigma_2\cdots\sigma_m = (k_{11}\ \cdots\ k_{1r_1})(k_{21}\ \cdots\ k_{2r_2})\cdots(k_{m1}\ \cdots\ k_{mr_m})$$

y diremos que la (única) permutación $\alpha = \sigma_1\sigma_2\cdots\sigma_m \in S_n$, es un **producto de ciclos** *de S_n y que $(k_{11}\ \cdots\ k_{1r_1})(k_{21}\ \cdots\ k_{2r_2})\cdots(k_{m1}\ \cdots\ k_{mr_m})$ es una* **descomposición en ciclos** *de α en S_n.*

Teorema 2.2.1. *Sea S_n el grupo simétrico en $n \geq 1$ elementos. Toda permutación en S_n se puede escribir, o bien como un ciclo, o como un producto de ciclos disjuntos de S_n.* ■

Recordemos que, dados conjuntos no vacíos A y B, toda función $f : A \to B$, con dominio finito $A = \{a_1, a_2, ..., a_n\}$, se puede expresar usando la *notación matricial* como

$$f = \begin{pmatrix} a_1 & a_2 & \cdots & a_n \\ f(a_1) & f(a_2) & \cdots & f(a_n) \end{pmatrix}, \text{ donde } f(a_i) \in B \text{ para toda } i \in [n]$$

Así, como ejemplo del **Teorema 2.2.1**, considérese $\sigma \in S_{10}$ dada primero en notación matricial y posteriormente escrita como producto de ciclos disjuntos de S_{10},

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 2 & 3 & 4 & 1 & 6 & 7 & 5 & 9 & 8 & 10 \end{pmatrix} = (1\,2\,3\,4)(5\,6\,7)(8\,9),$$

donde en particular se omite el 1-ciclo dado por (10), ya que este valor queda fijo bajo σ y por lo tanto es redundante indicar su imagen. Omitir los 1-ciclos es común en la práctica y usaremos esta notación en toda la tesis. Sin embargo, como veremos a continuación, cuando estos sí son considerados podemos dar a cada $\sigma \in S_n$ una representación en ciclos única.

Definición 2.2.15. *Sea S_n el grupo simétrico en $n \geq 1$ elementos. Se dice que una descomposición en ciclos disjuntos para una permutación $\sigma \in S_n$, es una* **factorización completa** *de σ, si dicha descomposición incluye a todos los 1-ciclos que representan a todos los elementos que son fijados por σ. Cada ciclo en la descomposición es conocido como* **factor** *de la descomposición.*

Teorema 2.2.2. *Sea S_n el grupo simétrico en $n \geq 1$ elementos. La factorización completa de cualquier $\sigma \in S_n$ es única salvo por el orden de sus factores.* ■

De todo lo anterior se siguen algunas propiedades de interés para el grupo S_n. La demostración del siguiente resultado se puede encontrar en [14].

Teorema 2.2.3. *Sea S_n el grupo simétrico en $n \geq 2$ elementos. Todos los siguientes son generadores de S_n,*

$$\begin{aligned} S_n &= \langle (1\,2), (1\,2\,\cdots\,n) \rangle \\ &= \langle (1\,2), (1\,3), ..., (1\,n) \rangle \\ &= \langle (1\,2), (2\,3), (3\,4)..., (n-1\;\;n) \rangle \end{aligned}$$

■

Observación 2.2.3. *El grupo simétrico S_n en $n \geq 1$ elementos, es isomorfo al subgrupo del grupo S_{n+1}, compuesto por todas las permutaciones de S_{n+1} que dejan fijo al elemento $n+1$.* ■

Proposición 2.2.11. *Sea S_n el grupo simétrico en $n \geq 1$ elementos considerado como subgrupo del grupo S_{n+1}. Sea además $\mathcal{H} \leq S_{n+1}$. Si se tiene $S_n \leq \mathcal{H} \leq S_{n+1}$, entonces $\mathcal{H} = S_n$ o bien $\mathcal{H} = S_{n+1}$.*

Demostración. Si todas las permutaciones en $\mathcal{H}$ dejan fijo al elemento $n+1$, por la **Observación 2.2.3** tendríamos $\mathcal{H} = S_n$. Supongamos entonces que existe en $\mathcal{H}$ por lo menos una permutación σ que mueve al $n+1$. Luego, por el **Teorema 2.2.1** sabemos que σ es, un ciclo que mueve a $n+1$, o bien se puede escribir como un producto de ciclos disjuntos, digamos $\sigma = \beta_1\beta_2\cdots\beta_k(a\cdots c\, b\ n+1)$ para una cantidad $k \geq 1$ de ciclos β_i que no mueven a $n+1$ y un único ciclo $(a\cdots c\, b\ \ n+1)$ que sí mueve a $n+1$. Pero de esta forma $\beta_i \in S_n$ para toda $1 \leq i \leq k$, por lo que $\beta_i^{-1} \in S_n \leq \mathcal{H}$ y consecuentemente $(a\cdots c\, b\ n+1) = [\beta_1\beta_2\cdots\beta_k(a\cdots c\, b\ n+1)]\beta_1^{-1}\beta_2^{-1}\cdots\beta_k^{-1} \in \mathcal{H}$. Consideremos sin pérdida de generalidad al ciclo $(a\cdots c\, b\ \ n+1) \in \mathcal{H}$ que sí mueve a $n+1$ y también a la transposición $(a\, b) \in S_n \leq \mathcal{H}$. Con estos se tiene $(a\cdots c)(b\ n+1) = (a\, b)(a\cdots c\, b\ n+1) \in \mathcal{H}$. Así, nuevamente $(a\cdots c) \in S_n$ y entonces $(b\ n+1) = [(a\cdots c)(b\ n+1)](a\cdots c)^{-1} \in \mathcal{H}$. Más aún, como $S_n \leq \mathcal{H}$, entonces $\{(b\,1), (b\,2), ..., (b\,n), (b\ n+1)\} \subseteq \mathcal{H}$. Pero por el **Teorema 2.2.3** sabemos que sin pérdida de generalidad $S_{n+1} = \langle (b\,1), (b\,2), ..., (b\,n), (b\ n+1)\rangle$ y de esta forma, de la **Proposición 2.2.5** se sigue que $S_{n+1} = \langle (b\,1), (b\,2), ..., (b\,n), (b\ n+1)\rangle \leq \mathcal{H} \leq S_{n+1}$ y por lo tanto $\mathcal{H} = S_{n+1}$. ■

Para concluir este apartado incluimos la definición de dos subgrupos de S_n, específicamente el *Grupo Alternante* y el *Grupo Dihédrico*. Para poder hablar del primero de estos subgrupos, a continuación definimos lo que se conoce como el *signo* de una permutación. Debemos mencionar que existen varias definiciones equivalentes para tal concepto, sin embargo sólo trataremos con una de estas, que es la más apropiada para los propósitos de esta tesis.

Definición 2.2.16. *Sea S_n el grupo simétrico en $n \geq 1$ elementos y sea además $\alpha \in S_n$. Dada la (única) factorización completa de α en S_n, digamos $\alpha = \sigma_1\sigma_2\cdots\sigma_m$ con $\sigma_i \in S_n$ para $i = 1, ..., m$, se define el* **signo** *de α para ser el valor $sgn(\alpha) \in \{-1, +1\}$, dado por $sgn(\alpha) = (-1)^{n-m}$.*

Proposición 2.2.12. *Sea S_n el grupo simétrico en $n \geq 1$ elementos. Para cualesquiera $\sigma, \rho \in S_n$ se tiene $sgn(\sigma\rho) = sgn(\sigma)sgn(\rho)$.* ■

Proposición 2.2.13. *Sea S_n el grupo simétrico en $n \geq 1$ elementos. El subconjunto A_n de S_n dado por $A_n = \{\sigma \in S_n \mid sgn(\sigma) = +1\}$, es un subgrupo de S_n.*

Demostración. Verificaremos las tres propiedades en la **Proposición 2.2.3**. La permutación identidad $I_n \in S_n$ tiene factorización completa $I_n = (1)(2)\cdots(n)$, de donde $sgn(I_n) = (-1)^{n-n} = +1$, por lo que $I_n \in A_n$. Por otro lado, usando la **Proposición 2.2.12**, respecto a cualquier $\sigma \in A_n$ se tiene $+1 = sgn(I_n) = sgn(\sigma\sigma^{-1}) = sgn(\sigma)sgn(\sigma^{-1}) = (+1)sgn(\sigma^{-1}) = sgn(\sigma^{-1})$, por lo que $\sigma^{-1} \in A_n$. De forma similar, para cualesquiera $\sigma, \rho \in A_n$ se tiene $sgn(\sigma\rho) = sgn(\sigma)sgn(\rho) = (+1)(+1) = +1$ y entonces $\sigma\rho \in A_n$. Con todas estas propiedades se sigue que $A_n \leq S_n$, como requerido. ■

Definición 2.2.17. *Sea S_n el grupo simétrico en $n \geq 1$ elementos. El subgrupo A_n de S_n dado por $A_n = \{\sigma \in S_n \mid sgn(\sigma) = +1\}$, es llamado* **subgrupo alternante** *de S_n.*

Proposición 2.2.14. *Sea S_n el grupo simétrico en $n \geq 1$ elementos y sea A_n el subgrupo alternante de S_n. Para $n \geq 2$ se tiene $|A_n| = n!/2$, y para $n = 1$ se tiene $|S_1| = |A_1| = 1$.* ■

Por último definiremos el *Grupo Dihédrico* haciendo referencia a conceptos fundamentales de geometría en el plano, y finalizamos este apartado relacionando este grupo con Teoría de Gráficas.

Definición 2.2.18. *Sea S_n el grupo simétrico en $n \geq 3$ elementos y sea P^n un polígono regular con n lados y con vértices en $\{1, ..., n\}$. Se llama* **grupo dihédrico** *al subgrupo D_n de S_n, formado por las simetrías de P^n dadas como permutaciones de sus vértices, y con la composición de biyecciones como operación binaria.*

Observación 2.2.4. *El grupo dihédrico D_n tiene orden $2n$.*

Demostración. El polígono tiene exactamente n simetrías rotacionales y n simetrías de reflexión. ■

Observación 2.2.5. *Sea G una gráfica de orden $n \geq 3$. Si G es un ciclo, entonces $Aut(G) \cong D_n$.*

Demostración. En la siguiente sección veremos que $Aut(G)$ es un grupo. Por otro lado, el isomorfismo de grupos entre $Aut(G)$ y D_n se sigue al mapear los vértices y aristas de G hacia el plano, sobreponiéndolo con un polígono regular P^n. Así, todas las simetrías de P^n se corresponden una a una con todos los automorfismos de G, respetándose además la composición de biyecciones. ■

Acción de un grupo sobre un conjunto.

Con las definiciones en este apartado veremos que los elementos de un grupo $\mathcal{G}$ no sólo se pueden "operar" entre sí, sino también con los de un conjunto X que inclusive sea disjunto de $\mathcal{G}$.

Definición 2.2.19. *Sea $(\mathcal{G}, \bullet)$ un grupo con neutro e y sea X cualquier conjunto no vacío. Diremos que una función $\blacksquare : \mathcal{G} \times X \to X$ es una* **acción a izquierda** *(o simplemente* **acción***) de $\mathcal{G}$ sobre X, y denotaremos por $g \blacksquare x$ a la imagen $\blacksquare((g, x)) \in X$ de $(g, x) \in \mathcal{G} \times X$ bajo $\blacksquare$, si se cumplen las siguientes dos propiedades,*

i) Acc 1: *$e \blacksquare x = x$ para todo $x \in X$.*

ii) Acc 2: *$(g \bullet g') \blacksquare x = g \blacksquare (g' \blacksquare x)$ para cualesquiera $g, g' \in \mathcal{G}$ y todo $x \in X$.*

Observación 2.2.6. *Sean $(\mathcal{G}, \bullet)$ un grupo, X un conjunto no vacío y $\blacksquare$ una acción de $\mathcal{G}$ sobre X. Para cualesquiera $x, y \in X$ y todo $g \in \mathcal{G}$ se tiene $x = g \blacksquare y$ si y sólo si $g^{-1} \blacksquare x = y$.*

Demostración. Si $x = g \blacksquare y$, de la definición $g^{-1} \blacksquare x = g^{-1} \blacksquare (g \blacksquare y) = (g^{-1} \bullet g) \blacksquare y = e \blacksquare y = y$, es decir, $g^{-1} \blacksquare x = y$. La implicación en el otro sentido se obtiene de forma análoga. ■

Ahora definiremos a las *órbitas* de los elementos de un conjunto X obtenidas bajo la acción de un grupo $\mathcal{G}$. Es común encontrar en la literatura el uso de $\mathcal{G}(x)$, o bien $\mathcal{G}x$, para hablar de la órbita de $x \in X$ bajo alguna acción de $\mathcal{G}$. No obstante, buscando mantener claridad en la notación, en esta tesis pondremos $\mathcal{G}[x]$ para referirnos a dicho conjunto, ya que nuestro interés en el uso de las órbitas es estudiar las acciones del grupo $\mathcal{G} = Aut(G)$ sobre los vértices y aristas de una gráfica G.

Definición 2.2.20. *Sean $\mathcal{G}$ un grupo, X un conjunto no vacío y $\blacksquare$ una acción de $\mathcal{G}$ sobre X. Para cada $x \in X$, nos referiremos como* **órbita** *de x bajo $\blacksquare$, al conjunto $\mathcal{G}[x] = \{g \blacksquare x \in X \mid g \in \mathcal{G}\}$.*

Proposición 2.2.15. *Sean $(\mathcal{G}, \bullet)$ un grupo con neutro e, X un conjunto no vacío y $\blacksquare$ una acción de $\mathcal{G}$ sobre X. La colección $\{\mathcal{G}[x]\}_{x \in X}$ de las órbitas de elementos de X bajo $\blacksquare$ es una partición de X.*

Demostración. Por definición $e \blacksquare x = x$ para todo $x \in X$, por lo que $x \in \mathcal{G}[x]$ y entonces $\mathcal{G}[x] \neq \varnothing$. De aquí además se obtiene $X = \bigcup_{x \in X} \mathcal{G}[x]$ y sólo hace falta mostrar que cualesquiera dos órbitas distintas son también disjuntas. Considérense las órbitas $\mathcal{G}[x]$ y $\mathcal{G}[y]$ de distintos $x, y \in X$ (de existir). Supóngase que $\mathcal{G}[x] \cap \mathcal{G}[y] \neq \varnothing$ y escójase $z \in \mathcal{G}[x] \cap \mathcal{G}[y]$. Entonces debe haber distintos $g_x, g_y \in \mathcal{G}$ tales que $g_x \blacksquare x = z = g_y \blacksquare y$. Pero por la **Observación 2.2.6** podemos también escribir $x = g_x^{-1} \blacksquare (g_y \blacksquare y) = (g_x^{-1} \bullet g_y) \blacksquare y$. Así, dado cualquier $w \in \mathcal{G}[x]$, como $w = g_w \blacksquare x$ para algún $g_w \in \mathcal{G}$, se sigue que $w = g_w \blacksquare x = g_w \blacksquare ((g_x^{-1} \bullet g_y) \blacksquare y) = (g_w \bullet g_x^{-1} \bullet g_y) \blacksquare y$, por lo que $w \in \mathcal{G}[y]$. Luego, al ser $w \in \mathcal{G}[x]$ arbitrario, debemos tener $\mathcal{G}[x] \subseteq \mathcal{G}[y]$. Más aún, por argumentos análogos $\mathcal{G}[y] \subseteq \mathcal{G}[x]$, y en consecuencia $\mathcal{G}[x] = \mathcal{G}[y]$. Pero esto quiere decir que si $\mathcal{G}[x] \neq \mathcal{G}[y]$, entonces tendremos $\mathcal{G}[x] \cap \mathcal{G}[y] = \varnothing$. Como x y y fueron arbitrarios, podemos concluir que esto se cumple para todo par de órbitas, como queríamos, y por lo tanto $\{\mathcal{G}[x]\}_{x \in X}$ forma una partición de X. ■

2.3. Grupo de Automorfismos de una Gráfica y sus Acciones sobre Vértices y Aristas

En esta sección relacionaremos lo visto hasta ahora sobre Teoría de Gráficas y Teoría de Grupos, para hablar sobre la colección de automorfismos $Aut(G)$ de una gráfica G (ver **Definición 2.1.15**). Primero estableceremos algunas definiciones y posteriormente veremos que $Aut(G)$ tiene estructura algebraica de grupo. Concluiremos definiendo dos acciones para $Aut(G)$, una dada sobre el conjunto de vértices y otra sobre las aristas de G misma.

Definición 2.3.1. *Sea $G = (V, E)$ una gráfica. El isomorfismo $I_G : V \to V$ de G en sí misma, dado por $I_G(v) = v$ para todo $v \in V$, es llamado* **automorfismo identidad** *de G.*

Ahora bien, cada uno de los automorfismos de una gráfica G, refleja una simetría particular de G, es decir, un mapeo de los vértices de G en sí mismos que preserva su estructura combinatoria. Más aún, algunos de estos pueden relacionarse con rotaciones, reflexiones y ejes de simetría de las representaciones visuales de G en el plano. De esta forma, si una gráfica G tiene como único automorfismo a la identidad, podremos concluir que dicha gráfica no posee simetrías.

Definición 2.3.2. *Sea $G = (V, E)$ una gráfica. Si la colección $Aut(G)$ de automorfismos de G contiene sólo al automorfismo identidad I_G de G, entonces decimos que G es una* **gráfica asimétrica**.

Por otro lado, también podemos encontrar ejemplos de gráficas que tienen todas sus simetrías.

Observación 2.3.1. *Sea $G = (V, E)$ una gráfica de orden n. Si G es vacía, o la gráfica completa con vértices en V, entonces $Aut(G) \simeq S_n$.* ■

A continuación mostraremos que $Aut(G)$ es en efecto un grupo.

Proposición 2.3.1. *Sea $G = (V, E)$ una gráfica. La colección $Aut(G)$ de automorfismos de G constituye un grupo junto con la composición de automorfismos como operación binaria.*

Demostración. Se verifican las propiedades **i)-iv)** de la **Definición 2.2.1**. La **i) Cerradura** se cumple por la **Proposición 2.1.2**, ya que la composición de isomorfismos φ y φ' de G en G, es otro isomorfismo $\varphi'\varphi$ de G en sí misma, es decir, $\varphi'\varphi \in Aut(G)$. Además, la **ii) Asociatividad** de esta operación se sigue inmediatamente de la asociatividad para la composición de biyecciones. Por otro lado, el automorfismo identidad I_G de G es tal que para cualquier $\varphi \in Aut(G)$ se tiene $\varphi I_G(v) = \varphi(I_G(v)) = \varphi(v) = I_G(\varphi(v)) = I_G\varphi(v)$ para todo $v \in V$, por lo que $\varphi I_G = I_G\varphi = \varphi$ y entonces I_G es el **iii) Elemento Neutro** de este grupo. Finalmente, por la **Proposición 2.1.1**, podemos ver que dado cualquier $\varphi \in Aut(G)$, la biyección inversa φ^{-1} de φ es igualmente un isomorfismo de G en G, esto es, $\varphi^{-1} \in Aut(G)$ y además $\varphi\varphi^{-1} = \varphi^{-1}\varphi = I_G$, por lo que este es el **iv) Elemento Inverso** de φ en $Aut(G)$. Todas esto muestra que $Aut(G)$ es un grupo. ■

Por las **Proposiciones 2.2.2** y **2.3.1**, el automorfismo identidad I_G de una gráfica G será el único elemento neutro de $Aut(G)$, y todo $\varphi \in Aut(G)$ tendrá un único inverso $\varphi^{-1} \in Aut(G)$.

Definición 2.3.3. *Sea G una gráfica. Nos referiremos como* **grupo de automorfismos** *de G al grupo formado por $Aut(G)$ junto con la composición de automorfismos como operación binaria y con neutro en el* **automorfismo identidad** *I_G de G. Adicionalmente, dada $\varphi \in Aut(G)$, nos referiremos como* **automorfismo inverso** *de φ en $Aut(G)$, al isomorfismo inverso φ^{-1} de φ.*

Nótese que, al ser $Aut(G)$ un grupo, la **Definición 2.3.2** es equivalente a definir una gráfica asimétrica G como aquella cuyo grupo de automorfismos es el grupo trivial $Aut(G) = \{I_G\}$.

Definición 2.3.4. *Sea $G = (V, E)$ una gráfica. Nos referiremos como* **acción natural de automorfismos sobre los vértices** *de G, a la acción $\odot : Aut(G) \times V \to V$, de $Aut(G)$ sobre V, dada por $\varphi \odot v = \varphi(v)$ para cualesquiera $\varphi \in Aut(G)$ y $v \in V$. Similarmente, y si G es no vacía, definimos como* **acción natural de automorfismos sobre las aristas** *de G, a la acción $\ominus : Aut(G) \times E \to E$, de $Aut(G)$ sobre E, dada por $\varphi \ominus uv = \varphi(u)\varphi(v)$ para cualesquiera $\varphi \in Aut(G)$ y $uv \in E$.*

Proposición 2.3.2. *Sea $G = (V, E)$ una gráfica. La acción natural $\odot$ de automorfismos sobre los vértices de G, está bien definida y es en efecto una acción del grupo $Aut(G)$ sobre el conjunto V. Igualmente, y si G es no vacía, la acción natural $\ominus$ de automorfismos sobre las aristas de G, está bien definida y es en efecto una acción del grupo $Aut(G)$ sobre el conjunto E.*

Demostración. Sin pérdida de generalidad supóngase que G es no vacía. Por definición, respecto a cada $\varphi \in Aut(G)$ todo $v \in V$ tiene asociado un único $\varphi(v) \in V$, y entonces $\odot$ asocia un único vértice $\varphi(v) \in V$ a cada par $(\varphi, v) \in Aut(G) \times V$. De forma similar, respecto a cualquier $\varphi \in Aut(G)$, a cada $uv \in E$ podemos asociar una única $\varphi(u)\varphi(v) \in E$, ya que φ mapea de forma única a los vértices de G y por definición preserva adyacencia, de donde $\ominus$ asocia una sola arista $\varphi(u)\varphi(v)$ a cada para $(\varphi, uv) \in Aut(G) \times E$. Por otro lado, respecto al automorfismo identidad I_G de G tenemos $I_G \odot v = I_G(v) = v$ y también $I_G \ominus uv = I_G(u)I_G(v) = uv$, con lo que se verifica **Acc 1** en la **Definición 2.2.19**. Pero por la definición de estas funciones tenemos,

$$(\varphi\varphi') \odot v = \varphi\varphi'(v) = \varphi(\varphi'(v)) = \varphi \odot \varphi'(v) = \varphi \odot (\varphi' \odot v),$$
$$(\varphi\varphi') \ominus uv = \varphi\varphi'(u)\varphi\varphi'(v) = \varphi(\varphi'(u))\varphi(\varphi'(v)) = \varphi \ominus \varphi'(u)\varphi'(v) = \varphi \ominus (\varphi' \ominus uv),$$

i.e., $(\varphi\varphi') \odot v = \varphi \odot (\varphi' \odot v)$ y además $(\varphi\varphi') \ominus uv = \varphi \ominus (\varphi' \ominus uv)$, que es **Acc 2** en la **Definición 2.2.19**. Por lo tanto $\odot$ y $\ominus$ son acciones del grupo $Aut(G)$ sobre, respectivamente, V y E en G. ■

Definición 2.3.5. *Sea $G = (V, E)$ una gráfica. Para cada $v \in V$, definimos la* **órbita natural del vértice** *v, como el conjunto $Aut(G)[v] = \{\varphi \odot v \in V \mid \varphi \in Aut(G)\}$, es decir, la órbita de v bajo la acción natural $\odot$ de $Aut(G)$ sobre V. Igualmente, y si G es no vacía, para cada $uv \in E$, definimos la* **órbita natural de la arista** *uv, como el conjunto $Aut(G)[uv] = \{\varphi \ominus uv \in E \mid \varphi \in Aut(G)\}$, es decir, la órbita de uv bajo la acción natural $\ominus$ de $Aut(G)$ sobre E.*

Recuérdese además que la colección de las órbitas de todos los elementos en un conjunto X, obtenidas bajo la acción de un grupo $\mathcal{G}$, forman una partición de X mismo (ver **Proposición 2.2.15**). Con base en esto se tiene la siguiente definición.

Definición 2.3.6. *Sea $G = (V, E)$ una gráfica. Si dados dos vértices u y v de G, se tiene $u \in Aut(G)[v]$ para la órbita natural $Aut(G)[v]$ de v, entonces decimos que u y v son* **vértices semejantes**. *De igual forma, y si G es no vacía, dadas dos aristas uv y xy de G, si se tiene $uv \in Aut(G)[xy]$ para la órbita natural $Aut(G)[xy]$ de xy, entonces diremos que uv y xy son* **aristas semejantes**.

Por último incluimos un par de propiedades que tratan con los grados de vértices semejantes. Recuérdese que se denota por $d_G(v)$ al grado de todo vértice v en una gráfica G.

Proposición 2.3.3. *Sea $G = (V, E)$ una gráfica. Si $u, v \in V$ son semejantes, entonces $d_G(u) = d_G(v)$.*

Demostración. Sea $Aut(G)[v]$ la órbita natural de un vértice $v \in V$. Considérese cualquier $u \in V$ con $u \in Aut(G)[v]$. Entonces existe $\varphi \in Aut(G)$ tal que $\varphi \odot v = u$, es decir, $u = \varphi(v)$. Además, ya que φ preserva adyacencia tenemos $vx \in E$ si y sólo si $\varphi(v)\varphi(x) \in E$ para todo $x \in V$. Pero φ es una biyección, por lo que v y u deben tener la misma cantidad de vecinos, esto es, $d_G(u) = d_G(v)$. ■

Proposición 2.3.4. *Sea $G = (V, E)$ una gráfica no vacía. Si uv y $u'v'$ son aristas semejantes de G, entonces $\{d_G(u), d_G(v)\} = \{d_G(u'), d_G(v')\}$.*

Demostración. Sea $Aut(G)[uv]$ la órbita natural de la arista $uv \in E$. Considérese cualquier $u'v' \in E$ con $u'v' \in Aut(G)[uv]$. Entonces existe $\varphi \in Aut(G)$ tal que $\varphi \ominus uv = u'v'$. Así, $u'v' = \varphi(u)\varphi(v)$, digamos con $u' = \varphi(u)$ y $v' = \varphi(v)$. Pero de esta forma, por la **Proposición 2.3.3** se tiene además $d_G(u) = d_G(u')$ y $d_G(v) = d_G(v')$, de donde finalmente $\{d_G(u), d_G(v)\} = \{d_G(u'), d_G(v')\}$. ■

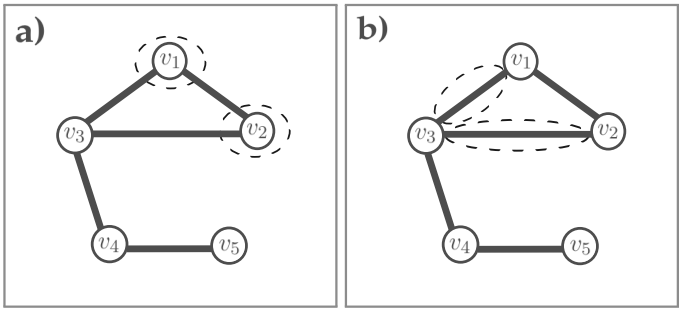

Figura 2.3: Una gráfica donde se encierran con círculos punteados **a)** dos vértices semejantes y **b)** dos aristas semejantes, obtenidos en ambos casos respecto a las acciones naturales de los automorfismos de esta gráfica.

2.4. Algoritmos, Complejidad Computacional e Isomorfismos entre Gráficas

Un algoritmo es, intuitivamente, una secuencia finita de instrucciones que se deben seguir para completar una tarea específica [10]. El funcionamiento de estos objetos se encuentra formalmente sustentado en la Teoría de la Computabilidad [15]. Sin embargo dicha área de estudios está fuera del alcance de esta tesis. Debido a esto, en el primer apartado de esta sección establecemos algunos términos y notación para describir algoritmos por medio de lo que se conoce como *pseudocódigo*, comparándolo con el lenguaje coloquial para facilitar la comprensión de los conceptos. En un segundo apartado tratamos con las definiciones de *tiempo de ejecución* y con la llamada *notación de la O grande*. Incluimos en un tercer apartado un criterio teórico que nos será de utilidad para aproximar el tiempo de ejecución de los algoritmos mostrados en esta tesis. Por último hablaremos sobre los algoritmos para identificar isomorfismos entre gráficas arbitrarias y entre árboles. Todo en esta sección es material usual de cursos de Computación, Programación y Diseño de Algoritmos [15–18], así como de las referencias sobre algoritmos de isomorfismos entre gráficas [19–25].

Expresar Algoritmos en Pseudocódigo.

Dos formas comunes para representar algoritmos son aquellas llamadas *diagramas de flujo* y *pseudocódigo*. La primera comprende el uso de gráficas dirigidas con vértices etiquetados, y facilita visualizar explícitamente todas las *decisiones* e *iteraciones* que suceden al desarrollarse un algoritmo. Sin embargo, ya que esto conlleva dibujar apropiadamente una digráfica conexa que resuma todo el procedimiento, el uso de esta representación puede resultar abrumador en comparación con el uso del pseudocódigo, en particular para algoritmos con una mayor cantidad de instrucciones.

Por su parte el pseudocódigo es, en esencia, un listado de las instrucciones en un algoritmo, expresadas en lenguaje coloquial e incluyendo lenguaje matemático. Este cuenta con una *sintáxis* propia para representar las decisiones e iteraciones en el algoritmo, así como para describir el *manejo* de las *variables* en este, en otras palabras, el pseudocódigo cuenta con términos y símbolos propios para denotar a las instrucciones conocidas como *estructuras de control* (e.g., If, While, For, entre otras), y para expresar la *asignación* y *modificación* de los valores de una variable por medio del llamado *operador asignación*, resultando ser una representación más versátil que los diagramas de flujo. Por ello recurriremos al pseudocódigo para representar los algoritmos en esta tesis.

Buscando facilitar la comprensión de la sintáxis del pseudocódigo, en este capítulo exhibiremos las semejanzas de todas estas instrucciones con expresiones del lenguaje coloquial, y mostraremos adicionalmente ejemplos del uso del operador asignación, que denotaremos por $\leftarrow$. Sin embargo en capítulos posteriores nos limitaremos a la sintáxis común del pseudocódigo, ya que esta refleja con mayor parentesco a las implementaciones computacionales de los algoritmos desarrollados.

A continuación mostramos una expresión en pseudocódigo para un algoritmo arbitrario, incluyendo los términos que hemos de usar en esta tesis. Esta se lee por renglones de arriba hacia abajo. Es importante mencionar que las instrucciones que aparecen en ella no tienen un papel necesariamente formal y sólo expresan lo que intuitivamente se puede entender por una **instrucción**.

Algoritmo 1: Ejemplo de pseudocódigo

Entrada: información inicial para el algoritmo.
Respuesta: información que el algoritmo garantiza obtener.

1	*instrucción*$_1$
2	*instrucción*$_2$
⋮	⋮
n-1	*instrucción*$_{n-1}$
n	*instrucción*$_n$

En el primer renglón escribiremos el nombre del algoritmo, seguido por una breve descripción de este o de su función. Después de esto aparecen las líneas de **Entrada** y **Respuesta**. Como sus nombres lo indican, en estas líneas se describen cualesquiera datos, objetos matemáticos o valores que, respectivamente, el algoritmo ha de recibir como variables para poder funcionar, o bien, que el algoritmo tiene como objetivo construir o determinar, y devolver como respuesta.

Después se tienen $n \geq 1$ renglones donde se enumeran las n instrucciones que componen al algoritmo. El índice (**k**) a la izquierda de la *instrucción*$_k$ se llamará **número** de línea y diremos que la *instrucción*$_k$ se **ejecuta** en la **línea** número k del algoritmo, o simplemente en la línea k del algoritmo. De esta forma, lo que esta representación indica es que para que el algoritmo funcione apropiadamente, las instrucciones se deben ejecutar siguiendo el orden de sus índices, es decir que para todo $i, j \in \{1, 2, ..., n\}$, la instrucción en la línea i se **ejecuta antes** de la instrucción en la línea j, o bien la línea j se **ejecuta después** de la línea i, siempre que $i \leq j$.

Cabe mencionar que el término *ejecutar*, a pesar de ser parte del lenguaje usual de la computación, no se incluye aquí como un verbo con una connotación práctica, sino simplemente se usará para indicar la existencia y posición relativa de cada instrucción en la expresión en pseudocódigo. No obstante, sí haremos uso intuitivo de la palabra **variable** para referirnos a objetos computacionales que pueden **guardar** valores durante la **ejecución** o **lectura** de un algoritmo. Así, diremos que una variable se **inicializan** o **define** en cierta línea k, si esta es la primera linea donde se le asigna un valor a la variable, y en cualquier otro caso diremos que su valor se ve **modificado** en esa línea.

Ahora mostraremos a las instrucciones llamadas **estructuras de control** en pseudocódigo, comparándolas con su expresión en lenguaje coloquial. Estas son de tres tipos fundamentales: las lineales, de decisión y de iteración. No daremos una lista exhaustiva de lo que se puede construir con ellas, pero sí presentamos lo necesario para comprender los algoritmos en esta tesis.

Por **estructura de control lineal** nos referimos a una sucesión finita de instrucciones consecutivas que han de ejecutarse dentro de un algoritmo, de manera independiente a los valores que puedan adquirir las variables en este. Esto parece redundante ya que se asemeja a la expresión en pseudocódigo para un algoritmo. Sin embargo, aclararemos su utilidad cuando hablemos de las *estructuras de control anidadas*.

A su vez, las **estructuras de control de decisión** son instrucciones que permiten a un algoritmo "escoger" ejecutar una instrucción en vez de otra, o bien, no ejecutar una instrucción dada. Esto se logra al revisar condiciones booleanas al respecto de los valores que tengan las variables en ese punto del algoritmo. Estas se pueden construir por medio de la llamada **estructura If**, que permite ejecutar una instrucción sólo si cierta condición booleana adquiere un valor verdadero, y simplemente omite dicha instrucción si tal valor es falso. Más aún, la **estructura If** puede ser dotada de una instrucción alternativa, para ejecutar cuando la condición booleana sea falsa, dando lugar a la llamada **estructura If ... Else**, cuya expresión en pseudocódigo mostramos a continuación.

Algoritmo 2: If...Else con sintáxis propia del pseudocódigo

Entrada: ...
Respuesta: ...
⋮
if (*condición booleana* **) then**
| *instrucción*$_T$
else
| *instrucción*$_F$
end
⋮

Se omiten los números de linea y se ejecuta una instrucción If...Else. Los puntos suspensivos verticales denotan instrucciones ejecutadas antes y después de ella. Esta indica que si la condición booleana es verdadera, se deberá ejecutar la *instrucción*$_T$, y al terminar el algoritmo deberá ignorar a la *instrucción*$_F$ y continuar con la instrucción después del último **end**, de existir. Similarmente, si la condición booleana es falsa, se ejecutará la *instrucción*$_F$ y se ignorará la *instrucción*$_T$. De no haber incluido el bloque **else** y la *instrucción*$_F$, diríamos que sólo se está ejecutando una **instrucción If**. Además, podemos expresar la instrucción If...Else en lenguaje coloquial de la siguiente forma.

Algoritmo 3: If...Else con lenguaje coloquial

Entrada: ...
Respuesta: ...
⋮
¿es cierta la **condición booleana**?
Sí: *instrucción*$_T$
No: *instrucción*$_F$
⋮

De esta forma, la instrucción If sería equivalente a no realizar ninguna acción bajo la respuesta **No** de la expresión anterior, en otras palabras, al poner *instrucción*$_F$ = "Continuar sin hacer nada" para la respuesta **No** de la pregunta "¿es cierta la **condición booleana**?", podemos reproducir el comportamiento de la estructura If con un If...Else para ignorar o sólo ejecutar la *instrucción*$_T$.

Por otro lado tenemos las **estructuras de control de iteración**, que permiten repetir una misma instrucción un número finito de veces. Con estas es posible además ejecutar múltiples ocasiones una misma instrucción que depende de los elementos de un conjunto finito dado. Estas estructuras tienen su base en lo que se conoce como **estructura While**, que repite una instrucción mientras se cumpla una condición booleana. No obstante, sólo trataremos con la **estructura For**, que de forma general depende de **recorrer** todos los elementos en un conjunto finito X, como mostramos ahora.

Algoritmo 4: For con sintáxis propia del pseudocódigo

Entrada: ...
Respuesta: ...
⋮
forall $x \in X$ **do**
  | *instrucción*$_R$
end
⋮

La instrucción For en esta tesis comienza con la palabra reservada **forall**. Esta estructura indica que por cada elemento x de un conjunto finito X se debe ejecutar la *instrucción*$_R$ que está **dentro** del For, y al terminar con todos los elementos de X se debe continuar con la instrucción después del último **end**, de existir. Si X es vacío, entonces se ignora el For y la *instrucción*$_R$ nunca se ejecuta. Además, en esta tesis no requeriremos que X sea un conjunto ordenado, por lo que la instrucción For puede escribirse en lenguaje coloquial como a continuación.

Algoritmo 5: For con lenguaje coloquial

Entrada: ...
Respuesta: ...
⋮
k Escoger $x \in X$ y hacer lo siguiente:
k+1 *instrucción*$_R$
k+2 Repetir desde la línea k hasta terminar con todo $x \in X$.
⋮

Cabe mencionar que el uso de instrucciones como la de la línea $k+2$, que indica el regreso a la línea k para realizar la iteración del For, resulta antinatural en la lectura de un algoritmo en pseudocódigo. En general, la acción de "forzar" la repetición de una línea, es conocida como **función go-to**, que incluso es posible utilizar fuera de un For. No obstante, tal uso no se recomienda, ya que abusar de esta puede producir algoritmos ambiguos o ilegibles, y sólo aparece aquí debido a las expresiones en lenguaje coloquial escogidas para facilitar la presentación de estos conceptos.

Es importante remarcar que todas las estructuras de control son a la vez instrucciones, por lo que estas mismas pueden aparecer dentro de otras estructuras de control, del mismo o distinto tipo, mientras que se respeten las posiciones destinadas para las instrucciones dentro de cada una de ellas. Por ejemplo, se puede contener un If o un If...Else dentro de un For para revisar condiciones sobre los elementos de un conjunto finito X, como veremos más adelante. Así, al componer un tipo de estructuras dentro de otras decimos que se obtienen **estructuras anidadas**.

En particular, se debe notar que el If, el If...Else, y el For, ejecutan todos a la vez una única instrucción, lo que puede resultar confuso si se buscan realizar más operaciones consecutivas dentro de estas. Esta es la razón por la que se consideran las estructuras de control lineales, de modo que se puedan construir apropiadamente expresiones en pseudocódigo como las siguientes.

Algoritmo 6: If...Else con múltiples instrucciones internas

Entrada: ...
Respuesta: ...
⋮
if (*condición booleana* **) then**
  instrucción$_{T_1}$
  instrucción$_{T_2}$
  ⋮
  instrucción$_{T_m}$
else
  instrucción$_F$
end
⋮

Algoritmo 7: For con múltiples instrucciones internas

Entrada: ...
Respuesta: ...
⋮
forall $x \in X$ **do**
  instrucción$_{R_1}$
  instrucción$_{R_2}$
  ⋮
  instrucción$_{R_m}$
end
⋮

Las sucesiones $\{$*instrucción*$_{T_1}$, ...,*instrucción*$_{T_m}\}$ y $\{$*instrucción*$_{R_1}$, ...,*instrucción*$_{R_m}\}$ están formadas por instrucciones consecutivas en sus respectivos algoritmos, por lo que pueden ser consideradas como estructuras lineales. Así, estas dos expresiones constituyen en realidad ejemplos básicos de estructuras anidadas, específicamente, estructuras lineales anidadas en un If...Else y en un For. Como ejemplo de un If anidado dentro de un For considérese el siguiente algoritmo, que determina el máximo elemento de un conjunto finito X de enteros positivos.

Algoritmo 8: If anidado en For y máximo entero positivo

Entrada: un conjunto finito y no vacío X de enteros positivos.
Respuesta: el máximo elemento de X.

```
1 inicializar variable max ↤ 0
2 forall x ∈ X do
3 |  if ( x > max ) then
4 |  |  max ↤ x
5 |  end
6 end
7 Devolver ( max )
```

Veamos lo que hace este algoritmo paso a paso. Recordemos que el símbolo $\mapsto$ denotará en esta tesis al *operador asignación*, definido en computación para **asignar** el valor ubicado a su derecha, a la variable que está a su izquierda. Así, en la línea 1 del algoritmo se está asignando el valor 0 a la variable **max**. Pero está es además la primera línea donde se le asigna un valor a **max**, por lo que decimos que esta variable se está **inicializando** con el valor 0 en el algoritmo. En la línea 2 comienza un For y se escoge algún $x \in X$. La primera instrucción dentro del For es la evaluación del If. Si el valor x escogido en el For es menor o igual que **max**, entonces repetiremos el For sin hacer nada. Por otro lado, si el x escogido es mayor que **max**, entonces **asignamos** el valor del x seleccionado a la variable **max**, **modificando** o **sobrescribiendo** el valor guardado por esta variable. Así, el For se repite modificando **max**, siempre que encuentra un entero mayor a su valor **actual**.

Debemos añadir que con el operador asignación se pueden construir expresiones de **actualización de variables**, por ejemplo de la forma "**n** $\mapsto$ **n** $+1$", donde primero se suma 1 al valor actual de una variable **n** y después se asigna el resultado a **n** misma, por lo que la función del operador asignación $\mapsto$ no debe ser confundida con la función de las igualdades matemáticas ($=$).

Finalmente, habiendo **recorrido** todos los valores en X, se ejecuta en la línea 7 la **Función Devolver(•)** (en inglés **return(•)** o también **exit(•)**). Esta es una función computacional que indica explícitamente la **respuesta** esperada de un algoritmo, que en este caso fue el valor de la variable **max**. Sin embargo, esta función no sólo permite **devolver** un número, sino también se puede usar para devolver cadenas de texto, es decir, respuestas en lenguaje coloquial, como a continuación.

Algoritmo 9: Función Devolver y entero par o impar

Entrada: un entero arbitrario n.
Respuesta: decir si n es par o impar.

```
1 if ( n/2 ∈ ℤ ) then
2 |  Devolver ( "El entero dado es par" )
3 else
4 |  Devolver ( "El entero dado es impar" )
5 end
6 instrucción que siempre será ignorada
```

Nótese que Devolver(•) no tiene porque ser la última instrucción del algoritmo, y de hecho puede aparecer varias veces en este. De ser el caso, se dice que la primera ejecución de cualquier aparición de Devolver(•), **termina** la lectura del algoritmo y regresa la respuesta correspondiente, por lo que la instrucción en la línea 6 nunca será ejecutada, independientemente del valor de n.

Tiempo de Ejecución y su Dependencia en la Entrada.

Habiendo establecido los conceptos básicos sobre algoritmos en el apartado anterior, ahora hablaremos sobre la *cantidad de operaciones* que estos realizan al ejecutarse. Consideremos el **Algoritmo 8** que devuelve el máximo elemento en un conjunto finito y no vacío X de enteros positivos. Si X tiene un único elemento, el For en este algoritmo se ejecutará una sola vez, y entonces el algoritmo hará un total de cinco *operaciones*: asignar el valor cero a la variable **max**, escoger al $x \in X$, evaluar la condición booleana del If, asignar el nuevo valor a **max** y devolver **max**.

Supóngase ahora que $|X| = n$ para alguna $n \geq 1$. Digamos que X se puede escribir como $X = \{x_1, x_2, ..., x_n\}$ con $x_1 < x_2 < \cdots < x_n$. La cantidad *exacta* de operaciones que debe hacer ahora el **Algoritmo 8** puede variar dependiendo del orden en que este escoja los elementos de X, ya que para algunas selecciones tendrá que modificar el valor de la variable **max** y para otras no.

Sin embargo, en un *peor caso*, este escogería primero a x_1, luego a x_2 y así sucesivamente hasta x_n, de modo que por cada selección también se deberá sobrescribir el valor de **max**. De esta forma, es posible *estimar* que el algoritmo realizará a lo más $T(n) = 3n + 2$ operaciones, específicamente: 1 operación en la línea 1, otra operación en la línea 7 y $3n$ operaciones en las líneas 2 a 6.

Así, este ejemplo nos permite ver que la cantidad estimada de operaciones $T(n)$ de un algoritmo, no sólo depende de las instrucciones que lo componen, sino también del **tamaño** de su entrada, i.e., $T(n)$ no siempre es una función constante y en general depende de la cantidad n de *datos* que se le proporcione al algoritmo. Pero este mismo ejemplo nos muestra que la cantidad $T(n)$ para un algoritmo dado, se puede calcular por conteo sobre sus instrucciones, ya sean estas expresiones aritméticas, estructuras lineales, de decisión, de iteración o incluso anidadas.

No obstante, para hacer dicho cálculo requerimos primero definir lo que se entiende por una *operación* realizada por un algoritmo, que como vimos difiere de los conceptos de línea y de instrucción. Por ejemplo, en la literatura usualmente se consideran como operaciones todas o algunas de las siguientes: la evaluación de expresiones booleanas, operaciones aritméticas, escoger un elemento de un conjunto, agregar o remover un único elemento de un conjunto, inicializar o modificar una variable y funciones como Devolver (•) que entregan una respuesta o *solicitan* una entrada.

Sin embargo, no existe una definición formal, ni una convención globalmente aceptada, para el concepto de operación. Esto se debe, en particular, a que la cantidad de operaciones en un algoritmo no se considera como una medida absoluta asociada a estos, sino simplemente como una estimación útil para comparar algoritmos que realizan tareas semejantes. Por este motivo, en esta tesis entenderemos por **operación** a aquellas antes enlistadas y sustentadas en la literatura. Con base en esto, damos ahora dos primeras definiciones que nos serán de utilidad para hablar de la cantidad de operaciones realizadas por los algoritmos a estudiar.

Definición 2.4.1. *Considérese cualquier algoritmo en pseudocódigo como el mostrado a continuación, compuesto por $m \geq 1$ instrucciones y con una entrada de tamaño $n \geq 1$,*

Algoritmo 10: Definición de tiempo estimado de ejecución

Entrada: *entrada con n datos.*
Respuesta: *...*
1 *instrucción$_1$*
$\vdots$ $\vdots$
m *instrucción$_m$*

Para toda $i \in \{1, ..., m\}$, nos referiremos como **tiempo estimado de ejecución** *de la instrucción$_i$, a la cantidad $T_i(n)$ de operaciones realizadas en la instrucción$_i$ calculada en función del tamaño n de la entrada.*

Definición 2.4.2. *Definimos el* **tiempo de ejecución** *de un algoritmo con $m \geq 1$ instrucciones y con una entrada de tamaño $n \geq 1$, para ser la cantidad $T(n)$ dada por*

$$T(n) = T_1(n) + T_2(n) + \cdots + T_m(n),$$

donde $T_i(n)$, con $i \in \{1, ..., m\}$, es el tiempo estimado de ejecución de la instrucción$_i$ en el algoritmo.

Nótese que hemos nombrado como tiempo *estimado* de ejecución, a la cantidad de operaciones realizadas por una sola instrucción en un algoritmo. Esto se debe en parte a la falta de una definición formal para el concepto de operación, pero además se expresa así ya que esta cantidad puede cambiar dependiendo de los criterios que se consideren para determinarla. Algunos de los criterios más comunes indicados en la literatura son, por ejemplo, calcularla respecto al **peor caso** como con el **Algoritmo 8**, o bien, obtenerla respecto a un **caso promedio** o un **mejor caso**.

En esta tesis recurriremos únicamente al criterio del *peor caso*, además de utilizar el **Criterio del Mayor Término 2.4.1**, que presentaremos en el siguiente apartado. En general no trataremos con los *casos promedio*, y sólo hablaremos del *mejor caso* si esto es de relevancia práctica. Más aún, por estas razones y ya que la cantidad de operaciones es una herramienta para comparar algoritmos semejantes, es común que no nos interese siempre el valor exacto de esta cantidad, sino únicamente una cota superior para ella. Debido a esto, ahora daremos una definición formal que después usaremos para calcular dichas cotas como funciones del tamaño de la entrada de un algoritmo.

En las siguientes definiciones denotaremos por $\mathbb{R}^+$ al conjunto de los números reales positivos, es decir, no negativos y distintos de cero, y denotaremos por $\mathbb{N}$ al conjunto de los números naturales comenzando en 1.

Definición 2.4.3. *Sean f y g cualesquiera funciones $f, g : \mathbb{N} \to \mathbb{R}^+$. Diremos que f es de* **orden** *g, escrito $f(n) = O(g(n))$, si existen dos enteros positivos c y n_0 tales que se tenga*

$$f(n) \leq cg(n) \text{ para toda } n \geq n_0$$

De ser el caso diremos además que $g(n)$ es una **cota superior asintótica** *para $f(n)$.*

La anterior definición es conocida coloquialmente como **notación de la O grande**, y poner $f(n) = O(g(n))$ indica, intuitivamente, que la función f es menor o igual a la función g salvo por un factor constante, y excepto posiblemente para un número finito de valores. Ahora veremos dos propiedades que se siguen directamente de la definición. Más adelante estas nos permitirán simplificar y analizar las expresiones para el orden de una función dada.

Observación 2.4.1. *Sean f y g cualesquiera funciones $f, g : \mathbb{N} \to \mathbb{R}^+$. Si $f(n) = O(k\, g(n))$ para algún entero positivo k, entonces $f(n) = O(g(n))$.*

Demostración. Sabemos que existen enteros positivos n_0 y c tales que $f(n) \leq c[k\, g(n)]$ para todo $n \geq n_0$. Pero de aquí, para n_0 y $c' = ck$ se tiene $f(n) \leq c'g(n)$ para toda $n \geq n_0$. Así $f = O(g(n))$. ■

Observación 2.4.2. *Sean f, g y h cualesquiera funciones $f, g, h : \mathbb{N} \to \mathbb{R}^+$. Si $f(n) = O(g(n))$ y a la vez $g(n) = O(h(n))$, entonces $f(n) = O(h(n))$.*

Demostración. Sean c, c' y n_0, n_0' enteros positivos tales que $f(n) \leq cg(n)$ para toda $n \geq n_0$ y además $g(n) \leq c'h(n)$ para toda $n \geq n_0'$. Supóngase primero que $n_0' \geq n_0$. Luego, si $n \geq n_0' \geq n_0$, entonces $f(n) \leq cg(n) \leq cc'h(n)$, es decir, $f(n) \leq c''h(n)$ con $c'' = cc'$, para toda $n \geq n_0'$, por lo que en este caso $f(n) = O(h(n))$. Pero igualmente, de tener $n \geq n_0 \geq n_0'$ se sigue $f(n) \leq c''h(n)$ con $c'' = cc'$, para toda $n \geq n_0$, por lo que en este caso también $f(n) = O(h(n))$ y se verifica el resultado. ■

Concluimos este apartado definiendo la complejidad computacional de un algoritmo, que es la cota que nos permitirá compararlos sin tener que conocer con exactitud sus tiempo de ejecución.

Definición 2.4.4. *Sea g cualquier función $g : \mathbb{N} \to \mathbb{R}^+$. Decimos que un algoritmo con una entrada de tamaño $n \geq 1$, tiene una* **complejidad computacional en tiempo** *de $g(n)$, o simplemente que tiene* **complejidad** *$g(n)$, si su tiempo de ejecución T es de orden g, es decir, $T(n) = O(g(n))$.*

Un Criterio Práctico para determinar la Complejidad de un Algoritmo.

Así, al ser la complejidad de un algoritmo una cota superior para su tiempo de ejecución, podremos usar esta como referencia para estimar la máxima cantidad de operaciones que este puede llegar a hacer. Sin embargo, aún requerimos determinar una forma para calcular en la práctica a dicha complejidad computacional. Para ello utilizaremos el siguiente resultado.

Proposición 2.4.1. ***(Criterio del Mayor Término)***
Supóngase que el tiempo de ejecución $T(n)$ de un algoritmo con $m \geq 1$ instrucciones y una entrada de tamaño $n \geq 1$, se puede escribir como

$$T(n) = a_1T_1(n) + a_2T_2(n) + \cdots + a_mT_m(n),$$

para cualesquiera enteros positivos $a_1, ..., a_n$, y donde para toda $i \in \{1, ..., m\}$, el valor $a_iT_i(n)$ es el tiempo estimado de ejecución de la instrucción$_i$ del algoritmo. Si existe un entero positivo n_0 tal que

$$T_1(n) \geq T_2(n) \geq \cdots \geq T_m(n) \text{ para toda } n \geq n_0,$$

entonces dicho algoritmo tiene complejidad $T_1(n)$, es decir, $T(n) = O(T_1(n))$.

Demostración. Como tenemos $T(n) = a_1T_1(n) + a_2T_2(n) + \cdots + a_mT_m(n)$ para toda $n \geq 1$, y al mismo tiempo $T_1(n) \geq T_2(n) \geq \cdots \geq T_m(n)$ para toda $n \geq n_0$, podemos poner

$$T(n) = a_1T_1(n) + a_2T_2(n) + \cdots + a_mT_m(n) \leq a_1T_1(n) + a_2T_1(n) + \cdots + a_mT_1(n) \text{ para } n \geq n_0,$$

es decir, $T(n) \leq (a_1 + a_2 + \cdots + a_m)T_1(n)$ para toda $n \geq n_0$, por lo que $T(n) = O(k\, T_1(n))$ con $k = a_1 + a_2 + \cdots + a_m$. Así, por la **Observación 2.4.1** tenemos $T(n) = O(T_1(n))$, como requerido. ■

El resultado anterior implica que, por ejemplo, si el tiempo de ejecución de un algoritmo se puede escribir como $T(n) = n! + n^4 + 3n^2 + 5n + 1$, podremos deducir que dicho algoritmo tiene complejidad $n!$, es decir, $T(n) = O(n!)$, ya que $n! \geq n^4 \geq n^2 \geq n \geq 1$ para toda $n \geq n_0 = 7$. Más aún, en Teoría de la Computación ya se conoce y utiliza regularmente, una jerarquía de funciones que facilita deducir la complejidad de un algoritmo, sin tener que buscar en cada caso un entero positivo n_0 que nos permita usar el **Criterio del Mayor Término 2.4.1**. Expresados en notación de la *O grande*, y para un entero $k \geq 4$, algunos de los elementos en esta jerarquía son,

$$O(n!) > O(n^k) > \cdots > O(n^3) > O(n^2) > O(n\log_2(n)) > O(n) > O(\log_2(n)) > O(1)$$

De esta forma, si los términos $T_i(n)$ en el tiempo de ejecución $T(n)$ de un algoritmo (o bien sus múltiplos positivos), pertenecen todos a esta jerarquía , siempre podremos concluir que dicho algoritmo tiene complejidad igual al mayor de estos términos.

En particular, dependiendo del tiempo de ejecución $T(n)$ de un algoritmo y la complejidad computacional que se le pueda asociar, podremos decir que este **es**, o que este **corre en tiempo**:

−**constante**, si $T(n) = O(1)$	−**cúbico**, si $T(n) = O(n^3)$
−**lineal**, si $T(n) = O(n)$	−**polinomial**, si $T(n) = O(n^k)$ con $k \geq 1$
−**cuadrático**, si $T(n) = O(n^2)$	−**factorial**, si $T(n) = O(n!)$

Ahora bien, notemos que es posible argumentar que el Criterio del Mayor Término puede llegar a dar resultados que exceden la complejidad que realmente tendría un algoritmo, cuando este se combina con el criterio del *peor caso*. Por ejemplo, supongamos que un algoritmo en realidad tiene un tiempo de ejecución $T(n) = O(n \log_2(n))$, pero que al considerar un *caso extremo*, digamos muy poco común entre las *instancias* que nos interesa analizar con dicho algoritmo, se obtenga para $T(n)$ un mayor término igual a n^2. De esta forma, como a la vez podemos poner $n \log_2(n) = O(n^2)$, tanto por la **Observación 2.4.2** como por el Criterio del Mayor Término deduciríamos que $T(n) = O(n^2)$.

Sin embargo, para los propósitos de esta tesis, ambos criterios son válidos e incluso necesarios. Esto se debe a que todos los algoritmos a estudiar, dependen en gran medida del *Problema del Isomorfismo de Gráficas*, para el que a la fecha no se conoce ninguna solución algorítmica que corra en tiempo polinomial, o menor, sobre todas las instancias posibles.

Algoritmos de Isomorfismos entre Gráficas Arbitrarias y entre Árboles.

Existen dos problemas computacionales relacionados con la definición de un isomorfismo entre gráficas. El primero de ellos es decidir si dos gráficas arbitrarias son isomorfas o no, conocido como el **Problema del Isomorfismo de Gráficas**. El segundo es, sabiendo de antemano que dos gráficas son isomorfas, determinar todos los posibles isomorfismos entre ellas.

De forma teórica, ambos problemas tienen una solución sin importar que par de gráficas se estén analizando, ya que resolverlos sólo dependería de revisar exhaustivamente todas las biyecciones entre sus vértices, y guardar aquellas que preserven adyacencia. Luego, si no existen biyecciones con esta propiedad, podemos concluir que las gráficas no son isomorfas, y de cualquier otra forma, ya habríamos determinado todos los isomorfismos entre ellas.

Más aún, es posible intuir que dicha revisión exhaustiva constituiría un algoritmo con complejidad de por lo menos $O(n!n^2)$ para dos gráficas G y H de orden n, asumiendo que se cuenta con suficiente *espacio* o *capacidad de memoria* para almacenar todas las biyecciones de los vértices de G en los de H, lo que a su vez es semejante a obtener las $n!$ permutaciones de los n vértices de H.

Por otro lado, revisar si una de estas biyecciones φ preserva adyacencia, conlleva evaluar si todo par no ordenado de vértices distintos u y v de G, tienen imágenes $\varphi(u)$ y $\varphi(v)$ adyacentes en H cuando uv es una arista en G, e imágenes no adyacentes si u y v no son vecinos en G. Así, como la cantidad de pares que se pueden hacer con n elementos es $n(n-1)/2$, haríamos aproximadamente n^2 revisiones (o un múltiplo constante de esto), por cada biyección, de donde se sigue el $n!n^2$. Sin embargo, asumir una capacidad de memoria para las $n!$ biyecciones no es real en la práctica. Aún así, los algoritmos más **eficientes**, esto es, de menor complejidad con los que se cuenta a la fecha para resolver este problema, funcionan de forma semejante a la revisión exhaustiva de antes, pero haciendo modificaciones *inteligentes* a esta para optimizar su tiempo de ejecución.

Por ejemplo, en vez de obtener todas las biyecciones posibles entre los vértices de G y H, para después evaluarlas, estos algoritmos las construyen *emparejando* vértices poco a poco por un procedimiento llamado *backtracking*, semejante al algoritmo de *búsqueda por profundidad* en un árbol. En este se comienza proponiendo una imagen en los vértices de H para un vértice arbitrario de G, y luego para uno de sus vecinos, continuando de esta forma mientras sea posible preservar la adyacencia. Luego, si en este procedimiento no existen vértices en H que constituyan una imagen apropiada para algún vértice v de G, se descartará la imagen del vértice u de G que fue emparejado antes de v, y se buscará una nueva imagen en H, primero para u y luego para v, repitiendo esto hasta poder determinar la biyección completa, o bien hasta llegar a un "callejón sin salida".

Lo antes descrito es la esencia del algoritmo **VF2**, que es uno de los algoritmos más conocidos para evaluar y obtener isomorfismos entre cualesquiera dos gráficas de orden n [20–23]. En este se realizan distintos tipos de revisiones durante el backtracking para poder descartar imágenes candidatas de cada vértice, incluso antes de tener que evaluarlas. Por ello recurrimos a las ya existentes implementaciones [24] de este algoritmo en lenguaje Python para poder realizar la detección computacional de las amoebas. No obstante, debemos señalar que aún con todas estas mejoras, este algoritmo tiene una complejidad de $O(n!n)$ respecto al peor caso, y de $O(n^2)$ para un mejor caso.

A pesar de esto, es posible obtener algoritmos más eficientes al restringir el Problema del Isomorfismo a ciertas subfamilias de gráficas. Tal es el caso del isomorfismo entre árboles de orden n, para el que hay un algoritmo ya implementado en lenguaje Python [25], basado en el algoritmo mostrado en [19]. Dicho algoritmo no tiene nombre (a diferencia del **VF2**), por lo que nos referiremos a su implementación como el algoritmo **IsoTree**. Estos funcionan determinando el o los *centros* de los árboles dados y comparando las distintas trayectorias que hay desde los centros hasta las hojas. Utilizaremos este algoritmo para identificar amoebas que al mismo tiempo son árboles, pero debemos agregar que dicha implementación tienen complejidad $O(n\log_2(n))$, según indica su documentación [25], por lo que haremos uso de esta complejidad para nuestros fines teóricos.

Capítulo 3

Amoebas: Definiciones, Propiedades Básicas y Ejemplos

Aquí presentaremos el objeto fundamental de estudio de esta tesis, es decir, a la familia de las amoebas. Comenzaremos estableciendo una definición para estas gráficas por medio de la noción de los *reemplazos admisibles de aristas*. Aunque dicha representación motiva el análisis de las amoebas, sustituiremos esta enseguida por un marco teórico basado en grupos de permutaciones, que simplificará modelar las propiedades con las que cuenta una amoeba y sentará las bases para el resto de la tesis. Al final de este capítulo daremos una clasificación para gráficas en cuatro colecciones distintas, específicamente tres subfamilias de amoebas y una de gráficas que no cumplen ser amoebas, y concluiremos revisando ejemplos de cada uno de estos conjuntos. Antes de comenzar, debemos añadir que entre los resultados de este capítulo se exhiben algunas *Observaciones*, que a pesar de deducirse en su mayoría directamente de las definiciones, se incluyen aquí junto con sus pruebas con propósito de hacer explícita la construcción de las ideas abordadas en esta tesis.

3.1. Reemplazos Admisibles de Aristas y una Primera Definición de Amoebas

A pesar de que repetiremos explícitamente y en toda ocasión este mismo aspecto primordial de la teoría, debemos comenzar señalando que todas las gráficas a estudiar de aquí en adelante, serán consideradas como subgráficas generadoras de la gráfica completa K_n, con conjunto de vértices $V = \{v_1, ..., v_n\}$ para algún entero $n \geq 1$. De esta forma, tenemos una primera definición, que además jugará un papel de gran importancia durante toda la tesis.

Definición 3.1.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $v_r v_s$ una arista en G. Dada $v_k v_l \in E(\overline{G}) \cup \{v_r v_s\}$, se dice que $rs \to kl$ es un* **reemplazo admisible de aristas** *en G, si se tiene $G - v_r v_s + v_k v_l \simeq G$. En particular, si $v_k v_l = v_r v_s$ decimos que $rs \to kl$ es un* **reemplazo trivial**.

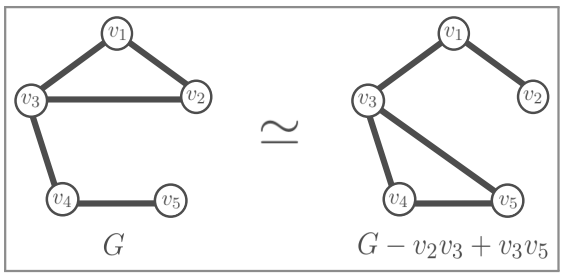

Figura 3.1: Ejemplo donde a una gráfica G se le aplica el reemplazo admisible de aristas $23 \to 35$, para obtener una nueva gráfica $G - v_2v_3 + v_3v_5$ que es en efecto isomorfa a G.

Ahora utilizaremos los reemplazos admisibles de aristas para relacionar dos subgráficas isomorfas de la gráfica completa. La siguiente definición resumirá el mecanismo principal que motiva la definición de las amoebas. Para comprenderla es necesario además recordar lo que se entiende por *copia encajada* de una gráfica G en una gráfica H (ver **Definición 2.1.14**).

Definición 3.1.2. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$, y sean G_1 y G_2 dos copias de G encajadas en la gráfica completa K_n también con $V(K_n) = \{v_1, ..., v_n\}$. Decimos que G_2 se obtiene desde G_1 por una* **sucesión de reemplazos admisibles de aristas**, *si existe una secuencia $\{H_1, H_2, ..., H_m\}$ de $m \geq 2$ copias de G encajadas en K_n, con $H_1 = G_1$ y $H_m = G_2$, tal que para cada $i \in \{1, ..., m-1\}$ hay un reemplazo admisible de aristas $r_i s_i \to k_i l_i$ de H_i, respecto al que se tiene $H_{i+1} = H_i - v_{r_i} v_{s_i} + v_{k_i} v_{l_i}$.*

Observación 3.1.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$, y sean G_1 y G_2 dos copias de G encajadas en la gráfica completa K_n también con $V(K_n) = \{v_1, ..., v_n\}$. Si es posible obtener G_2 a partir de G_1 por medio de una sucesión de reemplazos admisibles de aristas, entonces también es posible obtener G_1 desde G_2 por medio de una sucesión de reemplazos admisibles de aristas.*

Demostración. Supongamos que G_2 se obtiene de G_1 respecto a $m \geq 2$ copias $\{H_1, H_2, ..., H_{m-1}, H_m\}$ de G encajadas en K_n, con $H_1 = G_1$, $H_m = G_2$ y $H_{i+1} = H_i - v_{r_i} v_{s_i} + v_{k_i} v_{l_i}$. Inmediatamente vemos que G_1 se obtiene desde G_2 respecto a $\{H_m, H_{m-1}, ..., H_2, H_1\}$ con $H_i = H_{i+1} - v_{k_i} v_{l_i} + v_{r_i} v_{s_i}$. ■

La **Observación 3.1.1** nos será de utilidad después de que veamos el **Lema de la Composición 3.4.1**, que nos ayudará a expresar a las amoebas en términos de permutaciones. Por ahora daremos una primera versión para las definiciones de las gráficas que conforman la familia de las amoebas. Estas son de dos tipos, unas llamadas *amoebas locales* y otras *amoebas globales*. En cada caso indicamos aspectos importantes de la definición y damos ejemplos de estas.

Definición 3.1.3. ***(versión basada en sucesiones de reemplazos admisibles)***
Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea K_n la gráfica completa teniendo también $V(K_n) = \{v_1, ..., v_n\}$. Decimos que G es una **amoeba local**, *si dadas cualesquiera dos copias G_1 y G_2 de G encajadas en K_n, se puede obtener G_2 a partir de G_1 por una sucesión de reemplazos admisibles de aristas.*

Es importante notar que esta definición requiere que toda copia de G encajada en K_n, se pueda obtener a partir de cualquier otra copia de G en K_n, por medio de una sucesión de reemplazos admisibles de aristas. Es decir que esta definición no se verificará, por ejemplo, de cumplirse tal propiedad para un único par de copias y no para las demás.

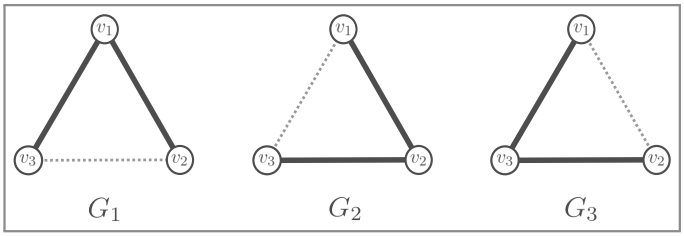

Figura 3.2: Se muestran las 3 copias de P_3 encajadas en K_3. De estas, G_2 se obtiene de G_1 por el reemplazo $13 \to 23$, G_3 de G_2 por $12 \to 13$ y G_1 de G_3 por $23 \to 13$. Luego, por la definición de sucesión de reemplazos admisibles de aristas, y usando la **Observación 3.1.1**, podemos concluir que P_3 es una amoeba local.

Definición 3.1.4. ***(versión basada en sucesiones de reemplazos admisibles)***
Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$. Decimos que G es una **amoeba global***, si existe un entero $T \geq 0$ tal que la gráfica $G \dot{\cup} tK_1$, esto es, la unión disjunta de G con t vértices aislados, es una amoeba local para todo $t \geq T$.*

Nuevamente hacemos hincapié en que la definición anterior requiere que la propiedad planteada se cumpla para todo entero $t \geq T$. Es decir que no es suficiente verificar esta propiedad para un único valor de t, por lo menos en principio.

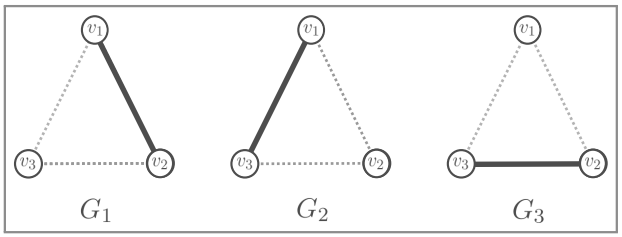

Figura 3.3: Se muestran las copias de $K_2 \cup tK_1$, para $t = 1$, encajadas en K_3. Podemos ver que cada una se obtiene de las otras por medio de un reemplazo admisible de aristas, por lo que $K_2 \cup K_1$ es una amoeba local. Pero como esto mismo sucede para $K_2 \cup tK_1$ con $t \geq 1$, entonces K_2 misma es una amoeba global.

Por último debemos notar que la gráfica G en las definiciones anteriores no puede ser vacía. Esto se debe a que todas dependen de los reemplazos admisibles, que a su vez requieren la existencia de por lo menos una arista en G. Así, buscando complementar estas definiciones, de aquí en adelante consideraremos que toda gráfica vacía es tanto una amoeba local, como una amoeba global.

3.2. Las σ-Representaciones de una Gráfica en K_n

Recuérdese que S_n es el grupo de permutaciones del conjunto $[n] = \{1, 2, ..., n\}$ (ver **Definición 2.2.9**), y además que todas las gráficas a tratar tienen vértices en $V = \{v_1, v_2, ..., v_n\}$ para $n \geq 1$. En esta sección se establecen conceptos que más adelante nos permitirán sustituir las definiciones de las amoebas, basadas en reemplazos admisibles de aristas, por definiciones basadas en permutaciones. Específicamente veremos que dada una gráfica arbitraria G, a toda permutación $\sigma \in S_n$ se le puede asociar una copia G_σ de G encajada en K_n, y viceversa.

Definición 3.2.1. *Sean G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y K_n la gráfica completa también con $V(K_n) = \{v_1, ..., v_n\}$. Sea además σ una permutación en S_n. Se define la* **σ-representación** *de G en K_n, como la subgráfica generadora G_σ de K_n con $E(G_\sigma) = \{v_{\sigma^{-1}(i)}v_{\sigma^{-1}(j)} \mid v_iv_j \in E(G)\}$.*

Es importante notar que toda σ-representación de una gráfica G en K_n está bien definida ya que $v_{\sigma^{-1}(i)}$ y $v_{\sigma^{-1}(j)}$ son distintos elementos de $V = \{v_1, ..., v_n\}$, al ser los extremos de la arista $v_iv_j \in E(G)$ distintos entre sí. Ahora haremos explícito el hecho de que G_σ es isomorfa a G.

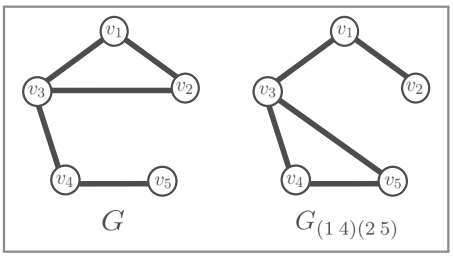

Figura 3.4: Se muestra la σ-representación G_σ de G en K_n para $\sigma = (14)(25) \in S_n$, expresando σ en notación de ciclos como haremos en toda la tesis. Se puede verificar que $v_iv_j \in E(G)$ si y sólo si $v_{\sigma^{-1}(i)}v_{\sigma^{-1}(j)} \in E(G_\sigma)$.

Observación 3.2.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$. Para toda $\sigma \in S_n$ se tiene $G_\sigma \simeq G$ y $E(G_\sigma) = \{v_iv_j \mid v_{\sigma(i)}v_{\sigma(j)} \in E(G)\}$. Además $G = G_{I_n}$ para la identidad $I_n \in S_n$.*

Demostración. Dada $v_iv_j \in E(G)$, como $i = \sigma(i')$ y $j = \sigma(j')$ para únicos $i', j' \in [n]$, por la definición $v_{\sigma(i')}v_{\sigma(j')} \in E(G)$ si y sólo si $v_{i'}v_{j'}$ en $E(G_\sigma)$. De aquí que la biyección $\widehat{\sigma} : V(G_\sigma) \to V(G)$ dada por $\widehat{\sigma}(v_i) = v_{\sigma(i)}$ para todo $v_i \in V(G_\sigma)$ sea un isomorfismo de G_σ en G. Es inmediato que $G = G_{I_n}$. ■

Definición 3.2.2. *Sean G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea σ una permutación en S_n. Diremos que la biyección $\widehat{\sigma} : V(G_\sigma) \to V(G)$ dada por $\widehat{\sigma}(v_i) = v_{\sigma(i)}$ para todo $v_i \in V(G_\sigma)$, es el isomorfismo, de G_σ en G,* **derivado** *de σ.*

A continuación veremos cómo toda copia G' de G encajada en K_n, se puede escribir también como una σ-representación de G para alguna $\sigma \in S_n$.

Observación 3.2.2. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea K_n la gráfica completa también con $V(K_n) = \{v_1, ..., v_n\}$. Para toda copia G' de G encajada en K_n existe por lo menos una permutación $\sigma \in S_n$ tal que $G' = G_\sigma$.*

Demostración. Por hipótesis existe un isomorfismo, digamos $\widehat{\sigma}$, de G' en G. Luego, podemos definir una permutación $\sigma \in S_n$ tal que $\sigma(i) = j$ si y sólo si $\widehat{\sigma}(v_i) = v_j$, de donde $\widehat{\sigma}(v_i) = v_{\sigma(i)}$. Así, tendremos $v_iv_j \in E(G')$, si y sólo si $\widehat{\sigma}(v_i)\widehat{\sigma}(v_j) \in E(G)$, es decir, si y sólo si $v_{\sigma(i)}v_{\sigma(j)} \in E(G)$. Luego, por la **Observación 3.2.1**, se sigue que $E(G') = \{v_iv_j \mid v_{\sigma(i)}v_{\sigma(j)} \in E(G)\} = E(G_\sigma)$, y como estas gráficas tienen también el mismo conjunto de vértices, podemos concluir que $G' = G_\sigma$. ■

Proposición 3.2.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea K_n la gráfica completa también con $V(K_n) = \{v_1, ..., v_n\}$. Toda copia G' de G encajada en K_n es una σ-representación de G en K_n para alguna $\sigma \in S_n$, y viceversa.*

Demostración. Por la **Observación 3.2.2**, toda copia G' de G encajada en K_n es al mismo tiempo una σ-representación de G. El sentido contrario se sigue ya que por definición, toda σ-representación de G es una subgráfica de K_n, y por la **Observación 3.2.1** estas son isomorfas a G. ■

La **Proposición 3.2.1** nos permitirá estudiar a las amoebas en términos de las permutaciones $\sigma \in S_n$ asociadas a cada σ-representación de G. Esta información extra cobrará relevancia cuando relacionemos dichas permutaciones con los reemplazos admisibles de aristas en G y veamos el **Lema de la Composición 3.4.1**. Por el momento concluimos esta sección utilizando estas permutaciones para calcular la cantidad de σ-representaciones distintas entre sí que tiene G en K_n.

Definición 3.2.3. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$. Denotaremos por $\overset{G}{\sim}$, a la relación sobre S_n asociada a G, tal que para cualesquiera $\sigma, \rho \in S_n$ se tiene $\sigma \overset{G}{\sim} \rho$ si y sólo si $G_\sigma = G_\rho$.*

Observación 3.2.3. *La relación $\overset{G}{\sim}$ de una gráfica G, es una relación de equivalencia sobre S_n.*

Demostración. Debemos mostrar que $\overset{G}{\sim}$ es reflexiva, simétrica y transitiva. Lo primero se sigue ya que $G_\sigma = G_\sigma$ para toda $\sigma \in S_n$. Por otro lado, la simetría se verifica ya que si $\sigma \overset{G}{\sim} \rho$, entonces $G_\sigma = G_\rho$, de donde inmediatamente $\rho \overset{G}{\sim} \sigma$. Finalmente, como tener $\sigma \overset{G}{\sim} \rho$ y $\rho \overset{G}{\sim} \pi$ implica que $G_\sigma = G_\rho = G_\pi$, en consecuencia $\sigma \overset{G}{\sim} \pi$, es decir que $\overset{G}{\sim}$ es también transitiva, como requerido. ■

Proposición 3.2.2. *Respecto a cada gráfica G de orden n con $V(G) = \{v_1, ..., v_n\}$, existen $n!/aut(G)$ $\overset{G}{\sim}$-clases de equivalencia sobre S_n*

Demostración. Considérese una copia G' de G encajada en K_n y denótese por $Iso(G', G)$ el conjunto de todos los isomorfismos de G' en G. Por definición hay por lo menos un isomorfismo φ en $Iso(G', G)$. Con base en las **Proposiciones 2.1.1** y **2.1.2** definamos $\Phi : Aut(G') \to Iso(G', G)$ como $\Phi(\alpha) = \varphi\alpha$ para todo $\alpha \in Aut(G')$ y veamos que es una biyección. De tener $\Phi(\alpha) = \Phi(\beta)$ se sigue $\varphi\alpha = \varphi\beta$ y entonces $\alpha = \beta$, por lo que Φ es inyectiva. Por otro lado, respecto a todo $\theta \in Iso(G', G)$ y con base en las proposiciones dadas, se tiene $\varphi^{-1}\theta \in Aut(G')$, y además $\Phi(\varphi^{-1}\theta) = \varphi\varphi^{-1}\theta = \theta$, por lo que Φ es sobreyectiva y así $|Iso(G', G)| = aut(G')$. Más aún, como $G' \simeq G$ por la **Observación 2.1.1** tenemos $|Iso(G', G)| = aut(G)$. Ahora bien, por la **Observación 3.2.2** y su prueba sabemos que para todo $\widehat{\sigma} \in Iso(G', G)$, se tiene $G' = G_\sigma$ respecto a $\sigma \in S_n$ definida como $\sigma(i) = j$ si y sólo si $\widehat{\sigma}(v_i) = v_j$. Por otro lado, si σ cumple $G' = G_\sigma$, el isomorfismo $\widehat{\sigma}$ derivado de σ estará en $Iso(G', G)$. Así, hay exactamente $m = aut(G)$ permutaciones distintas $\sigma_1, \sigma_2, ..., \sigma_m$ en S_n tales que $G' = G_{\sigma_1} = G_{\sigma_2} = \cdots = G_{\sigma_m}$ y por ende el conjunto $\{\sigma_1, \sigma_2, ..., \sigma_m\}$ es una $\overset{G}{\sim}$-clase de equivalencia que ya integra a todos sus elementos. Luego, ya que G' fue arbitraria, por la **Proposición 3.2.1** esto mismo pasará para cualquier $\overset{G}{\sim}$-clase de equivalencia, y como estas forman una partición de S_n, tenemos $[k][aut(G)] = n!$ para la cantidad k de $\overset{G}{\sim}$-clases, por lo que finalmente $k = n!/aut(G)$. ■

Corolario 3.2.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea K_n la gráfica completa también con $V(K_n) = \{v_1, ..., v_n\}$. Existen $n!/aut(G)$ σ-representaciones de G en K_n distintas entre sí.*

Demostración. El número de σ-representaciones distintas entre sí que tiene G en K_n, es igual a la cantidad $n!/aut(G)$ de $\overset{G}{\sim}$-clases en las que se puede partir S_n, ya que por definición todas las permutaciones en una misma $\overset{G}{\sim}$-clase tienen asociadas σ-representaciones idénticas para G. ■

3.3. Etiquetamientos y las Permutaciones Asociadas a Reemplazos Admisibles

Primero revisaremos brevemente un método para visualizar la σ-representación de una gráfica G dada cualquier $\sigma \in S_n$. Posteriormente estudiamos aquellas σ-representaciones de G que al mismo tiempo se pueden obtener por medio de un reemplazo admisible de aristas de G misma.

Definición 3.3.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea K_n la gráfica completa también con $V(K_n) = \{v_1, ..., v_n\}$. Dada una σ-representación G_σ de G en K_n, llamaremos σ-**etiquetamiento** de G_σ, a la biyección $\lambda_\sigma : V(G_\sigma) \to \{1, 2, ..., n\}$ dada por $\lambda_\sigma(v_i) = \sigma(i)$. Adicionalmente, nos referiremos a la imagen $\lambda_\sigma(v_i)$ de v_i bajo λ_σ, como la **etiqueta** del vértice v_i en G_σ.*

Observación 3.3.1. *Sean G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$. Sea además $\sigma \in S_n$. Para cualesquiera vértices v_i y v_j de G_σ, se tiene $v_iv_j \in E(G_\sigma)$, si y sólo si, las etiquetas $\lambda_\sigma(v_i)$ de v_i y $\lambda_\sigma(v_j)$ de v_j en G_σ, son iguales a los índices de vértices adyacentes en G.*

Demostración. De la **Observación 3.2.1** tenemos que $v_iv_j \in E(G_\sigma)$ si y sólo si $v_{\sigma(i)}v_{\sigma(j)} \in E(G)$. Pero por definición $\lambda_\sigma(v_i) = \sigma(i)$ y $\lambda_\sigma(v_j) = \sigma(j)$, de donde se sigue lo requerido. ■

De la observación anterior se sigue inmediatamente un método general, ejemplificado en la siguiente figura, para poder visualizar toda G_σ sin importar que $\sigma \in S_n$ se escoja.

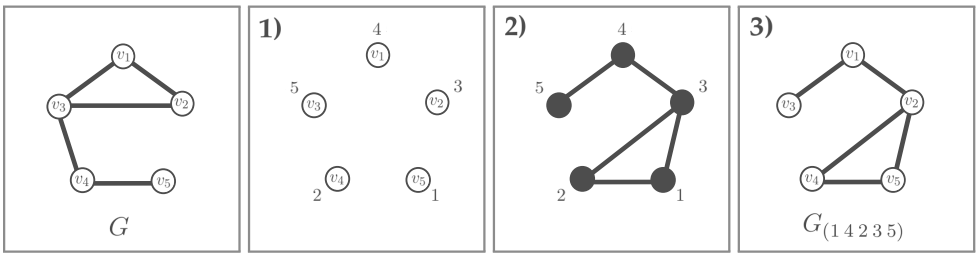

Figura 3.5: Considérense G y $\sigma = (1\ 4\ 2\ 3\ 5)$. Para visualizar G_σ seguimos 3 pasos: **1)** se asigna la etiqueta $\lambda_\sigma(v_i) = \sigma(i)$ a cada vértice v_i, **2)** redibujamos las aristas de G basándonos ahora no en los v_i, sino en las etiquetas $\lambda_\sigma(v_i)$ como si estas fueran los índices de vértices de G, y **3)** borramos todas las etiquetas.

Ahora bien, aunque que es posible recuperar toda σ-representación de una gráfica G usando el método en la figura anterior, no siempre será cierto que G_σ corresponda a un reemplazo admisible de aristas en G. Sin embargo, toda $\sigma \in S_n$ cuya G_σ sí cumpla esto será de gran utilidad para formar un subgrupo de S_n que nos facilitará decir si G es, o no, una amoeba, y si esta es local o global.

Definición 3.3.2. *Sean G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y K_n la gráfica completa también con $V(K_n) = \{v_1, ..., v_n\}$. Sea además $rs \to kl$ un reemplazo admisible de aristas en G. De aquí en adelante denotaremos por $S_G(rs \to kl)$ al conjunto $S_G(rs \to kl) = \{\sigma \in S_n \mid G_\sigma = G - v_rv_s + v_kv_l\}$.*

Nótese que de la definición se sigue $S_G(rs \to kl) \cap S_G(r's' \to k'l') = \varnothing$, para todo par de reemplazos admisibles distintos y no triviales $rs \to kl$ y $r's' \to k'l'$ de aristas de una gráfica G.

Veamos ahora un ejemplo de permutaciones que están, y otras que no están, en un conjunto $S_G(rs \to kl)$ de una gráfica G respecto algún reemplazo admisible de aristas $rs \to kl$ en G.

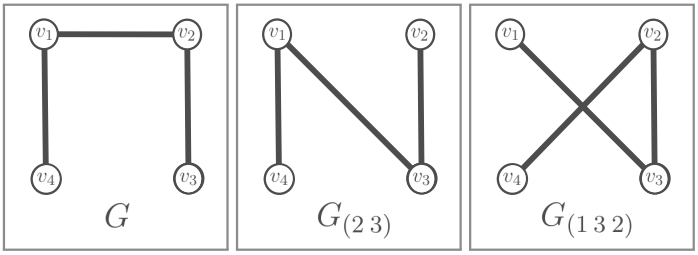

Figura 3.6: Las gráficas $G_{(2\,3)}$ y $G_{(1\,3\,2)}$ se puede recuperar con el método de la **Fig. 3.5**. Más aún, $G_{(2\,3)}$ se puede obtener al aplicar el reemplazo $12 \to 13$ a G, por lo que $(2\,3) \in S_G(12 \to 13)$. Pero $G_{(1\,3\,2)}$ no se puede obtener de G por un solo reemplazo admisible de aristas, ya que necesitaríamos aplicar los dos reemplazos $12 \to 13$ y $14 \to 24$ a G para obtenerla, i.e., no existe reemplazo $rs \to kl$ de G tal que $(1\,3\,2) \in S_G(rs \to kl)$.

Observación 3.3.2. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $rs \to kl$ un reemplazo admisible de G. Dada $\sigma \in S_G(rs \to kl)$, para toda $\rho \in S_n$, se tiene $\rho \in S_G(rs \to kl)$ si y sólo si $\rho \overset{G}{\sim} \sigma$.*

Demostración. Por hipótesis se tiene $G_\sigma = G - v_rv_s + v_kv_l$. Si además $\rho \in S_G(rs \to kl)$, entonces $G_\rho = G - v_rv_s + v_kv_l = G_\sigma$, por lo que $\rho \overset{G}{\sim} \sigma$. Por otro lado, de tener $\rho \overset{G}{\sim} \sigma$, se sigue inmediatamente que $G_\rho = G_\sigma = G - v_rv_s + v_kv_l$ y en consecuencia $\rho \in S_G(rs \to kl)$. ■

Corolario 3.3.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$. Para todo reemplazo admisible de aristas $rs \to kl$ de G se tiene $|S_G(rs \to kl)| = aut(G)$.*

Demostración. Se sigue de la **Observación 3.3.2** y la prueba de la **Proposición 3.2.2**. ■

Por último incluimos las siguientes dos observaciones, que nos serán de ayuda junto con el **Lema de la Composición 3.4.1**, para desarrollar la prueba del **Teorema 3.4.1**.

Observación 3.3.3. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $\sigma \in S_n$. Sea además G_σ la σ-representación de G en K_n también con $V(K_n) = \{v_1, ..., v_n\}$. Si $rs \to kl$ es un reemplazo admisible de aristas de G_σ, entonces $\sigma(r)\sigma(s) \to \sigma(k)\sigma(l)$ es un reemplazo admisible de aristas de G.*

Demostración. De la **Observación 3.2.1** sabemos que $v_iv_j \in E(G_\sigma)$ si y sólo si $v_{\sigma(i)}v_{\sigma(j)} \in E(G)$. Así, tendremos $v_iv_j \in [E(G_\sigma) \setminus \{v_rv_s\}] \cup \{v_kv_l\}$ si y sólo si $v_{\sigma(i)}v_{\sigma(j)} \in [E(G) \setminus \{v_{\sigma(r)}v_{\sigma(s)}\}] \cup \{v_{\sigma(k)}v_{\sigma(l)}\}$, que es $v_iv_j \in E(G_\sigma - v_rv_s + v_kv_l)$ si y sólo si $v_{\sigma(i)}v_{\sigma(j)} \in E(G - v_{\sigma(r)}v_{\sigma(s)} + v_{\sigma(k)}v_{\sigma(l)})$, y como por hipótesis $G_\sigma - v_rv_s + v_kv_l \simeq G_\sigma$, entonces $G - v_{\sigma(r)}v_{\sigma(s)} + v_{\sigma(k)}v_{\sigma(l)} \simeq G_\sigma \simeq G$. ■

Observación 3.3.4. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sean $\sigma, \rho \in S_n$. Sea además G_σ la σ-representación de G en K_n también con $V(K_n) = \{v_1, ..., v_n\}$. Si $rs \to kl$ es un reemplazo admisible de aristas de G_σ y $\rho \overset{G}{\sim} \sigma$, entonces $\rho(r)\rho(s) \to \rho(k)\rho(l)$ es un reemplazo admisible de G.* ■

3.4. El Lema de la Composición y su Consecuencia para las Amoebas

Ya sabemos que dada un gráfica G, toda copia G' de G encajada en K_n será representada por un conjunto de permutaciones en S_n, con propósito de sustituir las definiciones actuales de amoebas por unas basadas en dichas permutaciones. Sin embargo, las definiciones dadas para las amoebas también dependen de la existencia de sucesiones de reemplazos admisibles de aristas entre cualesquiera copias G_1 y G_2 de G. Luego, resulta natural preguntarse si existe una representación para estas sucesiones de reemplazos admisibles, basada también en permutaciones. El siguiente lema, formulado originalmente en [1], nos permitirá dar en esta misma sección una respuesta afirmativa a esta pregunta, y por lo tanto constituye una de las herramientas más importantes en la construcción y estudio de las amoebas. Recuérdese que dadas cualesquiera $\sigma, \rho \in S_n$, se escribe $\sigma\rho(k) = \sigma(\rho(k))$.

Lema 3.4.1. (Lema de la Composición)
Sean G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y K_n la gráfica completa teniendo igualmente $V(K_n) = \{v_1, ..., v_n\}$. Sean además $rs \to kl$ un reemplazo admisible de aristas de G, $\sigma \in S_G(rs \to kl)$ y $\rho \in S_n$ una permutación arbitraria. La $\sigma\rho$-representación $G_{\sigma\rho}$ de G en K_n se puede escribir como

$$G_{\sigma\rho} = G_\rho - v_{\rho^{-1}(r)}v_{\rho^{-1}(s)} + v_{\rho^{-1}(k)}v_{\rho^{-1}(l)}$$

Demostración. Ambas gráficas en la igualdad tienen el mismo conjunto de vértices, por lo que sólo hace falta mostrar que tienen las mismas aristas. Nótese primero que por la **Observación 3.2.1**,

$$E(G_{\sigma\rho}) = \{v_iv_j \mid v_{\sigma\rho(i)}v_{\sigma\rho(j)} \in E(G)\}$$

Por otro lado, siguiendo las definiciones y ya que por hipótesis $\sigma \in S_G(rs \to kl)$, podemos poner,

$$\begin{aligned}
E(G_\rho - v_{\rho^{-1}(r)}v_{\rho^{-1}(s)} + v_{\rho^{-1}(k)}v_{\rho^{-1}(l)}) &= [E(G_\rho) \setminus \{v_{\rho^{-1}(r)}v_{\rho^{-1}(s)}\}] \cup \{v_{\rho^{-1}(k)}v_{\rho^{-1}(l)}\} \\
&= [\{v_{\rho^{-1}(i)}v_{\rho^{-1}(j)} \mid v_iv_j \in E(G)\} \setminus \{v_{\rho^{-1}(r)}v_{\rho^{-1}(s)}\}] \cup \{v_{\rho^{-1}(k)}v_{\rho^{-1}(l)}\} \\
&= \{v_{\rho^{-1}(i)}v_{\rho^{-1}(j)} \mid v_iv_j \in [E(G) \setminus \{v_rv_s\}] \cup \{v_kv_l\}\} \\
&= \{v_{\rho^{-1}(i)}v_{\rho^{-1}(j)} \mid v_iv_j \in E(G - v_rv_s + v_kv_l)\} \\
&= \{v_{\rho^{-1}(i)}v_{\rho^{-1}(j)} \mid v_iv_j \in E(G_\sigma)\}
\end{aligned}$$

Pero si se tiene $v_{i'}v_{j'} \in \{v_{\rho^{-1}(i)}v_{\rho^{-1}(j)} \mid v_iv_j \in E(G_\sigma)\}$, habrá $i, j \in [n]$ con $v_iv_j \in E(G_\sigma)$ y con $i' = \rho^{-1}(i)$, $j' = \rho^{-1}(j)$, de donde $v_{\rho(i')}v_{\rho(j')} \in E(G_\sigma)$ y nuevamente por la **Observación 3.2.1** se sigue que $v_{\sigma\rho(i')}v_{\sigma\rho(j')} \in E(G)$. Así, finalmente tenemos

$$\begin{aligned}
v_{i'}v_{j'} \in E(G_\rho - v_{\rho^{-1}(r)}v_{\rho^{-1}(s)} + v_{\rho^{-1}(k)}v_{\rho^{-1}(l)}) &\text{ si y sólo si } v_{\rho(i')}v_{\rho(j')} \in E(G_\sigma), \\
&\text{que equivalentemente es } v_{\sigma\rho(i')}v_{\sigma\rho(j')} \in E(G), \\
&\text{e igualmente } v_{i'}v_{j'} \in E(G_{\sigma\rho}),
\end{aligned}$$

donde la última equivalencia se sigue por lo comentado al inicio de la prueba. ■

Más adelante veremos una figura que ilustra este lema al ejemplificar el **Teorema 3.4.1**.

Recuérdese que por la **Observación 3.2.1** se tiene $G = G_{I_n}$. Intuitivamente, el **Lema de la Composición 3.4.1** indica que dada G_ρ para ρ arbitraria, la gráfica $G_{\sigma\rho}$ con σ asociada a un reemplazo admisible $rs \to kl$ de G, es obtenida al aplicar a G_ρ un nuevo reemplazo admisible que "guarda la información" de $rs \to kl$. Luego, si al mismo tiempo G_ρ se obtiene de G por un reemplazo admisible de aristas, entonces $G_{\sigma\rho}$ se podrá obtener desde G por medio de una sucesión de dos reemplazos admisibles. Debido a esto presentamos el siguiente resultado, que a pesar de pertenecer implícitamente al trabajo en [1], se incluye aquí de forma explícita para motivar la definición de las amoebas basadas en permutaciones, al evidenciar el uso del Lema de la Composición para obtener ciertas σ-representaciones de G, desde G misma por medio de sucesiones de reemplazos admisibles.

Teorema 3.4.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea ρ una permutación en S_n. Sea además G_ρ la ρ-representación de G en K_n también con $V(K_n) = \{v_1, ..., v_n\}$. Es posible obtener G_ρ desde G por medio de una sucesión de reemplazos admisibles de aristas, si y sólo si, existen $m \geq 1$ permutaciones $\sigma_1, \sigma_2, ..., \sigma_m$ en S_n tales que $G_\rho = G_{\sigma_m \cdots \sigma_2 \sigma_1}$ y tales que se tiene $\sigma_i \in S_G(r_i s_i \to k_i l_i)$ para cada $i \in \{1, ..., m\}$, respecto a alguna colección de reemplazos admisibles $r_i s_i \to k_i l_i$ de G.*

Demostración. Supongamos que G_ρ se obtiene desde G por una sucesión de reemplazos admisibles de aristas, digamos respecto a $\{G, G_{\alpha_1}, ..., G_{\alpha_m}\}$ con $G_{\alpha_m} = G_\rho$ y donde $G_{\alpha_1} = G - v_{r_0}v_{s_0} + v_{k_0}v_{l_0}$, además de $G_{\alpha_{i+1}} = G_{\alpha_i} - v_{r_i}v_{s_i} + v_{k_i}v_{l_i}$ para $i \in \{1, ..., m-1\}$. Procedemos por inducción sobre $m \geq 1$. Si $m = 1$, entonces $G_{\alpha_1} = G_\rho$ satisface lo requerido ya que $\alpha_1 \in S_G(r_0 s_0 \to k_0 l_0)$. Más aún, si $m = 2$, es decir $G_\rho = G_{\alpha_2}$, podemos denotar $\sigma_1 = \alpha_1$ y por la **Observación 3.3.3** podemos escribir

$$G_{\sigma_1} = G - v_{r_0}v_{s_0} + v_{k_0}v_{l_0} = G_{\alpha_1},$$
$$G_{\sigma_2} = G - v_{\sigma_1(r_1)}v_{\sigma_1(s_1)} + v_{\sigma_1(k_1)}v_{\sigma_1(l_1)},$$

para alguna $\sigma_2 \in S_n$, ya que $r_1 s_1 \to k_1 l_1$ es un reemplazo admisible de $G_{\alpha_1} = G_{\sigma_1}$ y por ende $\sigma_1(r_1)\sigma_1(s_1) \to \sigma_1(k_1)\sigma_1(l_1)$ es un reemplazo admisible de G. Así, por el **Lema de la Composición 3.4.1** tendremos $G_{\sigma_2\sigma_1} = G_{\sigma_1} - v_{\sigma_1^{-1}\sigma_1(r_1)}v_{\sigma_1^{-1}\sigma_1(s_1)} + v_{\sigma_1^{-1}\sigma_1(k_1)}v_{\sigma_1^{-1}\sigma_1(l_1)} = G_{\sigma_1} - v_{r_1}v_{s_1} + v_{k_1}v_{l_1}$, que es $G_{\sigma_2\sigma_1} = G_{\alpha_1} - v_{r_1}v_{s_1} + v_{k_1}v_{l_1} = G_{\alpha_2}$ y entonces $G_\rho = G_{\alpha_2} = G_{\sigma_2\sigma_1}$.

Supongamos ahora que la proposición es cierta para $m \geq 2$ arbitraria y busquemos mostrarla para $m + 1$, es decir, con $G_\rho = G_{\alpha_{m+1}}$ y donde hay un reemplazo admisible de aristas $r_m s_m \to k_m l_m$ de G_{α_m} tal que $G_{\alpha_{m+1}} = G_{\alpha_m} - v_{r_m}v_{s_m} + v_{k_m}v_{l_m}$. En este caso tenemos $G_{\alpha_m} = G_{\sigma_m \cdots \sigma_1}$ con $\sigma_i \in S_G(r_i s_i \to k_i l_i)$. Luego, $\alpha_m \overset{G}{\sim} \sigma_m \cdots \sigma_1$ y por la **Observación 3.3.4**,

$$\sigma_m \cdots \sigma_1(r_m)\sigma_m \cdots \sigma_1(s_m) \to \sigma_m \cdots \sigma_1(k_m)\sigma_m \cdots \sigma_1(l_m)$$

es un reemplazo admisible de aristas de G. Con esto podemos definir

$$G_{\sigma_{m+1}} = G - v_{\sigma_m \cdots \sigma_1(r_m)}v_{\sigma_m \cdots \sigma_1(s_m)} + v_{\sigma_m \cdots \sigma_1(k_m)}v_{\sigma_m \cdots \sigma_1(l_m)}$$

para alguna $\sigma_{m+1} \in S_n$, y por el Lema de la Composición, denotando $\beta = \sigma_m \cdots \sigma_1$, obtenemos,

$$\begin{aligned} G_{\sigma_{m+1}\beta} &= G_\beta - v_{\beta^{-1}\beta(r_m)}v_{\beta^{-1}\beta(s_m)} + v_{\beta^{-1}\beta(k_m)}v_{\beta^{-1}\beta(l_m)} \\ &= G_{\sigma_m\cdots\sigma_1} - v_{r_m}v_{s_m} + v_{k_m}v_{l_m} \\ &= G_{\alpha_m} - v_{r_m}v_{s_m} + v_{k_m}v_{l_m} \\ &= G_{\alpha_{m+1}} \end{aligned}$$

Por lo tanto $G_\rho = G_{\alpha_{m+1}} = G_{\sigma_{m+1}\sigma_m\cdots\sigma_1}$ donde además σ_{m+1} está también asociada a un reemplazo admisible de aristas de G, con lo que se concluye la inducción.

Supongamos por otro lado que $G_\rho = G_{\sigma_m\sigma_{m-1}\cdots\sigma_2\sigma_1}$ con $\sigma_i \in S_G(r_is_i \to k_il_i)$ para $i \in \{1, ..., m\}$ y considérese la sucesión $\{G, G_{\sigma_1}, G_{\sigma_2\sigma_1}, ..., G_{\sigma_{m-1}\cdots\sigma_2\sigma_1}, G_{\sigma_m\sigma_{m-1}\cdots\sigma_2\sigma_1}\}$. Por definición y usando el **Lema de la Composición 3.4.1**, para estas gráficas inmediatamente se obtiene,

$$\begin{aligned} G_{\sigma_1} &= G - v_{r_1}v_{s_1} + v_{k_1}v_{l_1} \\ G_{\sigma_2\sigma_1} &= G_{\sigma_1} - v_{\sigma_1^{-1}(r_2)}v_{\sigma_1^{-1}(s_2)} + v_{\sigma_1^{-1}(k_2)}v_{\sigma_1^{-1}(l_2)} \\ &\vdots \\ G_{\sigma_i\sigma_{i-1}\cdots\sigma_1} &= G_{\sigma_{i-1}\cdots\sigma_1} - v_{[\sigma_{i-1}\cdots\sigma_1]^{-1}(r_i)}v_{[\sigma_{i-1}\cdots\sigma_1]^{-1}(s_i)} + v_{[\sigma_{i-1}\cdots\sigma_1]^{-1}(k_i)}v_{[\sigma_{i-1}\cdots\sigma_1]^{-1}(l_i)} \\ &\vdots \end{aligned}$$

$$G_\rho = G_{\sigma_m\sigma_{m-1}\cdots\sigma_1} = G_{\sigma_{m-1}\cdots\sigma_1} - v_{[\sigma_{m-1}\cdots\sigma_1]^{-1}(r_m)}v_{[\sigma_{m-1}\cdots\sigma_1]^{-1}(s_m)} + v_{[\sigma_{m-1}\cdots\sigma_1]^{-1}(k_m)}v_{[\sigma_{m-1}\cdots\sigma_1]^{-1}(l_m)},$$

es decir que entre gráficas sucesivas hay un reemplazo admisible y por lo tanto G_ρ se puede obtener desde G por una sucesión de reemplazos admisibles de aristas, como requerido. ■

Nótese que por la **Observación 3.1.1**, al obtener una copia G' desde G por medio de una sucesión de reemplazos admisibles de aristas, G también se podrá obtener desde G' de la misma forma. Así, componer permutaciones asociadas a reemplazos admisibles de G como indicado por el **Teorema 3.4.1**, nos permitirá conocer todas las copias de G encajadas en K_n, que se pueden *alcanzar* desde G, y a la vez, desde las que se puede *alcanzar* G, por una sucesión de reemplazos admisibles de aristas. Luego, todo par de copias G_1 y G_2 de G obtenidas de esta forma, se podrán *alcanzar* entre sí, dado que existe una sucesión que va desde G_1 hacia G, y otra que va desde G hacia G_2.

Por otro lado, se debe considerar que las definiciones de amoebas en términos de las sucesiones de reemplazos admisibles, requieren que toda copia G' de G encajada en K_n, sea *alcanzable* no sólo desde $G = G_{I_n}$, sino desde cualquier otra copia de G. Sin embargo, esto mismo facilita la formalización de las amoebas, ya que si una copia particular no se pudiera obtener desde G por una sucesión de reemplazos admisibles, o viceversa, entonces dicha G simplemente no sería una amoeba. Está será la clave para comprender las definiciones que daremos en la siguiente sección. A continuación incluimos una figura que ejemplifica lo establecido por el **Teorema 3.4.1** y su prueba.

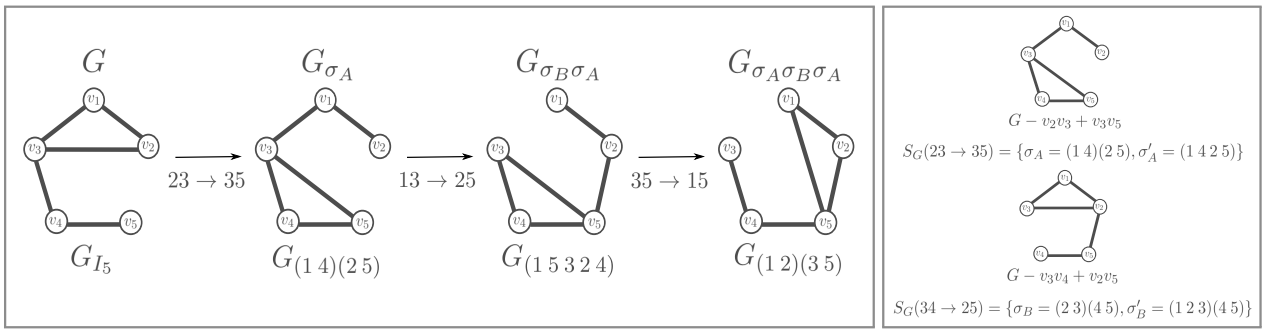

Figura 3.7: La gráfica G_ρ para $\rho = (1\,2)(3\,5)$, se obtiene desde G por una sucesión de reemplazos admisibles de aristas, específicamente: $23 \to 35$ es un reemplazo de G con el que se obtiene $G_{(14)(25)}$, después $G_{(15324)}$ se obtiene de $G_{(14)(25)}$ por $13 \to 25$, y por último $G_{(12)(35)}$ se obtiene de $G_{(15324)}$ por el reemplazo $35 \to 15$. Luego, el **Teorema 3.4.1** indica que podemos encontrar permutaciones $\sigma_A = (1\,4)(2\,5)$ y $\sigma_B = (2\,3)(4\,5)$, con $\sigma_A \in S_G(23 \to 35)$ y $\sigma_B \in S_G(34 \to 25)$, tales que se tenga $G_\rho = G_{\sigma_A\sigma_B\sigma_A}$. Debe notarse que en este caso se tiene además $\rho = \sigma_A\sigma_B\sigma_A$ debido a la elección particular de σ_A y σ_B, más no debido al **Teorema 3.4.1**. Este último hecho cobrará importancia después de haber visto la definición del grupo S_G en la siguiente sección.

3.5. Reemplazos Inversos, el Grupo S_G y Amoebas en Términos de Permutaciones

Antes de poder hablar de las amoebas en términos de permutaciones, y buscando complementar las ideas vistas hasta ahora, mostraremos como es que la colección de todas las permutaciones asociadas a reemplazos admisibles de una gráfica G, es cerrada bajo inversos. Esto nos facilitará introducir la definición del grupo de permutaciones S_G de G, del que a su vez dependen las amoebas. Debemos agregar que aunque el siguiente resultado se sigue directamente del trabajo en [1], se incluye aquí explícitamente junto con la **Definición 3.5.1**, que es es contribución original de esta tesis, para motivar las definiciones de las amoebas basadas en grupos de permutaciones.

Proposición 3.5.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $rs \to kl$ un reemplazo admisible de aristas en G. Sea además $\sigma \in S_G(rs \to kl)$. El reemplazo $\sigma(k)\sigma(l) \to \sigma(r)\sigma(s)$ es un reemplazo admisible de aristas de G y más aún $\sigma^{-1} \in S_G(\sigma(k)\sigma(l) \to \sigma(r)\sigma(s))$.*

Demostración. De la **Observación 3.2.1** tenemos que $v_iv_j \in E(G_\sigma)$ si y sólo si $v_{\sigma(i)}v_{\sigma(j)} \in E(G)$. Así, al tener $v_kv_l \in E(G_\sigma)$, se sigue que $v_{\sigma(k)}v_{\sigma(l)} \in E(G)$, e igualmente ya que $v_rv_s \notin E(G_\sigma)$, entonces $v_{\sigma(r)}v_{\sigma(s)} \notin E(G)$ y por ende $v_{\sigma(r)}v_{\sigma(s)} \in E(\overline{G})$. Luego, $\sigma(k)\sigma(l) \to \sigma(r)\sigma(s)$ es un reemplazo en G y sólo hace falta verificar que sea admisible. En efecto, por el **Lema de la Composición 3.4.1** se tiene

$$G_{\sigma\sigma^{-1}} = G_{\sigma^{-1}} - v_{(\sigma^{-1})^{-1}(r)}v_{(\sigma^{-1})^{-1}(s)} + v_{(\sigma^{-1})^{-1}(k)}v_{(\sigma^{-1})^{-1}(l)},$$

de donde $G = G_{I_n} = G_{\sigma\sigma^{-1}} = G_{\sigma^{-1}} - v_{\sigma(r)}v_{\sigma(s)} + v_{\sigma(k)}v_{\sigma(l)}$. Pero de esta forma, para $G_{\sigma^{-1}}$ tenemos,

$$G_{\sigma^{-1}} = G - v_{\sigma(k)}v_{\sigma(l)} + v_{\sigma(r)}v_{\sigma(s)},$$

y como por la **Observación 3.2.1** tenemos en particular $G_{\sigma^{-1}} \simeq G$, se sigue que el reemplazo $\sigma(k)\sigma(l) \to \sigma(r)\sigma(s)$ es admisible en G y al mismo tiempo $\sigma^{-1} \in S_G(\sigma(k)\sigma(l) \to \sigma(r)\sigma(s))$. ■

Considérese entonces un reemplazo admisible de aristas $rs \to kl$ en G y una permutación $\sigma \in S_G(rs \to kl)$. De forma intuitiva pensaríamos en el "reemplazo inverso" de $rs \to kl$, como un reemplazo ahora hecho en G_σ que nos permite recuperar G, digamos $kl \to rs$ aplicado a G_σ. No obstante, los reemplazos que nos interesa estudiar son todos reemplazos que se hacen sobre G, y no sobre las σ-representaciones (distintas) de G. Por ello debemos buscar un reemplazo admisible de aristas de G, que se comporte de forma análoga al reemplazo $kl \to rs$ mencionado.

En este trabajo proponemos a $\sigma(k)\sigma(l) \to \sigma(r)\sigma(s)$ para ser dicho *reemplazo inverso*, ya que no sólo es un reemplazo admisible de aristas en G con $\sigma^{-1} \in S_G(\sigma(k)\sigma(l) \to \sigma(r)\sigma(s))$, sino que por el **Lema de la Composición 3.4.1**, respecto a este reemplazo se verifica que

$$G = G_{I_n} = G_{\sigma^{-1}\sigma} = G_\sigma - v_{\sigma^{-1}(\sigma(k))}v_{\sigma^{-1}(\sigma(l))} + v_{\sigma^{-1}(\sigma(r))}v_{\sigma^{-1}(\sigma(s))} = G_\sigma - v_k v_l + v_r v_s,$$

es decir que este reemplazo "compuesto" con G_σ, corresponde precisamente a aplicar $kl \to rs$ sobre G_σ para recuperar G, como queríamos. Con base en esto tenemos la siguiente definición.

Definición 3.5.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $rs \to kl$ un reemplazo admisible de aristas de G. Para cada $\sigma \in S_G(rs \to kl)$, nos referiremos al reemplazo admisible de aristas $\sigma(k)\sigma(l) \to \sigma(r)\sigma(s)$ de G, como el* **reemplazo inverso** *de $rs \to kl$ correspondiente a σ.*

Por ahora los reemplazos inversos nos ayudarán únicamente a introducir de forma natural la definición para un grupo de permutaciones del que dependerá, en particular, la definición de amoeba local. Sin embargo, volveremos a tratar con ellos en el capítulo 5 cuando veamos su relación con lo que llamaremos *Reemplazos Raros*.

Ya vimos que las composiciones entre permutaciones asociadas a reemplazos admisibles de una gráfica G, nos permite determinar todos los pares de copias de G que se pueden obtener una a partir de otra por medio de, por lo menos, una sucesión de reemplazos admisibles de aristas, i.e., la que incluye a G misma. Más aún, se debe recordar que la colección de todas las composiciones entre permutaciones de un conjunto, que además es cerrado bajo inversos como en este caso, es igual al grupo generado por dicho conjunto (ver **Proposición 2.2.7**). De aquí la siguiente definición.

Definición 3.5.2. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea R_G el conjunto de los reemplazos admisibles de aristas de G. Si G es no vacía, nos referiremos como* **grupo de alcanzabilidad** *de G, al grupo S_G generado por las permutaciones asociadas a reemplazos admisibles de aristas de G, esto es,*

$$S_G = \left\langle \bigcup_{rs \to kl \,\in\, R_G} S_G(rs \to kl) \right\rangle$$

Por otro lado, si G es vacía escribiremos $S_G = S_n$ para el grupo de alcanzabilidad de G. Adicionalmente, para cualquier permutación $\sigma \in S_n$, si tenemos $\sigma \in S_G$ diremos que G_σ se puede **alcanzar** *desde G y viceversa.*

La siguiente observación nos ayudará a mostrar algunas propiedades que relacionan el grupo S_G, con el grupo de automorfismos $Aut(G)$ de una gráfica G.

Observación 3.5.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $\alpha \in S_n$. Sea además $\widehat{\alpha} : V(G_\alpha) \to V(G)$ la biyección definida como $\widehat{\alpha}(v_i) = v_{\alpha(i)}$ para todo $v_i \in V(G_\alpha)$. Entonces se tiene $G_\alpha = G$, si y sólo si, $\widehat{\alpha} \in Aut(G)$.*

Demostración. Dada $\alpha \in S_n$ tal que $G_\alpha = G$, es inmediato que $\widehat{\alpha} \in Aut(G)$, ya que $\widehat{\alpha}$ es el isomorfismo de G_α en G, derivado de α. Por otro lado, dado $\widehat{\alpha} \in Aut(G)$, podemos definir $\alpha \in S_n$ para ser $\alpha(i) = j$ si y sólo si $\widehat{\alpha}(v_i) = v_j$, es decir, $\widehat{\alpha}(v_i) = v_{\alpha(i)}$. Luego, por la **Observación 3.2.1** tenemos $v_iv_j \in E(G_\alpha)$ si y sólo si $v_{\alpha(i)}v_{\alpha(j)} \in E(G)$, que es $\widehat{\alpha}(v_i)\widehat{\alpha}(v_j) \in E(G)$, y al ser $\widehat{\alpha}$ un automorfismo, equivalentemente $v_iv_j \in E(G)$, por lo que $E(G_\alpha) = E(G)$ y consecuentemente $G_\alpha = G$. ■

Proposición 3.5.2. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $Aut(G)$ el grupo de automorfismos de G. Entonces $Aut(G) \cong \{\alpha \in S_n \mid G_\alpha = G\} \leq S_G$.*

Demostración. Si G es vacía $Aut(G) \cong S_n = \{\alpha \in S_n \mid G_\alpha = G\} = S_G$. Supongamos que G no es vacía y veamos que $X = \{\alpha \in S_n | G_\alpha = G\}$ es un subgrupo de S_G. Como $G_{I_n} = G$, entonces $I_n \in X$. Además, ya que $G_\alpha = G$ para $\alpha \in X$, se sigue $\alpha \in S_G(rs \to kl)$ para cualquier reemplazo trivial $rs \to kl$ de G, digamos con $k = r$ y $l = s$. Luego, para $\alpha, \rho \in X$, por el **Lema de la Composición 3.4.1** se tiene $G_{\alpha\rho} = G_\rho - v_{\rho^{-1}(r)}v_{\rho^{-1}(s)} + v_{\rho^{-1}(r)}v_{\rho^{-1}(s)} = G_\rho = G$, de donde $\alpha\rho \in X$. Pero por la **Proposición 3.5.1** $\alpha^{-1} \in S_G(\alpha(r)\alpha(s) \to \alpha(r)\alpha(s))$, por lo que $G_{\alpha^{-1}} = G$ y entonces $\alpha^{-1} \in X$. Así, por la **Proposición 2.2.3** se sigue que X es subgrupo de S_G. Finalmente, usando la **Observación 3.5.1**, vemos que $f : X \to Aut(G)$ con $f(\alpha) = \widehat{\alpha}$ para toda $\alpha \in X$ es una biyección, y además $f(\alpha\rho)(v_i) = \widehat{\alpha\rho}(v_i) = v_{\alpha\rho(i)} = v_{\alpha(\rho(i))} = \widehat{\alpha}(v_{\rho(i)}) = \widehat{\alpha}\widehat{\rho}(v_i) = f(\alpha)f(\rho)(v_i)$ para todo $v_i \in V(G)$, por lo que $f(\alpha\rho) = f(\alpha)f(\rho)$ y entonces f es un isomorfismo de grupos entre X y $Aut(G)$. ■

Proposición 3.5.3. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$. Se tiene $S_G \cong Aut(G)$, si y sólo si, G es vacía, o bien todo reemplazo admisible de aristas de G es trivial.*

Demostración. Si G es vacía $Aut(G) \cong S_n = S_G$. Supongamos que G es no vacía. De $Aut(G) \cong S_G$, por la **Proposición 3.5.2** se sigue $S_G = \{\alpha \in S_n \mid G_\alpha = G\}$, que son todas permutaciones asociadas a reemplazos triviales de G. Más aún, dada cualquier $\alpha \in S_G$, no podemos tener $\alpha \in S_G(rs \to kl)$ para algún reemplazo no trivial $rs \to kl$ de G, ya que en tal caso $E(G_\alpha) \neq E(G)$ y así $G_\alpha \neq G$, una contradicción. Por otro lado, si G sólo tiene reemplazos triviales, para todo reemplazo admisible $rs \to kl$ de G y toda $\alpha \in S_G(rs \to kl)$ se tiene $G_\alpha = G - v_rv_s + v_kv_l = G - v_rv_s + v_rv_s = G$ y entonces $S_G = \langle\{\alpha \in S_n \mid G_\alpha = G\}\rangle$. Pero por la **Proposición 3.5.2** sabemos que $\{\alpha \in S_n \mid G_\alpha = G\}$ es un subgrupo de S_G, y al mismo tiempo S_G es el mínimo subgrupo de S_n que contiene a este conjunto (ver **Proposición 2.2.5**). Así, necesariamente $S_G = \{\alpha \in S_n \mid G_\alpha = G\} \cong Aut(G)$. ■

Observación 3.5.2. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$. Si G es una gráfica vacía, o bien la gráfica completa con vértices en V, entonces $S_G = S_n$.*

Demostración. Si G es vacía el resultado se sigue por definición. Si G es completa, todo reemplazo admisible de aristas de G es trivial y por la **Proposición 3.5.3** tenemos $S_G \cong Aut(G) \cong S_n$. ■

Ahora bien, si se tiene $\rho \in S_G$, entonces esta se puede escribir como un producto de permutaciones asociadas a reemplazos admisibles de aristas de G. De esta forma, por el **Teorema 3.4.1**, G_ρ podrá ser obtenida desde G por medio de una sucesión de reemplazos admisibles. Sin embargo, el recíproco de esto no se sigue inmediatamente. Es decir, si G_ρ se puede obtener desde G por una sucesión de reemplazos admisibles, por el **Teorema 3.4.1** existirán permutaciones $\sigma_1, ..., \sigma_m$ asociadas a reemplazos admisibles de aristas de G tales que la gráfica $G_\rho = G_{\sigma_m \cdots \sigma_1}$ se puede alcanzar desde G por una sucesión de reemplazos, de donde se tiene $\rho \overset{G}{\sim} \sigma_m \cdots \sigma_1$, más no necesariamente $\rho = \sigma_m \cdots \sigma_1$. Luego, si quisiéramos usar la membresía de S_G como equivalente de la existencia de las sucesiones de reemplazos admisibles de aristas, aún faltaría mostrar que en este último caso ρ sí pertenece a S_G. Por esta razón incluimos los siguientes dos resultados.

Proposición 3.5.4. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $\sigma \in S_n$. Si se tiene $\sigma \in S_G$, igualmente se tiene $\rho \in S_G$ para toda permutación ρ con $\rho \overset{G}{\sim} \sigma$.*

Demostración. Por la **Observación 3.5.1**, las permutaciones $\alpha \in S_n$ con $G_\alpha = G$ se corresponden una a una con los automorfismos de G. Pero además se tiene $\alpha \in S_G(rs \to kl)$ respecto a cualquier reemplazo trivial $rs \to kl$ de G, digamos con $k = r$ y $l = s$, por lo que $\alpha \in S_G$. Buscamos probar que cada $\rho \in S_n$ con $\rho \overset{G}{\sim} \sigma$ se puede escribir como $\rho = \alpha\sigma \in S_G$ para una única α de este tipo. En efecto, por el **Lema de la Composición 3.4.1** tenemos $G_{\alpha\sigma} = G_\sigma - v_{\sigma^{-1}(r)}v_{\sigma^{-1}(s)} + v_{\sigma^{-1}(r)}v_{\sigma^{-1}(s)} = G_\sigma$, por lo que $\alpha\sigma \overset{G}{\sim} \sigma$. Luego, para distintas α, α' con $G_\alpha = G_{\alpha'} = G$, se tiene $\alpha\sigma \neq \alpha'\sigma$, y entonces existen $aut(G)$ permutaciones que se pueden construir de la forma $\alpha\sigma \in S_G$, incluyendo $I_n\sigma = \sigma$ misma. Pero esta es exactamente la cardinalidad de la $\overset{G}{\sim}$-clase de σ, de donde se sigue el resultado. ■

Corolario 3.5.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $\rho \in S_n$. Sea además G_ρ la ρ-representación de G en K_n también con $V(K_n) = \{v_1, ..., v_n\}$. Se tiene $\rho \in S_G$, si y sólo si, G_ρ se puede obtener desde G por medio de una sucesión de reemplazos admisibles de aristas.*

Demostración. Usando el **Teorema 3.4.1** se sigue que si $\rho \in S_G$, entonces G_ρ se puede obtener desde G por medio de una sucesión de reemplazos admisibles de aristas. Por otro lado, si G_ρ se obtiene de esta forma desde G, entonces existen $m \geq 1$ permutaciones $\sigma_1, ..., \sigma_m$ asociadas a reemplazos admisibles de G con las que $G_\rho = G_{\sigma_m \cdots \sigma_1}$ se puede alcanzar desde G por una sucesión de reemplazos. Pero como $\sigma_m \cdots \sigma_1 \in S_G$ y a la vez $\rho \overset{G}{\sim} \sigma_m \cdots \sigma_1$, por la **Proposición 3.5.4** concluimos $\rho \in S_G$. ■

Así, dada una gráfica G de orden n, para que cualesquiera dos copias de G encajadas K_n (incluida G misma), se puedan obtener mutuamente por medio de sucesiones de reemplazos admisibles de aristas, el grupo S_G deberá ser en general igual al grupo S_n. Con esto finalmente podemos expresar a las amoebas en términos de permutaciones como se muestra a continuación.

Definición 3.5.3. ***(versión basada en permutaciones)***
Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea S_G el grupo de alcanzabilidad de G. Se dice que G es una **amoeba local** *si se tiene $S_G = S_n$. Por otro lado, se dice que G es una* **amoeba global***, si existe un entero $T \geq 0$ tal que la gráfica $G \dot{\cup} tK_1$ es una amoeba local para todo $t \geq T$.*

Esta definición proporciona un criterio computacional sencillo para decidir si G es amoeba local o no, ya que sólo deberemos revisar si la cardinalidad de S_G es $n!$ o no, respectivamente, siendo esta la cardinalidad del grupo S_n. Esta será además la definición que utilizaremos de ahora en adelante para referirnos a esta familia de gráficas. Más aún, con base en estas definiciones podemos clasificar a toda gráfica en alguna de las siguientes 4 colecciones: gráficas que son amoebas locales y globales, gráficas que sólo son amoebas locales, otras sólo globales, y por último gráficas que no satisfacen ninguna de las dos definiciones. Abajo se presentan ejemplos de cada una de estas clases, mostrándose así que ninguna de las definiciones de amoebas implica a la otra. Es importante mencionar que en el siguiente capítulo veremos propiedades que simplificarán la detección computacional de amoebas, como el **Teorema 4.1.1** que establece que para poder determinar si una gráfica G es amoeba global, es suficiente y necesario verificar que la gráfica $G \dot{\cup} K_1$ sea amoeba local.

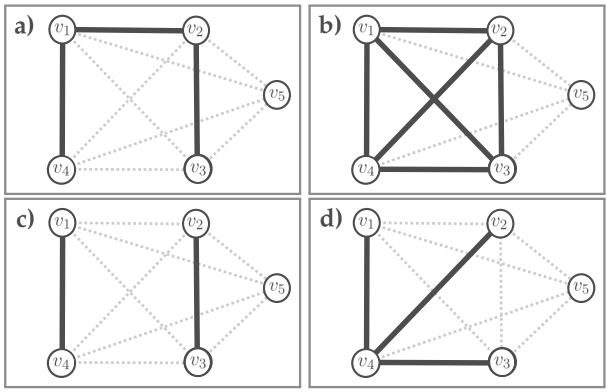

Figura 3.8: Cuatro gráficas encajadas en K_5. Con los algoritmos que estudiaremos en esta tesis podemos analizar exhaustivamente los reemplazos admisibles de cada gráfica para obtener su grupo S_G, con propósito de verificar lo que se describe a continuación. En **a)** se muestra P_4, que de forma similar a P_3 en la **Figura. 3.2**, puede "recorrer" la gráfica completa de 4 vértices por medio de reemplazos admisibles, y como esto mismo sucede al agregar cualquier cantidad de $t \geq 1$ vértices aislados a P_4, se sigue que esta es una amoeba local y global. Por otro lado, la gráfica K_4 en **b)** es amoeba local por la **Observación 3.5.2**, pero como ninguna de sus aristas se puede cambiar por otra que tenga a v_5 como un extremo, esta no puede ser amoeba global. Luego, $K_2 \dot{\cup} K_2$ en **c)** es amoeba global pero no local, ya que esta puede "recorrer" K_5, pero estando encajada en K_4 sólo tiene reemplazos triviales. Finalmente, $K_{1,3}$ mostrada en **d)** no es amoeba de ningún tipo, ya que no es posible "intercambiar" el vértice v_4, de grado 3, por ningún otro vértice en ella.

Capítulo 4

Propiedades Importantes para Detectar Amoebas Computacionalmente

Uno de los objetivos principales de esta tesis es estudiar y comprender los fundamentos matemáticos de las amoebas para poder llevar a cabo su detección computacional. En el capítulo anterior se establecieron las bases teóricas para poder hablar de esta familia de gráficas. Por otro lado, en el capítulo 5 trataremos con los *reemplazos raros*, aportación original de esta tesis con la que se busca complementar a los algoritmos que usaremos para su detección, mostrados en el capítulo 6. Sin embargo, el presente capítulo comprende una recopilación de resultados ya conocidos sobre estas gráficas. No obstante, por medio de estos resultados buscaremos establecer una conexión entre la construcción teórica de las amoebas y las condiciones que se han de revisar sobre una gráfica para poder decidir si esta es una amoeba o no, por lo que este capítulo representa una primera discusión dentro de esta tesis, sobre las propiedades de las amoebas teniendo en mente un contexto computacional. Debemos enfatizar que todos los resultados mostrados en este capítulo son parte del trabajo original de la Dra. Adriana Hansberg, el Dr. Yair Caro y la Dra. Amanda Montejano sobre la construcción y estudio de las amoebas [1], así como del trabajo de la Mat. Jennifer Lilith Espinosa [8], quien realizó sus tesis de licenciatura bajo la dirección de la Dra. Adriana Hansberg.

4.1. Problemas de Decisión de Amoebas Locales (PDLA) y Amoebas Globales (PDGA)

Uno de nuestros objetivos principales es poder desarrollar un algoritmo que, al revisar ciertas propiedades y condiciones de una gráfica G, sea capaz de decirnos si G es una amoeba o no, y de serlo, que además indique si esta es global, local o ambas. No obstante, debemos notar que desarrollar un algoritmo con tal capacidad, en realidad conllevaría encontrar la solución a dos problemas computacionales distintos, llámese uno **Problema de Decisión de las Amoebas Locales (PDLA)**, y al otro **Problema de Decisión de las Amoebas Globales (PDGA)**.

Un problema computacional es llamado *problema de decisión*, cuando su solución conlleva una respuesta del tipo "sí" o "no" [15,18]. Así, al hablar sobre el PDLA y el PDGA, simplemente nos referimos a los problemas de indicar si una gráfica arbitraria es amoeba local, o no, y respectivamente, amoeba global, o no. De la teoría desarrollada hasta ahora ya se sigue una solución al PDLA, mientras que para resolver el PDGA todavía hace falta contar con otras propiedades de las amoebas.

Veamos primero como funcionaría a grandes rasgos un algoritmo para resolver el PDLA. La **Definición 3.5.3** nos dice que para decidir si una gráfica G de orden n es amoeba local o no, debemos revisar que su grupo de alcanzabilidad S_G (ver **Definición 3.5.2**) sea igual al grupo simétrico S_n en n elementos, lo que se puede hacer al checar que se cumpla $|S_G| = |S_n| = n!$, ya que no hay ningún otro grupo de permutaciones en n elementos, que contenga a S_n como subgrupo propio.

Así, para resolver el PDLA sobre una gráfica G no vacía, primero debemos obtener S_G. Pero obtener dicho grupo generado no es eficiente en la práctica, en particular si se deben calcular y almacenar $n!$ permutaciones. Por ello deberemos obtener la cardinalidad de S_G, habiendo calculado únicamente el conjunto de permutaciones asociadas a reemplazos admisibles de aristas de G, llámese este $\mathcal{X}_G \subseteq S_n$. Más aún, obtener $\mathcal{X}_G$ sí es viable en la práctica, ya que estas permutaciones se pueden determinar como isomorfismos entre las representaciones computacionales que daremos a G, y a $G - v_rv_s + v_kv_l$ para cada reemplazo admisible $rs \to kl$ de G, en el capítulo 7.

Luego, para obtener la cardinalidad de $S_G = \langle \mathcal{X}_G \rangle$ sin obtener explícitamente a S_G, haremos uso de las ya existentes implementaciones en Python del algoritmo Schreier-Sims [26] en la biblioteca **Sympy**, funciones ampliamente utilizadas y construidas con el propósito de recibir un subconjunto arbitrario $S \subseteq S_n$ y devolver $|\langle S \rangle|$. No obstante, el estudio de dicho algoritmo pertenece al área de Teoría Computacional de Grupos [27], y como tal queda fuera del alcance de esta tesis.

Todo lo anterior nos permitirá dar en el capítulo 6 un algoritmo para resolver el PDLA. Ahora bien, ya que la definición de amoeba global depende explícitamente de la definición de amoeba local, estaríamos inclinados a pensar que todo algoritmo que resuelve el PDLA, es capaz de resolver también el PDGA. Más abajo veremos que este ciertamente es el caso. Sin embargo, debemos notar que para poder determinar si una gráfica G es amoeba global siguiendo la definición, requeriríamos evaluar que la gráfica $G \dot\cup tK_1$, fuera una amoeba local para todo entero $t \geq T$, dado algún $T \geq 0$. Pero esto no es computacionalmente posible, ya que en principio exige realizar dicha evaluación para un conjunto infinito de valores. No obstante, la solución a este problema ya fue determinada por la Dra. Adriana Hansberg, el Dr. Yair Caro y la Dra. Amanda Montejano, y comprende una caracterización de las amoebas globales. A continuación presentamos dicho resultado, y haremos uso de este a lo largo de la tesis, pero omitimos su demostración y para ello referimos a [1].

Teorema 4.1.1. (Hansberg, Caro, Montejano, [1]) *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$. Las siguientes propiedades son equivalentes,*

i) G es amoeba global.
ii) Para todo entero $i \in [n]$, existe $\sigma \in S_G$ tal que para la imagen $j = \sigma(i)$ se tiene $d_G(v_j) = 1$.
iii) $G \dot\cup K_1$ es amoeba local. ■

Luego, esto implica inmediatamente que para poder determinar si una gráfica G es amoeba global, debemos simplemente evaluar si $G \dot\cup K_1$, es decir, la unión disjunta de G con un único vértice aislado, es una amoeba local. Así, con motivo de hacer explícito uno de los usos que daremos en esta tesis al **Teorema 4.1.1**, incluimos también el siguiente corolario.

Corolario 4.1.1. *Todo algoritmo que resuelve el PDLA, resuelve también el PDGA.* ■

Más aún, en el trabajo en [1] también se presentan otros resultados como el siguiente lema, cuya relación con el **Teorema 4.1.1** nos permitirá deducir inmediatamente si una amoeba local G con ciertas propiedades, es también una amoeba global. Esto será de utilidad, por ejemplo, para poder evitar la evaluación del PDLA sobre $G \dot\cup K_1$, o por lo menos evitar el cálculo del tamaño del grupo de alcanzabilidad $S_{G \dot\cup K_1}$, después de ya haber invertido recursos para evaluar el PDLA sobre G misma. Nuevamente omitimos la demostración de este lema y para ello referimos a [1].

Lema 4.1.1. (Hansberg, Caro, Montejano, [1]) *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y grado mínimo $\delta \in \{0, 1\}$. Si G es amoeba local, entonces $G \dot\cup K_1$ es amoeba local.* ■

Así, finalmente podremos contar con estrategias para reducir, por lo menos en la práctica, la cantidad de operaciones que deben desarrollar lo algoritmos de detección de amoebas. Esto es de especial importancia, ya que estas estrategias no sólo complementarán a los algoritmos que resuelvan estos problemas sobre gráficas arbitrarias, sino que también podremos proponer con ellas algoritmos más eficientes para la detección de árboles que son amoebas, como el siguiente corolario, que es una implicación directa del **Teorema 4.1.1** junto con el **Lema 4.1.1**.

Corolario 4.1.2. *Sea T un árbol de orden n con $V(T) = \{v_1, ..., v_n\}$. Si T es una amoeba local, entonces T es también una amoeba global.* ■

4.2. Grados de Todos los Vértices en una Amoeba

Continuamos exhibiendo más propiedades de las amoebas que nos ayudarán en su detección computacional. Aquí mostramos algunas que nos hablan sobre los grados de todos los vértices, en gráficas que cumplen ser amoebas de algún tipo. Veremos ahora una que es consecuencia directa del **Lema de la Composición 3.4.1**. Recuérdese que todas estas son originales de [1].

Observación 4.2.1. (Hansberg, Caro, Montejano, [1]) *Sean G una gráfica de orden n con vértices $V(G) = \{v_1, ..., v_n\}$ y $rs \to kl$ un reemplazo admisible de aristas de G. Sean además $\sigma \in S_G(rs \to kl)$ y $\rho \in S_n$ una permutación arbitraria. Para cualquier vértice $v_i \in \{v_1, ..., v_n\}$ se tiene,*

$$d_{G_{\sigma\rho}}(v_i) = \begin{cases} d_{G_\rho}(v_i) - 1, & si\ \ \rho(i) \in \{r, s\} \setminus \{k, l\} \\ d_{G_\rho}(v_i) + 1, & si\ \ \rho(i) \in \{k, l\} \setminus \{r, s\} \\ d_{G_\rho}(v_i), & cualquier\ otro\ caso \end{cases}$$

Demostración. Del **Lema de la Composición 3.4.1** tenemos $G_{\sigma\rho} = G_\rho - v_{\rho^{-1}(r)}v_{\rho^{-1}(s)} + v_{\rho^{-1}(k)}v_{\rho^{-1}(l)}$. Luego, dado $v_i \in \{v_1, ..., v_n\}$, si $i = \rho^{-1}(r)$ o bien $i = \rho^{-1}(s)$, pero $i \neq \rho^{-1}(k)$ y a la vez $i \neq \rho^{-1}(l)$, entonces $d_{G_{\sigma\rho}}(v_i) = d_{G_\rho}(v_i) - 1$. Los demás casos se siguen de forma análoga. ■

De aquí se sigue un propiedad sobre la secuencia de grados de toda amoeba local.

Proposición 4.2.1. (Hansberg, Caro, Montejano, [1]) *Sea G una gráfica de orden n con conjunto de vértices $V(G) = \{v_1, ..., v_n\}$, con grado mínimo δ y grado máximo Δ. Si G es una amoeba local, entonces para cada entero r con $\delta \leq r \leq \Delta$ existe por lo menos un vértice v_i de G tal que $d_G(v_i) = r$.*

Demostración. Escójanse vértices v_a y v_b de G con $d_G(v_a) = \delta$ y $d_G(v_b) = \Delta$. Considérese la transposición $\rho = (a\ b) \in S_n$. Como G es amoeba local se tiene $S_G = S_n$ y en consecuencia $\rho \in S_G$. Pero entonces existen permutaciones $\sigma_1, ..., \sigma_m$ con $\sigma_i \in S_G(r_i s_i \to k_i l_i)$ para toda $i \in [m]$ tales que $\rho = \sigma_m \sigma_{m-1} \cdots \sigma_1$. Defínase $\beta_i = \sigma_i \sigma_{i-1} \cdots \sigma_1$ para cada i, donde en particular se tiene $\beta_m = \rho$, y además póngase $\beta_0 = I_n$. Considérese la sucesión $(d_{G_{\beta_0}}(v_a), d_{G_{\beta_1}}(v_a), ..., d_{G_{\beta_m}}(v_a))$ formada por los grados de v_a en cada β_i-representación G_{β_i} de G en K_n con $V(K_n) = \{v_1, ..., v_n\}$, para toda $i \in \{0, ..., m\}$. Luego, tenemos $d_{G_{\beta_0}}(v_a) = d_{G_{I_n}}(v_a) = d_G(v_a) = \delta$ y por el isomorfismo $\widehat{\rho}(v_i) = v_{\rho(i)}$ de G_ρ en G derivado de ρ, tenemos también $d_{G_{\beta_m}}(v_a) = d_{G_\rho}(v_a) = d_G(v_{\rho(a)}) = d_G(v_b) = \Delta$. Supóngase ahora que existe un entero r con $\delta < r < \Delta$ tal que $d_{G_{\beta_i}}(v_a) \neq r$ para toda $i \in \{0, ..., m\}$ y considérese el primer índice j tal que $d_{G_{\beta_j}}(v_a) \geq r + 1$, que sabemos que existe ya que $d_{G_{\beta_m}}(v_a) = \Delta$. De aquí necesariamente $d_{G_{\beta_{j-1}}}(v_a) \leq r - 1$. Pero como $G_{\beta_j} = G_{\sigma_j \sigma_{j-1} \cdots \sigma_1} = G_{\sigma_j \beta_{j-1}}$ y a la vez $\sigma_j \in S_G(r_j s_j \to k_j l_j)$, por la **Observación 4.2.1** se sigue que $d_{G_{\beta_j}}(v_a) = d_{G_{\beta_{j-1}}}(v_a) + q$ con $q \in \{-1, 0, +1\}$ y así $d_{G_{\beta_j}}(v_a) \leq r < r + 1$, que nos da una contradicción. Por lo tanto, debe existir un índice $i \in \{1, ..., m - 1\}$ tal que $d_{G_{\beta_i}}(v_a) = r$. Pero $G_{\beta_i} \simeq G$ para toda i, por lo que existe en G un vértice de grado r, y como esto no dependió del valor particular de r, se sigue el resultado. ■

La ventaja computacional de la propiedad anterior radica en que, si el conjunto de los grados de todo vértice en una gráfica G, con grado mínimo δ y grado máximo Δ, no es igual al intervalo de enteros $\{\delta, \delta + 1, ..., \Delta\}$, podremos concluir sin más que G no es una amoeba local, evitando de nueva cuenta gastar recursos innecesariamente.

Más aún, ya que una gráfica G es amoeba global si y sólo si $G \dot{\cup} K_1$ es una amoeba local, podemos también enunciar la siguiente propiedad de las amoebas globales, que se deriva directamente del **Teorema 4.1.1** y de la **Proposición 4.2.1**.

Corolario 4.2.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y grado máximo Δ. Si G es una amoeba global, entonces i) existe por lo menos un vértice $v_r \in V(G)$ con $d_G(v_r) = 1$, y ii) para el conjunto $D_G = \{d_G(v_i) \mid v_i \in V(G)\}$, de grados de los vértices de G, se tiene $D_G = \{0\} \cup [\Delta]$, o bien $D_G = [\Delta]$.* ■

Adicionalmente incluimos la propiedad análoga pero ahora aplicada a árboles.

Proposición 4.2.2. *Sea T un árbol de orden n con $V(T) = \{v_1, ..., v_n\}$ y con grado máximo Δ. Si T es una amoeba local o global, entonces para el conjunto $D_T = \{d_T(v_i) \mid v_i \in V\}$ se cumple $D_T = [\Delta]$.* ■

4.3. Grados de los Vértices Involucrados en un Reemplazo Admisible

Por último incluimos una propiedad presentada originalmente en la tesis de licenciatura de la Mat. Jennifer Lilith Espinosa [8]. Aunque inicialmente parece una propiedad simple, esta ha probado ser un instrumento poderoso en la implementación de los algoritmos de detección de amoebas, ya que permite descartar reemplazos $rs \to kl$ de G sin tener que evaluar el isomorfismo entre $G - v_r v_s + v_k v_l$ y G, durante la búsqueda de reemplazos admisibles de G. Omitimos su prueba, que sigue directamente del isomorfismo esperado entre $G - v_r v_s + v_k v_l$ y G, para todo reemplazo admisible de aristas $rs \to kl$ de G. Finalmente, su corolario nos será de ayuda en el capítulo 6.

Proposición 4.3.1. (Espinosa [8]) *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$. Para todo reemplazo admisible de aristas $rs \to kl$ de G se cumplen las siguientes condiciones,*

i) Si $|\{r, s\} \cap \{k, l\}| = 0$, entonces $\{d_G(v_k) + 1, d_G(v_l) + 1\} = \{d_G(v_r), d_G(v_s)\}$.

ii) Si $|\{r, s\} \cap \{k, l\}| = 1$ digamos con $k = r$, entonces $\{d_G(v_k), d_G(v_l) + 1\} = \{d_G(v_r), d_G(v_s)\}$.

iii) Si $|\{r, s\} \cap \{k, l\}| = 2$, entonces $\{d_G(v_k), d_G(v_l)\} = \{d_G(v_r), d_G(v_s)\}$. ■

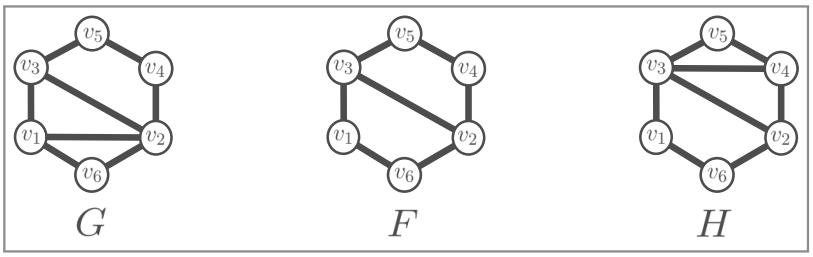

Figura 4.1: Ejemplo de una gráfica G sobre la que se aplica un reemplazo admisible de aristas $12 \to 34$. En la imagen se denota $F = G - v_1 v_2$ y $H = G - v_1 v_2 + v_3 v_4$. Al ser G y H isomorfas, estas deben tener la misma secuencia de grados. Ya que todo vértice v_i de G distinto de v_1, v_2, v_3 y v_4 "conserva" su grado durante el reemplazo, los únicos dos vértices que pueden "recuperar" los grados de v_1 y v_2, serán los vértices v_3 y v_4 al agregar la nueva arista a F para obtener H. Por ello $\{d_G(v_3) + 1, d_G(v_4) + 1\} = \{d_G(v_1), d_G(v_2)\}$, como indicado por el inciso i) de la **Proposición 4.3.1**.

Corolario 4.3.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $rs \to kl$ un reemplazo cualquiera de aristas de G. Sea además $F = G - v_r v_s$. Para $rs \to kl$ se mantienen todas las condiciones en la* **Proposición 4.3.1**, *si y sólo si, se tiene $\{d_F(v_k), d_F(v_l)\} = \{d_F(v_r), d_F(v_s)\}$.*

Demostración. Considérese el caso i) de la **Proposición 4.3.1**. Aquí se tiene $d_F(v_k) = d_G(v_k)$ y $d_F(v_l) = d_G(v_l)$, y a la vez $d_F(v_r) = d_G(v_r) - 1$ y $d_F(v_s) = d_G(v_s) - 1$. Si ocurre $d_G(v_k) + 1 = d_G(v_r)$ y $d_G(v_l) + 1 = d_G(v_s)$, entonces $d_F(v_k) + 1 = d_G(v_k) + 1 = d_G(v_r) = d_F(v_r) + 1$ y similarmente $d_F(v_l) + 1 = d_G(v_l) + 1 = d_G(v_s) = d_F(v_s) + 1$, de donde $d_F(v_k) = d_F(v_r)$ y $d_F(v_l) = d_F(v_s)$. Pero si $d_F(v_k) = d_F(v_r)$ y $d_F(v_l) = d_F(v_s)$, también se sigue $d_G(v_k) + 1 = d_F(v_k) + 1 = d_F(v_r) + 1 = d_G(v_r)$ y $d_G(v_l) + 1 = d_F(v_l) + 1 = d_F(v_s) + 1 = d_G(v_s)$. Los otros casos se siguen de forma análoga. ■

Capítulo 5

Principal Contribución Teórica de esta Tesis: Reemplazos Raros

Hasta ahora hemos presentado las bases para poder hablar de las amoebas, así como mostrado propiedades que nos serán de utilidad para simplificar su detección computacional. Pese a esto, es natural preguntarse si existen más condiciones con las que se puedan descartar reemplazos no admisibles de aristas en una gráfica G, sin tener que evaluar el isomorfismo para todos ellos. Una idea (que trataremos más a detalle en la **Sección 5.4** de este capítulo) podría ser, por ejemplo, restringir dicha evaluación a los reemplazos $rs \to kl$, donde los vértices v_k y v_l sean escogidos dentro de las órbitas naturales (ver **Definición 2.3.5**) de los vértices v_r y v_s, pero obtenidas en la gráfica $G - v_r v_s$ respecto a su grupo de automorfismos $Aut(G - v_r v_s)$. En este trabajo no sólo hemos encontrado ejemplos de que tal estrategia no es viable en lo general, sino que estos ejemplos nos han permitido formular un tipo de reemplazo admisible que se comporta de forma contraintuitiva. Nuevamente debemos añadir que entre los resultados de este capítulo se exhiben algunas *Observaciones*, que a pesar de deducirse en su mayoría directamente de las definiciones, se incluyen aquí junto con sus pruebas con propósito de hacer explícita la construcción de las ideas abordadas en esta tesis.

5.1. Noción de Reemplazo Raro

Ahora introduciremos de forma intuitiva los conceptos de *reemplazo raro* y *reemplazo ordinario*, pero debemos hacer notar que inmediatamente en la siguiente sección se presentan las definiciones formales de las ideas aquí planteadas. Considérese entonces la gráfica G mostrada en la **Figura 5.1**, sobre la que se realiza el reemplazo admisible $23 \to 35$, dando como resultado la copia $G_{(14)(25)}$ de G encajada en K_5. Una propiedad importante a resaltar de dicho reemplazo es que la nueva arista v_3v_5 se escoge de tal forma que "juega" en $G_{(14)(25)}$, el mismo "papel que jugaba" la arista v_2v_3 en G.

Pero ¿será cierto que todos los reemplazos admisibles cumplen con esta propiedad?, es decir ¿si $rs \to kl$ es un reemplazo admisible en G, entonces $v_r v_s$ y $v_k v_l$ "juegan siempre el mismo papel", respectivamente, en G y en $G - v_r v_s + v_k v_l$?

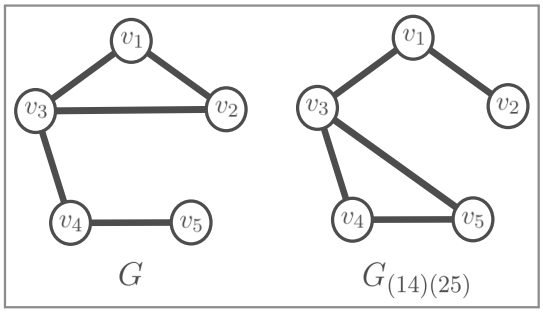

Figura 5.1: Las aristas v_2v_3 de G y v_3v_5 de $G_{(14)(25)}$, "juegan el mismo papel" en su respectiva gráfica, siendo parte del único ciclo en ellas e incidentes con el único vértice de grado 3.

Podríamos pensar que en efecto todos los reemplazos admisibles de una gráfica se comportan de esta forma, que llamaremos natural u *ordinaria*. Sin embargo este no es el caso, ya que en la **Figura 5.2** se muestra un tipo de reemplazo admisible de aristas que no cumple con dicha propiedad, al que en contraste nos referiremos como *reemplazo raro*.

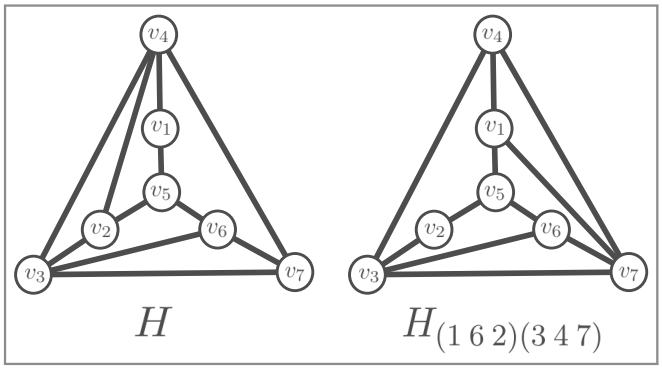

Figura 5.2: Denotemos $H' = H_{(1\,6\,2)(3\,4\,7)}$. Aunque H' se obtiene de H por el reemplazo admisible $24 \to 17$, no existe forma de mapear los vértices de H' en H preservando adyacencia y a la vez enviando la arista v_1v_7 de H' a v_2v_4 en H, es decir, v_1v_7 no puede "volver a jugar el papel" de v_2v_4. De hecho, es posible verificar que H (igualmente H') es una gráfica asimétrica al comparar las vecindades de vértices del mismo grado, para ver que no existen dos vértices distintos que sean vértices semejantes. Esto implica que $\sigma = (1\,6\,2)(3\,4\,7)$ es el único elemento de $S_G(24 \to 17)$ por el **Corolario 3.3.1**. Así, $\widehat{\sigma}(v_i) = v_{\sigma(i)}$ es el único isomorfismo de H' en H, y con este se tiene $\widehat{\sigma}(v_1)\widehat{\sigma}(v_7) = v_3v_6$, es decir que v_1v_7 en H', "juega ahora el papel" de la arista v_3v_6 en G.

Aquí definiremos y analizaremos a los reemplazos raros y sus propiedades. Debemos señalar que todo el material presentado en este capítulo es una aportación original de esta tesis, ya que este tipo de reemplazos no se había estudiado con anterioridad en el contexto de las amoebas. Comenzaremos dando una definición para los reemplazos raros, con la que se establecerá formalmente lo que se entiende cuando se menciona que las aristas v_rv_s y v_kv_l, dadas por un reemplazo admisible $rs \to kl$, "juegan el mismo papel" en sus respectivas gráficas. Luego, veremos algunas características de estos reemplazos, que nos permitirán en particular contar con una alternativa a las condiciones que se deben evaluar para determinar si un reemplazo admisible particular es raro o no. Posteriormente veremos cómo estas propiedades tienen implicaciones importantes para la detección computacional de las amoebas. Por último presentaremos una familia infinita de árboles que poseen reemplazos raros, y que al mismo tiempo son amoebas locales y globales.

5.2. Definiciones Formales de Reemplazo Raro y Reemplazo Ordinario

En la **Figura 5.2** se sugiere la idea de que, para que un reemplazo admisible de aristas $rs \to kl$ de una gráfica G sea raro, no debe existir ningún isomorfismo de la gráfica $G' = G - v_rv_s + v_kv_l$ en G, tal que envíe los extremos de la arista $v_kv_l \in E(G')$ hacia los extremos de $v_rv_s \in E(G)$.

Esta idea será nuestro punto de partida para estudiar a los reemplazos raros, más no procuraremos inicialmente a los isomorfismos entre dichas gráficas, sino a las permutaciones en S_n de las que se derivan tales isomorfismos (ver **Definición 3.2.2**). Recuérdese que dada una gráfica G y un reemplazo admisible $rs \to kl$ de G, se denota por $S_G(rs \to kl)$ al conjunto de todas las permutaciones $\sigma \in S_n$ tales que $G_\sigma = G - v_r v_s + v_k v_l$. De esto se tiene una primera definición.

Definición 5.2.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $rs \to kl$ un reemplazo admisible de aristas de G. Diremos que $rs \to kl$ es un* **reemplazo ordinario de aristas** *de G, si existe por lo menos una permutación $\sigma \in S_G(rs \to kl)$ tal que $\{\sigma(k), \sigma(l)\} = \{r, s\}$. De lo contrario, diremos que $rs \to kl$ es un* **reemplazo raro de aristas** *de G.*

Observación 5.2.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $rs \to kl$ un reemplazo admisible de aristas de G. El reemplazo $rs \to kl$ es ordinario, si y sólo si, existe por lo menos un isomorfismo $\widehat{\sigma}$ de la gráfica $G' = G - v_r v_s + v_k v_l$ en G, tal que $\widehat{\sigma}(v_k)\widehat{\sigma}(v_l) = v_r v_s$.*

Demostración. Si $rs \to kl$ es ordinario, existe $\sigma \in S_G(rs \to kl)$ tal que $\{\sigma(k), \sigma(l)\} = \{r, s\}$, digamos con $\sigma(k) = r$ y $\sigma(l) = s$. Luego, con el isomorfismo $\widehat{\sigma}$ de G' en G derivado de σ, se tiene enseguida $\widehat{\sigma}(v_k) = v_{\sigma(k)} = v_r$ y $\widehat{\sigma}(v_l) = v_{\sigma(l)} = v_s$. Lo recíproco se sigue inmediatamente para $\sigma \in S_n$ definida como $\sigma(i) = j$ si y sólo si $\widehat{\sigma}(v_i) = v_j$, para cualquier isomorfismo $\widehat{\sigma}$ tal que $\widehat{\sigma}(v_k)\widehat{\sigma}(v_l) = v_r v_s$. ■

Esto quiere decir que si $rs \to kl$ es un reemplazo raro en G, no habrá forma de mapear la arista $v_k v_l$ de $G - v_r v_s + v_k v_l$, hacia $v_r v_s$ en G preservando adyacencia, a pesar de que ambas gráficas son en efecto isomorfas. En otras palabras, $v_k v_l$ no puede "volver a jugar el papel" que $v_r v_s$ tenía en G, como planteado inicialmente. Ahora damos nombre a las gráficas con estos reemplazos.

Definición 5.2.2. *Decimos que una gráfica G de orden n con $V(G) = \{v_1, ..., v_n\}$, es una* **gráfica rara**, *si G tiene por lo menos un reemplazo raro. En caso contrario, o si G es vacía, esta será llamada* **gráfica ordinaria**. *Si además G es una amoeba, diremos que G es una* **amoeba rara (resp. amoeba ordinaria)**.

Observación 5.2.2. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$. Todo reemplazo trivial de G es un reemplazo ordinario de aristas de G.*

Demostración. Para todo reemplazo trivial $rs \to kl$ de G, sin pérdida de generalidad con $k = r$ y $l = s$, se tiene $I_n \in S_G(rs \to kl)$, $I_n(k) = k = r$ y $I_n(l) = l = s$, por lo que $rs \to kl$ es ordinario. ■

Observación 5.2.3. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$. Si todo reemplazo admisible de aristas de G es trivial, entonces G es una gráfica ordinaria.* ■

Para ejemplificar lo visto hasta ahora, en la **Figura 5.3** se exponen dos gráficas, una con ciclos y otra sin ciclos, que tienen tanto reemplazos raros, como reemplazos ordinarios no triviales.

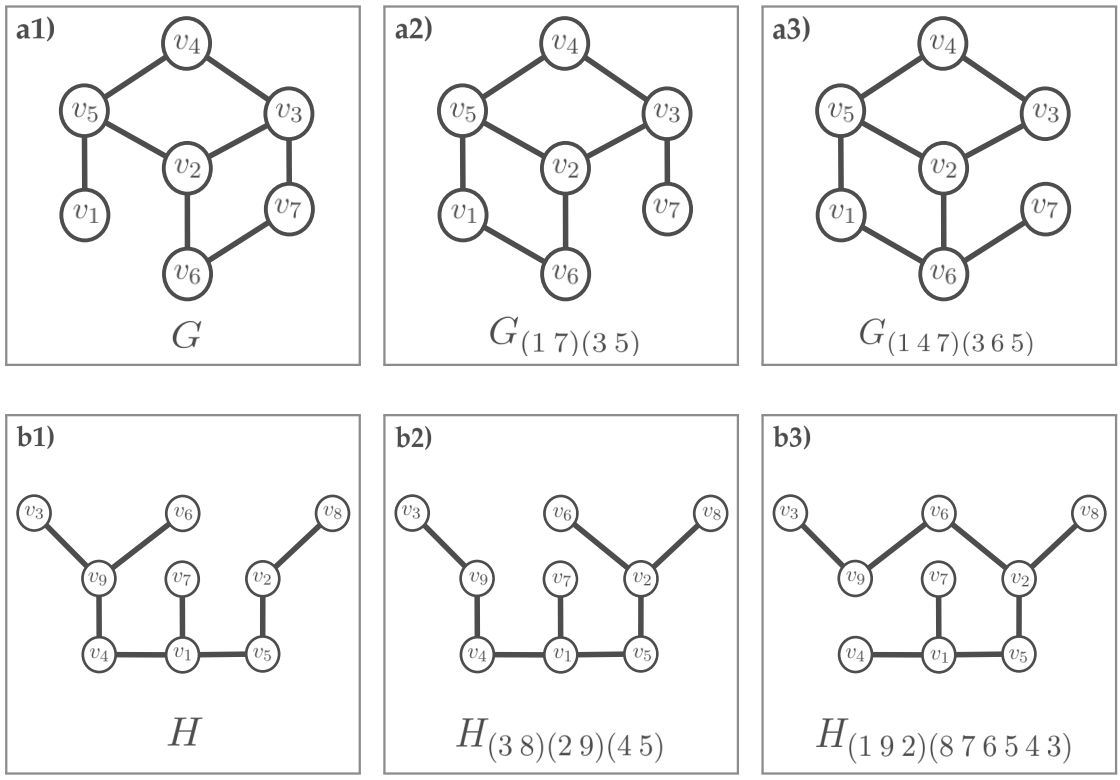

Figura 5.3: En **a1)** la gráfica G tiene un reemplazo ordinario $67 \to 16$ que da lugar en **a2)** a la copia $G_{(1\,7)(3\,5)}$ y un reemplazo raro $37 \to 16$ que devuelve en **a3)** la copia $G' = G_{(1\,4\,7)(3\,6\,5)}$. Podemos ver que $37 \to 16$ es raro usando la **Observación 5.2.1**, al notar que para cualquier isomorfismo φ de G' en G se tiene $\varphi(v_7) = v_1$ y por ende $\varphi(v_1)\varphi(v_6) = v_4v_5$. Igualmente, **b1)** de H se obtiene **b2)** $H_{(3\,8)(2\,9)(4\,5)}$ por el reemplazo ordinario $69 \to 26$ y **b3)** $H' = H_{(1\,9\,2)(8\,7\,6\,5\,4\,3)}$ por el reemplazo raro $49 \to 26$. Esto último se verifica también por la **Observación 5.2.1**, al ver que para todo isomorfismo φ de H' en H se tiene $\varphi(v_1) = v_9$ y así $\varphi(v_2)\varphi(v_6) = v_1v_5$.

5.3. Caracterización por Órbitas de Aristas y Reemplazo Inverso del Reemplazo Raro

De la **Observación 5.2.1** se sigue un algoritmo exhaustivo para reconocer reemplazos raros en una gráfica G. Específicamente, la observación nos dice que dado un reemplazo admisible de aristas $rs \to kl$ de G, debemos revisar todos los isomorfismos de $G' = G - v_rv_s + v_kv_l$ en G para buscar si existe o no, uno que envíe los extremos de la arista v_kv_l de G', hacia los extremos de la arista v_rv_s de G. Sin embargo, en esta sección desarrollaremos algunos resultados que nos permitirán determinar si $rs \to kl$ es raro o no, al conocer un único isomorfismo de $G - v_rv_s + v_kv_l$ en G.

Consideramos además que algunos de estos resultados tienen relevancia teórica, ya que de ellos se sigue una relación entre los reemplazos admisibles de aristas, sean estos raros u ordinarios, y las órbitas naturales de las aristas de una gráfica G, es decir, sus órbitas obtenidas respecto a la acción natural $\ominus$, del grupo $Aut(G)$ sobre $E(G)$ dada por $\varphi \ominus v_iv_j = \varphi(v_i)\varphi(v_j)$ para todo $\varphi \in Aut(G)$ y cualquier $v_iv_j \in E(G)$ (ver **Definiciones 2.3.4** y **2.3.5**).

Más aún, de esto se sigue el **Teorema 5.3.2** sobre el reemplazo inverso de un reemplazo raro, que no sólo complementa las aportaciones teóricas de la **Proposición 3.5.1** y de la **Definición 3.5.1**, o las incluidas en este capítulo, sino que además nos permitirá explicar el comportamiento de algunos resultados de las aportaciones computacionales, que veremos en la **Sección 7.5** del capítulo 7.

Comenzamos motivando este análisis con el ejemplo de la **Figura 5.4**. En esta se sugiere la relación entre las órbitas naturales de aristas de un gráfica G y el tipo de reemplazo admisible en cuestión. Más abajo resumimos esta relación en dos problemas de interés.

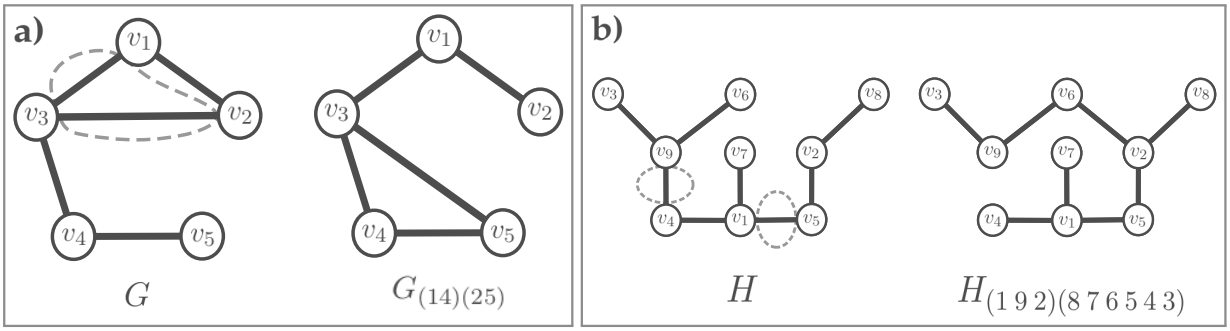

Figura 5.4: Dos gráficas G y H con dos automorfismos cada una. En **a)** se obtiene $G' = G_{(1\,4)(2\,5)}$ por el reemplazo ordinario $23 \to 35$ de G y se encierra en un contorno punteado la órbita natural que incluye a las aristas v_1v_3 y v_2v_3 de G, mientras que todas las demás aristas están solas en sus propias órbitas. Nótese que uno de los dos isomorfismos φ_1 de G' en G cumple $\varphi_1(v_3)\varphi_1(v_5) = v_2v_3$ y el otro isomorfismo de G' en G envía los extremos de v_3v_5, a los extremos de una arista en la órbita de v_2v_3. Por otro lado, en **b)** se obtiene $H' = H_{(1\,9\,2)(8\,7\,6\,5\,4\,3)}$ por el reemplazo raro $49 \to 26$ de H y se encierran con líneas punteadas dos órbitas distintas, la de la arista v_4v_9 y la de v_1v_5. En este caso, los dos isomorfismos de H' en H no sólo no envían v_2v_6 hacia v_4v_9, sino que mapean esta arista de H' en una misma órbita natural de H distinta de la de v_4v_9, es decir la de v_1v_5.

El primero de los dos problemas a estudiar consiste en, dado un reemplazo admisible de aristas $rs \to kl$, determinar si de todas las permutaciones en $S_G(rs \to kl)$ se derivan isomorfismos que envían v_kv_l a una misma órbita natural de aristas en G, independientemente de si el reemplazo $rs \to kl$ es raro u ordinario. El segundo problema es saber si, al existir un isomorfismo que en efecto mande v_kv_l a la órbita de v_rv_s, se puede además deducir que $rs \to kl$ es ordinario. Los dos siguientes lemas dan, respectivamente, respuesta a cada uno de estos problemas. Recuérdese que se denota por $Aut(G)[v_iv_j]$ a la órbita natural (**Definición 2.3.5**) de $v_iv_j \in E(G)$.

Lema 5.3.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $rs \to kl$ un reemplazo admisible de aristas de G. Para cualesquiera $\sigma, \rho \in S_G(rs \to kl)$ se tiene $Aut(G)[v_{\sigma(k)}v_{\sigma(l)}] = Aut(G)[v_{\rho(k)}v_{\rho(l)}]$.*

Demostración. Considérense cualesquiera $\sigma, \rho \in S_G(rs \to kl)$ y denótese $G' = G - v_rv_s + v_kv_l$. Recuérdese que el isomorfismo $\widehat{\sigma} : V(G') \to V(G)$ derivado de σ está dado por $\widehat{\sigma}(v_i) = v_{\sigma(i)}$. Más aún, ya que $\sigma(i) = j$ si y sólo si $\widehat{\sigma}(v_i) = v_j$, el isomorfismo inverso $\widehat{\sigma}^{-1} : V(G) \to V(G')$ de $\widehat{\sigma}$, se puede escribir como $\widehat{\sigma}^{-1}(v_i) = v_{\sigma^{-1}(i)}$. Luego, por las **Proposiciones 2.1.1** y **2.1.2**, la composición $\widehat{\rho}\widehat{\sigma}^{-1} : V(G) \to V(G)$ es un automorfismo de G. Pero entonces con este se sigue

$$\widehat{\rho}\widehat{\sigma}^{-1} \ominus v_{\sigma(k)}v_{\sigma(l)} = \widehat{\rho}\widehat{\sigma}^{-1}(v_{\sigma(k)})\widehat{\rho}\widehat{\sigma}^{-1}(v_{\sigma(l)}) = \widehat{\rho}(v_{\sigma^{-1}\sigma(k)})\widehat{\rho}(v_{\sigma^{-1}\sigma(l)}) = \widehat{\rho}(v_k)\widehat{\rho}(v_l) = v_{\rho(k)}v_{\rho(l)},$$

por lo que debemos tener $v_{\rho(k)}v_{\rho(l)} \in Aut(G)[v_{\sigma(k)}v_{\sigma(l)}]$, y finalmente usando la **Proposición 2.2.15** podemos concluir $Aut(G)[v_{\sigma(k)}v_{\sigma(l)}] = Aut(G)[v_{\rho(k)}v_{\rho(l)}]$, para $\sigma, \rho \in S_G(rs \to kl)$ arbitrarias. ■

Lema 5.3.2. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $rs \to kl$ un reemplazo admisible de aristas de G. Si existe una permutación $\sigma \in S_G(rs \to kl)$ tal que $v_{\sigma(k)}v_{\sigma(l)} \in Aut(G)[v_rv_s]$, entonces $rs \to kl$ es un reemplazo ordinario en G.*

Demostración. Por la **Proposición 2.2.15** tenemos $Aut(G)[v_{\sigma(k)}v_{\sigma(l)}] = Aut(G)[v_rv_s]$. Luego, debe existir $\widehat{\alpha} \in Aut(G)$ tal que $\widehat{\alpha} \ominus v_{\sigma(k)}v_{\sigma(l)} = v_rv_s$. Así, con $\alpha \in S_n$ dada por $\alpha(i) = j$ si y sólo si $\widehat{\alpha}(v_i) = v_j$, se tiene $\widehat{\alpha}(v_{\sigma(k)})\widehat{\alpha}(v_{\sigma(l)}) = v_{\alpha\sigma(k)}v_{\alpha\sigma(l)} = v_rv_s$ y entonces $\{\alpha\sigma(k), \alpha\sigma(l)\} = \{r, s\}$. Más aún, α está asociada a cualquier reemplazo trivial, digamos $\alpha \in S_G(rs \to rs)$. De esta forma, por el **Lema de la Composición 3.4.1** tenemos $G_{\alpha\sigma} = G_\sigma - v_{\sigma^{-1}(r)}v_{\sigma^{-1}(s)} + v_{\sigma^{-1}(r)}v_{\sigma^{-1}(s)} = G_\sigma$, y como $G_\sigma = G - v_rv_s + v_kv_l$, igualmente $G_{\alpha\sigma} = G - v_rv_s + v_kv_l$, o lo que es lo mismo $\alpha\sigma \in S_G(rs \to kl)$. Con estas propiedades $rs \to kl$ satisface la definición de reemplazo ordinario, como requerido. ■

El siguiente resultado depende de los **Lemas 5.3.1** y **5.3.2**. Aunque este, y su corolario, pueden parecer redundantes, ambos nos permitirán hacer explícitas propiedades que tienen tanto consecuencias teóricas, como aplicaciones computacionales en la detección de los reemplazos raros.

Teorema 5.3.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $rs \to kl$ un reemplazo admisible de aristas de G. Las siguientes propiedades son equivalentes,*

[O1] *Existe una permutación $\sigma \in S_G(rs \to kl)$ tal que $Aut(G)[v_{\sigma(k)}v_{\sigma(l)}] = Aut(G)[v_rv_s]$.*

[O2] *Para toda permutación $\sigma \in S_G(rs \to kl)$ se tiene $Aut(G)[v_{\sigma(k)}v_{\sigma(l)}] = Aut(G)[v_rv_s]$.*

[O3] *El reemplazo $rs \to kl$ es ordinario.*

Demostración. La implicación de **[O1]** a **[O2]** se sigue por el **Lema 5.3.1**. Luego **[O2]** implica **[O3]** usando el **Lema 5.3.2**. Finalmente **[O3]** implica **[O1]**, ya que al ser $rs \to kl$ ordinario, por definición debe existir $\sigma \in S_G(rs \to kl)$ tal que $\{\sigma(k), \sigma(l)\} = \{r, s\}$, y por lo tanto $v_{\sigma(k)}v_{\sigma(l)} = v_rv_s$. ■

Corolario 5.3.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $rs \to kl$ un reemplazo admisible de aristas de G. Las siguientes propiedades son equivalentes,*

[R1] *Existe una permutación $\sigma \in S_G(rs \to kl)$ tal que $Aut(G)[v_{\sigma(k)}v_{\sigma(l)}] \neq Aut(G)[v_rv_s]$.*

[R2] *Para toda permutación $\sigma \in S_G(rs \to kl)$ se tiene $Aut(G)[v_{\sigma(k)}v_{\sigma(l)}] \neq Aut(G)[v_rv_s]$.*

[R3] *El reemplazo $rs \to kl$ es raro.*

Demostración. **[R1]** es la negación de **[O2]**. A su vez **[R2]** es la negación de **[O1]**. Igualmente **[R3]** es la negación de **[O3]**. Su equivalencia se sigue por la equivalencia entre **[O1]**, **[O2]** y **[O3]**. ■

Así, **[O1]** y **[R1]** tienen una aplicación computacional, ya que nos dicen que basta con conocer la órbita $Aut(G)[v_rv_s]$ y a $v_{\sigma(k)}v_{\sigma(l)}$ dada una única $\sigma \in S_G(rs \to kl)$, para determinar si el reemplazo $rs \to kl$ es **[O1]** ordinario si $v_{\sigma(k)}v_{\sigma(l)} \in Aut(G)[v_rv_s]$ y **[R1]** raro si $v_{\sigma(k)}v_{\sigma(l)} \notin Aut(G)[v_rv_s]$. Por otro lado, **[O2]** y **[R2]** tienen importancia teórica, ya que junto con el **Lema 5.3.1**, nos dicen que dado $rs \to kl$, los isomorfismos derivados de las permutaciones en $S_G(rs \to kl)$ mapean v_kv_l, todos a la órbita $Aut(G)[v_rv_s]$ si este es ordinario, o todos a una misma órbita distinta de la de v_rv_s si es raro. Esto último prohíbe, en particular, la existencia de casos como el esbozado en la **Figura 5.5**.

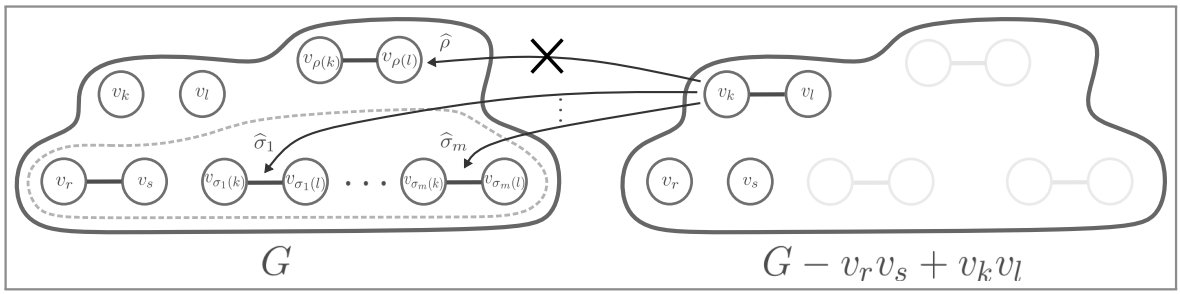

Figura 5.5: Representación de una gráfica G y su copia $G' = G - v_rv_s + v_kv_l$ obtenida por un reemplazo ordinario $rs \to kl$. El inciso **[O2]** del **Teorema 5.3.1** nos dice que todos los isomorfismos $\widehat{\sigma}_1, ..., \widehat{\sigma}_m$ (mostrados en la imagen como flechas) derivados de cualesquiera permutaciones $\sigma_1, ..., \sigma_m \in S_G(rs \to kl)$, envían $v_kv_l \in E(G')$ hacia la órbita de $v_rv_s \in E(G)$, delimitada aquí por un contorno punteado, sin que pueda existir un isomorfismo $\widehat{\rho}$ de G' en G mapeando v_kv_l hacia una arista $v_{\rho(k)}v_{\rho(l)}$ fuera de la órbita de v_rv_s en G.

Finalmente, apoyados en el **Teorema 5.3.1**, podemos probar ahora que los reemplazos inversos (ver **Definición 3.5.1**) de cualquier reemplazo raro son igualmente reemplazos raros.

Teorema 5.3.2. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $rs \to kl$ un reemplazo admisible de aristas de G. Si $rs \to kl$ es un reemplazo raro, entonces el reemplazo inverso $\sigma(k)\sigma(l) \to \sigma(r)\sigma(s)$ de $rs \to kl$ correspondiente a cualquier $\sigma \in S_G(rs \to kl)$, es también un reemplazo raro de G.*

Demostración. Escójase cualquier $\sigma \in S_G(rs \to kl)$ y procediendo por contradicción supongamos que el reemplazo $\sigma(k)\sigma(l) \to \sigma(r)\sigma(s)$ es un reemplazo ordinario de G.

Denotemos $r' = \sigma(k)$, $s' = \sigma(l)$, $k' = \sigma(r)$ y $l' = \sigma(s)$. Luego, por **[O2]** en el **Teorema 5.3.1**, para toda $\rho \in S_G(r's' \to k'l')$ tenemos $Aut(G)[v_{\rho(k')}v_{\rho(l')}] = Aut(G)[v_{r'}v_{s'}]$, de donde sustituyendo se sigue $Aut(G)[v_{\rho\sigma(r)}v_{\rho\sigma(s)}] = Aut(G)[v_{\sigma(k)}v_{\sigma(l)}]$. Sin embargo, por la **Proposición 3.5.1** sabemos que además se tiene $\sigma^{-1} \in S_G(\sigma(k)\sigma(l) \to \sigma(r)\sigma(s))$. De esta forma, para $\rho = \sigma^{-1}$ se se seguiría $Aut(G)[v_rv_s] = Aut(G)[v_{\sigma^{-1}\sigma(r)}v_{\sigma^{-1}\sigma(s)}] = Aut(G)[v_{\sigma(k)}v_{\sigma(l)}]$. Pero entonces la permutación σ, que está asociada a $rs \to kl$, sería tal que $Aut(G)[v_{\sigma(k)}v_{\sigma(l)}] = Aut(G)[v_rv_s]$, y de ser el caso, por **[O1]** en el **Teorema 5.3.1** el reemplazo $rs \to kl$ mismo tendría que ser un reemplazo ordinario de G, que es la contradicción buscada. Por lo tanto $\sigma(k)\sigma(l) \to \sigma(r)\sigma(s)$ debe ser raro, y como esto no dependió de la elección particular de $\sigma \in S_G(rs \to kl)$, entonces se sigue lo enunciado por el teorema. ■

Corolario 5.3.2. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $rs \to kl$ un reemplazo admisible de aristas de G. Sea además $\sigma \in S_G(rs \to kl)$ arbitraria. El reemplazo $rs \to kl$ es un reemplazo raro, si y sólo si, el reemplazo inverso $\sigma(k)\sigma(l) \to \sigma(r)\sigma(s)$ de $rs \to kl$ es también un reemplazo raro de G.*

Demostración. Por la **Proposición 3.5.1** tenemos $\sigma^{-1} \in S_G(\sigma(k)\sigma(l) \to \sigma(r)\sigma(s))$. Pero el reemplazo inverso de $\sigma(k)\sigma(l) \to \sigma(r)\sigma(s)$ correspondiente a σ^{-1} es $\sigma^{-1}\sigma(r)\sigma^{-1}\sigma(s) \to \sigma^{-1}\sigma(k)\sigma^{-1}\sigma(l)$, es decir $rs \to kl$. De esta forma, el resultado en ambas direcciones se sigue por el **Teorema 5.3.2**. ■

5.4. Caracterización por Órbitas de Vértices y su Efecto en la Detección de Amoebas

En la sección anterior vimos una conexión entre los reemplazos admisibles de una gráfica G y las órbitas de sus aristas. Aquí estableceremos una relación entre cada reemplazo admisible $rs \to kl$ de aristas de G y los automorfismos de la gráfica $G - v_rv_s$. Buscando motivar esto abordaremos ahora en detalle la estrategia esbozada en el párrafo introductorio de este capítulo.

Para poder determinar si una gráfica G es una amoeba o no, necesitamos primero identificar todos los reemplazos admisibles de aristas en ella. Una forma de hacer esto es seleccionando exhaustivamente todo par de aristas $v_rv_s \in E(G)$ y $v_kv_l \in E(\overline{G})$, para después revisar si $G - v_rv_s + v_kv_l$ es isomorfa a G por medio de algoritmos como el **VF2**, comentado al final de la **Sección 2.4** en el capítulo 2 sobre los conceptos preliminares.

No obstante, realizar dicha búsqueda exhaustiva puede resultar computacionalmente demandante. Por ello es recomendable desarrollar criterios bajo los que seleccionar $v_kv_l \in E(\overline{G})$, tales que permitan reducir la cantidad de veces que se debe de ejecutar un algoritmo para evaluar el isomorfismo entre G y sus copias obtenidas por cada reemplazo.

Un criterio que inicialmente se consideró en esta tesis, consiste en quitar v_rv_s de G, y escoger los extremos de v_kv_l dentro de las órbitas naturales de v_r y v_s en $G - v_rv_s$ (e.g., **Figura 5.6**), es decir, obtenidas respecto a la acción $\odot$ del grupo $Aut(G-v_rv_s)$ sobre $V(G-v_rv_s)$, dada por $\varphi \odot v_i = \varphi(v_i)$ para cada $\varphi \in Aut(G - v_rv_s)$ y todo $v_i \in V(G - v_rv_s)$ (ver **Definiciones 2.3.4** y **2.3.5**).

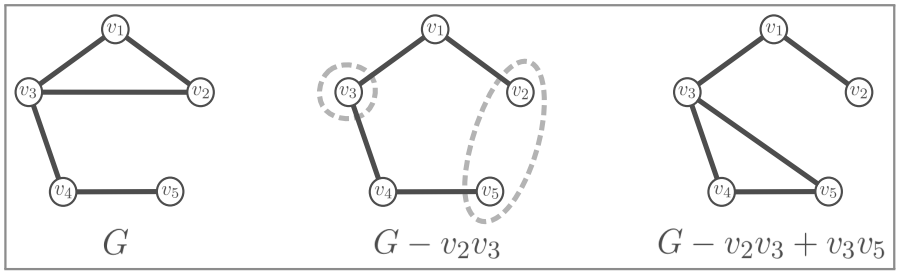

Figura 5.6: Una gráfica G sobre la que se realiza un reemplazo admisible de aristas $23 \to 35$. Se encierran con líneas punteadas las órbitas naturales de los vértices v_2 y v_3 en $G - v_2v_3$. En este caso v_3 está solo en su órbita, y se escoge además v_5 en la órbita de v_2, obteniéndose así un reemplazo ordinario de G.

Ahora bien, aunque este criterio no nos permite asegurar que $G - v_rv_s + v_kv_l$ sea siempre isomorfa a G como es el caso de la gráfica en la **Figura 5.7**, con esta sí podemos garantizar que respecto a $G - v_rv_s$, los nuevos vértices v_k y v_l tengan en conjunto los mismos grados que v_r y v_s (ver **Proposición 2.3.3**). Así, esto permitiría a $G - v_rv_s + v_kv_l$ conservar la secuencia de grados de G, que es una condición necesaria del isomorfismo que buscamos entre estas gráficas.

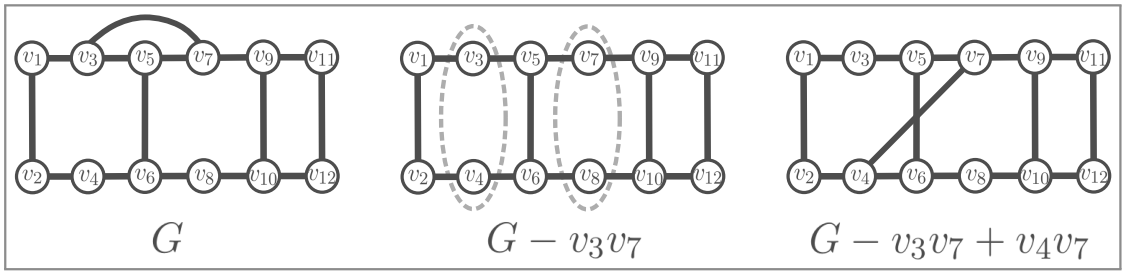

Figura 5.7: Se realiza un reemplazo de aristas $37 \to 47$ que no es admisible sobre una gráfica G, y se encierran con líneas punteadas las órbitas de v_3 y v_7 en $G - v_3v_7$. Es posible ver que G y $G' = G - v_3v_7 + v_4v_7$ no son isomorfas ya que G contiene un ciclo con 3 vértices mientras que G' sólo tiene ciclos con más de 4 vértices. Esto sucede a pesar de que v_4 se escoge en la órbita de v_3 y de que el extremo v_7 se conserva en el reemplazo, lo que indica que el criterio planteado no garantiza en general el isomorfismo entre estas gráficas.

Sin embargo, utilizando dicha estrategia no sólo se producen reemplazos de aristas que no son admisibles, sino que con ella también se falla en recuperar todos los reemplazo admisibles de aristas de una gráfica. Esto se puede ver en ejemplos como el de la **Figura 5.8**, de reemplazos admisibles de aristas $rs \to kl$ que no cuentan con esta propiedad, es decir, donde los extremos de v_kv_l no pertenecen en $G - v_rv_s$ a las órbitas de v_r y v_s. Es importante notar que el ejemplo en esta figura es además un reemplazo raro de la gráfica en cuestión.

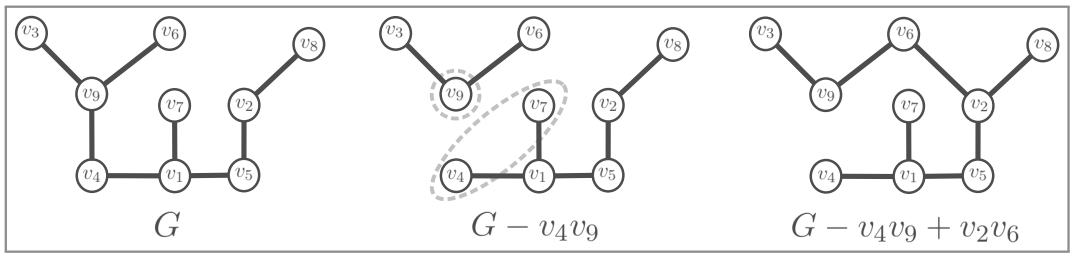

Figura 5.8: Se realiza un reemplazo admisible de aristas $49 \to 26$ sobre una gráfica G y se encierran con líneas punteadas las órbitas de los vértices v_4 y v_9 en $G - v_4v_9$. No obstante, ninguno de los extremos de la nueva arista v_2v_6 pertenecen a las órbitas de v_4 o v_9, y aún así este es un reemplazo admisible de G. Más aún, por lo comentado en la **Figura 5.3**, sabemos que este es también un reemplazo raro de esta gráfica.

Luego, resulta natural preguntarse si el criterio mencionado antes garantiza únicamente que el reemplazo admisible $rs \to kl$ sea ordinario. La respuesta se da en el siguiente resultado.

Teorema 5.4.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $rs \to kl$ un reemplazo admisible de aristas de G. El reemplazo $rs \to kl$ es un reemplazo ordinario de G, si y sólo si, existe por lo menos un automorfismo $\widehat{\alpha} \in Aut(G - v_rv_s)$ tal que $\{\widehat{\alpha}(v_k), \widehat{\alpha}(v_l)\} = \{v_r, v_s\}$.*

Demostración. Supongamos primero que $rs \to kl$ es un reemplazo ordinario de G. Luego, debe haber $\sigma \in S_G(rs \to kl)$ tal que $\{\sigma(k), \sigma(l)\} = \{r, s\}$. Consideremos el isomorfismo $\widehat{\sigma}$ de G_σ en G derivado de σ, dado por $\widehat{\sigma}(v_i) = v_{\sigma(i)}$ para todo $v_i \in V(G_\sigma)$. Como para todo par v_i, v_j de vértices (distintos) se tiene $v_iv_j \in E(G_\sigma) \leftrightarrow \widehat{\sigma}(v_i)\widehat{\sigma}(v_j) \in E(G)$, podemos poner

$$v_iv_j \in E(G_\sigma) \setminus \{v_kv_l\} \leftrightarrow \widehat{\sigma}(v_i)\widehat{\sigma}(v_j) \in E(G) \setminus \{\widehat{\sigma}(v_k)\widehat{\sigma}(v_l)\}.$$

Pero ambos conjuntos en la equivalecia son iguales a $E(G - v_rv_s)$. Para ver esto nótese que al tener $G_\sigma = G - v_rv_s + v_kv_l$, se sigue que $G_\sigma - v_kv_l = G - v_rv_s$, con lo que

$$E(G_\sigma) \setminus \{v_kv_l\} = E(G_\sigma - v_kv_l) = E(G - v_rv_s), \text{ mientras que}$$
$$E(G) \setminus \{\widehat{\sigma}(v_k)\widehat{\sigma}(v_l)\} = E(G) \setminus \{v_{\sigma(k)}v_{\sigma(l)}\} = E(G) \setminus \{v_rv_s\} = E(G - v_rv_s),$$

y podemos reescribir la equivalencia de antes como $v_iv_j \in E(G - v_rv_s) \leftrightarrow \widehat{\sigma}(v_i)\widehat{\sigma}(v_j) \in E(G - v_rv_s)$, es decir, $\widehat{\sigma}$ es un automorfismo de $G - v_rv_s$ tal que $\{\widehat{\sigma}(v_k), \widehat{\sigma}(v_l)\} = \{v_{\sigma(k)}v_{\sigma(l)}\} = \{v_r, v_s\}$.

Por otro lado supongamos que existe $\widehat{\alpha} \in Aut(G - v_rv_s)$ tal que $\{\widehat{\alpha}(v_k), \widehat{\alpha}(v_l)\} = \{v_r, v_s\}$. Sabemos que se cumple $v_iv_j \in E(G - v_rv_s) \leftrightarrow \widehat{\alpha}(v_i)\widehat{\alpha}(v_j) \in E(G - v_rv_s)$. Pero entonces podemos poner $v_iv_j \in E(G - v_rv_s) \cup \{v_kv_l\} \leftrightarrow \widehat{\alpha}(v_i)\widehat{\alpha}(v_j) \in E(G - v_rv_s) \cup \{\widehat{\alpha}(v_k)\widehat{\alpha}(v_l)\}$. No obstante,

$$E(G - v_rv_s) \cup \{v_kv_l\} = E(G - v_rv_s + v_kv_l) \text{ y}$$
$$E(G - v_rv_s) \cup \{\widehat{\alpha}(v_k)\widehat{\alpha}(v_l)\} = E(G - v_rv_s) \cup \{v_rv_s\} = E(G).$$

Así, $v_iv_j \in E(G - v_rv_s + v_kv_l) \leftrightarrow \widehat{\alpha}(v_i)\widehat{\alpha}(v_j) \in E(G)$, es decir, $\widehat{\alpha}$ es un isomorfismo de $G - v_rv_s + v_kv_l$ en G tal que $\widehat{\alpha}(v_k)\widehat{\alpha}(v_l) = v_rv_s$, y por la **Observación 5.2.1** concluimos que $rs \to kl$ es ordinario. ■

Esto indica que los reemplazos ordinarios, y sólo los ordinarios, se construyen al realizar un reemplazo admisible $rs \to kl$ escogiendo v_k y v_l en las órbitas $Aut(G-v_rv_s)[v_r]$ y $Aut(G-v_rv_s)[v_s]$, pero con ambos v_r y v_s como imágenes de v_k y v_l bajo un mismo automorfismo $\widehat{\alpha} \in Aut(G-v_rv_s)$. Así, de no existir dicho automorfismo podremos concluir que el reemplazo admisible es raro.

Pero esto también tiene implicaciones computacionales, ya que no es posible, en lo general, determinar todos los reemplazos admisibles de una gráfica G, simplemente restringiendo la evaluación del isomorfismo a reemplazos donde v_kv_l tenga sus extremos en las órbitas de v_r y v_s en $G-v_rv_s$, como planteado inicialmente. No obstante, por el **Corolario 4.3.1** sabemos que en $G-v_rv_s$, los vértices v_k y v_l deben tener en conjunto los mismos grados que v_r y v_s independientemente de si $rs \to kl$ es raro u ordinario, ya que de otra forma este reemplazo no sería admisible.

De esta manera, la estrategia discutida en esta sección puede modificarse para elegir los extremos de v_kv_l, no en las órbitas de v_r y v_s en $G - v_rv_s$ como antes, sino ahora en cada uno de los conjuntos de vértices que tengan los mismos grados que v_r y v_s en esa misma gráfica, que adicionalmente incluyen a las órbitas naturales $Aut(G-v_rv_s)[v_r]$ y $Aut(G-v_rv_s)[v_s]$ debido a la **Proposición 2.3.3**, como se ejemplifica en la **Figura 5.9**.

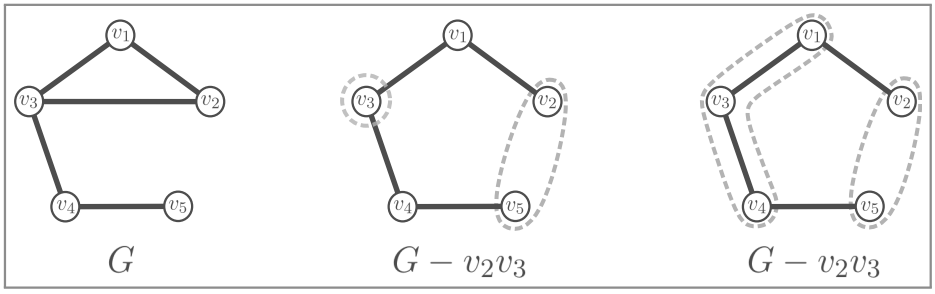

Figura 5.9: Una gráfica G y dos copias de $G - v_2v_3$. En la copia de la izquierda se encierran con líneas punteadas las órbitas $Aut(G-v_2v_3)[v_2]$ y $Aut(G-v_2v_3)[v_3]$, mientras que en la copia de la derecha se encierran los conjuntos de vértices de $G - v_2v_3$ que tienen los mismo grados que v_2 y v_3. La **Proposición 2.3.3** nos dice que las órbitas mencionadas están contenidas en los conjuntos de vértices del mismo grado.

Más adelante en el capítulo 6, que engloba a los algoritmos que se derivan de la teoría vista hasta ahora, discutiremos a detalle esta modificación para seleccionar la arista $v_kv_l \in E(\overline{G})$ por medio de los grados de los vértices en $G - v_rv_s$, y veremos entonces la repercusión de esta idea sobre la detección computacional de amoebas. Por ahora concluiremos este capítulo estudiando una familia infinita de árboles raros que además son amoebas locales y globales. Debemos señalar que esta familia no sólo es una contribución teórica, sino que muestra que en realidad existe un conjunto infinito de gráficas con reemplazos raros, es decir, de gráficas con reemplazos admisibles que no se pueden detectar usando el **Teorema 5.4.1**, ni la estrategia planteada inicialmente.

5.5. Una Familia Infinita de Árboles Amoeba Raros

Como parte de nuestra contribución teórica definiremos y analizaremos ahora una familia infinita de árboles, a la que de aquí en adelante denotaremos por $\mathcal{T}$, que cuentan con reemplazos raros. En esta sección desarrollaremos resultados que nos permitirán probar que estos árboles son además amoebas locales y globales. No obstante, debemos agregar que en la **Sección 7.5** del capítulo 7 se exhiben aún más gráficas con reemplazos raros, obtenidas computacionalmente, incluyendo amoebas raras minimales y otros ejemplos que cuentan con cierta relevancia teórica.

Ahora bien, para comenzar considérese el árbol de la **Figura 5.8**, que ahora denotaremos por T^1, ya que este será el mínimo elemento de la familia de árboles $\mathcal{T} = \{T^k\}_{k=1}^{\infty}$ a construir aquí. Es importante mencionar que la notación T^k, i.e., enumerando estos elementos con un superíndice k, se escoge así ya que más adelante trataremos con las σ-representaciones de estos árboles, para las que ya se usa la notación T_σ^k como definido en el capítulo 3 (ver **Definición 3.2.1**). Por lo tanto T^k no debe ser confundida con una notación de potencias.

Cabe mencionar que T^1 es además el árbol más pequeño, esto es, con la menor cantidad de vértices de entre todo árbol, que tiene reemplazos raros, como veremos en la **Sección 7.5**. Primero estudiaremos las propiedades de T^1 y con ello motivaremos una definición para todo árbol en la familia $\mathcal{T}$. Sin embargo, posteriormente deberemos analizar el caso T^2, ya que las diferencias entre este último con T^1 nos facilitarán preparar las ideas necesarias para poder hablar del caso general T^k en $\mathcal{T}$. Consideremos entonces el reemplazo admisible de aristas $15 \to 23$ para T^1 mostrado en la **Figura 5.10**, y recordemos que dada una gráfica G se denota por S_G al grupo de alcanzabilidad de G y se escribe $S_G(rs \to kl) = \{\sigma \in S_n \mid G_\sigma = G - v_r v_s + v_k v_l\}$ (ver **Definiciones 3.5.2** y **3.3.2**).

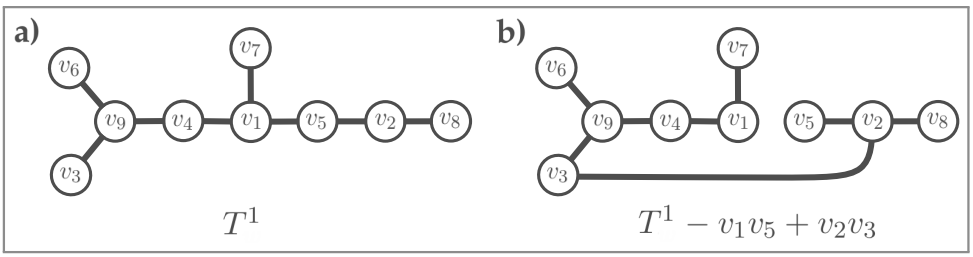

Figura 5.10: Se muestran **a)** el árbol T^1 de orden 9 y **b)** $T^1 - v_1v_5 + v_2v_3$. Más adelante veremos cómo la numeración escogida para los vértices de T^1, facilitará el análisis de este y de los demás árboles en $\mathcal{T}$.

Veamos primero que $15 \to 23$ es un reemplazo raro de T^1. Para ello nótese que los vértices v_2 en $T^1 - v_1v_5 + v_2v_3$ y v_9 en T^1 son, en sus respectivos árboles, los únicos vértices de grado 3 adyacentes con dos hojas. Así, la arista $v_2v_3 \in E(T^1 - v_1v_5 + v_2v_3)$ será mapeada hacia $v_4v_9 \in E(T^1)$ por todo isomorfismo entre estos árboles, y por la **Observación 5.2.1** se concluye que $15 \to 23$ es raro.

Por otro lado busquemos mostrar que T^1 es una amoeba local, es decir, que para su grupo de alcanzabilidad S_{T^1} se tiene $S_{T^1} = S_9$, donde S_9 es el grupo simétrico de los enteros en $\{1, ..., 9\}$. En efecto, nótese que respecto al reemplazo $15 \to 23$ de T^1 se tiene $(1\,2\,9)(3\,4\,5\,6\,7\,8) \in S_{T^1}(15 \to 23)$, mientras que respecto al reemplazo admisible $25 \to 58$ también tenemos $(2\,8) \in S_{T^1}(25 \to 58)$. Pero como $S_{T^1}(15 \to 23)$ y $S_{T^1}(25 \to 58)$ son subconjuntos de S_{T^1}, entonces $\{(129)(345678), (28)\} \subseteq S_{T^1}$ y consecuentemente por su producto se tiene $(1\,2\,3\,4\,5\,6\,7\,8\,9) = [(1\,2\,9)(3\,4\,5\,6\,7\,8)](2\,8) \in S_{T^1}$.

Ahora bien, del reemplazo admisible $14 \to 13$ de T^1 también podríamos obtener $(3\,4) \in S_{T^1}$, y por el párrafo anterior se tendría $\{(3\,4), (3\,4\,5\,6\,7\,8\,9\,1\,2)\} \subseteq S_{T^1}$. Luego, como por el **Teorema 2.2.3** sabemos que sin pérdida de generalidad $S_9 = \langle (3\,4), (3\,4\,5\,6\,7\,8\,9\,1\,2)\rangle$, usando la **Proposición 2.2.5** podríamos poner $S_9 = \langle (3\,4), (3\,4\,5\,6\,7\,8\,9\,1\,2)\rangle \leq S_{T^1} \leq S_9$, de donde $S_{T^1} = S_9$, es decir, T^1 es una amoeba local, y por el **Corolario 4.1.2** vemos que T^1 es igualmente una amoeba global.

Más aún, en la **Figura 5.11** se muestra una sucesión de 4 reemplazos admisibles de aristas de la que se deduce que $(1\,2) \in S_{T^1}$. Dicha sucesión no sólo nos facilitará construir el generador $S_9 = \langle (1\,2), (1\,2\,3\,4\,5\,6\,7\,8\,9)\rangle \leq S_{T^1}$, sino que basados en ella, o bien una versión de ella motivada por el caso T^2, obtendremos el generador análogo contenido en S_{T^k} para cada árbol T^k de $\mathcal{T}$.

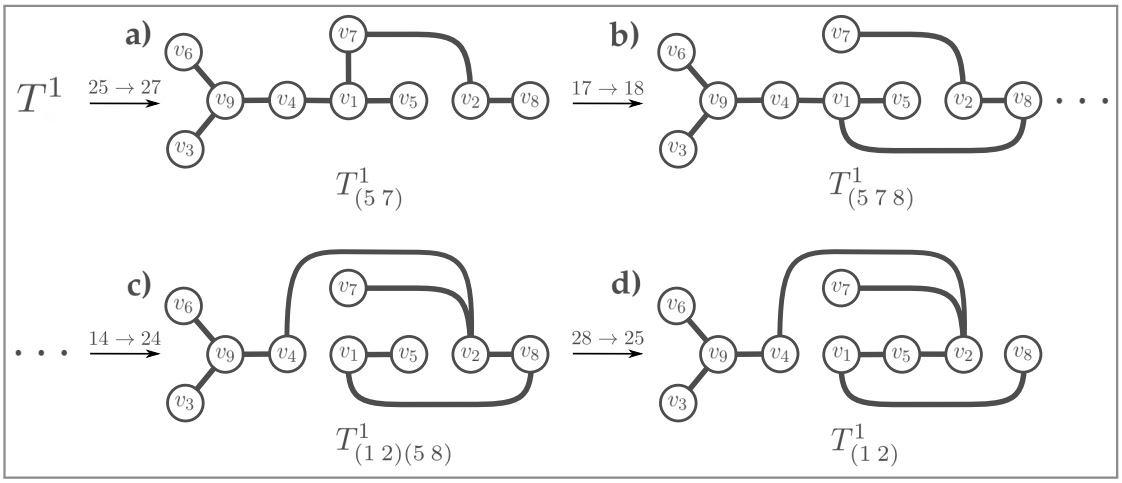

Figura 5.11: Aquí se muestra una sucesión de cuatro reemplazos admisibles de aristas que nos permite obtener la $(1\,2)$-representación de T^1 en K_9 con $V(K_9) = \{v_1, ..., v_9\}$, desde T^1 mismo. Primero obtenemos **a)** $T^1_{(5\,7)}$ a partir de T^1 por el reemplazo $25 \to 27$. Luego se obtiene **b)** $T^1_{(5\,7\,8)}$ de $T^1_{(5\,7)}$ por $17 \to 18$. Después tenemos **c)** $T^1_{(1\,2)(5\,8)}$ de $T^1_{(5\,7\,8)}$ por medio de $14 \to 24$, y finalmente **d)** $T^1_{(1\,2)}$ de $T^1_{(1\,2)(5\,8)}$ por $28 \to 25$.

Luego, ya que es posible obtener $T^1_{(1\,2)}$ desde T^1 por una sucesión de reemplazos admisibles de aristas, por el **Corolario 3.5.1** se sigue $(1\,2) \in S_{T^1}$, como queríamos. Así $\{(12), (123456789)\} \subseteq S_{T^1}$ por lo que $S_9 = \langle (12), (123456789)\rangle \leq S_{T^1} \leq S_9$ y nuevamente verificamos $S_{T^1} = S_9$. Continuamos ahora con la definición general para los árboles en $\mathcal{T}$.

Definición 5.5.1. *Sea k un entero positivo. Definimos el k-ésimo árbol en la familia $\mathcal{T}$, como el árbol T^k de orden $3+6k$ con $V(T^k) = \{v_1, v_2, ..., v_{3+6k}\}$, formado por 6 trayectorias inducidas $\{A_1, A_2, ..., A_6\}$, con k vértices cada una y disjuntas entre sí, dadas por $A_{r-2} = \{v_{r+6(0)}, ..., v_{r+6(i)}, v_{r+6(i+1)}, ..., v_{r+6(k-1)}\}$ para cada $3 \leq r \leq 8$, y tal que además tenga las siguientes 8 aristas,*

$$\{v_1v_5, v_1v_{4+6(k-1)}, v_1v_{7+6(k-1)}, v_2v_{5+6(k-1)}, v_2v_{8+6(k-1)}, v_{3+6k}v_4, v_{3+6k}v_{3+6(k-1)}, v_{3+6k}v_{6+6(k-1)}\} \subseteq E(T^k).$$

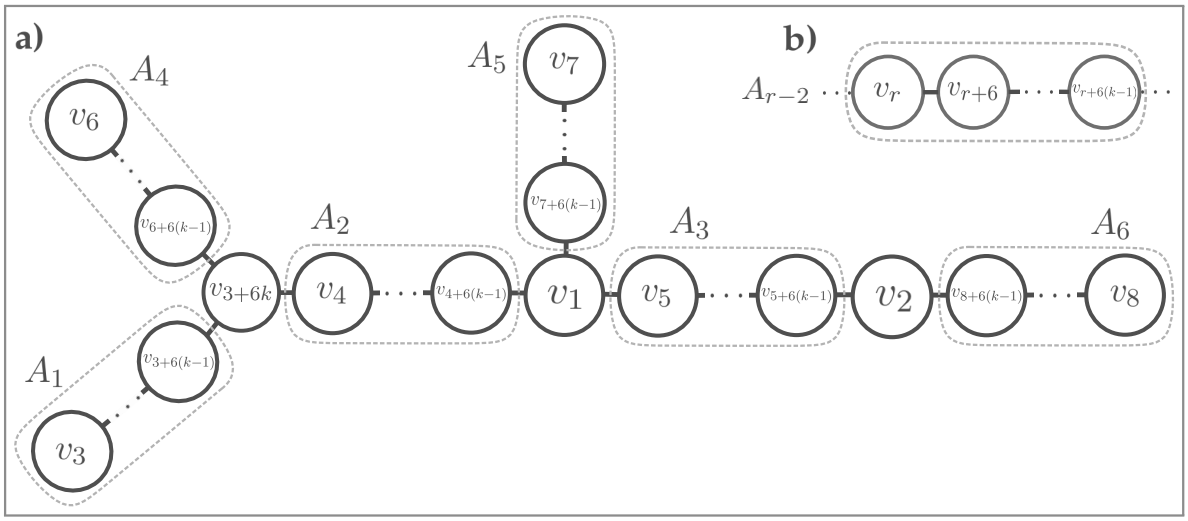

Figura 5.12: Se muestra en **a)** el k-ésimo árbol T^k en $\mathcal{T}$ y se encierran con líneas punteadas las trayectorias $\{A_1, ..., A_6\}$. En **b)** se muestra la numeración general que tienen los vértices en las trayectorias $\{A_1, ..., A_6\}$.

Con base en esto estudiaremos ahora el caso T^2, y con ello veremos la utilidad de la numeración escogida para los vértices de estos árboles. Como se mencionó antes, las propiedades de T^2 difieren ligeramente de las de T^1, y como tal, este caso nos será de utilidad para motivar algunos resultados generales, que finalmente usaremos para mostrar las propiedades requeridas de todos los árboles en $\mathcal{T}$. Siguiendo la **Definición 5.5.1** podemos visualizar el caso T^2 como en la **Figura 5.13**, donde se considera nuevamente el reemplazo admisible de aristas $15 \to 23$ pero ahora sobre T^2.

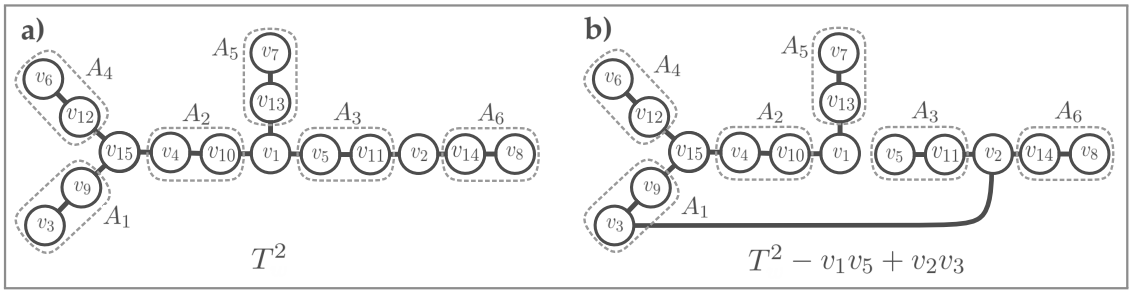

Figura 5.13: Se muestran **a)** el árbol T^2 y **b)** $T^2 - v_1v_5 + v_2v_3$. En ambos casos se encierran con líneas punteadas las trayectorias $\{A_1, ..., A_6\}$. Recuérdese que el reemplazo $15 \to 23$, pero hecho sobre T^1, fue el mismo reemplazo que nos permitió deducir las propiedades de T^1.

De nueva cuenta veamos primero que $15 \to 23$ es un reemplazo raro de T^2. Para ello nótese que la arista $v_2v_3 \in E(T^2 - v_1v_5 + v_2v_3)$ será mapeada a $v_4v_{15} \in E(T^2)$ por todo isomorfismo entre estos árboles, ya que de forma similar al caso T^1, los vértices $v_2 \in V(T^2 - v_1v_5 + v_2v_3)$ y $v_{15} \in V(T^2)$ son, cada uno en su respectivo árbol, los únicos vértices de grado 3 incidentes con los vértices finales de 2 de las trayectorias $\{A_1, ..., A_6\}$, que al mismo tiempo tienen vértices iniciales de grado 1, y en consecuencia por la **Observación 5.2.1** se concluye que $15 \to 23$ es un reemplazo raro de T^2.

Por otro lado veamos que el ciclo de todos los índices $(1\ 2\ 3\ \cdots\ 14\ \ 15)$, acomodados siguiendo su orden natural, también pertenece al grupo de alcanzabilidad S_{T^2} de T^2. Para ello debemos notar que $(1\ \ 2\ \ 15)(3\ 4\ 5\ 6\ 7\ 8)(9\ \ 10\ \ 11\ \ 12\ \ 13\ \ 14) \in S_{T^2}(15 \to 23)$, y entonces esta permutación pertenece a S_{T^2}. De aquí la importancia de la numeración escogida, ya que esta nos permite obtener una permutación que preserva el orden natural entre los índices, al permutar cíclicamente los vértices *fijos*, es decir v_1, v_2 y v_{15} entre sí, y al mismo tiempo permutar en ciclos disjuntos a los vértices en las trayectorias $\{A_1, ..., A_6\}$, dependiendo de su cercanía con el vértice inicial de cada trayectoria.

Ahora hace falta mostrar que se tiene $(1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10\ \ 11\ \ 12\ \ 13\ \ 14\ \ 15) \in S_{T^2}$. Para este fin veremos que $(2\ 8) \in S_{T^2}$ y $(8\ 14) \in S_{T^2}$. En efecto, de la **Figura 5.13** vemos que se pueden obtener dichas transposiciones asociadas a reemplazos admisibles de aristas de T^2, específicamente $(2\ 8) \in S_{T^2}(2\ \ 11 \to 8\ \ 11)$ y a la vez $(8\ \ 14) \in S_{T^2}(2\ \ 14 \to 2\ \ 8)$. Luego, usando estas tenemos,

$$[(1\ \ 2\ \ 15)(3\ 4\ 5\ 6\ 7\ 8)(9\ \ 10\ \ 11\ \ 12\ \ 13\ \ 14)](2\ 8) = (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ \ 15)(9\ \ 10\ \ 11\ \ 12\ \ 13\ \ 14) \in S_{T^2},$$

y en consecuencia,

$$[(1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ \ 15)(9\ \ 10\ \ 11\ \ 12\ \ 13\ \ 14)](8\ \ 14) = (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ \ 10\ \ 11\ \ 12\ \ 13\ \ 14\ \ 15) \in S_{T^2}.$$

Igualmente, podemos obtener $(1\ 2) \in S_{T^2}$, por medio de una sucesión análoga a la sucesión de los 4 reemplazos admisibles de aristas para T^1 mostrada en la **Figura 5.11**. No obstante, en este caso deberemos realizar otros 3 reemplazos que nos ayudarán a *reordenar* 3 trayectorias $\{A_1, ..., A_6\}$, que son *invertidas* por los primeros 4 reemplazos. Más aún, esta nueva sucesión de 7 reemplazos admisibles de aristas servirá también para todo T^k con $k \geq 2$, por lo que en la **Figura 5.14** se muestra el caso general haciendo referencia únicamente a los vértices inicial y final de cada trayectoria.

Luego, al obtener $T^2_{(1\ 2)}$ de T^2 por una sucesión de reemplazos admisibles de aristas, nuevamente por el **Corolario 3.5.1** deducimos que $(1\ 2) \in S_{T^2}$ y así $\{(1\ 2), (1\ 2\ 3\ \cdots\ 14\ \ 15)\} \subseteq S_{T^2}$, por lo que $S_{15} = \langle (1\ 2), (1\ 2\ 3\ \cdots\ 14\ \ 15)\rangle \leq S_{T^2} \leq S_{15}$, es decir, $S_{T^2} = S_{15}$, y por lo tanto T^2 es una árbol raro que además es amoeba local, mientras que por el **Corolario 4.1.2** podemos concluir que T^2 es también una amoeba global. Veamos ahora el primer resultado general para la familia $\mathcal{T}$.

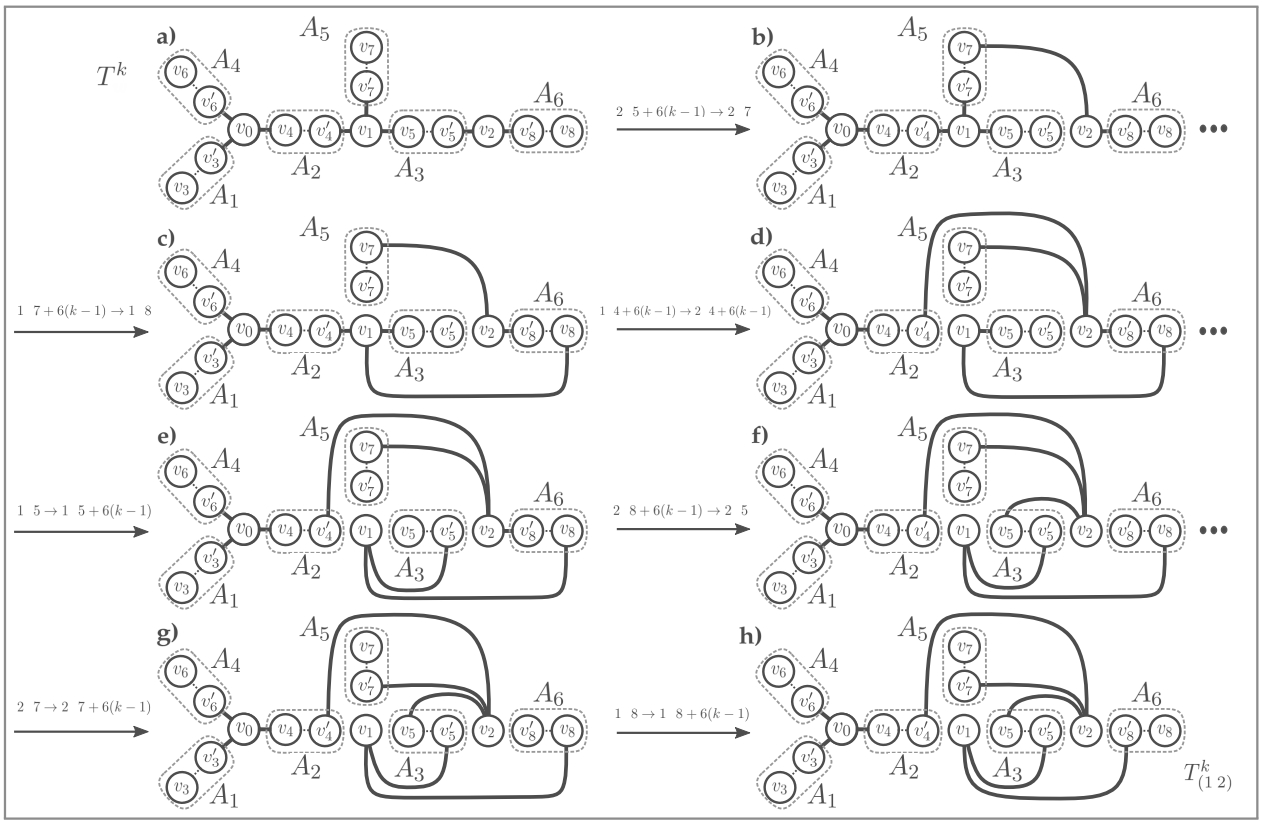

Figura 5.14: Se muestra en **a)** el árbol T^k para $k \geq 2$, y en **h)** la $(1\,2)$-representación $T^k_{(1\,2)}$ de T^k. Para poder apreciar correctamente la estructura de estos árboles, en este dibujo se denota por v_0 al vértice v_{3+6k} y por v'_i a $v_{i+6(k-1)}$ para $3 \leq i \leq 8$. Sin embargo sobre las flechas se muestran reemplazos admisibles escritos con los índices originalmente asignados a cada vértice. De aquí vemos que $T^k_{(1\,2)}$ se puede obtener desde T^k por una sucesión de 7 reemplazos admisibles de aristas similar a la de la **Figura 5.11**. Sin embargo, en este caso, de **d)** a **e)** se *invierte* la trayectoria A_3, de **f)** a **g)** se invierte A_5 y de **g)** a **h)** se invierte A_6 para obtener $T^k{}_{(1\,2)}$.

Lema 5.5.1. *Para todo árbol T^k en $\mathcal{T}$ se tiene $(1\,2) \in S_{T^k}$.*

Demostración. Para $k = 1$, la **Figura 5.11** muestra una sucesión de 4 reemplazos admisibles de aristas que nos permiten obtener $T^1{}_{(1\,2)}$ a partir de T^1. De forma similar, para $k \geq 2$ se exhibe en la **Figura 5.14** una sucesión de 7 reemplazos admisibles con la que se puede obtener $T^k{}_{(1\,2)}$ desde T^k. Así, en todo caso se sigue que $(1\,2) \in S_{T^k}$ debido al **Corolario 3.5.1**. ■

Para poder deducir más atributos al respecto de la familia $\mathcal{T}$, demostraremos primero la **Proposición 5.5.1** y su corolario. Es importante notar que estos resultados no sólo corresponden a los árboles en $\mathcal{T}$, sino que son aplicables a toda gráfica G que contenga una trayectoria *pendiente* o *colgante* con dos o más vértices, es decir, cuyo vértice inicial tenga grado 1 y tal que todos los demás vértices en ella no tengan -excepto por su vértice final- vecinos fuera de la trayectoria.

Debemos mencionar que la prueba de la **Proposición 5.5.1** se realiza por inducción y por ello en la **Figura 5.15** se ilustran sus casos base. Es necesario añadir que esta y su corolario forman parte de las aportaciones originales de esta tesis, y en particular se incluyen en esta sección ya que enseguida nos apoyaremos en ellos para formular otro lema para tratar con los árboles en la familia $\mathcal{T}$.

Proposición 5.5.1. *Sea G una gráfica de orden $n \geq 3$ con $V(G) = \{v_1, ..., v_n\}$ y sea S_G el grupo de alcanzabilidad de G. Sea además P_r con $n > r \geq 2$ una trayectoria inducida de G, cuyos vértices están enumerados sin pérdida de generalidad como $P_r = \{v_1, v_2, ..., v_{r-1}, v_r\}$, tal que $d_G(v_1) = 1$ y $d_G(v_i) = 2$ para toda $i \in \{2, ..., r\}$. Entonces se tiene $\{(1\ 2), (1\ 3), ..., (1\ \ r-1), (1\ r)\} \subseteq S_G$.*

Demostración. Procedemos por inducción sobre la cantidad de vértices $r \geq 2$ en P_r. Nótese que v_r en P_r debe ser adyacente con un único vértice v_b fuera de P_r, además de ser adyacente con v_{r-1} en P_r. Los casos $r = 2$ y $r = 3$ son inmediatos, como mostrado en la **Figura 5.15**. Consideremos el caso $r = 4$ para analizar casos no triviales. Aquí $\{(1\ 2), (1\ 3)\} \subseteq S_G$, pero $(1\ 4)(2\ 3) \in S_G(4b \to 1b) \subseteq S_G$, ya que el reemplazo $4b \to 1b$ intercambia por pares a los vértices en P_r. No obstante, por el **Teorema 2.2.3** sabemos que $S_3 = \langle (1\ 2), (1\ 3) \rangle$ y por la **Proposición 2.2.5** tenemos $S_3 = \langle (1\ 2), (1\ 3) \rangle \leq S_G$. Luego $(2\ 3) \in S_G$ y consecuentemente $(1\ 4) = [(1\ 4)(2\ 3)](2\ 3) \in S_G$ como se requiere.

Ahora supóngase que el resultado es cierto para $r \geq 4$ arbitraria, y mostrémoslo para una trayectoria inducida P_{r+1} de G satisfaciendo las hipótesis de la proposición, cuyos vértices estén enumerados como $\{v_1, ..., v_r, v_{r+1}\}$ y tal que v_{r+1} es adyacente a un único vértice v_b fuera de P_{r+1}. Considérese primero $r + 1$ par. En este caso, para el reemplazo admisible de $r + 1\ \ b \to 1b$ se tiene,

$$\alpha = (1\ \ r+1)(2\ \ r) \cdots (\tfrac{r+1}{2}\ \ \ \tfrac{r+1}{2} + 1) \in S_G(r + 1b \to 1b) \subseteq S_G,$$

ya que esto intercambia los vértices en P_{r+1} de dos en dos. Por otro lado, si $r + 1$ es impar entonces,

$$\alpha = (1\ \ r+1)(2\ \ r) \cdots (\tfrac{r}{2}\ \ \ \tfrac{r}{2} + 2) \in S_G(r + 1b \to 1b) \subseteq S_G,$$

ya que este reemplazo intercambia todos los vértices de P_{r+1} de dos en dos, excepto por el vértice central de P_{r+1}. Pero la trayectoria $P_r = G[\{v_1, ..., v_r\}]$ cumple las mismas condiciones que P_{r+1} en G, ya que v_r es adyacente fuera de P_r sólo con v_{r+1}. De esta forma $\{(1\ 2), (1\ 3), ..., (1\ r)\} \subseteq S_G$ por hipótesis de inducción. Luego, por el **Teorema 2.2.3** sabemos que $S_r = \langle (1\,2), (1\,3), ..., (1\,r-1), (1\,r) \rangle$ y por la **Proposición 2.2.5** se sigue que $S_r = \langle (1\ 2), (1\ 3), ..., (1\ \ r-1), (1\ r) \rangle \leq S_G$. De aquí que,

$$\beta = (2\ \ r) \cdots (\tfrac{r+1}{2}\ \ \ \tfrac{r+1}{2} + 1) \in S_G \text{ si } r + 1 \text{ es par, o bien}$$
$$\beta = (2\ \ r) \cdots (\tfrac{r}{2}\ \ \ \tfrac{r}{2} + 2) \in S_G \text{ si } r + 1 \text{ es impar,}$$

y por ello en ambos casos se obtiene $(1\ \ r + 1) = \alpha\beta \in S_G$, con lo que se concluye la inducción. ■

Corolario 5.5.1. *Sea G una gráfica de orden $n \geq 3$ con $V(G) = \{v_1, ..., v_n\}$ y sea S_G el grupo de alcanzabilidad de G. Sea P_r con $n > r \geq 2$ una trayectoria inducida de G, cuyos vértices están enumerados sin pérdida de generalidad como $P_r = \{v_1, v_2, ..., v_{r-1}, v_r\}$, tal que $d_G(v_1) = 1$ y $d_G(v_i) = 2$ para toda $i \in \{2, ..., r\}$. Para el grupo S_G se cumple $S_r \leq S_G$.*

Demostración. Por la **Proposición 5.5.1** se tiene $\{(1\ 2), (1\ 3), ..., (1\ \ r-1), (1\ r)\} \subseteq S_G$. Pero por el **Teorema 2.2.3** sabemos que $S_r = \langle (1\ 2), (1\ 3), ..., (1\ \ r-1), (1\ r) \rangle$. Por lo tanto, con base en la **Proposición 2.2.5** podemos poner en general $S_r = \langle (1\ 2), (1\ 3), ..., (1\ \ r-1), (1\ r) \rangle \leq S_G$. ■

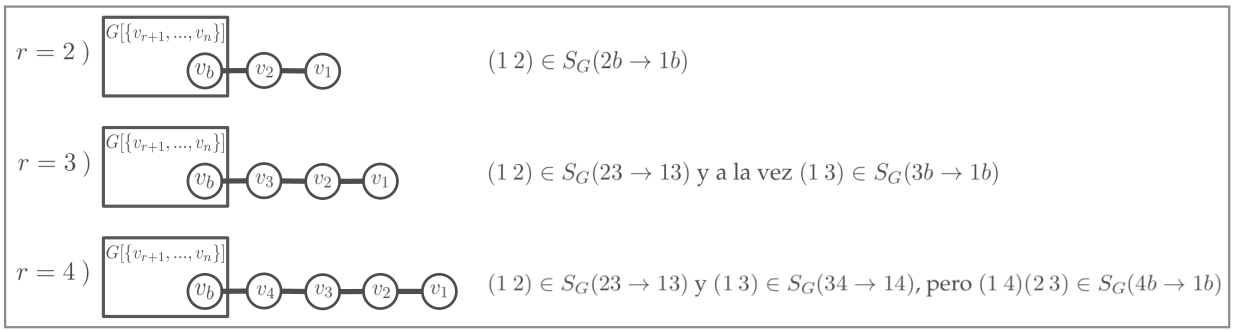

Figura 5.15: Una gráfica G, con una trayectoria *pendiente* o *colgante* formada por $r \geq 2$ vértices. Se muestran los casos base $r \in \{2, 3, 4\}$ para la inducción en la **Proposición 5.5.1**. El resultado de dicha proposición para $r \in \{2, 3\}$ es inmediato, mientras que $r = 4$ conlleva otras consideraciones.

Con base en estos resultados podemos deducir un último lema que será necesario para analizar a la familia de árboles $\mathcal{T}$. Inmediatamente después concluimos mostrando lo requerido para $\mathcal{T}$.

Lema 5.5.2. *Para todo T^k en $\mathcal{T}$ se tiene las transposiciones $\{(2\,8)\} \cup \{(8+6r\ \ 8+6(r+1))\}_{r=0}^{r=k-2} \subseteq S_{T^k}$.*

Demostración. Se sigue inmediatamente del **Corolario 5.5.1** al renombrar a los vértices de la trayectoria $T^k[A_6 \cup \{v_2\}] = \{v_{8+6(0)}, v_{8+6(1)}, v_{8+6(2)}, ..., v_{8+6(k-2)}, v_{8+6(k-1)}, v_2\}$ con $k+1$ vértices, que es pendiente o colgante ya que el vértice v_2 siempre es adyacente a un único vértice fuera de ella. ■

Teorema 5.5.1. *Todo árbol T^k en la familia $\mathcal{T}$ es un árbol raro y a la vez es un árbol amoeba local y global.*

Demostración. Ya mostramos en particular estas propiedades para los árboles T^1 y T^2. Escojamos ahora cualquier $k \geq 3$ y probemos que T^k es un árbol raro y que además es una amoeba local, siendo que de esto último se sigue que T^k es también una amoeba global.

Para ver que T^k es raro considérese la **Figura 5.16**, donde se muestra el reemplazo admisible de aristas $15 \to 23$ hecho sobre T^k. Nótese que el vértice $v_2 \in V(T^k - v_1v_5 + v_2v_3)$ y el vértice $v_{3+6k} \in V(T^k)$ son, en sus respectivos árboles, los únicos vértices de grado 3 adyacentes con los vértices finales de dos de las trayectorias $\{A_1, ..., A_6\}$, que al mismo tiempo tienen vértices iniciales de grado 1. De aquí que se tenga $\varphi(v_2) = v_{3+6k}$ para cualquier isomorfismo φ de $T^k - v_1v_5 + v_2v_3$ en T^k. Pero de esta forma los extremos de la arista $v_2v_3 \in E(T^k - v_1v_5 + v_2v_3)$ son mapeados a los extremos de $v_{3+6k}v_4 \in E(T^k)$ por todo isomorfismo entre estos árboles, y en base a la **Observación 5.2.1** podemos concluir que $15 \to 23$ es un reemplazo raro de T^k, es decir, T^k es un árbol raro.

Por otro lado, para ver que T^k es una amoeba local, probaremos que para el grupo de alcanzabilidad S_{T^k} de T^k se tiene $S_n = \langle (1\ 2), (1\ 2\ \cdots\ 3+6k) \rangle \leq S_{T^k} \leq S_n$. Por el **Lema 5.5.1** tenemos $(1\,2) \in S_{T^k}$ y sólo hace falta construir el ciclo de todos los índices $(1\ \ 2\ \cdots\ \ 3+6k)$ en este mismo grupo. Para ello nótese que el mismo reemplazo raro $15 \to 23$ tiene asociado el producto de ciclos disjuntos,

$$\alpha = (1\ \ 2\quad 3+6k)\beta_0\beta_1\beta_2\beta_3\cdots\beta_{k-2}\beta_{k-1}, \text{ donde se denota}$$
$$\beta_q = (3+6q\quad 4+6q\quad 5+6q\quad 6+6q\quad 7+6q\quad 8+6q) \text{ para toda } 0 \leq q \leq k-1,$$

es decir $\alpha \in S_{T^k}(15 \to 23) \subseteq S_{T^k}$, ya que los vértices v_1, v_2 y v_{3+6k} son intercambiados entre sí bajo este reemplazo, mientras que los vértices en las trayectorias $\{A_1, ..., A_6\}$ se pueden intercambiar en ciclos de longitud 6, disjuntos dos a dos, dependiendo de su cercanía con el vértice inicial de cada trayectoria y siguiendo el orden natural de sus índices. Pero por el **Lema 5.5.2** también contamos con las transposiciones $\{(2\,8)\} \cup \{(8+6r\ \ 8+6(r+1))\}_{r=0}^{r=k-2} \subseteq S_{T^k}$ y como $\beta_0 = (3\cdots 8)$ tenemos,

$$\alpha(2\,8) = [(1\ \ 2\quad 3+6k)(3\cdots 8)\beta_1\beta_2\beta_3\cdots\beta_{k-1}](2\,8) = (1\,2\cdots 8\quad 3+6k)\beta_1\beta_2\beta_3\cdots\beta_{k-1} \in S_{T^k}.$$

Pero ya que $\beta_1 = (9\cdots 14)$ y tomando en cuenta $(8+6(0)\ \ 8+6(1)) = (8\ \ 14)$ entonces,

$$[\alpha(2\,8)](8\ \ 14) = [(1\,2\cdots 8\quad 3+6k)(9\cdots 14)\beta_2\beta_3\cdots\beta_{k-1}](8\ \ 14) = (1\,2\cdots 14\quad 3+6k)\beta_2\beta_3\cdots\beta_{k-1} \in S_{T^k},$$

y como $\beta_2 = (15\cdots 20)$ y considerando que $(8+6(1)\ \ 8+6(2)) = (14\ \ 20)$, de igual forma,

$$[\alpha(2\,8)(8\ \ 14)](14\ \ 20) = [(1\,2\cdots 14\ \ 3+6k)(15\cdots 20)\beta_3\cdots\beta_{k-1}](14\ \ 20) = (1\,2\cdots 20\ \ 3+6k)\beta_3\cdots\beta_{k-1} \in S_{T^k}.$$

Pero continuando de esta forma, en general tenemos

$$[\alpha(2\,8)(8\ \ 14)(14\ \ 20)\cdots(8+6(r-1)\ \ 8+6r)](8+6r\ \ 8+6(r+1)) =$$
$$[(1\,2\cdots 8+6r\quad 3+6k)(3+6(r+1)\ \cdots\ 8+6(r+1))\beta_{r+2}\cdots\beta_{k-1}](8+6r\ \ 8+6(r+1)) =$$
$$(1\,2\cdots 8+6(r+1)\quad 3+6k)\beta_{r+2}\cdots\beta_{k-1} \in S_{T^k},$$

donde se debe notar que $3+6(r+1) = 9+6r = (8+6r)+1$ y por ende $(1\,2\cdots 8+6(r+1)\ \ 3+6k)$ es el ciclo formado por elementos consecutivos $\{1\,2\cdots 8+6(r+1)\}$, incluyendo al $3+6k$. Así, finalmente,

$$[\alpha(2\,8)(8\ 14)(14\ 20)\cdots(8+6(k-3)\ \ 8+6(k-2))](8+6(k-2)\ \ 8+6(k-1)) = (1\,2\cdots 8+6(k-1)\ \ 3+6k) \in S_{T^k},$$

donde nuevamente es importante notar que $8 + 6(k-1) = 2 + 6k = (3 + 6k) - 1$, es decir que el ciclo de todos los elementos consecutivos $(1\ 2 \cdots 3 + 6k)$ también pertenece a S_{T^k}, como queríamos. De esta forma $\{(1\ 2), (1\ 2 \cdots 3 + 6k)\} \subseteq S_{T^k}$ y por la **Proposición 2.2.5** vemos que $S_{3+6k} = \langle (1\ \ 2), (1\ \ 2\ \cdots\ \ 3 + 6k) \rangle \leq S_{T^k} \leq S_{3+6k}$, de donde $S_{T^k} = S_{3+6k}$, es decir, T^k es una amoeba local, mientras que por el **Corolario 4.1.2** deducimos que T^k es también una amoeba global. Finalmente, como T^k fue arbitrario, estos resultados se siguen para todo árbol en la familia $\mathcal{T}$. ■

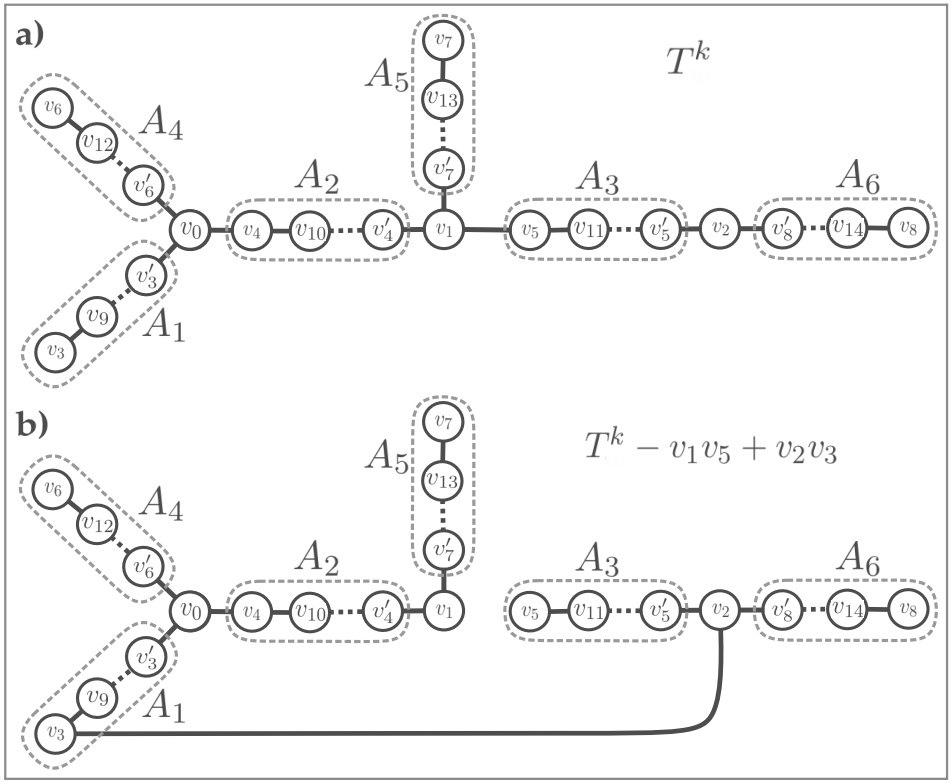

Figura 5.16: Se muestran **a)** T^k para $k \geq 3$ y **b)** $T^k - v_1v_5 + v_2v_3$. Para poder apreciar la estructura de estos árboles, aquí se denota por v_0 a v_{3+6k} y por v'_i a $v_{i+6(k-1)}$ para $3 \leq i \leq 8$. Finalmente, debe notarse que los vértices v_2 y v_3 estarían fuera de las órbitas naturales de los vértices v_1 y v_5 en la gráfica $T^k - v_1v_5$, por lo que el reemplazo $15 \to 23$ de T^k tampoco se podría detectar con la estrategia discutida en la sección anterior.

Capítulo 6

Principal Contribución Computacional de esta Tesis: Algoritmos para Detectar Amoebas

Aquí desarrollaremos, basándonos en la teoría presentada anteriormente, varios algoritmos para la detección y estudio de las amoebas. Retomaremos en la primera sección la discusión sobre la modificación a la estrategia, inicialmente planteada en el capítulo 5, para escoger la arista $v_kv_l \in E(\overline{G})$ durante la búsqueda de reemplazos admisibles y no triviales $rs \to kl$ de una gráfica G. De esto se seguirá un algoritmo general para resolver el **Problema de Decisión de las Amoebas Locales (PDLA)**, visto en el capítulo 4, que sentará las bases para los algoritmos en las demás secciones. Debemos mencionar que todo lo incluido en este capítulo es material original de esta tesis, y por lo tanto presentamos en cada caso una discusión sobre el tiempo de ejecución de cada uno de los algoritmos, así como de los aspectos teóricos que garantizan su correcto funcionamiento.

6.1. Estrategia de Optimización por los Grados de Vértices en el Reemplazo

Determinar si una gráfica no vacía G de orden n es amoeba local, conlleva evaluar que para su grupo de alcanzabilidad S_G se cumpla $|S_G| = n!$. Para ello debemos obtener todas las permutaciones asociadas a reemplazos admisibles de aristas de G, lo que por supuesto requiere identificar primero a dichos reemplazos admisibles. Ahora bien, nótese que para todo reemplazo trivial $rs \to kl$ de G, digamos con $r = k$ y $s = l$, se tiene $S_G(rs \to rs) = \{\alpha \in S_n \mid G_\alpha = G\}$, por lo que nuestros algoritmos deberán obtener este conjunto una única vez. Más aún, por la **Observación 3.5.1** sabemos que todas estas permutaciones se corresponden una a una con los automorfismos de G, lo que será de utilidad junto con la representación computacional que daremos a G en el capítulo 7.

Por otro lado, para determinar si un reemplazo arbitrario y no trivial de aristas $rs \to kl$, es en efecto un reemplazo admisible de aristas de G, debemos evaluar la existencia de un isomorfismo de la gráfica $G - v_rv_s + v_kv_l$ en G. No obstante, como planteado en el capítulo 5, en la **Sección 5.4**, antes de invertir recursos computacionales en evaluar dicho isomorfismo para todas y cada una de las aristas $v_kv_l \in E(\overline{G})$, es deseable contar con una estrategia para elegir la nueva arista v_kv_l de una manera más precavida, es decir, buscamos reservar la evaluación de dichos isomorfismos únicamente para aristas v_kv_l en un subconjunto más apropiado de $E(\overline{G})$.

Consideremos una arista $v_rv_s \in E(G)$ y considérese el subconjunto $\mathcal{A}_G(v_rv_s) \subseteq E(\overline{G})$ dado por $\mathcal{A}_G(v_rv_s) = \{v_kv_l \mid G - v_rv_s + v_kv_l \simeq G\}$, esto es, de aristas $v_kv_l \in E(\overline{G})$ que hacen que $rs \to kl$ sea admisible en G. Con esta notación, lo que la estrategia discutida hasta ahora plantea, es en realidad poder determinar si existe un subconjunto $\mathcal{P}_G(v_rv_s) \subseteq E(\overline{G})$, construible o identificable en tiempo polinomial (ver **Sección 2.4**), que sea lo más parecido posible a $\mathcal{A}_G(v_rv_s)$, i.e., tal que se tenga

$$\mathcal{A}_G(v_rv_s) \subseteq \mathcal{P}_G(v_rv_s) \subseteq E(\overline{G})$$

Como parte del trabajo mostrado en el capítulo 5, inicialmente se propuso tomar los extremos de $v_k v_l$ dentro de las órbitas naturales $Aut(G - v_r v_s)[v_r]$ y $Aut(G - v_r v_s)[v_s]$, de los vértices v_r y v_s en $G - v_r v_s$ (ver **Definiciones 2.3.4** y **2.3.5**). Si bien es cierto que ejemplos como la **Figura 5.7**, dejan ver que no todo reemplazo de esta forma es admisible, se había considerado en un inicio que todo reemplazo admisible sí verificara estas condiciones. Sin embargo, se encontraron ejemplos de reemplazos admisibles que no cumplen con esta propiedad, específicamente, aquellos que llamamos reemplazos raros (ver **Definición 5.2.1**). Luego, dada $v_r v_s \in E(G)$ y retomando al conjunto $\mathcal{A}_G(v_r v_s)$ de antes, vemos que para el subconjunto $O_G(v_r v_s) \subseteq \mathcal{A}_G(v_r v_s)$ de dichos pares de vértices, i.e., dado por $O_G(v_r v_s) = \{v_k v_l \mid v_k \in Aut(G - v_r v_s)[v_r] \text{ y } v_l \in Aut(G - v_r v_s)[v_s]\}$, los resultados del capítulo 5 indican que existen gráficas G con $O_G(v_r v_s) \subsetneq \mathcal{A}_G(v_r v_s) \subseteq E(\overline{G})$, y por ende no podemos usar $O_G(v_r v_s)$ para identificar todos los reemplazos admisibles de una gráfica arbitraria G.

No obstante, como indicado al final de la **Sección 5.4**, una alternativa al uso de $O_G(v_r v_s)$, conlleva ahora escoger los extremos de $v_k v_l$ entre los vértices de la gráfica $G - v_r v_s$, tales que en conjunto tengan los mismos grados que v_r y v_s. Esto se debe en particular a que por la **Proposición 2.3.3**, sabemos que todos los vértices en $Aut(G - v_r v_s)[v_r]$ y en $Aut(G - v_r v_s)[v_s]$, tienen respectivamente, el mismo grado que v_r y que v_s. De esta forma, para el subconjunto $\mathcal{F}_G(v_r v_s) \subseteq E(\overline{G})$, dado por $\mathcal{F}_G(v_r v_s) = \{v_k v_l \mid \{d_F(v_k), d_F(v_l)\} = \{d_F(v_r), d_F(v_s)\} \text{ con } F = G - v_r v_s\}$, por lo mencionado antes se cumple $O_G(v_r v_s) \subseteq \mathcal{F}_G(v_r v_s)$. Más aún, si $rs \to kl$ es admisible en G, sabemos que para los vértices v_r, v_s, v_k y v_l se cumplen las condiciones en la **Proposición 4.3.1**. Pero el **Corolario 4.3.1** nos dice que satisfacer dichas condiciones, es equivalente a tener $\{d_F(v_k), d_F(v_l)\} = \{d_F(v_r), d_F(v_s)\}$ en la gráfica $F = G - v_r v_s$. Así, todo lo comentado antes implica que en general se tiene

$$O_G(v_r v_s) \subseteq \mathcal{A}_G(v_r v_s) \subseteq \mathcal{F}_G(v_r v_s) \subseteq E(\overline{G}),$$

es decir que estos conjuntos nos permiten hasta ahora, restringir o aproximar $\mathcal{A}_G(v_r v_s)$ para una gráfica G arbitraria. Sin embargo, el mismo caso de la **Figura 5.7** muestra que existen gráficas G para las que se tiene $\mathcal{A}_G(v_r v_s) \subsetneq \mathcal{F}_G(v_r v_s)$. Debido a todo esto, la estrategia discutida en un inicio puede replantearse para determinar si existe un subconjunto $\mathcal{P}_G(v_r v_s) \subseteq E(\overline{G})$, nuevamente identificable en tiempo polinomial, pero tal que para toda gráfica G se cumpla ahora

$$O_G(v_r v_s) \subseteq \mathcal{A}_G(v_r v_s) \subseteq \mathcal{P}_G(v_r v_s) \subsetneq \mathcal{F}_G(v_r v_s) \subseteq E(\overline{G})$$

A pesar del trabajo desarrollado como parte de esta tesis, no se ha podido encontrar una solución a esta nueva pregunta. Por ello, determinar el conjunto $\mathcal{F}_G(v_r v_s)$ comprende actualmente la mejor estrategia para determinar los reemplazos admisibles de aristas de una gráfica arbitraria G. Mostraremos en la siguiente sección un algoritmo, que se sigue directamente de la **Proposición 4.3.1**, útil para identificar $\mathcal{F}_G(v_r v_s)$, y veremos además que este algoritmo corre en tiempo constante.

6.2. Un Algoritmo para resolver el PDLA

El siguiente algoritmo revisa que las condiciones en la **Proposición 4.3.1** se cumplan para un reemplazo de aristas arbitrario y no trivial $rs \to kl$ respecto a una gráfica G. Nótese que de no cumplirse la correspondiente condición, la arista v_kv_l no puede pertenecer al conjunto $\mathcal{F}_G(v_rv_s)$ dado antes debido al **Corolario 4.3.1**, y por ende $rs \to kl$ no puede ser un reemplazo admisible de G.

Algoritmo 11: CondicionesDeGrados(G, v_rv_s, v_kv_l)

Entrada: una gráfica G con $V(G) = \{v_1, ..., v_n\}$, una arista $v_rv_s \in E(G)$ y una arista $v_kv_l \in E(\overline{G})$.
Respuesta: valor booleano indicando si $rs \to kl$ cumple o no en G, las condiciones de la **Proposición 4.3.1**.

```
1  inicializar variable booleana condición ↤ False
2  if ( |{r, s} ∩ {k, l}| = 0 ) then
3  |   if ( {d_G(v_k) + 1, d_G(v_l) + 1} = {d_G(v_r), d_G(v_s)} ) then
4  |   |   condición ↤ True
5  |   end
6  end
7  if ( |{r, s} ∩ {k, l}| = 1 ) then
8  |   obtener único vértice v_c conservado en el reemplazo, es decir v_c ∈ {v_r, v_s} ∩ {v_k, v_l}
9  |   obtener único v_a ∈ {v_r, v_s} \ {v_c}
10 |   obtener único v_b ∈ {v_k, v_l} \ {v_c}
11 |   if ( d_G(v_b) + 1 = d_G(v_a) ) then
12 |   |   condición ↤ True
13 |   end
14 end
15 Devolver ( condición )
```

Ejemplificaremos ahora por medio del **Algoritmo 11**, la manera en la que se analizará el tiempo de ejecución y se justificará el funcionamiento de los algoritmos a desarrollar más adelante.

Proposición 6.2.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$. Sea $rs \to kl$ un reemplazo arbitrario y no trivial de aristas de G. El* **Algoritmo 11** *determina correctamente y en tiempo constante si $rs \to kl$ satisface o no las condiciones en la* **Proposición 4.3.1** *respecto a G.*

Demostración. Sólo existen 3 posibilidades para $rs \to kl$ derivadas de la **Proposición 4.3.1**. Tener $|\{r, s\} \cap \{k, l\}| = 2$ no se puede ya que $rs \to kl$ es no trivial. Sin embargo, es inmediato que el algoritmo realizará la mayor cantidad de operaciones cuando se tenga $|\{r, s\} \cap \{k, l\}| = 1$. En efecto, en la línea 1 se inicializa la variable **condición**. Luego, se evalúa en la linea 2 si $|\{r, s\} \cap \{k, l\}| = 0$, que no es el caso y entonces se continúa a la línea 7. Ya que la condición ahí es cierta, se obtienen los vértices v_c, v_a y v_b y se evalúa para ellos el inciso $ii)$ de la **Proposición 4.3.1**, modificándose, de ser caso, el valor de la variable **condición**. Así, el algoritmo terminará devolviendo el valor True en dicho caso. Pero esto llevó un total de 9 operaciones. Más aún, ya que de forma análoga podemos mostrar el correcto funcionamiento respecto a los demás casos, así como probar que estos realizan menos de 9 operaciones, vemos que este tiene un tiempo de ejecución $T(n) = 9$ en el peor caso, por lo que $T(n) = O(9)$, y finalmente con base en la **Observación 2.4.1** concluimos $T(n) = O(1)$. ■

Por medio del **Algoritmo 11** y con base en la teoría vista hasta ahora, podemos construir un nuevo algoritmo que determina si una gráfica arbitraria G es o no, una amoeba local, i.e., un algoritmo que resuelve el PDLA. Buscando mantener la claridad en el funcionamiento de este algoritmo, lo mostramos compuesto por 2 etapas: una que analiza los reemplazos triviales de G, y otra donde se analizan los reemplazos no triviales y se obtiene el grupo de alcanzabilidad S_G de G.

Es importante remarcar que este nuevo algoritmo **manda llamar** en la línea 12, al **Algoritmo 11**, recibiendo el valor obtenido por este último en su evaluación. Tal comportamiento es ampliamente conocido y utilizado en el estudio de los algoritmos [15–18], sin embargo no ahondaremos más en sus propiedades. Recuérdese que se denota por $\leftarrow\!\!\shortmid$ al operador asignación, y nótese que se comienza con el símbolo "**;** " los comentarios en el algoritmo, es decir, texto en el pseudocódigo que no forma parte de la ejecución del algoritmo, y sólo se incluye a modo de aclaraciones y descripciones.

Algoritmo 12: IdentificarAmoebaLocal(G)

Entrada: una gráfica no vacía G de orden n con conjunto de vértices $V(G) = \{v_1, ..., v_n\}$.
Respuesta: texto indicando si G es o no una amoeba local.

```
1  inicializar conjunto vacío X_G = {}; para guardar permutaciones asociadas a reemplazos admisibles de G
2  inicializar variable booleana viable ←\| False
3  ———————————————— ETAPA 1 ————————————————
4  obtener conjunto D_G = {d_G(v_i) | v_i ∈ V(G)}, y obtener los grados máximo Δ y mínimo δ de G
5  if ( |D_G| ≠ Δ − (δ − 1) ) then
6  |  Devolver ( "G no es una amoeba local" )
7  end
8  X_G ←\| X_G ∪ {α ∈ S_n | G_α = G}
9  ———————————————— ETAPA 2 ————————————————
10 forall v_r v_s ∈ E(G) do
11 |  forall v_k v_l ∈ E(Ḡ) do
12 |  |  viable ←\| CondicionesDeGrados(G, v_r v_s, v_k v_l)
13 |  |  if ( viable ) then
14 |  |  |  if ( G − v_r v_s + v_k v_l ≃ G ) then
15 |  |  |  |  X_G ←\| X_G ∪ S_G(rs → kl)
16 |  |  |  end
17 |  |  end
18 |  end
19 end
20 obtener grupo de alcanzabilidad S_G = ⟨X_G⟩ de G
21 if ( |S_G| = n! ) then
22 |  Devolver ( "G sí es una amoeba local" )
23 end
24 Devolver ( "G no es una amoeba local" )
```

Debemos mencionar que nuestro interés en el **Algoritmo 12** no es únicamente su implementación computacional, sino que por medio de este buscamos sentar las bases para los demás algoritmos a desarrollar. Por ello incluimos a continuación una demostración de su correcto funcionamiento, junto con una discusión sobre su tiempo de ejecución. Posteriormente veremos como construir los demás algoritmos a partir de este haciendo uso, en particular, del **Corolario 4.1.1**.

Teorema 6.2.1. *El* **Algoritmo 12** *resuelve correctamente el PDLA para toda gráfica no vacía G de orden n con conjunto de vértices $V(G) = \{v_1, ..., v_n\}$.*

Demostración. Toda gráfica vacía es amoeba local por definición de S_G, razón por la que este algoritmo sólo admite gráficas no vacías. En la línea 4 se obtiene el conjunto D_G de grados de los vértices de G, además de los elementos máximo Δ y mínimo δ de dicho conjunto. Por la **Proposición 4.2.1** sabemos que si G es amoeba local, D_G debe ser igual al intervalo $\{\delta, \delta + 1, ..., \Delta\}$. Luego, si $|D_G| \neq \Delta - (\delta - 1)$, entonces el algoritmo termina, correctamente, en la línea 6 respondiendo que G no es amoeba local. En caso de continuar, aún se deben determinar todas las permutaciones asociadas a reemplazos admisibles de aristas de G. La Etapa 1 del algoritmo finaliza en la línea 8 al guardar en $\mathcal{X}_G$ todas las permutaciones asociadas a reemplazos triviales de G. Después, para cada $v_rv_s \in E(G)$ y $v_kv_l \in E(\overline{G})$ se revisan en la línea 12 las condiciones en la **Proposición 4.3.1**, reservando la evaluación del isomorfismo $G - v_rv_s + v_kv_l \simeq G$ sólo para las aristas que satisfacen dichas condiciones, i.e., sólo para cada conjunto $\mathcal{F}_G(v_rv_s)$, que sabemos contiene a las aristas $v_kv_l \in \mathcal{A}_G(v_rv_s) \subseteq E(\overline{G})$ por lo señalado al inicio de esta sección. De esta forma, al llegar a la línea 20, el algoritmo ya ha guardado en $\mathcal{X}_G$ todas las permutaciones asociadas a reemplazos admisibles triviales de G por la Etapa 1, y no triviales de G por la Etapa 2, y en consecuencia es posible obtener S_G. Finalmente, para verificar la definición de amoeba local sólo hace falta revisar si $|S_G| = n!$, respondiendo apropiadamente según sea el caso, como se hace en las líneas 21 a 24 del algoritmo. ■

Ahora bien, por la **Proposición 7.1.2** sabemos que calcular el conjunto $\{\alpha \in S_n \mid G_\alpha = G\}$ en la línea 8 y los conjuntos $S_G(rs \to kl)$ en la línea 15 del **Algoritmo 12**, son tareas equivalentes a determinar isomorfismos entre gráficas, por lo que el tiempo que toma realizarlas dependen del algoritmo utilizado para obtener dichos isomorfismos. Con tal propósito, en esta tesis se hizo uso de las ya existentes implementaciones del algoritmo **VF2** [24] para isomorfismos entre gráficas arbitrarias, y del algoritmo que llamaremos **IsoTree** [25] para evaluar isomorfismos entre árboles.

Luego, considerando un peor caso y el uso del algoritmo **VF2** para el **Algoritmo 12**, podemos estimar que este realizará $3+n+n!n$ operaciones dada una gráfica G de orden n, sólo entre las líneas 1 y 8, que son: n en la línea 4, $n!n$ en la línea 8, y 3 en las demás líneas. Después, un peor caso para los ciclos For anidados en la Etapa 2, conllevaría iterar sobre $|E(G)| = n^2/2$ y $|E(\overline{G})| = n^2/2$ pares de aristas, evaluando el isomorfismo para todos y cada uno de dichos pares, entre lo que consideramos ya la obtención de $S_G(rs \to kl)$. Así, despreciando la llamada al **Algoritmo 11** en la línea 12 (que corre en tiempo constante), vemos que en este caso se realizarían cerca de $2 + (n!n)n^4 = 2 + n!n^5$ operaciones en la Etapa 2 del algoritmo, sin considerar todavía el tiempo requerido para obtener el grupo de alcanzabilidad S_G de G. No obstante, estas consideraciones son suficientes para ver que el tiempo de ejecución $T(n)$ del **Algoritmo 12** se encuentra ya acotado inferiormente por un total de $n!n^5$ operaciones, ya que hasta ahora se tiene $T(n) \geq n!n^5 + n!n + n + 5$.

Aunado a lo anterior, para llevar a cabo la obtención de S_G, en el presente trabajo se usaron las ya existentes implementaciones del algoritmo Schreier-Sims [26] en la biblioteca **Sympy** del lenguaje Python. Sin embargo, el estudio de dicho algoritmo pertenece al área de Teoría Computacional de Grupos [27], y como tal queda fuera del alcance de esta tesis. Por esta última razón, y ya que la cota inferior $n!n^5$ dada antes para el tiempo de ejecución del **Algoritmo 12**, pertenece a las complejidades computacionales más demandantes reportadas en la literatura [15–25], en esta tesis determinaremos para el tiempo de ejecución de nuestros algoritmos únicamente cotas inferiores, buscando así mostrar la cantidad mínima de operaciones que requieren realizar para resolver sus respectivas tareas considerando instancias de peor caso. Por ello resumimos lo discutido antes en el siguiente resultado, y continuamos con la construcción de los algoritmos para detectar amoebas locales y globales basados en el **Algoritmo 12**.

Proposición 6.2.2. *El tiempo de ejecución del* **Algoritmo 12**, *dada una gráfica de orden n, está acotado inferiormente por $n!n^5$, en un peor caso y usando el algoritmo* **VF2** *para evaluar isomorfismos.* ■

6.3. Algoritmo para Detección de Amoebas entre Gráficas Arbitrarias

Debido al **Corolario 4.1.1**, podríamos usar ya el **Algoritmo 12**, no sólo para identificar amoebas locales, sino también amoebas globales, específicamente, al agregar un vértice aislado a cada gráfica G, y analizar la gráfica resultante $G\dot\cup K_1$ con este mismo algoritmo. Sin embargo, debe notarse antes que para la gráfica $G\,\dot\cup\, K_1$, digamos con un nuevo vértice aislado v_{n+1}, se cumple,

$$E(\overline{G\,\dot\cup\, K_1}) = E(\overline{G})\,\dot\cup\,\{v_iv_{n+1} \mid v_i \in V(G)\},$$

es decir que al iterar sobre todas las aristas de $\overline{G\,\dot\cup\, K_1}$, se itera ya sobre las aristas de $\overline{G}$, requeridas en la Etapa 2 del **Algoritmo 12**. Por esta razón, y con base en siguiente resultado, mostramos a continuación un algoritmo que realiza el análisis de todos los reemplazos de aristas $rs \to kl$, respecto a toda $v_rv_s \in E(G)$ y $v_kv_l \in E(\overline{G\,\dot\cup\, K_1})$, comprendiéndolo ya dentro de un mismo ciclo For anidado sin tener que repetir la evaluación del **Algoritmo 12** para $G\,\dot\cup\, K_1$.

Proposición 6.3.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $G\,\dot\cup\, K_1$ la gráfica obtenida al agregar el vértice aislado v_{n+1} a G. Sea además $rs \to kl$ un reemplazo de aristas de $G\,\dot\cup\, K_1$ tal que $v_{n+1} \notin \{v_k, v_l\}$. Entonces $rs \to kl$ es un reemplazo admisible de aristas de $G\,\dot\cup\, K_1$, si y sólo si, $rs \to kl$ es un reemplazo admisible de aristas de G.*

Demostración. Nótese que en general se tiene $E(G\dot\cup K_1) = E(G)$ y además que en este caso también $E(G\dot\cup K_1 - v_rv_s + v_kv_l) = E(G - v_rv_s + v_kv_l)$. Así, si $rs \to kl$ es admisible en $G\,\dot\cup\, K_1$, la restricción $\varphi|_{V(G)}$ de cualquier isomorfismo φ de $G\dot\cup K_1 - v_rv_s + v_kv_l$ en $G\dot\cup K_1$ tal que $\varphi(v_{n+1}) = v_{n+1}$, es a su vez un isomorfismo de $G - v_rv_s + v_kv_l$ en G. Por otro lado, si φ es un isomorfismo de $G - v_rv_s + v_kv_l$ en G, la biyección $\varphi' : V(G\,\dot\cup\, K_1 - v_rv_s + v_kv_l) \to V(G\,\dot\cup\, K_1)$ dada por $\varphi'(v_i) = \varphi(v_i)$ para todo $v_i \in V(G)$ y tal que $\varphi'(v_{n+1}) = v_{n+1}$, es un isomorfismo de $G\,\dot\cup\, K_1 - v_rv_s + v_kv_l$ en $G\,\dot\cup\, K_1$. ■

Algoritmo 13: IdentificarAmoebaLocalyGlobal(G)

Entrada: una gráfica no vacía G de orden n con conjunto de vértices $V(G) = \{v_1, ..., v_n\}$.
Respuesta: texto indicando si G: es sólo amoeba local, sólo amoeba global, ambas local y global, o no es amoeba.

obtener la gráfica $G \dot{\cup} K_1$ al agregar el vértice aislado v_{n+1} a G
inicializar variable booleana **viable** $\leftarrow$ False
inicializar variable booleana **esLA** $\leftarrow$ False
inicializar variable booleana **esGA** $\leftarrow$ False
inicializar conjunto vacío $\mathcal{X}_G = \{\}$; para permutaciones asociadas a reemplazos admisibles de G
inicializar conjunto vacío $\mathcal{X}_{G\dot{\cup}K_1} = \{\}$; para permutaciones asociadas a reemplazos admisibles de $G \dot{\cup} K_1$
——————————— ETAPA 1 ———————————
obtener conjunto $D_G = \{d_G(v_i) \mid v_i \in V(G)\}$, y obtener los grados máximo Δ y mínimo δ de G
if ($|D_G| \neq \Delta - (\delta - 1)$ **) then**
    Devolver ("G no es amoeba" **)**
end
$\mathcal{X}_G \leftarrow \mathcal{X}_G \cup \{\alpha \in S_n \mid G_\alpha = G\}$
$\mathcal{X}_{G\dot{\cup}K_1} \leftarrow \mathcal{X}_{G\dot{\cup}K_1} \cup \{\alpha \in S_n \mid (G\dot{\cup}K_1)_\alpha = G\dot{\cup}K_1\}$
——————————— ETAPA 2 ———————————
forall $v_r v_s \in E(G)$ **do**
    forall $v_k v_l \in E(\overline{G\dot{\cup}K_1})$ **do**
        viable $\leftarrow$ CondicionesDeGrados($G\dot{\cup}K_1$, $v_r v_s$, $v_k v_l$)
        if (viable **) then**
            if ($G\dot{\cup}K_1 - v_r v_s + v_k v_l \simeq G\dot{\cup}K_1$ **) then**
                $\mathcal{X}_{G\dot{\cup}K_1} \leftarrow \mathcal{X}_{G\dot{\cup}K_1} \cup S_{G\dot{\cup}K_1}(rs \to kl)$
                if ($v_{n+1} \notin \{v_k, v_l\}$ **) then**
                    $\mathcal{X}_G \leftarrow \mathcal{X}_G \cup S_G(rs \to kl)$
                end
            end
        end
    end
end
——————————— ETAPA 3 ———————————
obtener grupo de alcanzabilidad $S_G = \langle \mathcal{X}_G \rangle$ de G
if ($|S_G| = n!$ **) then**
    esLA $\leftarrow$ True
end
if (**esLA** y $\delta \in \{0, 1\}$ **) then**
    esGA $\leftarrow$ True
else
    if ($1 \in D_G$ **) then**
        obtener grupo de alcanzabilidad $S_{G\dot{\cup}K_1} = \langle \mathcal{X}_{G\dot{\cup}K_1} \rangle$ de $G\dot{\cup}K_1$
        if ($|S_{G\dot{\cup}K_1}| = (n+1)!$ **) then**
            esGA $\leftarrow$ True
        end
    end
end
if (**esLA** y **esGA** **) then**
    Devolver ("G es amoeba local y global" **)**
end
if (**isLA** y *no* **esGA) then**
    Devolver ("G es amoeba local pero no global" **)**
end
if (*no* **esLA** y **esGA) then**
    Devolver ("G es amoeba global pero no local" **)**
end
Devolver ("G no es amoeba" **)**

Nótese que el **Algoritmo 13** igualmente manda llamar al **Algoritmo 11** en la línea 17, para evaluar que $rs \to kl$ cumpla las condiciones de la **Proposición 4.3.1**, pero ahora respecto a la gráfica $G\dot{\cup}K_1$. Esto se debe a que si v_k y v_l no son, ninguno de los dos, iguales al vértice aislado v_{n+1}, entonces es inmediato que v_r, v_s, v_k y v_l satisfacen las condiciones de la **Proposición 4.3.1** en G, si y sólo si, las satisfacen en $G\dot{\cup}K_1$. Por otro lado, si $v_{n+1} \in \{v_k, v_l\}$, entonces sólo estamos interesados en evaluar dichas condiciones respecto a $G\dot{\cup}K_1$ misma. De esta forma, vemos que las aristas v_kv_l se escojen de nueva cuenta, apropiadamente para poder reducir la cantidad de operaciones que realiza este algoritmo en la práctica. A continuación demostramos el correcto funcionamiento de este algoritmo, seguido por un resultado indicando una cota inferior para su tiempo de ejecución.

Teorema 6.3.1. *El* **Algoritmo 13** *resuelve correctamente el PDLA y PDGA para toda gráfica no vacía G de orden n con conjunto de vértices $V(G) = \{v_1, ..., v_n\}$.*

Demostración. La **Proposición 4.2.1** y el **Corolario 4.2.1** indican, respectivamente, que si el conjunto D_G de grados de los vértices de G, no es igual al intervalo $\{\delta, ..., \Delta\}$ independientemente del valor de δ, entonces G no puede ser una amoeba local, ni tampoco una amoeba global, por lo que el algoritmo terminaría en la línea 10 indicando que G no es amoeba de ningún tipo. Por otro lado, en las líneas 12 y 13 se calculan todas las permutaciones asociadas a los reemplazos triviales de G, y luego de $G\dot{\cup}K_1$, concluyendo así la Etapa 1. Luego, en la Etapa 2 se determinan todas las permutaciones asociadas a reemplazos no triviales de estas gráficas, donde todas las decisiones están sustentadas en la discusión al inicio de esta página, así como en la **Proposición 6.3.1**. Con todo lo anterior, es posible al inicio de la Etapa 3 obtener S_G y modificar el valor de la variable **esLA** si G es en efecto amoeba local. Seguido a esto, por el **Lema 4.1.1** y el **Teorema 4.1.1**, podemos deducir que G es también amoeba global, modificando el valor de **esGA**, siempre y cuando G sea amoeba local y tenga grado mínimo $\delta \in \{0, 1\}$. De cualquier otra forma se debe obtener $S_{G\dot{\cup}K_1}$ y modificar **esGA** correspondientemente. No obstante, antes de ello debemos revisar que G tenga por lo menos un vértice de grado 1, como hecho en la línea 36, debido al **Corolario 4.2.1**. De esta forma, el algoritmo concluye apropiadamente dependiendo del valor de **esLA** y **esGA** en este punto. ■

Proposición 6.3.2. *El tiempo de ejecución del* **Algoritmo 13**, *dada una gráfica de orden n, está acotado inferiormente por $n!n^6$, en un peor caso y usando el algoritmo* **VF2** *para evaluar isomorfismos.*

Demostración. La inicialización de variables y la Etapa 1 conllevan por lo menos, y bajo estas condiciones, $(n+1)!(n+1) = n!n^2 + n!2n + n!$ operaciones debido a la búsqueda de automorfismos de la gráfica $G\dot{\cup}K_1$. Por la misma razón y de forma similar al **Algoritmo 12**, la Etapa 2 de este algoritmo realizaría en un peor caso por lo menos $(n+1)!(n+1)n^4 = n!n^6 + n!2n^5 + n!n^4$ operaciones. Finalmente, como todas las instrucciones en la Etapa 3, excepto por el cálculo de los grupos de alcanzabilidad en la líneas 29 y 37, se evalúan todas en tiempo constante, podemos concluir que el tiempo de ejecución de este algoritmo está acotado inferiormente por $n!n^6$, como requerido. ■

6.4. Algoritmo para Detección de Amoebas entre Árboles

Ahora haremos uso de múltiples propiedades de los árboles para desarrollar un algoritmo más eficiente (en comparación con los **Algoritmos 12** y **13**) para la detección de amoebas locales y globales dentro de un conjunto de árboles no triviales. Comenzamos con una observación y una discusión derivadas de la **Proposición 4.3.1**, seguidas por el pseudocódigo de este nuevo algoritmo.

Observación 6.4.1. *Sea T un árbol de orden $n \geq 2$ con $V(T) = \{v_1, v_2, ..., v_n\}$ y sea $T \dot{\cup} K_1$ la gráfica obtenida al agregar el vértice aislado v_{n+1} a T. Sea además $rs \to kl$ un reemplazo de aristas de $T \dot{\cup} K_1$ tal que $v_{n+1} \in \{v_k, v_l\}$. Si para $rs \to kl$ se cumplen las condiciones en la* **Proposición 4.3.1**, *entonces se tiene $d_T(v_r) = d_{T\dot{\cup}K_1}(v_r) = 1$ o bien $d_T(v_s) = d_{T\dot{\cup}K_1}(v_s) = 1$.*

Demostración. Nótese que en general $d_T(v_r) = d_{T\dot{\cup}K_1}(v_r)$ y $d_T(v_s) = d_{T\dot{\cup}K_1}(v_s)$. Ahora bien, ya que $v_{n+1} \in \{v_k, v_l\}$, supongamos con $v_l = v_{n+1}$, entonces $rs \to kl$ es reemplazo no trivial de $T \dot{\cup} K_1$, y por ende de la **Proposición 4.3.1** se siguen sólo dos casos. Si $|\{r, s\} \cap \{k, l\}| = 1$, digamos con $r = k$, entonces $d_{T\dot{\cup}K_1}(v_s) = d_{T\dot{\cup}K_1}(v_l)+1 = d_{T\dot{\cup}K_1}(v_{n+1})+1 = 1$. De la misma forma, si $|\{r, s\} \cap \{k, l\}| = 0$, se tiene $\{d_{T\dot{\cup}K_1}(v_r), d_{T\dot{\cup}K_1}(v_s)\} = \{d_{T\dot{\cup}K_1}(v_k) + 1, d_{T\dot{\cup}K_1}(v_l) + 1\} = \{d_{T\dot{\cup}K_1}(v_k) + 1, 1\}$. ■

Esto nos dice que en todo reemplazo de aristas, admisible o no, $rs \to kl$ de $T \dot{\cup} K_1$ involucrando a v_{n+1} y satisfaciendo la **Proposición 4.3.1**, alguno de los dos extremos v_r o v_s de la arista original debe ser una hoja en $T \dot{\cup} K_1$ (es decir en T), implicando la existencia de los siguientes 3 casos.

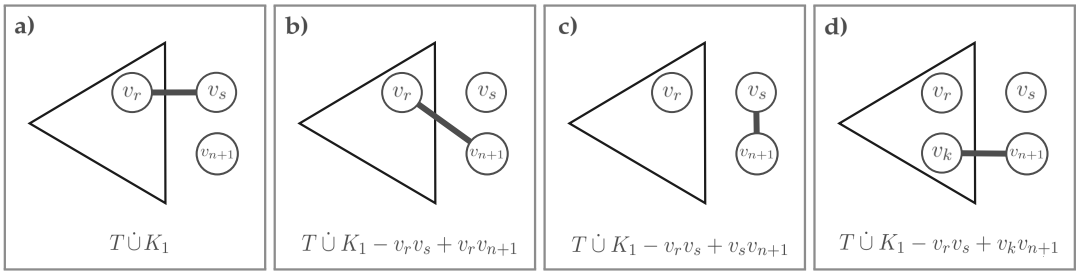

Figura 6.1: Se considera un árbol T de orden $n \geq 2$, mostrado aquí como un triángulo junto con una arista $v_r v_s$, donde v_s es una hoja y v_r es un vértice arbitrario de T. En **a)** se muestra la gráfica $T\dot{\cup}K_1$ obtenida al agregar un nuevo vértice v_{n+1} a T. Los demás casos corresponden a realizar un reemplazo admisible de aristas $rs \to kl$ sobre $T\dot{\cup}K_1$ con $l = n + 1$. En **b)** se tiene además $k = r$, mientras que en **c)** se tiene $k = s$ siendo que v_s es originalmente una hoja. Por último en **d)** se ejemplifica el caso $k \notin \{r, s\}$.

Nótese que dado un reemplazo de aristas $rs \to kl$ como el del caso **b)** en la **Figura 6.1**, es inmediato que $T\dot{\cup}K_1 - v_r v_s + v_k v_l \simeq T\dot{\cup}K_1$, incluso sin tener que evaluar el isomorfismo entre estas gráficas, ya que este reemplazo sólo intercambia la hoja v_s por una nueva hoja v_{n+1}. Podemos ver además que el reemplazo del caso **c)** en esa figura es admisible, si y sólo si, T mismo es K_2, ya que $T\dot{\cup}K_1$ consta de una única componente conexa no trivial y un vértice aislado. Finalmente, el caso **d)** siempre devuelve una componente árbol no trivial T' y un vértice aislado v_s, por lo que el isomorfismo $T\dot{\cup}K_1 - v_r v_s + v_k v_l \simeq T\dot{\cup}K_1$, depende únicamente del isomorfismo entre T' y T.

Algoritmo 14: IdentificarÁrbolAmoebaLocalyGlobal(T)

Entrada: un árbol T de orden $n \geq 2$ con conjunto de vértices $V(T) = \{v_1, ..., v_n\}$.
Respuesta: texto indicando si T: es sólo amoeba global, ambas local y global, o no es amoeba.

obtener la gráfica $T \dot{\cup} K_1$ al agregar el vértice aislado v_{n+1} a T, e inicializar variable booleana **viable** $\leftarrow$ False
inicializar conjunto vacío $\mathcal{X}_T = \{\}$; para permutaciones asociadas a reemplazos admisibles de T
inicializar conjunto vacío $\mathcal{X}_{T \dot{\cup} K_1} = \{\}$; para permutaciones asociadas a reemplazos admisibles de $T \dot{\cup} K_1$
———————————— ETAPA 1 ————————————
obtener conjunto $D_T = \{d_T(v_i) \mid v_i \in V(T)\}$, y obtener grado máximo Δ de T
if ($|D_T| \neq \Delta$ **) then**
    Devolver ("T no es amoeba" **)**
end
$\mathcal{X}_T \leftarrow \mathcal{X}_T \cup \{\alpha \in S_n \mid T_\alpha = T\}$
$\mathcal{X}_{T \dot{\cup} K_1} \leftarrow \mathcal{X}_{T \dot{\cup} K_1} \cup \{\alpha \in S_n \mid (T \dot{\cup} K_1)_\alpha = T \dot{\cup} K_1\}$
———————————— ETAPA 2 ————————————
forall $v_r v_s \in E(T)$ **do**
    obtener componentes conexas C_1 y C_2 de $T - v_r v_s$
    obtener conjunto de pares de vértices $E_{C_1 C_2} = \{v_i v_j \mid v_i \in C_1 \text{ y } v_j \in C_2\} \setminus \{v_r v_s\}$
    forall $v_k v_l \in E_{C_1 C_2}$ **do**
        viable $\leftarrow$ CondicionesDeGrados(T, $v_r v_s$, $v_k v_l$)
        if (viable **) then**
            if ($T - v_r v_s + v_k v_l \simeq T$ **) then**
                $\mathcal{X}_T \leftarrow \mathcal{X}_T \cup S_T(rs \to kl)$
                $\mathcal{X}_{T \dot{\cup} K_1} \leftarrow \mathcal{X}_{T \dot{\cup} K_1} \cup S_{T \dot{\cup} K_1}(rs \to kl)$
            end
        end
    end
end
obtener grupo de alcanzabilidad $S_T = \langle \mathcal{X}_T \rangle$ de T
if ($|S_T| = n!$ **) then**
    Devolver ("T es amoeba local y global" **)**
end
———————————— ETAPA 3 ————————————
obtener conjunto de pares de vértices $E_{ext} = \{v_i v_{n+1} \mid v_i \in V(T)\}$
forall $v_r v_s \in E(T)$ **do**
    forall $v_k v_l \in E_{ext}$ **do**
        viable $\leftarrow$ CondicionesDeGrados($T \dot{\cup} K_1$, $v_r v_s$, $v_k v_l$)
        if (viable **) then**
            ; suponer s.p.d.g que $v_l = v_{n+1}$ y notar que la **Proposición 4.3.1** indica en este punto del algoritmo,
            ; que v_r o v_s es una hoja, pero no ambos, ya que este algoritmo termina en la ETAPA 2 para $T = K_2$
            if ($v_k \in \{v_r, v_s\}$ **) then**
                $\mathcal{X}_{T \dot{\cup} K_1} \leftarrow \mathcal{X}_{T \dot{\cup} K_1} \cup S_{T \dot{\cup} K_1}(rs \to kl)$
            else
                obtener (única) componente conexa no trivial T' de $T \dot{\cup} K_1 - v_r v_s + v_k v_l$
                ; la existencia de T' se sigue de haber cumplido las condiciones de la **Proposición 4.3.1**
                if ($T' \simeq T$ **) then**
                    $\mathcal{X}_{T \dot{\cup} K_1} \leftarrow \mathcal{X}_{T \dot{\cup} K_1} \cup S_{T \dot{\cup} K_1}(rs \to kl)$
                end
            end
        end
    end
end
obtener grupo de alcanzabilidad $S_{T \dot{\cup} K_1} = \langle \mathcal{X}_{T \dot{\cup} K_1} \rangle$ de $T \dot{\cup} K_1$
if ($|S_{T \dot{\cup} K_1}| = (n+1)!$ **) then**
    Devolver ("T es amoeba global pero no local" **)**
end
Devolver ("T no es amoeba" **)**

Nótese que este nuevo algoritmo manda llamar en dos ocasiones distintas al **Algoritmo 11**. Esto se hace primero durante la Etapa 2 del algoritmo (en la línea 16), para evaluar las condiciones de la **Proposición 4.3.1** respecto a reemplazos de aristas de T y de $T \dot{\cup} K_1$, que no involucran al nuevo vértice v_{n+1}. El segundo momento donde se realiza dicha llamada es durante la Etapa 3, en la línea 33, con el motivo de evaluar tales condiciones respecto a $T \dot{\cup} K_1$, para un reemplazo $rs \to kl$ involucrando a v_{n+1} en esta gráfica. Es importante notar además que la línea 35 indica que se supone sin pérdida de generalidad $v_l = v_{n+1}$, lo que es posible ya que dicha condición en realidad se puede incorporar naturalmente dentro de la implementación computacional de este algoritmo. Basándonos en todo esto, mostramos ahora su correcto funcionamiento.

Teorema 6.4.1. *El* **Algoritmo 14** *resuelve correctamente el PDLA y PDGA para todo árbol T de orden $n \geq 2$ con conjunto de vértices $V(T) = \{v_1, v_2, ..., v_n\}$.*

Demostración. Por la **Proposición 4.2.2** sabemos que de no cumplirse $|D_T| = \Delta$, el algoritmo debe terminar en la línea 7 indicando que T no es amoeba de ningún tipo. Por otro lado se deben obtener todas las permutaciones asociadas a reemplazos admisibles. La Etapa 1 concluye obteniendo dichas permutaciones para reemplazos triviales de ambas gráficas. En la Etapa 2 se analizan los reemplazos no triviales que no involucran a v_{n+1}. Nótese que al remover $v_r v_s$ de T, siempre se obtienen dos componentes conexas árboles C_1 y C_2, y en consecuencia las únicas aristas $v_k v_l \in E(\overline{T})$ que pueden hacer que $T - v_r v_s + v_k v_l$ sea nuevamente un árbol, son aquellas con un extremo en C_1 y el otro en C_2. Luego, al iterar sobre dichas aristas, se usa la **Proposición 6.3.1**, así como el hecho de que las condiciones de la **Proposición 4.3.1** se satisfacen respecto a T, si y sólo si, se satisfacen respecto a $T \dot{\cup} K_1$. Así, como el grupo de alcanzabilidad de T mismo no depende del vértice v_{n+1}, podemos obtener S_T en la línea 25, y si T es amoeba local, por el **Corolario 4.1.2** podemos concluir en la línea 27 que T es ambas local y global. Posteriormente en la Etapa 3, como se tiene $E(\overline{T \dot{\cup} K_1}) = E(\overline{T}) \dot{\cup} \{v_i v_{n+1} \mid v_i \in V(T)\}$, vemos que sólo hace falta tomar $v_k v_l$ en este último conjunto. Si para uno de estos reemplazos $rs \to kl$ se mantiene la **Proposición 4.3.1**, por la **Observación 6.4.1** podemos suponer sin pérdida de generalidad que v_s es una hoja en T, y además sabemos que $d_{T \dot{\cup} K_1}(v_r) \geq 2$ ya que el algoritmo termina en la Etapa 2 para $T = K_2$ por la **Observación 3.5.2**. Luego, si $v_k \in \{v_r, v_s\}$, necesariamente $v_k = v_r$, ya que de tener $v_k = v_s$ habiendo cumplido la **Proposición 4.3.1**, por el inciso **ii)** de dicha proposición se seguiría $d_{T \dot{\cup} K_1}(v_r) = d_{T \dot{\cup} K_1}(v_l) + 1 = d_{T \dot{\cup} K_1}(v_{n+1}) + 1 = 1$, lo que es imposible. Es decir que si se cumple $v_k \in \{v_r, v_s\}$ en la línea 37 del algoritmo, $rs \to kl$ cumple el caso **b)** en la **Figura 6.1** y por ende es admisible. Por otro lado, si $v_k \notin \{v_r, v_s\}$, entonces se verifica el caso **d)** de la **Figura 6.1**, y sólo debemos evaluar el isomorfismo de árboles entre T y la única componente conexa T' no trivial de $T \dot{\cup} K_1 - v_r v_s + v_k v_l$. Con todo lo anterior, podemos obtener $S_{T \dot{\cup} K_1}$ en la línea 49 del algoritmo, y concluir que T es sólo amoeba global, o bien que no es amoeba ya que no existe otra posibilidad. ■

Más aún, como en el **Algoritmo 14** sólo se evalúan isomorfismos entre árboles, podremos recurrir al algoritmo en [25] que en esta tesis hemos llamado **IsoTree**, en vez de utilizar el algoritmo **VF2**, reduciendo así la cantidad de operaciones requeridas para detectar árboles amoeba.

Proposición 6.4.1. *El tiempo de ejecución del* **Algoritmo 14**, *dado un árbol T de orden $n \geq 2$, está acotado inferiormente por $n^4 \log_2(n)$, en un peor caso y usando el algoritmo* **IsoTree** *para evaluar isomorfismos.*

Demostración. Recuérdese que las contribuciones de todas las etapas en el algoritmo se calculan sin considerar la obtención de los grupos de alcanzabilidad, y en este caso tampoco consideraremos las contribuciones (todas positivas) de las líneas 9, 10, 19, 20, 38 y 43, ya que el algoritmo usado para determinar las permutaciones requeridas en esas líneas puede variar en la práctica, mientras que el propósito de esta estimación es dar una cota inferior teórica para el tiempo de ejecución del algoritmo. Así, consideraremos que en la Etapa 1 se realizarán por lo menos n operaciones para determinar D_T. Luego, la Etapa 2 realizará $[n-1][n+n^3\log_2(n)] = n^2 + n^4\log_2(n) - n - n^3\log_2(n)$ operaciones. Esto se sigue ya que por cada una de las $n-1$ aristas de T, se deben determinar primero las componentes conexas C_1 y C_2 en la línea 13, lo que es realizable por medio del algoritmo de búsqueda por profundidad (o bien por búsqueda por anchura), ambos con complejidad en tiempo $O(|V(G)| + |E(G)|)$ para una gráfica G, y que en nuestro caso corresponde a $O(n + n - 1) = O(n)$. Pero seguido a esto, en un peor caso se deberá evaluar el isomorfismo entre árboles, de orden $n\log_2(n)$, para n^2 pares de vértices en $E_{C_1C_2}$, dando lugar al término $n^3\log_2(n)$ de antes. Finalmente, la Etapa 3 hará $[n-1][n][n+n\log_2(n)] = n^3 + n^3\log_2(n) - n^2 - n^2\log_2(n)$ operaciones. Sumando todas estas contribuciones tenemos,

$$[n] + [n^2 + n^4\log_2(n) - n - n^3\log_2(n)] + [n^3 + n^3\log_2(n) - n^2 - n^2\log_2(n)] = \\ [n^4\log_2(n)] + [n^3 - n^2\log_2(n)],$$

donde $n^3 \geq n^2\log_2(n)$ para toda $n \geq 1$ y por ende dicha suma está acotada inferiormente por el mayor término $n^4\log_2(n)$, siendo además que las otras contribuciones son todas positivas. ■

Cabe remarcar que todas las cotas inferiores para los tiempos de ejecución de los **Algoritmos 12**, **13** y **14**, se determinaron al considerar para ellos los peores casos. Sin embargo, se debe notar que la elección del peor caso no sólo influye sobre los algoritmos de isomorfismos escogidos, sino que esto repercute, por ejemplo, en estimar la cantidad de aristas a recorrer en un ciclo For anidado para ser del orden n^4, como en el **Algoritmo 12**. Luego, esto deja ver que en general, la elección de un caso óptimo o promedio sería poco informativa en comparación con los peores casos ya que, por el mismo ejemplo de antes, se podría estimar una cantidad arbitrariamente optimista de aristas a recorrer con dicho ciclo For anidado. No obstante, por fines prácticos, se puede ver que los tres algoritmos terminan en un mejor caso en exactamente n operaciones, después de determinar el conjunto D_G (o bien D_T) para una gráfica G (resp. un árbol T) de orden n que no es amoeba, y que en particular, pueda identificarse únicamente utilizando dicho conjunto de grados.

Capítulo 7
Resultados de Ambas Contribuciones

Aquí mostramos los resultados obtenidos al implementar los algoritmos del capítulo 6, y analizar con ellos distintos conjuntos de gráficas disponibles en bases de datos públicas [6, 7, 28, 29]. Para comenzar, en la **Sección 7.1** se describe la representación computacional que se ha dado a las gráficas analizadas, y se incluyen dos resultados teóricos que fundamentan su uso para llevar a cabo la detección computacional de las amoebas. Luego, en la **Sección 7.2** describimos los aspectos prácticos de las implementaciones de los **Algoritmos 13** y **14**, al presentar al mismo tiempo nuestro repositorio de Github: **Detection of Amoeba Graphs** [3], disponible públicamente y que contiene todos los programas que hemos desarrollado como parte del trabajo en esta tesis. Luego, en la **Sección 7.3** se muestran los resultados de analizar con dichos programas todas las gráficas no isomorfas teniendo entre 1 y 10 vértices. Igualmente, en la **Sección 7.4** se incluyen los resultados de un análisis similar sobre todos los árboles no isomorfos con entre 1 y 22 vértices. Por último, en la **Sección 7.5** se exhiben ejemplos de gráficas teniendo reemplazos raros, junto con los resultados de un análisis computacional sobre la estructura del grupo de alcanzabilidad S_G de estas gráficas.

7.1. Aspectos Teóricos de la Implementación Computacional

Considérese una gráfica G de orden n con conjunto de vértices $V(G) = \{v_1, ..., v_n\}$. De aquí en adelante nos referiremos como la representación computacional de G, a la gráfica G' con conjunto de vértices en $V(G') = \{1, ..., n\}$ y con conjunto de aristas $E(G')$ tal que $ij \in E(G')$ si y sólo si $v_iv_j \in E(G)$, como ejemplificado en la **Figura 7.1**. Aquí veremos dos resultados que justifican el uso de esta representación computacional en la detección de las amoebas. En estos denotamos por $Iso(G, H)$ al conjunto de todos los isomorfismos de G, en otra gráfica H isomorfa a G.

Proposición 7.1.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $rs \to kl$ un reemplazo admisible de aristas de G. Sea además G' la gráfica de orden n con conjunto de vértices $V(G') = \{1, ..., n\}$ y con conjunto de aristas $E(G')$ tal que $ij \in E(G')$ si y sólo si $v_iv_j \in E(G)$. Para toda $\sigma \in S_n$ se tiene $\sigma \in S_G(rs \to kl)$ si y sólo si $\sigma \in Iso(G' - rs + kl, G')$.*

Demostración. De las hipótesis se sigue que $ij \in E(G' - rs + kl)$ si y sólo si $v_iv_j \in E(G - v_rv_s + v_kv_l)$. Supongamos primero que $\sigma \in S_G(rs \to kl)$, es decir, $G_\sigma = G - v_rv_s + v_kv_l$. Entonces tenemos,

$$ij \in E(G' - rs + kl) \text{ si y sólo si } v_iv_j \in E(G - v_rv_s + v_kv_l), \text{ que es } v_iv_j \in E(G_\sigma),$$
$$\text{y equivalentemente } v_{\sigma(i)}v_{\sigma(j)} \in E(G), \text{ e igualmente } \sigma(i)\sigma(j) \in E(G').$$

Por otro lado supóngase que $\sigma \in Iso(G' - rs + kl, G')$, entonces tenemos,

$$v_iv_j \in E(G - v_rv_s + v_kv_l) \text{ si y sólo si } ij \in E(G' - rs + kl), \text{ de donde } \sigma(i)\sigma(j) \in E(G'),$$
$$\text{o lo que es lo mismo } v_{\sigma(i)}v_{\sigma(j)} \in E(G), \text{ y equivalentemente } v_iv_j \in E(G_\sigma).$$

En consecuencia $G_\sigma = G - v_rv_s + v_kv_l$ y por lo tanto $\sigma \in S_G(rs \to kl)$, como requerido. ■

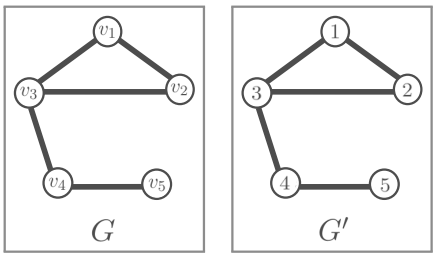

Figura 7.1: Se exhibe una gráfica G y su representación computacional G'. Aunque esta última es sencilla, se requiere la **Proposición 7.1.1** para justificar su uso en la obtención algorítmica de los conjuntos $S_G(rs \to kl)$, como isomorfismos de $G' - rs + kl$ en G', respecto a cada reemplazo admisible de aristas $rs \to kl$ de G.

La ventaja de usar esta representación radica en que ya existen algoritmos como el **VF2** [23], con implementaciones disponibles públicamente [24] en distintos lenguajes de programación, que nos permiten determinar isomorfismos entre dos gráficas con conjuntos arbitrarios de vértices. Luego, con esta representación computacional y por medio de dichos algoritmos, ya contamos con todo lo necesario para obtener los conjuntos $S_G(rs \to kl)$ dado cualquier reemplazo admisible de aristas $rs \to kl$ de una gráfica G, y consecuentemente llevar a cabo la detección de las amoebas. Ahora haremos explícita la repercusión de la **Proposición 7.1.1** sobre el grupo de alcanzabilidad S_G de una gráfica G. Omitimos su prueba ya que esta es implicación directa de dicha proposición.

Proposición 7.1.2. *Sea G una gráfica no vacía de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea R_G el conjunto de todos los reemplazos admisibles de aristas de G. Sea además G' la gráfica de orden n con conjunto de vértices $V(G') = \{1, ..., n\}$ y con conjunto de aristas $E(G')$ tal que $ij \in E(G')$ si y sólo si $v_iv_j \in E(G)$. Entonces,*

i) $S_G(rs \to kl) = Iso(G' - rs + kl, G')$ *para todo* $rs \to kl \in R_G$.

ii) $\{\alpha \in S_n \mid G_\alpha = G\} = S_G(rs \to kl) = Aut(G')$ *para todo reemplazo trivial* $rs \to kl$ *de* G.

iii) $S_G = \Big\langle \bigcup_{rs \to kl \,\in\, R_G} Iso(G' - rs + kl, G') \Big\rangle$. ■

Con base en esto, los **Algoritmos 13** y **14** pueden ser modificados para analizar la representación computacional G' de cada gráfica G, en vez de analizar a G misma. Esto se logra, por ejemplo, realizando la actualización $\mathcal{X}_{G'} \leftarrow\!\shortmid\; \mathcal{X}_{G'} \cup Aut(G')$, para guardar en un conjunto $\mathcal{X}_{G'}$ las permutaciones asociadas a reemplazos triviales de G en la línea 12 del **Algoritmo 13**, en lugar de la instrucción $\mathcal{X}_G \leftarrow\!\shortmid\; \mathcal{X}_G \cup \{\alpha \in S_n \mid G_\alpha = G\}$ en dicha línea. De forma similar se deben obtener los isomorfismos $Iso(G' - rs + kl, G')$, cada vez que estos algoritmos requieran obtener el conjunto $S_G(rs \to kl)$ para un reemplazo admisible $rs \to kl$ de G. Ya que todas estas modificaciones se pueden realizar prontamente y de forma muy natural, no ahondaremos más en los detalles de como reescribir dichos algoritmos para considerar estos dos nuevos resultados. Sin embargo, en la siguiente sección presentaremos un repositorio público que contiene todos los programas realizados como parte del trabajo en esta tesis, y que funcionan tomando en cuenta esta representación computacional.

7.2. Implementación en Python y Repositorio en Github

Python [4] es un lenguaje de programación muy versátil que cuenta con sintáxis y terminología muy similar al lenguaje coloquial, lo que permite implementar algoritmos legibles y fácilmente comprensibles. A su vez, Github [5] puede describirse como un sistema de almacenamiento, abarcando funcionalidades para la colaboración por internet y manejo de versiones, entre otras, con el que puede hacerse disponible públicamente un proyecto de programación si así se requiere.

En esta tesis no sólo llevamos a cabo la implementación de los **Algoritmos 13** y **14** en lenguaje Python, para la detección computacional de las amoebas, sino que además hicimos públicos estos programas al crear el repositorio de Github: **Detection of Amoeba Graphs** [3], al que se puede acceder usando el enlace `https://github.com/MarcosLaffitte/Amoebas`. Github además proporciona opciones para visualizar o descargar el contenido del repositorio.

En nuestro repositorio se pueden encontrar 6 carpetas, conteniendo cada una programas con una funcionalidad particular para realizar experimentos computacionales distintos. Estas carpetas son: G6 Builder, Detect Arbitrary Amoebas, Detect Tree Amoebas, Weird Edge Replacements, Group S_G Structure, y finalmente Examples. A continuación describimos el contenido de cada carpeta, y por medio de esto explicaremos además, las particularidades de la implementación computacional en lenguaje Python de nuestros algoritmos de detección de amoebas.

Contenido de nuestro repositorio: Detection of Amoeba Graphs.

- **1. G6 Builder -** Existen múltiples formas de procesar una gráfica computacionalmente. Una de ellas es conocida como formato **g6** cuya definición se puede encontrar en [28, 29]. Este conlleva, en esencia, expresar como una cadena de bits a la matriz de adyacencia de la gráfica, y después transformar dicha cadena a texto por medio del ampliamente conocido código ASCII [30]. Si bien es cierto que cada cadena de este tipo codifica a una única gráfica, se debe tomar en cuenta que dos gráficas isomorfas pueden tener asociadas distintas cadenas **g6**, como aquellas mostradas en la **Figura 7.2**. En esta tesis hicimos uso de dicho formato para poder almacenar todas las gráficas analizadas. Luego, cada cadena puede ser transformada a un objeto **Graph** de la biblioteca **NetworkX** [31] del lenguaje Python para su análisis, ya que este último modela a una gráfica simple con todas sus propiedades. Así, en esta carpeta es posible encontrar dos programas: uno que contiene ejemplos de como ingresar en el código mismo una gráfica -arista por arista- con propósito de obtener un archivo en formato **g6**, y otro que recibe una lista de gráficas todas dentro de un archivo con formato **g6** e imprime un dibujo en formato **PDF** de las gráficas proporcionadas. A la fecha, estos programas en nuestro repositorio cuentan con todas las gráficas mostradas en esta tesis como ejemplo de su uso.

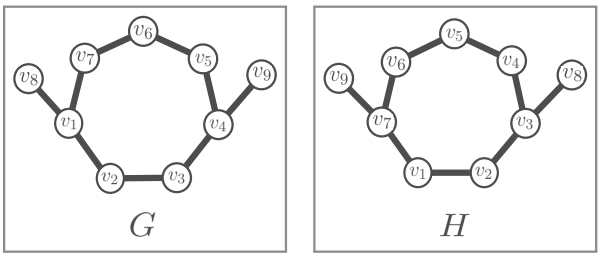

Figura 7.2: Se muestran dos gráficas distintas G y H. A pesar de ser isomorfas, estas tienen diferentes representaciones en formato **g6**. Por ejemplo, en la carpeta **G6 Builder** de nuestro repositorio **Detection of Amoeba Graphs** [3], se puede ver como a G se le asigna la cadena de texto "HhCKK?O" como representación en formato **g6**, mientras que a H se le asigna la cadena "HhCKH?A" en este mismo formato.

- **2. Detect Arbitrary Amoebas -** En esta carpeta se puede encontrar la implementación en lenguaje Python del **Algoritmo 13** para detectar amoebas dentro de una colección arbitraria de gráficas. Aquí hay dos programas. El primero recibe una lista de gráficas en formato **g6** (como la generada con los programas en la carpeta **G6 Builder** de antes) y ejecuta el **Algoritmo 13** para cada una de las gráficas en la lista. Al finalizar, dicho programa produce 4 archivos: uno de amoebas sólo globales, otro de sólo locales, uno con amoebas que son locales y globales, y uno con las gráficas que no fueron identificadas como amoebas. El segundo programa toma todos estos archivos y produce los dibujos de todas las gráficas según fueron clasificadas. Para lograr todo esto, se hace uso de la ya existente implementación del algoritmo **VF2** de la biblioteca **NetworkX** [24, 31], así como de las implementaciones del algoritmo Schreier-Sims en la biblioteca **Sympy** [26] para la obtención de los grupos de alcanzabilidad de las gráficas analizadas. Con estos programas se llevó a cabo la detección de amoebas dentro de la colección de todas las gráficas simples no isomorfas con entre 1 y 10 vértices, tomadas de la base de datos pública [6]. En la **Sección 7.3** mostramos los resultados de este análisis.

- **3. Detect Tree Amoebas -** Aquí se puede encontrar la implementación del **Algoritmo 14** en lenguaje Python, diseñado para la detección de amoebas dentro de un conjunto de árboles. Nuevamente se incluyen dos programas: uno que determina todos los árboles dentro de una lista de gráficas arbitrarias en formato **g6** para después realizar la detección de amoebas entre dichos árboles, y otro que dibuja los resultados obtenidos por este análisis, clasificando a tales árboles como: amoebas sólo globales, amoebas locales y globales, o árboles que no son amoebas. Para evaluar el isomorfismo entre árboles se utiliza el algoritmo en [25] de la biblioteca **NetworkX** [31] al que en esta tesis nos hemos referido como **IsoTree**, y de nueva cuenta se utiliza el algoritmo **VF2** [24] para determinar las permutaciones asociadas a reemplazos admisibles, así como las implementaciones del algoritmo Schreier-Sims de **Sympy** [26] para obtener los grupos de alcanzabilidad. Con esto se llevó a cabo la detección de amoebas dentro de la colección de todos los árboles no isomorfos teniendo entre 1 y 22 vértices, obtenidos de la base de datos [7]. En la **Sección 7.4** mostramos los resultados de este otro análisis.

- **4. Weird Edge Replacements -** Aunque los programas mencionados antes determinan durante su ejecución a todos los reemplazos admisibles de cada gráfica, ninguno de estos reporta como resultado final la lista de dichos reemplazos. Esta tarea se deja a cargo de los programas en esta nueva carpeta con motivo de clasificar tales reemplazos como reemplazos raros u ordinarios de la gráfica en cuestión, además de poder reportar finalmente a las gráficas analizadas en 4 categorías disjuntas: amoebas de todo tipo con reemplazos raros, gráficas que no son amoebas también con reemplazos raros, y las clases análogas pero sin reemplazos raros, i.e., ordinarias. Los programas aquí son semejantes a los de la carpeta **Detect Arbitrary Amoebas** de antes, y con estos obtuvimos los resultados mostrados en la **Sección 7.5**.

- **5. Group S_G Structure -** Si bien es cierto que los programas de antes determinan el grupo de alcanzabilidad S_G de cada gráfica G, debe notarse que estos sólo hacen uso de la cardinalidad de S_G, más no determinan su estructura algebraica como subgrupo de S_n, dada G de orden n. En esta carpeta se incluye un último par de programas con esta funcionalidad. Específicamente, el primero de estos determina el único subgrupo (salvo isomorfismos) de S_n isomorfo a S_G, y lo mismo para la gráfica $G\dot{\cup}K_1$ y su respectivo grupo de alcanzabilidad $S_{G\dot{\cup}K_1}$. Más aún, si G, o bien $G\dot{\cup}K_1$, poseen reemplazos raros, entonces también se determinan las estructuras que tendrían los grupos S_G y $S_{G\dot{\cup}K_1}$ de generarlos sin considerar a las permutaciones asociadas, primero a reemplazos raros de dichas gráficas, y luego sin aquellas asociadas a sus reemplazos ordinarios. Así, el segundo programa imprime todas estas estructuras junto con una imagen de cada gráfica, indicando además si G, o $G\dot{\cup}K_1$, fueron gráficas raras u ordinarias. Todo esto fue posible al implementar dentro de un programa similar al de la carpeta **Detect Arbitrary Amoebas**, funciones del software SageMath [32], tanto para generar los respectivos grupos de alcanzabilidad, como para obtener sus estructuras como subgrupos de S_n. Con estos programas se complementarán los resultados mostrados en la **Sección 7.5**.

- **6. Examples -** Finalmente, aquí se incluyen los resultados de ejecutar todos estos programas sobre distintos conjuntos de gráficas. En esta carpeta se pueden encontrar los resultados obtenidos respecto al conjunto de todas las gráficas incluidas en esta tesis, así como sobre todas las gráficas no isomorfas teniendo entre 1 y 7 vértices, siendo que ambas colecciones sirven como ejemplo del uso de estos programas. Además, se incluye un resumen de las cantidades de: amoebas entre todas las gráficas no isomorfas con 1 a 10 vértices, árboles amoeba con entre 1 y 22 vértices, y las amoebas y gráficas raras con entre 1 y 10 vértices.

Debemos agregar que todos los dibujos producidos por los programas en el repositorio, son generados usando la ampliamente conocida biblioteca Matplotlib [33] de Python, que cuenta con muchas facilidades para la creación de imágenes en formato **PDF**. Adicionalmente, en nuestro repositorio de Github se incluye también un manual con las instrucciones y ejemplos de uso de todos estos programas, indicando las versiones de las bibliotecas de Python requeridas para su ejecución.

De igual forma, cabe mencionar que las funciones de detección de amoebas en estos scripts, fueron programadas buscando que se parecieran lo más posible a los algoritmos mostrados en esta tesis, incluyendo cada una de las etapas que los componen, así como múltiples comentarios que describen las acciones en ellos, todo esto con propósito de facilitar su lectura y reusabilidad.

7.3. Amoebas en Conjuntos de Gráficas Arbitrarias

En la base de datos "Combinatorial Data - Graphs Page" del Prof. Brendan D. McKay de la Universidad Nacional Australiana [6], se pueden encontrar todas las gráficas (simples) no isomorfas teniendo entre 1 y 11 vértices, todas en formato **g6**. Para esta tesis hemos escogido analizar únicamente dichas gráficas hasta 10 vértices. Esto se determinó así en consideración al tiempo y la capacidad computacional con que se contó durante el desarrollo de este trabajo, aunado al hecho de que la cantidad de gráficas no isomorfas con 11 vértices, es aproximadamente 85 veces más grande que la de gráficas no isomorfas con 10 vértices (incluida en la tabla más abajo).

Todas estas gráficas fueron analizadas con programas que implementan el **Algoritmo 13**. Esto nos permitió determinar todas las gráficas en este conjunto que son o no amoebas, y su tipo de amoeba de ser el caso. En la **Tabla 7.1** resumimos las cantidades obtenidas por cada familia de gráficas. Específicamente, en esta tabla se denota: por **LA** a la familia de amoebas locales independientemente de si son globales o no, por **GA** a las amoebas globales independientemente de si también son locales o no, mientras que **LA∩GA** denota sólo a las gráficas cumpliendo ambas definiciones, y por último se nombra como **LA∪GA** a la columna que incluye el total de amoebas por cada orden. Así, para obtener la cantidad de amoebas sólo locales en cada orden, se debe restar el número en la columna **LA∩GA** a la cantidad en la columna **LA**, y de forma similar para **GA**.

Orden	LA	GA	LA∩GA	LA∪GA	Gráficas No Isomorfas
1	1	1	1	1	**1**
2	2	2	2	2	**2**
3	4	3	3	4	**4**
4	7	6	5	8	**11**
5	21	14	14	21	**34**
6	56	38	30	64	**156**
7	306	124	117	313	**1,044**
8	1,146	425	338	1,233	**12,346**
9	3,990	1,138	1,001	4,127	**274,668**
10	8,192	3,071	2,196	9,067	**12,005,168**
Total	13,725	4,822	3,707	14,840	**12,293,434**

Tabla 7.1: Cantidades de amoebas en el conjunto de gráficas no isomorfas de cada orden. El último renglón muestra el total de cada familia respecto a todas las gráficas no isomorfas teniendo entre 1 y 10 vértices.

Es de nuestro interés conocer no sólo las cantidades de estas amoebas, sino también su proporción respecto al total de gráficas no isomorfas de cada orden. A continuación se resumen en la **Tabla 7.2** tales proporciones expresadas como porcentajes, y en la **Figura 7.3** se grafican dichos porcentajes como función del orden de las amoebas en cada una de las familias.

Orden	[%] LA	[%] GA	[%] LA∩GA	[%] LA∪GA	Gráficas No Isomorfas
1	100.00	100.00	100.00	100.00	**1**
2	100.00	100.00	100.00	100.00	**2**
3	100.00	75.00	75.00	100.00	**4**
4	63.64	54.55	45.45	72.73	**11**
5	61.76	41.18	41.18	61.76	**34**
6	35.90	24.36	19.23	41.03	**156**
7	29.31	11.88	11.21	29.98	**1,044**
8	9.28	3.44	2.74	9.99	**12,346**
9	1.45	0.41	0.36	1.50	**274,668**
10	0.07	0.03	0.02	0.08	**12,005,168**
Total	0.11	0.04	0.03	0.12	**12,293,434**

Tabla 7.2: Proporciones de amoebas en el conjunto de gráficas no isomorfas de cada orden, expresadas como porcentajes redondeados a dos decimales (hacia arriba para tercer decimal ≥ 5). Nótese que el total de amoebas (**LA∪GA**) de orden 10 es menor al 1 % de las gráficas no isomorfas de ese orden.

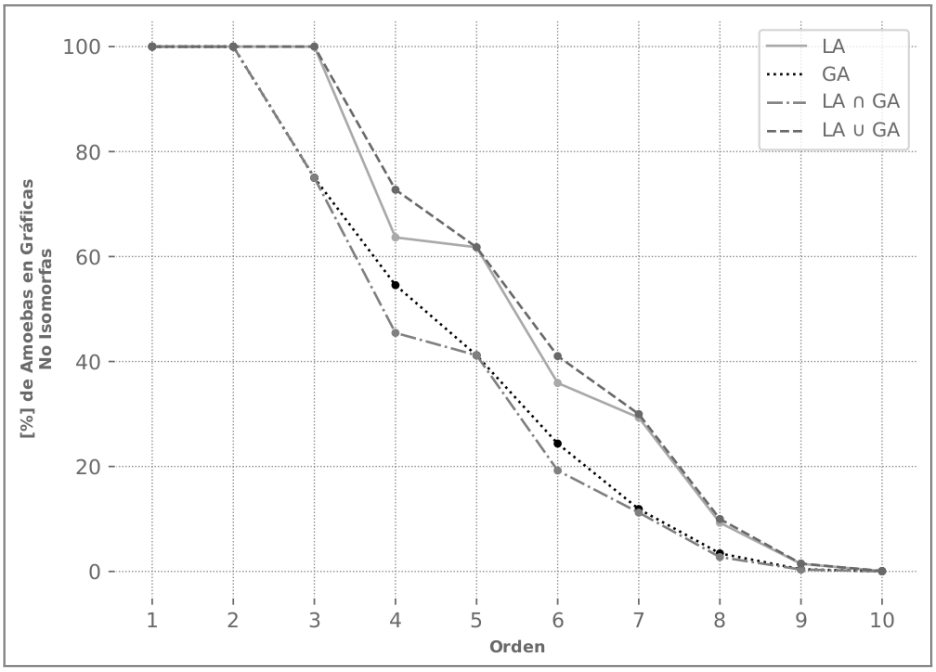

Figura 7.3: Porcentajes de la **Tabla 7.2** graficados como función del orden de las amoebas en cada familia. Debe notarse que las 4 curvas mostradas son todas estrictamente decrecientes.

Ahora bien, en la **Tabla 7.1** podemos ver que todas las gráficas teniendo entre 1 y 3 vértices, y que son amoebas globales, son también amoebas locales (i.e., **GA** $=$ **LA**$\cap$**GA** para las cantidades en todos estos órdenes). Luego, en 4 vértices aparece una primera amoeba que es global pero no local, específicamente, esta es la gráfica $K_2 \dot{\cup} K_2$ mostrada en el cuadro **c)** de la **Figura 3.8** del capítulo 3. Sin embargo, esta es una gráfica disconexa, por lo que un primer objetivo planteado al inicio de este trabajo fue determinar, haciendo uso de las implementaciones computacionales de nuestros algoritmos, a las mínimas gráficas conexas que fueran amoebas globales pero no locales. Estas se encontraron hasta el orden 6, y son las 4 gráficas mostradas en la **Figura 7.4**. Debemos añadir que las otras 4 gráficas globales pero no locales de este mismo orden son todas disconexas.

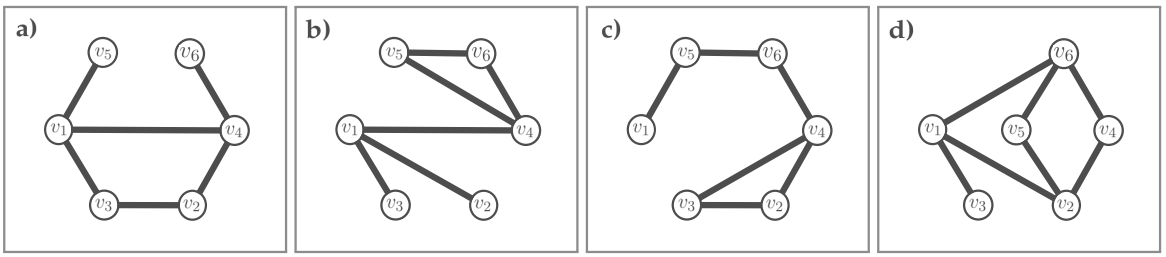

Figura 7.4: Gráficas conexas de menor orden que son amoebas globales pero no locales. Nótese que 3 de ellas **a)-c)** tienen tamaño 6, mientras que **d)** la última tiene tamaño 7.

Por último, es de importancia mencionar que el análisis de todas las gráficas con 1 a 10 vértices se completó en aproximadamente 2 a 3 semanas, usando un servidor del Laboratorio Nacional de Visualización Científica Avanzada (LAVIS) de la UNAM, que cuenta con una capacidad de 125GB de memoria RAM en su totalidad. Para lograr esto, adicionalmente se particionó la colección de las gráficas no isomorfas de grado 10 en 24 lotes, específicamente: 23 lotes con 500,000 gráficas cada uno, y otro lote de 505,168 gráficas, lo que nos permitió llevar a cabo el análisis simultáneo de estos lotes en dicho servidor, mientras que las gráficas de 1 a 9 vértices se analizaron en un mismo lote.

7.4. Amoebas en Conjuntos de Árboles

Es posible encontrar también en la base de datos pública "Combinatorial Data - Trees Page" del Prof. Brendan D. McKay de la Universidad Nacional Australiana [7], a todos los árboles no isomorfos teniendo entre 4 y 22 vértices. En esta tesis llevamos a cabo un análisis, similar al de la sección anterior, sobre todos estos árboles por medio de nuestras implementaciones para el **Algoritmo 14**, incluyendo también a los árboles con entre 1 y 3 vértices. No obstante, se debe recordar que por el **Corolario 4.1.2**, todo árbol amoeba local es también amoeba global. Así, una de las primeras tareas planteadas, fue determinar al mínimo árbol que es amoeba global pero no local. Este se encontró de orden 10 y se muestra en la **Figura 7.5**. Igualmente, dicho corolario tiene una repercusión sobre la manera de clasificar a todos los árboles amoeba, como veremos más adelante en la **Tabla 7.3**.

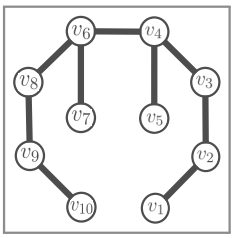

Figura 7.5: Se muestra el árbol de orden mínimo que es una amoeba global pero no local. A diferencia de las gráficas en la **Figura 7.4**, este es el único árbol de orden 10 con la propiedad buscada.

Ahora se muestran las cantidades de árboles amoeba teniendo entre 1 y 22 vértices. Denotamos nuevamente por **GA** a la familia de los árboles que son amoebas globales independientemente de si también son amoebas locales o no. Sin embargo, debe notarse que como todo árbol amoeba local es también global, la familia **LA** de árboles amoeba local es igual a la familia **LA∩GA**. Por esta misma razón, la familia **LA∪GA** de todas las amoebas es igual a la familia **GA** misma. Igualmente, es importante notar que en este caso se añade una última columna mostrando la cantidad de amoebas en la familia **GA** $\setminus$ **(LA∩GA)**, i.e., de árboles amoeba global que no son amoeba local.

Orden	LA = LA∩GA	GA = LA∪GA	GA $\setminus$ (LA∩GA)	Árboles No Isomorfos
1	1	1	0	**1**
2	1	1	0	**1**
3	1	1	0	**1**
4	1	1	0	**2**
5	2	2	0	**3**
6	3	3	0	**6**
7	4	4	0	**11**
8	9	9	0	**23**
9	16	16	0	**47**
10	27	28	1	**106**
11	48	48	0	**235**
12	76	80	4	**551**
13	150	152	2	**1,301**
14	243	252	9	**3,159**
15	410	430	20	**7,741**
16	681	750	69	**19,320**
17	1,161	1,269	108	**48,629**
18	1,800	2,109	309	**123,867**
19	2,919	3,372	453	**317,955**
20	4,531	5,593	1,062	**823,065**
21	6,992	8,616	1,624	**2,144,505**
22	10,976	14,224	3,248	**5,623,756**
Total	30,052	36,961	6,909	**9,114,285**

Tabla 7.3: Cantidades de amoebas en el conjunto de árboles no isomorfos de cada orden. El último renglón muestra el total de cada familia respecto a todos los árboles no isomorfos teniendo entre 1 y 22 vértices.

De nueva cuenta expresamos las cantidades anteriores como porcentajes del total de árboles no isomorfos de cada orden, mostradas en la **Tabla 7.4**. En la **Figura 7.6** se grafican dichos porcentajes como función del orden de las amoebas en cada una de las familias.

Orden	[%] LA = [%] LA∩GA	[%] GA = [%] LA∪GA	[%] GA \ (LA∩GA)	Árboles No Isomorfos
1	100.00	100.00	0.00	**1**
2	100.00	100.00	0.00	**1**
3	100.00	100.00	0.00	**1**
4	50.00	50.00	0.00	**2**
5	66.67	66.67	0.00	**3**
6	50.00	50.00	0.00	**6**
7	36.36	36.36	0.00	**11**
8	39.13	39.13	0.00	**23**
9	34.04	34.04	0.00	**47**
10	25.47	26.42	0.94	**106**
11	20.43	20.43	0.00	**235**
12	13.79	14.52	0.73	**551**
13	11.53	11.68	0.15	**1,301**
14	7.69	7.98	0.28	**3,159**
15	5.30	5.55	0.26	**7,741**
16	3.52	3.88	0.36	**19,320**
17	2.39	2.61	0.22	**48,629**
18	1.45	1.70	0.25	**123,867**
19	0.92	1.06	0.14	**317,955**
20	0.55	0.68	0.13	**823,065**
21	0.33	0.40	0.08	**2,144,505**
22	0.20	0.25	0.06	**5,623,756**
Total	0.33	0.41	0.08	**9,114,285**

Tabla 7.4: Proporciones de amoebas en el conjunto de árboles no isomorfos de cada orden, expresadas como porcentajes redondeados a dos decimales (hacia arriba para tercer decimal ≥ 5). Nótese que el total de amoebas (**LA∪GA**) de grado 20 en adelante es menor al 1 % de los árboles no isomorfos de cada orden.

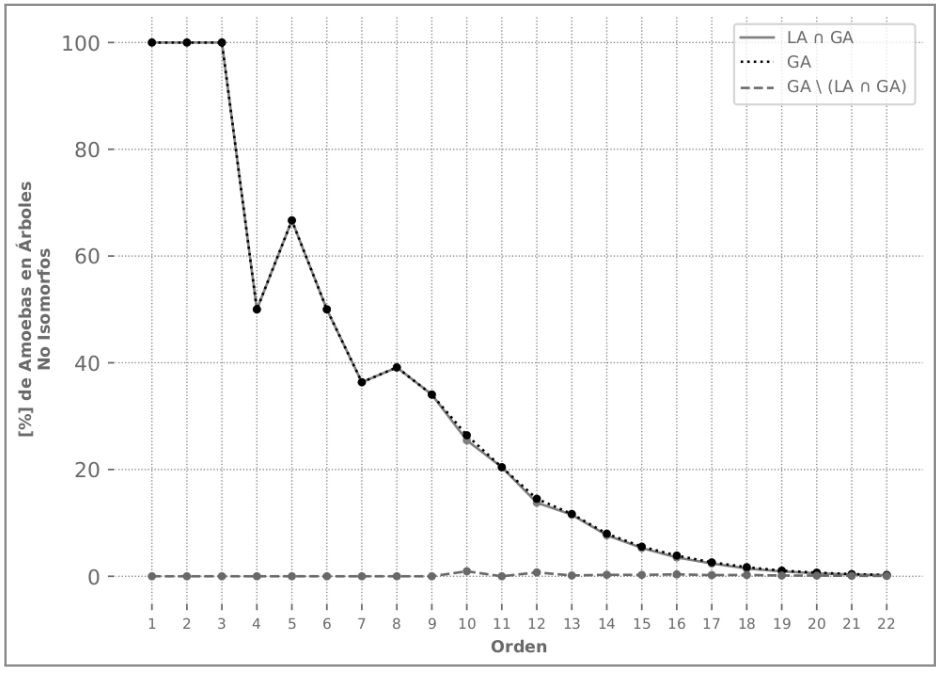

Figura 7.6: Porcentajes de la **Tabla 7.4** como función del orden. Los valores **[%] GA** y **[%] LA∩GA** son similares, con pequeñas diferencias desde el orden 10 en adelante reflejadas por el valor **[%] GA \ (LA∩GA)**.

Para concluir esta sección debemos añadir que el análisis de todos los árboles con 1 a 22 vértices se completó en aproximadamente 5 meses, haciendo uso nuevamente de un servidor del Laboratorio Nacional de Visualización Científica Avanzada (LAVIS) de la UNAM, contando con 125GB de memoria RAM. Específicamente, este análisis consistió de dos partes: la detección de árboles amoeba local que tomó alrededor de 3 meses, y por otro lado la detección de árboles amoeba sólo globales, que se completó en cerca de 2 meses. No obstante, se debe mencionar que para lograr esto, se invirtió más tiempo en particionar estos árboles en múltiples lotes (23 para los árboles locales y 37 para los sólo globales) todos con aproximadamente la misma cantidad de árboles, requiriendo sin embargo, volver a particionar algunos de estos lotes con propósito de mejorar el tiempo de análisis, y optimizar y redistribuir la carga computacional sobre el servidor utilizado.

7.5. Amoebas Raras y la Estructura del Grupo S_G

En esta sección mostramos los resultados de analizar las gráficas arbitrarias con 1 a 10 vértices, obtenidas de [6], pero esta vez haciendo uso de implementaciones del **Algoritmo 13** diseñadas, primero para determinar los reemplazos raros y ordinarios de una gráfica G, y luego para obtener la estructura del grupo de alcanzabilidad S_G de G como subgrupo de S_n, dada G de orden n.

Detección de Gráficas Raras.

Para poder determinar computacionalmente si cada reemplazo admisible, era ordinario o raro, se evaluó directamente la **Definición 5.2.1** de estos tipos de reemplazos. Para mostrar como funciona dicha evaluación, nótese que dadas una gráfica G de orden n con $V(G) = \{v_1, ..., v_n\}$ y su representación computacional G' con $V(G') = \{1, ..., n\}$ (como planteada en la **Sección 7.1**), se tiene que un reemplazo admisible de aristas $rs \to kl$ de G es un reemplazo ordinario en esta gráfica, si y sólo si, existe por lo menos un isomorfismo σ de $G' - rs + kl$ en G' tal que $\{\sigma(k), \sigma(l)\} = \{\sigma(r), \sigma(s)\}$, lo que se sigue inmediatamente de la **Definición 5.2.1** y de la **Proposición 7.1.1**.

Luego, como nuestros programas obtienen ya a todos los isomorfismos σ de $G' - rs + kl$ en G' para cada reemplazo admisible de aristas $rs \to kl$ de G con motivo de determinar si G misma es una amoeba o no, sólo se requirió revisar iterativamente la condición $\{\sigma(k), \sigma(l)\} = \{\sigma(r), \sigma(s)\}$ por cada uno de estos isomorfismos, terminando dicha iteración y concluyendo que el reemplazo $rs \to kl$ de G es ordinario si alguno la verificaba, o bien raro si ninguno de ellos la satisfacía. Así, al incorporar esta evaluación en nuestros programas, pudimos determinar las cantidades de gráficas raras y ordinarias (ver **Definición 5.2.2**) existentes entre las gráficas no isomorfas con 1 a 10 vértices, y más aún, pudimos decir si cada una de estas era una amoeba o no. Resumimos esta información en la **Tabla 7.5** usando la misma notación de columnas que la de la **Tabla 7.1**, pero agregamos la columna **No Amoeba** con la cantidad de gráficas raras que no son amoebas de ningún tipo.

Orden	No Amoeba	LA	GA	LA∩GA	LA∪GA	Gráficas No Isomorfas
1	0	0	0	0	0	**1**
2	0	0	0	0	0	**2**
3	0	0	0	0	0	**4**
4	0	0	0	0	0	**11**
5	0	0	0	0	0	**34**
6	0	1	1	1	1	**156**
7	0	6	3	3	6	**1,044**
8	24	28	9	7	30	**12,346**
9	358	74	19	14	79	**274,668**
10	3,859	173	54	41	186	**12,005,168**
Total	4,241	282	86	66	302	**12,293,434**

Tabla 7.5: Cantidades de gráficas raras en el conjunto de todas las gráficas no isomorfas teniendo entre 1 y 10 vértices. Nótese que el total de gráficas raras en cada orden se obtiene sumando la cantidad en la columna **No Amoeba** con la de la columna **LA∪GA**. Aunque no hay gráficas raras con entre 1 y 5 vértices, se incluyen estos renglones para mostrar la cantidad de gráficas no isomorfas, y por ende ordinarias, con esos vértices.

De esta tabla se sigue que las gráficas con 1 a 5 vértices son todas ordinarias. Luego, la primera gráfica rara se encuentra en el orden 6, y adicionalmente, esta cumple ser una amoeba local y global. Sin embargo, dicha gráfica no es un árbol, por lo que también se buscó determinar al mínimo árbol que tuviera reemplazos raros, que se encontró en el orden 9, y de nueva cuenta, este resultó ser además una amoeba local y global. Ambas gráficas se muestran en la **Figura 7.7**.

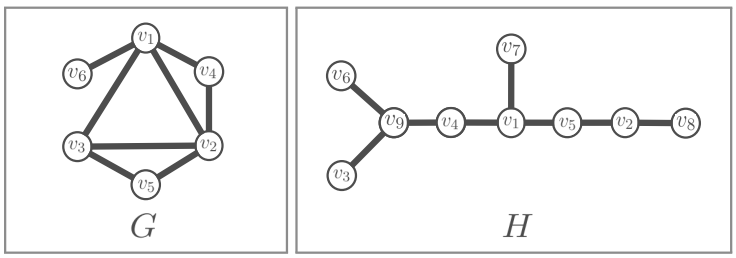

Figura 7.7: Se muestran la mínima gráfica rara G y el mínimo árbol raro H. Usando los programas desarrollados como parte del trabajo en esta tesis, se puede verificar que ambas son, adicionalmente, amoebas locales y globales. Aunado a esto, es posible ver que el reemplazo $25 \to 36$ es un reemplazo raro de G. Más aún, H es el mínimo árbol T^1 de la familia infinita $\mathcal{T}$ de árboles raros definida y analizada en la **Sección 5.5** del capítulo 5, por lo que sabemos que el reemplazo $15 \to 23$ es un reemplazo raro de H.

Ahora bien, dado que con nuestros programas se obtiene ya la lista de todos los reemplazos admisibles y no triviales de aristas de cada gráfica rara con 1 a 10 vértices, podemos también determinar cuántos de estos son ordinarios y cuántos de estos son raros. Un aspecto que llamó nuestra atención es que la cantidad de reemplazos ordinarios (no triviales) en estas gráficas raras, toma todos los valores desde 0 hasta 26 excepto por el 23, mientras que la cantidad de reemplazos raros en ellas toma sólo los valores 2, 4, 6, 8 y 12, es decir, más de un sólo reemplazo raro y únicamente cantidades pares de estos, sin que haya gráficas raras con un número impar de reemplazos raros.

Si bien esto puede ser clara consecuencia de la reducida cantidad de gráficas raras que se obtiene con entre 1 y 10 vértices, un factor que ayuda a explicar parcialmente estos resultados es el **Teorema 5.3.2**, aportación original de esta tesis y que indica que los reemplazos inversos (ver **Definición 3.5.1**) de todo reemplazo raro, son igualmente reemplazos raros de la gráfica a la que pertenecen, permitiéndonos obtener, por lo menos en estas gráficas, cantidades pares de reemplazos raros.

Por otro lado, debe notarse que en la **Figura 3.8** del capítulo 3, se presentan 4 gráficas, específicamente: una que no es amoeba, una amoeba sólo global, una sólo local, y una amoeba local y global. Más aún, como tales gráficas son de orden 4, con base en la **Tabla 7.5** podemos concluir que estas son todas ordinarias, i.e., gráficas que no tienen reemplazos raros. Adicionalmente, de la **Tabla 7.5** se sigue que también existen gráficas raras de todos los tipos, es decir: amoebas raras sólo locales, amoebas raras sólo globales, amoebas raras locales y globales, y gráficas raras que no son amoebas. En la **Figura 7.8** se incluyen ejemplos de todos estos tipo de gráficas raras.

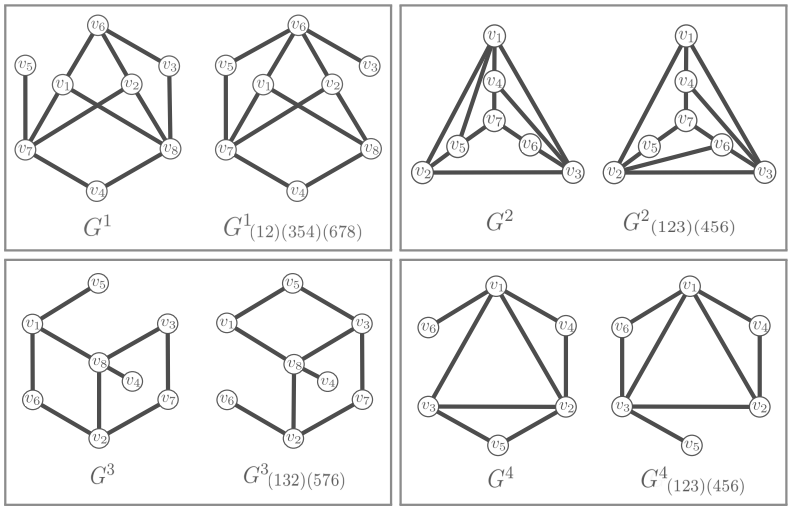

Figura 7.8: Las gráficas en esta figura se enumeran como G^i para $1 \leq i \leq 4$. La gráfica G^1 no es una amoeba, y además $G^1_{(12)(354)(678)}$ puede ser obtenida de G^1 por el reemplazo raro $38 \rightarrow 56$. Por otro lado G^2 es sólo amoeba local, y $G^2_{(123)(456)}$ puede ser obtenida de ella por el reemplazo raro $15 \rightarrow 26$. Igualmente G^3 es sólo amoeba global, y de esta podemos obtener $G^3_{(132)(576)}$ por el reemplazo raro $16 \rightarrow 35$. Finalmente G^4 es ambas amoeba local y global, mientras que $G^4_{(123)(456)}$ se obtiene de ella por el reemplazo raro $25 \rightarrow 36$.

Estudio de la Estructura del Grupo S_G.

Los resultados anteriores indican que en lo general, la propiedad de una gráfica G, de tener o no reemplazos raros, no tiene repercusión sobre si G es o no una amoeba, ni sobre su tipo de amoeba de ser el caso, y viceversa. Luego, es natural cuestionar la relevancia de considerar a las permutaciones asociadas a reemplazos raros de G al momento de generar el grupo de alcanzabilidad S_G.

Esta fue una de las motivaciones para implementar rutinas del software SageMath [32], dentro de nuestros programas de detección de amoebas con propósito de determinar las estructuras únicas (salvo isomorfismos) de los grupos S_G y $S_{G\dot\cup K_1}$ como subgrupos de S_n y de S_{n+1} respectivamente, dada G de orden n. Antes de continuar debemos enfatizar que estos programas no determinan la estructura de dichos grupos sólo por su cardinalidad, sino que SageMath devuelve cadenas de texto indicando tales estructuras después de completar una estrategia de búsqueda [32] basada en Teoría de Grupos, pero cuyo estudio queda fuera del alcance de esta tesis. Esto nos proporciona un medio alterno para analizar dichos grupos, y a la vez nos permite contrastar los grupos obtenidos exhaustivamente por nuestros algoritmos, contra la teoría esperada para ellos con motivo de verificar el correcto funcionamiento de nuestros programas. Por ejemplo, nótese que el grupo de alcanzabilidad de todo ciclo con $n \geq 4$ vértices (ver **Definición 2.1.19**), es isomorfo al grupo dihédrico D_n (ver **Definición 2.2.18**), lo que se sigue de la **Proposición 3.5.3** y la **Observación 2.2.5** al ver que estos sólo tienen reemplazos triviales. Luego, si sabemos que una gráfica G es un ciclo con $n \geq 4$ vértices, SageMath nos deberá devolver cadenas de texto indicando que en efecto $S_G \cong D_n$.

Así, con estos programas podemos finalmente determinar las estructuras, en principio desconocidas, que tendrían los grupos S_G y $S_{G\dot\cup K_1}$ de generarse considerando sólo las permutaciones asociadas a sus respectivos reemplazos raros, o bien sólo a sus reemplazos ordinarios, siempre que G o $G\dot\cup K_1$ sean gráficas raras. Debemos añadir que en primer lugar, para poder determinar los reemplazos raros de cada una de estas gráficas, nuestros programas hacen uso del siguiente resultado teórico, incluido aquí ya que su relevancia gira completamente en torno a estos programas.

Proposición 7.5.1. *Sea G una gráfica de orden n con $V(G) = \{v_1, ..., v_n\}$ y sea $G \dot\cup K_1$ la gráfica obtenida al agregar el vértice aislado v_{n+1} a G. Sea además $rs \to kl$ un reemplazo admisible de aristas de $G \dot\cup K_1$ tal que $v_{n+1} \notin \{v_k, v_l\}$. Entonces $rs \to kl$ es un reemplazo ordinario (respectivamente raro) de $G \dot\cup K_1$, si y sólo si, $rs \to kl$ es un reemplazo ordinario (resp. raro) de G.*

Demostración. Por la **Proposición 6.3.1** sabemos que $rs \to kl$ es admisible en ambas gráficas. Supóngase que $rs \to kl$ es ordinario en G. Por la **Observación 5.2.1** debe existir un isomorfismo φ de $G - v_rv_s + v_kv_l$ en G tal que $\varphi(v_k)\varphi(v_l) = v_rv_s$. Luego, como se indica en la **Proposición 6.3.1**, la biyección $\varphi' : V(G \dot\cup K_1 - v_rv_s + v_kv_l) \to V(G \dot\cup K_1)$ dada por $\varphi'(v_i) = \varphi(v_i)$ para todo $v_i \in V(G)$ y con $\varphi'(v_{n+1}) = v_{n+1}$, es un isomorfismo de $G \dot\cup K_1 - v_rv_s + v_kv_l$ en $G \dot\cup K_1$, y por la **Observación 5.2.1** misma, se sigue que $rs \to kl$ es ordinario en $G \dot\cup K_1$. Por otro lado, de suponer que $rs \to kl$ es ordinario en $G \dot\cup K_1$, debe existir un isomorfismo φ de $G \dot\cup K_1 - v_rv_s + v_kv_l$ en $G \dot\cup K_1$ tal que $\varphi(v_k)\varphi(v_l) = v_rv_s$, y sin pérdida de generalidad podemos suponer además que $\varphi(v_{n+1}) = v_{n+1}$. Pero como indicado en la **Proposición 6.3.1**, la restricción $\varphi|_{V(G)}$ es un isomorfismo de $G - v_rv_s + v_kv_l$ en G, y nuevamente por la **Observación 5.2.1** concluimos que $rs \to kl$ es ordinario en G. ■

Así, para determinar con el **Algoritmo 13**, si un reemplazo admisible de aristas $rs \to kl$ es raro u ordinario en estas gráficas, únicamente debemos revisar esto con respecto a la gráfica $G\dot{\cup}K_1$, y de ocurrir $v_{n+1} \notin \{v_k, v_l\}$ podremos inmediatamente considerar a $rs \to kl$ como un reemplazo raro u ordinario de G misma, según sea el caso. Con esto finalmente podemos obtener 4 conjuntos disjuntos de permutaciones, específicamente las asociadas a reemplazos raros y a reemplazos ordinarios de G, llámense respectivamente $\mathcal{X}_G^R$ y $\mathcal{X}_G^O$, y los conjuntos análogos $\mathcal{X}_{G\dot{\cup}K_1}^R$ y $\mathcal{X}_{G\dot{\cup}K_1}^O$ para $G\dot{\cup}K_1$.

Luego, con estos conjuntos todavía podemos obtener S_G y $S_{G\dot{\cup}K_1}$ como $S_G = \langle \mathcal{X}_G^R \dot{\cup} \mathcal{X}_G^O \rangle$ y $S_{G\dot{\cup}K_1} = \langle \mathcal{X}_{G\dot{\cup}K_1}^R \dot{\cup} \mathcal{X}_{G\dot{\cup}K_1}^O \rangle$. Pero además podemos determinar por medio de SageMath, la estructura de los grupos $S_G^R = \langle \mathcal{X}_G^R \rangle$ y $S_G^O = \langle \mathcal{X}_G^O \rangle$ de G, y la de los grupos $S_{G\dot{\cup}K_1}^R = \langle \mathcal{X}_{G\dot{\cup}K_1}^R \rangle$ y $S_{G\dot{\cup}K_1}^O = \langle \mathcal{X}_{G\dot{\cup}K_1}^O \rangle$ de $G\dot{\cup}K_1$, que representan la estructura que *tendrían* S_G y $S_{G\dot{\cup}K_1}$ de no considerar, repectivamente, las permutaciones asociadas a reemplazos ordinarios y a reemplazos raros de estas gráficas, como planteado en un inicio. Más aún, con base en esto podemos encontrar ejemplos de gráficas que dejan de ser amoebas al no considerar cada uno de estos tipos de reemplazos.

Para ver esto, finalizaremos esta sección analizando a las gráficas raras de la **Figura 7.9**. Ahí se muestra una gráfica G de orden 8, y otra H de orden 9. Además, con nuestros programas podemos determinar que G es una amoeba local y global, mientras que H es una amoeba local pero no global. Luego, SageMath debería devolver estructuras para sus grupos de alcanzabilidad, que fueran consistentes con estos resultados. En efecto, para G se obtiene $S_G \cong S_8$ y $S_{G\dot{\cup}K_1} \cong S_9$ como esperado. Por otro lado, nótese que para toda gráfica F de orden n, que es amoeba local pero no global, se tiene $S_F \cong S_{F\dot{\cup}K_1} \cong S_n$ debido a la **Observación 2.2.3** y la **Proposición 2.2.11**. Este es el caso de la gráfica H, para la que SageMath devuelve cadenas de texto indicando que $S_H \cong S_{H\dot{\cup}K_1} \cong S_9$.

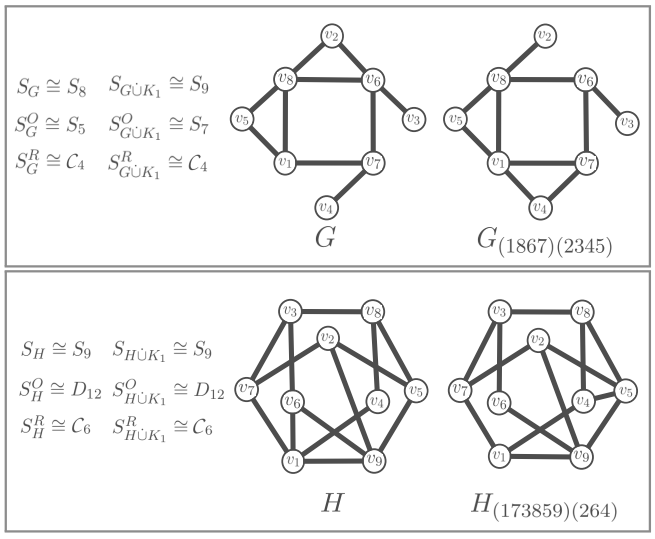

Figura 7.9: Se consideran dos gráficas raras G y H. La gráfica $G_{(1867)(2345)}$ se obtiene de G por el reemplazo raro $26 \to 14$, mientras que $H_{(173859)(264)}$ se obtiene de H por el reemplazos raro $16 \to 45$. Más adelante se discute el significado de los isomorfismos entre los grupos mostrados en esta figura.

Ahora bien, con nuestros programas es posible determinar que $G\dot{\cup}K_1$ y $H\dot{\cup}K_1$ no tienen reemplazos raros involucrando a sus respectivos vértices aislados. Es decir que los reemplazos raros de estas gráficas, como los descritos en la **Figura 7.9**, son todos reemplazos raros de G y de H, debido a la **Proposición 7.5.1**. Sin embargo, los cuatro grupos de alcanzabilidad: el de G, de H y los de $G\dot{\cup}K_1$ y $H\dot{\cup}K_1$, *cambiarían* todos su estructura de no considerar las permutaciones asociadas a los reemplazos raros, u ordinarios, de estas gráficas, y por ende G y H dejarían de ser amoebas.

En efecto, recuérdese que $\mathcal{C}_n$ es el grupo cíclico con n elementos, mientras que el grupo dihédrico se denota por D_n (ver **Definiciones 2.2.7** y **2.2.18**). Luego, al no considerar las permutaciones asociadas a reemplazos raros de G, con SageMath se obtiene $S_G^O \cong S_5$ y $S_{G\dot{\cup}K_1}^O \cong S_7$. Por otro lado, si no se consideran las permutaciones asociadas a sus reemplazos ordinarios tenemos $S_G^R \cong S_{G\dot{\cup}K_1}^R \cong \mathcal{C}_4$. Pero esto mismo pasa para H, ya que si se generaran estos grupos sin las permutaciones asociadas a reemplazos raros, se obtendría $S_H^O \cong S_{H\dot{\cup}K_1}^O \cong D_{12}$, o bien $S_H^R \cong S_{H\dot{\cup}K_1}^R \cong \mathcal{C}_6$ de generar estos grupos sin considerar las permutaciones asociadas a sus reemplazos ordinarios.

Debido a todo esto, podemos concluir que no es posible generar a las permutaciones asociadas a reemplazos raros de estas gráficas, sólo con las permutaciones asociadas a sus reemplazos ordinarios, ni viceversa. Así, estos resultados implican que, en lo general, es necesario considerar ambos tipos de reemplazos para llevar a cabo la detección de amoebas correctamente.

Finalmente, debemos mencionar que este análisis fue idéntico al planteado en la **Sección 7.3**, excepto por la tarea adicional de determinar si cada gráfica teniendo entre 1 y 10 vértices era rara u ordinaria. No obstante, esto se completó en el mismo tiempo que el requerido para la detección de amoebas en el conjunto de gráficas arbitrarias, es decir cerca de 2 a 3 semanas, usando el mismo servidor del Laboratorio Nacional de Visualización Científica Avanzada (LAVIS) de la UNAM, y la misma partición de estos datos para optimizar la carga computacional sobre este servidor.

Capítulo 8

Discusión y Conclusión General

Recordemos que uno de los objetivos principales de esta tesis fue desarrollar e implementar algoritmos para llevar a cabo la detección computacional de las amoebas. Otro de los objetivos fue analizar a la familia de las gráficas que tienen reemplazos raros y estudiar la repercusión de este nuevo tipo de reemplazos sobre la detección de amoebas. Los resultados obtenidos respecto a ambos objetivos se presentaron ya en los capítulos 5, 6 y 7. Por ello, el propósito del presente capítulo es resumir brevemente estos resultados para desarrollar una discusión en torno a ellos, así como presentar algunos aspectos que se pueden abordar como trabajo a futuro.

8.1. Discusión de Resultados

Habiendo implementado los algoritmos mostrados en el capítulo 6, se llevó a cabo la detección de amoebas en la colección de todas las gráficas no isomorfas teniendo entre 1 y 10 vértices, tomadas de la base de datos pública [6]. Posteriormente se obtuvo la proporción de amoebas detectadas, como porcentaje del total de dichas gráficas en cada orden. En la **Figura 7.3** del capítulo 7 se visualizan tales porcentajes. Esta figura sugiere que la proporción de amoebas dentro del total de gráficas no isomorfas de cada orden, tiende a cero conforme aumenta este valor, debido a que todos los porcentajes mostrados se aproximan a cero de forma estríctamente decreciente.

Similarmente, en la figura **Figura 7.6** se visualiza el porcentaje de árboles amoeba, nuevamente como función del orden, detectados en la colección de todos los árboles no isomorfos teniendo entre 1 y 22 vértices, obtenidos de la base de datos [7]. Aunque en este caso los porcentajes no decrecen de forma estricta para todos los ordenes, estos valores sí se aproximan a cero sin volver a crecer después del orden 8, alcanzando incluso valores menores al 1 % desde el orden 20 en adelante.

Si bien ambos casos exhiben, conforme crece el orden, una reducción en el porcentaje de las amoebas existentes en sus respectivas familias, consideramos que estos resultados no nos permiten conjeturar el desvanecimiento de este porcentaje en el caso general. Esto se debe en particular al relativamente reducido número de gráficas que fueron analizadas, y más aún, debido al claramente limitado número de vértices (1 a 10 para gráficas arbitrarias y 1 a 22 para árboles) de estas gráficas.

De esta forma, para poder ser concluyentes al respecto del comportamiento de dicho porcentaje como función del orden, consideramos recomendable analizar primero una muestra de tamaño apropiado, de gráficas de mayor orden a las estudiadas aquí. Como razón de esto, debe tomarse en cuenta que las gráficas de un mayor orden podrían contar con propiedades que no ocurrirían, o que por lo menos no resulten intuitivas, en gráficas con una menor cantidad de vértices, y que faciliten a una gráfica ser clasificada como amoeba de algún tipo.

En efecto, una propiedad que promueve comportamientos poco intuitivos en las amoebas es aquella de contar con reemplazos raros, contribución original de esta tesis, definida y analizada en el capítulo 5. Más aún, de forma consistente a lo comentado en el párrafo anterior, se tiene la **Tabla 7.5**, que resume la cantidad de gráficas con reemplazos raros detectadas entre todas las gráficas no isomorfas teniendo entre 1 y 10 vértices tomadas de [6], y que además muestra cómo ninguna de las gráficas de orden 5 o menor poseen reemplazos raros.

Aunado a esto, debe notarse que tanto la gráfica de menor orden ($n = 6$), así como el árbol de menor orden ($n = 9$), que tienen reemplazos raros (ver **Figura 7.7**), fueron ambas amoebas locales y globales. Asimismo, debemos agregar que se encontraron ejemplos de gráficas, mostradas en la **Figura 7.9**, que dejarían de ser clasificadas como amoebas en caso de no utilizar las permutaciones asociadas a sus reemplazos raros. Por ello consideramos que todo esto es ejemplo, no sólo de la necesidad de tomar en cuenta a los reemplazos raros al detectar amoebas, sino también de la diversa colección de propiedades que permiten a una gráfica comportarse como una amoeba.

8.2. Una Posible Conjetura y su Consecuencia

Aquí presentamos una conjetura derivada, de la inspección de las amoebas detectadas en estas tesis tanto entre las gráficas con 1 a 10 vértices, como entre los árboles con 1 a 22 vértices. Se incluye aquí ya que esta fue propuesta originalmente por la Dra. Adriana Hansberg Pastor durante un seminario de Matemáticas Discretas en Gráficas y Grupos con Énfasis en Amoebas, de la Maestría en Ciencias Matemáticas de la UNAM impartido por la Dra. Adriana Hansberg y la Dra. Amanda Montejano. Debemos enfatizar que lo enunciado por dicha conjetura se verificó computacionalmente, al revisar con nuestros programas a todas las amoebas, locales y \ o globales, encontradas en dichos conjuntos de gráficas. A la fecha esta no se ha podido demostrar, ni se ha podido encontrar un contraejemplo para ella. Más aún, debe notarse que esta tendría una repercusión positiva en la detección de amoebas, al proporcionarnos una condición necesaria para esta familia de gráficas.

Conjetura 8.2.1. ***(Conjetura determinada por la Dra. Adriana Hansberg Pastor)***

Sea G una gráfica conexa de orden n con $V(G) = \{v_1, ..., v_n\}$. Si G es una amoeba (local o global), entonces todo vértice de G es adyacente con a lo más dos hojas.

Veremos ahora una consecuencia que tendría la **Conjetura 8.2.1** de ser cierta. Esta es contribución original de esta tesis y habla sobre una colección de árboles que no podrían ser amoebas, nuevamente proporcionándonos más condiciones para detectar computacionalmente a esta familia de gráficas. Acompañamos su demostración con el ejemplo de la **Figura 8.1**.

Consecuencia 8.2.1. *Sea T un árbol de orden n con $V(T) = \{v_1, ..., v_n\}$ y grado máximo $\Delta \geq 5$. Si T tiene secuencia de grados S de la forma $S = (\Delta, \Delta - 1, ..., 3, 2, 1, 1, ..., 1)$, constituida por $n - \Delta$ repeticiones del número 1 y donde las demás Δ entradas toman todos y cada uno de los valores en el intervalo $\{1, 2, 3, ..., \Delta\}$, entonces T no puede ser amoeba (local o global).*

Demostración. Para ver que existe por lo menos un árbol de esta forma, considérense $n = 12$ y $\Delta = 5$, respecto a lo que se tiene $S = (5, 4, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1)$. Luego, la suma de los elementos en esta secuencia es $22 = 2(12) - 2 = 2n - 2$, y por la **Proposición 2.1.4** podemos concluir que existe un árbol con S como secuencia de grados.

Ahora bien, supongamos que existe un árbol T cumpliendo con estas propiedades, y procediendo por contradicción supongamos además que T es una amoeba (local o global). Considérese el (único) vértice de grado Δ, digamos v_Δ, y llámese v_k al vértice con grado $d_T(v_k) = k$, para cada k en el intervalo $\{2, ..., \Delta - 1\}$. Supongamos que v_Δ no es adyacente con ninguna hoja en T. Entonces tendría a lo más grado $\Delta - 2$ al ser adyacente con todo v_k para $k \in \{2, ..., \Delta - 1\}$. Más aún, al ser T una amoeba, por la **Conjetura 8.2.1** el vértice v_Δ no puede ser adyacente con más de dos hojas.

Por ello v_Δ debe ser vecino de todo v_k para $k \in \{2, ..., \Delta - 1\}$, y ser vecino con exactamente dos hojas de T. Luego, como los vecinos de v_Δ no pueden ser adyacentes entre sí, y entre estos están ya todos los vértices de grado mayor a 1, cada v_k debe ser adyacente con $k - 1$ hojas. Así, v_2 es adyacente a v_Δ y a un hoja. Igualmente v_3 es adyacente a v_Δ y a dos hojas. Continuando de esta forma $v_{\Delta-1}$ debe ser adyacente a v_Δ y a $\Delta - 2$ hojas. Pero ya que $\Delta \geq 5$, tal vértice debe ser adyacente con $\Delta - 2 \geq 3$ hojas, contradiciendo la **Conjetura 8.2.1**. Por lo tanto T no puede ser una amoeba. ■

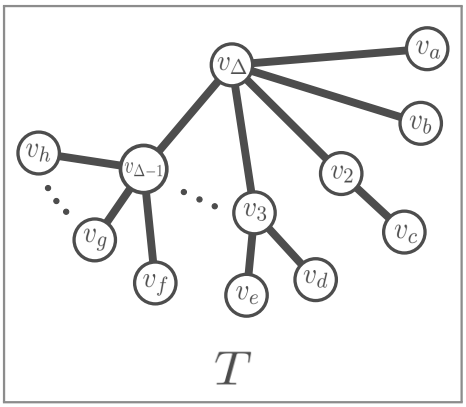

Figura 8.1: Se muestra el árbol T de la demostración para la **Consecuencia 8.2.1**. Nótese que el vértice v_Δ es el único vértice de grado Δ, y de igual forma se nombra como v_k al único vértice de grado k en T para cada $k \in \{2, ..., \Delta - 1\}$. Todos los demás vértices son hojas cuyo índice no es requerido por la demostración.

8.3. Trabajo a Futuro

Nótese que la importancia de la **Consecuencia 8.2.1**, radica en que no sólo nos permite descartar a una familia de árboles que no pueden ser amoebas, sino que nos permite hacer esto por medio de la secuencia de grados de dichos árboles, siendo que estos no podrían ser identificados sólo por el conjunto de grados de sus vértices, usando el **Algoritmo 14** como presentado actualmente en esta tesis. Por ello, dicha consecuencia junto con la **Conjetura 8.2.1** deben ser estudiadas a detalle en un futuro, ya que de ser ciertas ambas serían favorables para descartar árboles y gráficas arbitrarias que no pueden ser amoebas, sin tener que invertir más recursos computacionales en ellas.

A continuación se incluyen otras dos propiedades respecto a los grados de los vértices de un árbol, que sería beneficioso estudiar a futuro, ya que podrían ser utilizadas para optimizar todavía más la detección de amoebas de este tipo. Es importante notar que la primera propiedad se puede generalizar inmediatamente para árboles con grado máximo $\Delta \geq 3$.

Proposición 8.3.1. *Sea T un árbol de orden n con $V(T) = \{v_1, ..., v_n\}$ y grado máximo $\Delta = 3$. Sea n_k el número de vértices de T con grado k, para $k \in \{1, 2, 3\}$. Si T es amoeba (local o global), entonces se satisfacen las siguientes tres condiciones,*

$$n_1, n_2, n_3 \geq 1 \text{ para toda } k, \qquad n_2 + 2n_3 = n - 2, \qquad n_1 = n - n_2 - n_3$$

Demostración. La condición $n_k \geq 1$ se sigue de la **Proposición 4.2.2**. Por los **Teoremas 2.1.1** y **2.1.2** podemos poner $2n - 2 = \sum_{i=1}^{n} d_G(v_i) = n_1 + 2n_2 + 3n_3$. Pero al mismo tiempo $n_1 + n_2 + n_3 = n$, por lo que $n_1 = n - n_2 - n_3$. Luego, $2n - 2 = n_1 + 2n_2 + 3n_3 = [n_1 + n_2 + n_3] + n_2 + 2n_3 = n + n_2 + 2n_3$, y por lo tanto $n_2 + 2n_3 = n - 2$, con lo que se verifican las tres condiciones planteadas. ■

Luego, las ecuaciones en la **Proposición 8.3.1** (y sus generalizaciones), podrían ser usadas en un futuro, no sólo para descartar árboles que no son amoebas, sino para generar computacionalmente todos los árboles amoeba para un orden n y un grado máximo $\Delta \geq 3$, por medio de sus secuencias de grados, al reproducir la totalidad de secuencias de enteros que cumplan estas condiciones.

Por otro lado, debemos mencionar que la siguiente es una propiedad, no sólo de árboles, sino de toda amoeba. Esta fue planteada originalmente por la Dra. Adriana Hansberg, el Dr. Yair Caro y la Dra. Amanda Montejano [1] para amoebas arbitrarias. No obstante, aquí la demostramos sólo para árboles, con propósito de complementar lo presentado hasta ahora al respecto de estas gráficas.

Proposición 8.3.2. *Sea T un árbol de orden $n \geq 2$ con $V(T) = \{v_1, ..., v_n\}$ y con grado máximo Δ. Si T es una amoeba (local o global), entonces $\Delta \leq \frac{1}{2}(1 + \sqrt{8n - 15})$.*

Demostración. Por el **Teorema 2.1.2** sabemos que todo árbol de orden n tiene $m = n - 1$ aristas. Por la **Proposición 4.2.2** sabemos que además $\{d_T(v_i) \mid v_i \in V(T)\} = \{1, ..., \Delta\}$ al ser T una amoeba, por lo que sin pérdida de generalidad podemos poner $d_T(v_k) = k$ para $k \in \{1, ..., \Delta\}$, e igualmente $d_T(v_i) \geq 1$ para $\Delta < i \leq n$ al ser T una gráfica conexa. Luego, usando el **Teorema 2.1.1** se sigue,

$$2m = \sum_{i=1}^{n} d_T(v_i) = \sum_{k=1}^{\Delta} k + \sum_{i=\Delta+1}^{n} d_T(v_i) \geq \sum_{k=1}^{\Delta} k + (n - \Delta) = \frac{\Delta(\Delta+1)}{2} + (n - \Delta),$$

$$\text{de donde, } 2m \geq \frac{\Delta^2+\Delta}{2} + \frac{(2n-2\Delta)}{2}, \text{ que es } 0 \geq \Delta^2 - \Delta + 2n - 4m.$$

Pero nótese que si $0 = \Delta^2 - \Delta + 2n - 4m$, entonces

$$\Delta = \tfrac{1}{2}(1 \pm \sqrt{1 - 8n + 16m}) = \tfrac{1}{2}(1 \pm \sqrt{1 - 8n + 16n - 16}) = \tfrac{1}{2}(1 \pm \sqrt{8n - 15})$$

Así, para Δ positivo y considerando la desigualdad de antes concluimos $\Delta \leq \frac{1}{2}(1 + \sqrt{8n - 15})$. ■

Como consecuencia de todo esto podemos ver que sería deseable en un futuro, no sólo utilizar los resultados anteriores para descartar árboles que no pueden ser amoebas en base a los grados de sus vértices, sino que con estas propiedades y apoyados en la **Consecuencia 8.2.1**, se pudieran sentar las bases para una caracterización de los árboles amoeba por medio de sus secuencias de grados.

Ahora bien, es importante mencionar que una tarea que queda pendiente por realizar, posiblemente relacionada con lo comentado hasta ahora, es determinar todos los árboles raros, amoeba y no amoeba, teniendo desde 11 hasta 22 vértices. Esto podría ayudar a comprender de mejor forma los mecanismos y propiedades que facilitan a un árbol comportarse como una amoeba, y adicionalmente permitiría estudiar en más detalle los efectos de los reemplazos raros en estas gráficas.

Para concluir, debemos enfatizar que todos los programas en nuestro repositorio de Github: **Detection of Amoeba Graphs** [3] fueron desarrollados en lenguaje Python. Si bien este cuenta con una sintáxis que permite crear programas legibles, es sabido que la forma en que una computadora o servidor ejecutan programas en Python, puede resultar un poco lenta debido a las características propias del lenguaje. Una solución para esto podría ser recurrir a otros lenguajes de programación como C o C++. Sin embargo, es posible que tales lenguajes no cuenten con todas las facilidades de Python, por lo que en su lugar puede resultar más favorable llevar a cabo implementaciones en este mismo lenguaje, pero considerando ahora el procesamiento en paralelo con Python [34], por ejemplo, de los reemplazos de aristas que sí tienen posibilidad de ser reemplazos admisibles. Así, aunque dichas mejoras se encuentran alejadas de los aspectos teóricos de las amoebas, estas podrían ser herramientas beneficiosas para la detección computacional de esta familia de gráficas.

Referencias

[1] Yair Caro, Adriana Hansberg, Amanda Montejano, "Graphs isomorphisms under edge-replacements and the family of amoebas," p. 33, 2021. https://doi.org/10.48550/arXiv.2007.11769.

[2] Y. Caro, A. Hansberg, and A. Montejano, "Unavoidable chromatic patterns in 2-colorings of the complete graph," *Journal of Graph Theory*, vol. 97, pp. 123–147, 2021. https://onlinelibrary.wiley.com/doi/abs/10.1002/jgt.22645.

[3] "Github Repository: Detection of Amoeba Graphs - @MarcosLaffitte," https://github.com/MarcosLaffitte/Amoebas, Sitio visitado en: 10/06/2022.

[4] "Sitio de Python," https://www.python.org/, Sitio visitado en: 17/07/2021.

[5] "Sitio de Github," https://github.com/, Sitio visitado en: 17/07/2021.

[6] "Graphs Page - Prof. Brendan D. McKay, Universidad Nacional Australiana," https://users.cecs.anu.edu.au/~bdm/data/graphs.html, Sitio visitado en: 17/07/2021.

[7] "Trees Page - Prof. Brendan D. McKay, Universidad Nacional Australiana," https://users.cecs.anu.edu.au/~bdm/data/trees.html, Sitio visitado en: 17/07/2021.

[8] Mat. Jennifer Lilith Espinosa Hernández, "Gráficas Inevitables en 2-coloraciones de la Gráfica Completa: El Caso de las Amoebas," *Tesis de Licenciatura en Matemáticas, UNAM. Directora de Tesis: Dra. Adriana Hansberg Pastor*, p. 79, 2021. http://132.248.9.195/ptd2021/junio/0812763/Index.html, Sitio visitado en: 17/07/2021.

[9] J. A. Bondy and U. S. R. Murty, *Graph Theory*. Springer, 1 ed., 2008.

[10] Francesc Comellas et. al, *Matemática Discreta*. Edicions UPC, 1 ed., 2001.

[11] Sarada Herke, "Graph Theory: 44. Degree Sequence of a Tree." https://www.youtube.com/watch?v=cCG4_mj9TgM. Sitio visitado en: 17/07/2021.

[12] John B. Fraleigh, *A First Course in Abstract Algebra*. Addison-Wesley, 1 ed., 1988.

[13] Cárdenas et.al, *Álgebra Superior*. Editorial Trillas, 2 ed., 1995.

[14] Keith Conrad, "Generating Sets." https://kconrad.math.uconn.edu/blurbs/grouptheory/genset.pdf. Sitio visitado en: 17/07/2021.

[15] Michael Sipser, *Introduction to the Theory of Computation*. Cengage Learning, 3 ed., 2013.

[16] Deitel, Paul and Deitel, Harvey, *How to Program C*. Deitel, 6 ed., 2008.

[17] Brassard, Gilles and Bratley, Paul, *Fundamentals of Algorithmics*. Prentice Hall, 1 ed., 1995.

[18] Kleinberg, John and Tardos, Éva, *Algorithm Design*. Pearson, 1 ed., 2006.

[19] A. Aho, J. Hopcroft, and J. Ullman, *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1 ed., 1974. Example 3.2 pp. 84-86.

[20] Cordella et. al, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1367–1372, 2004. doi: 10.1109/TPAMI.2004.75.

[21] Cordella et.al, "Performance evaluation of the vf graph matching algorithm," *Proceedings 10th International Conference on Image Analysis and Processing*, pp. 1172–1177, 1999. doi: 10.1109/ICIAP.1999.797762.

[22] Cordella et.al, "An in-depth comparison of subgraph isomorphism algorithms in graph databases," *Proceedings of the VLDB Endowment*, p. 133–144, 2012. doi: 10.14778/2535568.2448946.

[23] Cordella et.al, "An improved algorithm for matching large graphs," p. 8, 2001. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.5342&rep=rep1&type=pdf`, Sitio visitado en: 17/07/2021.

[24] "NetworkX Analysis in Python - ver **Isomorphism**," `https://networkx.org/documentation/stable/reference/algorithms/isomorphism.html`, Sitio visitado en: 17/07/2021.

[25] "NetworkX Analysis in Python - ver **Tree Isomorphism** y notas en la documentación de la función `tree_isomorphism(t1,t2)`," `https://networkx.org/documentation/stable/reference/algorithms/isomorphism.html`, Sitio visitado en: 17/07/2021.

[26] "Sympy Documentation - buscar **PermutationGroup**," `https://docs.sympy.org/latest/index.html`, Sitio visitado en: 17/07/2021.

[27] Derek F. Holt et.al, *Handbook of Computational Group Theory*. CRC Press, 1 ed., 2005.

[28] "Combinatorial Data - Prof. Brendan D. McKay, Universidad Nacional Australiana," `https://users.cecs.anu.edu.au/~bdm/data/`, Sitio visitado en: 17/07/2021.

[29] "Description of graph6 encoding - Prof. Brendan D. McKay, Universidad Nacional Australiana," `https://users.cecs.anu.edu.au/~bdm/data/formats.txt`, Sitio visitado en: 17/07/2021.

[30] "Código ASCII," `https://www.asciitable.com/`, Sitio visitado en: 17/07/2021.

[31] "Biblioteca NetworkX del Lenguaje Python," `https://networkx.org/`, Sitio visitado en: 17/07/2021.

[32] "Sitio Principal de SageMath," `https://www.sagemath.org/`. Para la función que obtiene estructuras de grupos: `https://doc.sagemath.org/pdf/en/reference/groups/groups.pdf` al buscar **structure_description** y en el enlace `https://www.gap-system.org/Manuals/doc/ref/chap39.html#X87BF1B887C91CA2E` debajo de la sección **39.6 Structure Descriptions**. Sitios visitados en: 17/07/2021.

[33] "Sitio de Matplotlib," `https://matplotlib.org/`, Sitio visitado en: 17/07/2021.

[34] "Multiprocessing in Python," Machine Learning Mastery, by Jason Brownlee. `https://machinelearningmastery.com/multiprocessing-in-python/`, Sitio visitado en: 17/07/2021.